BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF COMPUTER SYSTEMS
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

AUTOMATED DESIGN METHODOLOGY FOR APPROXIMATE LOW POWER CIRCUITS

METODOLOGIE PRO AUTOMATICKÝ NÁVRH NÍZKOPŘÍKONOVÝCH APROXIMATIVNÍCH OBVODŮ

PHD THESIS
DISERTAČNÍ PRÁCE

AUTHOR **Ing. VOJTĚCH MRÁZEK**
AUTOR PRÁCE

SUPERVISOR **prof. Ing. LUKÁŠ SEKANINA, Ph.D.**
ŠKOLITEL

BRNO 2018

Abstract

The rapid expansion of modern embedded and battery-powered systems has brought new challenges for design methods oriented to low power circuits and systems. Although these methods systematically apply various power optimization techniques, the overall power requirements are still growing because of the increased complexity of integrated circuits. It has been shown that many applications are inherently error resilient and this property can be exploited for further power consumption reduction. This principle is systematically investigated in the nascent field of approximate computing. This thesis deals with efficient design methods for approximate circuits. The proposed methods are based on evolutionary algorithms (EAs). Although EAs have been applied in logic synthesis and optimization of common as well as approximate circuits, their scalability is limited in these areas. The goal of this dissertation is to show that approximate logic synthesis based on evolutionary algorithms (particularly on genetic programming) can provide excellent tradeoffs between the error and power consumption of complex digital circuits. We analyzed four different applications that use digital circuits described at three different levels of abstraction. By means of Cartesian genetic programming we reduced power consumption of small transistor-level circuits that are typically used in a technology library. We combined evolutionary approximation with formal verification techniques in order to evolve high quality gate-level approximate circuits such as adders and multipliers and provide formal guarantees on the approximation error. These circuits were employed to reduce power consumption in neural image classifiers and discrete cosine transform blocks of the HEVC encoder. We proposed a new data-independent error metric -– the distance error -– and used it in the evolutionary approximation of complex median circuits that are suitable for low power signal processing. This doctoral thesis presents a coherent methodology for the design of approximate circuits at different levels of description which is also capable of providing formal guarantees on the approximation error.

Keywords

Evolutionary algorithms, Cartesian Genetic Programming, low-power circuits, logic synthesis, approximate computing, median filters, neural network, technology library, relaxed equivalence.

Reference

MRÁZEK, Vojtěch. *Automated Design Methodology for Approximate Low Power Circuits.* Brno, 2018. PhD thesis. Brno University of Technology, Faculty of Information Technology. Supervisor prof. Ing. Lukáš Sekanina, Ph.D.

Abstrakt

Rozšiřování moderních vestavěných a mobilních systémů napájených bateriemi zvyšuje požadavky na návrh těchto systémů s ohledem na příkon. Přestože moderní návrhové techniky optimalizují příkon, elektrická spotřeba těchto obvodů stále roste díky jejich složitosti. Nicméně existuje celá řada aplikací, kde nepotřebujeme získat úplně přesný výstup. Díky tomu se objevuje technika zvaná aproximativní (přibližné) počítání, která umožňuje za cenu zanesení malé chyby do výpočtu významně redukovat příkon obvodů. V práci se zaměřujeme na použití evolučních algoritmů v této oblasti. Ačkoliv již tyto algoritmy byly úspěšně použity v syntéze přesných i aproximativních obvodů, objevují se problémy škálovatelnosti — schopnosti aproximovat složité obvody. Cílem této disertační práce je ukázat, že aproximační logická syntéza založená na genetickém programování umožňuje dosáhnout vynikajícího kompromisu mezi spotřebou a chybou. Byla provedena analýza čtyř různých aplikacích na třech úrovních popisu. Pomocí kartézského genetického programování s modifikovanou reprezentací jsme snížili spotřebu malých obvodů popsaných na úrovni tranzistorů použitelných například v technologické knihovně. Dále jsme zavedli novou metodu pro aproximaci aritmetických obvodů, jako jsou sčítačky a násobičky, popsaných na úrovni hradel. S využitím metod formální verifikace navíc celý návrhový proces umožňuje garantovat stanovenou chybu aproximace. Tyto obvody byly využity pro významné snížení příkonu v neuronových sítích pro rozpoznávání obrázků a v diskrétní kosinově transformaci v HEVC kodéru. Pomocí nové chybové metriky nezávislé na rozložení vstupních dat jsme navrhli komplexní aproximativní mediánové filtry vhodné pro zpracování signálů. Disertační práce reprezentuje ucelenou metodiku pro návrh aproximativních obvodů na různých úrovních popisu, která navíc garantuje nepřekročení zadané chyby aproximace.

Klíčová slova

Evoluční algoritmy, kartézské genetické programování, obvody s nízkým příkonem, logická syntéza, aproximační počítání, mediánové filtry, neuronová síť, technologická knihovna, přibližná ekvivalence.

Citace

MRÁZEK, Vojtěch. *Automated Design Methodology for Approximate Low Power Circuits.* Brno, 2018. Disertační práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Školitel prof. Ing. Lukáš Sekanina, Ph.D.

Rozšířený abstrakt

Rozšiřování moderních vestavěných a mobilních systémů napájených bateriemi zvyšuje požadavky na návrh těchto systémů s ohledem na příkon. Přestože moderní návrhové techniky příkon optimalizují, elektrická spotřeba těchto obvodů stále roste díky jejich složitosti. Nicméně existuje celá řada aplikací, kde nepotřebujeme získat úplně přesný výstup. Díky tomu se objevuje technika zvaná aproximativní (přibližné) počítání, která umožňuje za cenu zanesení malé chyby do výpočtu významně redukovat příkon obvodů.

V práci se zaměřujeme na použití evolučních algoritmů pro návrh těchto obvodů. Ačkoliv již tyto algoritmy byly úspěšně použity v syntéze přesných i aproximativních obvodů, objevují se problémy škálovatelnosti reprezentace i evaluace evolučního návrhu. Cílem této disertace je ukázat, že aproximativní logická syntéza založená na evoluci umožňuje dosáhnout významné úspory elektrické energie a vynikajícího kompromisu mezi spotřebou a chybou pro obvody, které jsou použitelné v reálných aplikacích. Proto byly zavedeny nové techniky vyhodnocování jak kvality navrhovaných obvodů, tak i rychlého odhadu elektrických parametrů, jako je spotřeba či plocha na čipu.

Práce je koncipovaná jako komentovaný soubor sedmi nejvýznamnějších prací autora. Cílem výzkumu bylo upravit syntézu přesných logických obvodů s využitím kartézského genetického programování (CGP) a vytvořit nástroj umožňující syntézu aproximativních obvodů tak, aby mohla (i) pracovat na různých úrovních popisu, (ii) zpracovat rozumně velké obvody popsané na různých úrovních popisu a (iii) optimalizovat parametry, které odpovídají výsledné obvodové implementaci. Navržená metoda je použita pro aproximaci kombinačních obvodů popsaných na třech různých úrovních popisu — na úrovni tranzistorů, hradel a funkčních bloků.

Popis na úrovni tranzistorů je vhodný pro návrh a optimalizaci malých obvodů použitelných například v technologických knihovnách podporující aproximativní počítání. V článku I jsme představili modifikovanou verzi reprezentace pro CGP, která je vhodná pro rychlou diskrétní simulaci obvodů. Později jsme do algoritmu zavedli přesnější simulaci pomoci SPICE simulátoru a rychlou metodu odhadu dynamického příkonu obvodů. S tímto přístupem se nám podařilo snížit spotřebu aproximačních aritmetických obvodů, jako jsou čtyřbitové násobičky.

S využitím vyšší, hradlové, úrovně popisu jsme vytvořili velkou knihovnu osmibitových aproximativních aritmetických obvodů (článek II). S využitím vícekriteriální implementace CGP byly nalezeny stovky aproximativních násobiček a sčítaček, jejichž softwarové a hardwarové modely jsou dispozici ke stažení na internetu. Pomocí navržených obvodů jsme vypozorovali, že při aproximaci násobení v neuronové síti klasifikující obrázky, je vhodné zachovat přesné násobení nulou. Toto omezení jsme zahrnuli do aproximační syntézy založené na CGP, která vygenerovala aproximativní 8- a 12-bitové násobičky. Tyto obvody jsme aplikovali ve dvou různých architekturách neuronových sítí. Výsledné neuronové sítě výborný kompromis mezi spotřebou násobení a celkovou klasifikační přesností (článek III).

Abychom mohli optimalizovat i větší obvody, kde je zjištění chyby obvodu pomocí simulace výpočetně nezvládnutelné, byly vytvořeny nové verifikační techniky vhodné pro aproximativní obvody. V článku IV jsme použili vyhodnocování chyby obvodu s použitím SAT-solveru. Díky zavedení časového omezení verifikace do fitness funkce se podařilo snížit časovou náročnost evaluace. To přispělo k nalezení velmi kvalitních obvodů s garantovanou chybou. V další práci jsme použili reprezentaci obvodů založenou na binárních rozhodovacích diagramech (BDD). Analýza BDD umožňuje vypočítat jak průměrnou a maximální aritmetickou chybu, tak i přepínací aktivitu hradel, která byla následně použita pro od-

had spotřeby obvodu. S využitím této analýzy v evaluaci obvodů jsme navrhli sčítačky a odčítačky použitelné v diskrétní kosinově transformaci v HEVC kodéru (článek V).

V článku VI byl zkoumán vliv množiny funkcí použitelných v CGP na rychlost konvergence evolučního algoritmu. Ukázalo se, že pro aproximaci násobiček je vhodné kromě běžných hradel zahrnout do této množiny i složitější obvody, jako jsou úplné a poloviční sčítačky. Evoluce konvergovala rychleji a následná syntéza produkovala kvalitnější výsledky než při použití jednoduchých dvouvstupých hradel.

V posledním článku VII jsme ukázali schopnost evolučního algoritmu aproximovat komplexní obvody využívající speciální stavební bloky. Zaměřili jsme se na aproximativní mediánové filtry. Pro jejich evaluaci jsme navrhli novou chybovou metriku, která je nezávislá na rozložení vstupních dat a umožňuje přesně a rychle určovat aproximační chybu obvodů. Za pomoci této chybové metriky a navržené reprezentace v CGP byla evolučně získána sada mediánových filtrů, která byla použita při filtraci obrazových dat i údajů získaných ze senzoru vestavěného systému.

Tato práce ukazuje, že evoluční algoritmy mohou být použity pro optimalizaci příkonu digitálních obvodů popsaných na různých úrovních — od nejnižšího propojení tranzistorů, přes zapojení hradel až po využití specializovaných stavebních bloků. Na čtyřech reálných aplikacích jsme ukázali, že řešení navržené evolučními algoritmy jsou kvalitnější než řešení získaná konvenčními přístupy.

Automated Design Methodology for Approximate Low Power Circuits

Declaration

Hereby I declare that this PhD thesis was prepared as an original author's work under the supervision of prof. Ing. Lukáš Sekanina, Ph.D. The supplementary information was provided by co-supervisor doc. Ing. Zdeněk Vašíček, Ph.D. All the relevant information sources, which were used during preparation of this thesis, are properly cited and included in the list of references.

.....
Vojtěch Mrázek
June 15, 2018

Acknowledgements

I would like to thank my dissertation and research advisor professor Lukáš Sekanina for his invaluable suggestions and advices during this research. I am most grateful to my co-supervisor associate professor Zdeněk Vašíček for sharing his expertise and intuition not only when I was working on the thesis. This project would not be nearly as good without their help.

I would especially like to thank my family. The encouragement and support from my beloved wife Milana is a powerful source of inspiration and energy. Special thanks are devoted to my parents for their never-ending support.

Contents

Chapter 1

Introduction

This chapter starts with a brief motivation for the research conducted as a part of my doctoral study at the Faculty of Information Technology, Brno University of Technology in 2014-2018. In this chapter, the open problems are identified, and the research objectives are formulated. Finally, the thesis outline is introduced.

1.1 Motivation

Despite the rapid developments in the very-large-scale integration (VLSI) circuit technologies and in modern circuit design techniques, the overall energy consumption of integrated circuits is rapidly growing mainly due to their increasing complexity needed in current computing systems. At the same time, many computationally intensive applications, e.g. image recognition, signal processing and data mining, are widely implemented in these systems. Moreover, the expansion of modern battery-powered and smart devices such as mobile systems, IoT nodes and wearable electronics emphasizes the low-power requirements put on the computer-based systems.

It is essential to improve the energy efficiency of these systems implemented. Fortunately, many of the computationally intensive applications feature an intrinsic error-resilience property [5]. Since they process noisy or redundant data and users are willing to accept certain errors in some cases, an emerging paradigm, the so-called *approximate computing*, can be used for design of energy-efficient applications. At the circuit level, approximations (circuit simplifications) are intentionally introduced to find a good tradeoff between power consumption and error.

The approximations can be introduced to the circuit in various steps of the standard circuit design flow. In this thesis, we primarily focus on the technology independent logic synthesis step because it has one important advantage — the approximate circuit can be used in arbitrary ASIC as well as FPGA technology, because we can assume the technology dependent synthesis is performed by means of some well-optimized open source or commercial tool after the approximation is conducted. The common logic synthesis typically starts with a circuit description given at the register-transfer-logic (RTL) level, in the form of a source code written in hardware description language (HDL). The goal of the logic synthesis is to transform a typically suboptimal solution into a close to optimal implementation with respect to given synthesis goals. For example, if a gate-level netlist is the starting point, we speak about logic optimization, because the circuit is described at the gate level which directly represents logic expressions specifying its behavior. The ultimate goal of the logic

optimization is thus to transform this netlist into an optimal gate-level implementation with respect to given synthesis goals.

The performance (delay) and area parameters are usually considered as the key design objectives in the traditional design flow. Since the power consumption becomes more and more important, the power consumption objective was included to the goals in modern circuit design. In the emerging approximate circuit design flow, the error (accuracy) is a new design objective (Fig. 1.1) [58].

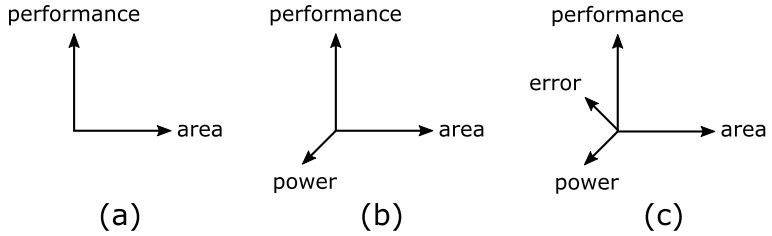

Figure 1.1: Design parameters considered in (a) traditional, (b) low-power and (c) approximate design flows [58]

Recently, a new approach to the synthesis of digital circuits has been proposed. It exploits evolutionary algorithms (EA) — a promising class of optimization algorithms inspired in evolutionary biology. The evolutionary logic synthesis is a method utilizing EAs to design and optimize logic circuits. Many papers showing the merits of the EA in this domain have already been published in the literature [57, 63]. The thesis deals with the use of EA for approximate logic synthesis.

1.2 Open problems

Although the EAs have already been successfully applied to the optimization of complex logic circuits, some open problems still exist. The main issue of the traditional (accurate) logic synthesis based on EA is the limited scalability. It means that EA typically produces a high-quality solution for low complex problem instances, but an insufficient (or even none) solution is provided for complex problem instances. There are two main types of the scalability problem: (i) the scalability of representation — the search-space grows rapidly with the representation size, and (ii) the scalability of evaluation — the increasing number of inputs and circuit components causes longer evaluation time. These problems contribute to a very limited acceptance of the method by the community of hardware designers. Another issue is that the optimization criteria, typically the number of active nodes, do not usually correspond with the hardware parameters such as power consumption or area, and hence, the resulting circuits are not acceptable in real-world applications. Open problems of the EA-based logic synthesis are how to: (i) effectively represent the circuits for the EA, (ii) quickly evaluate the functionality of the circuit, and (iii) perform fast estimation of hardware parameters (delay, power consumption etc.).

Regarding approximate circuits, some of the open problems are how to determine the error of an approximate circuit and which error metric is suitable for a given application. It is also unclear how to perform desired approximations.

1.3 Research objectives

The main objective of the research presented in this thesis is to show that

it is possible to reduce power dissipation of digital combinational circuits described on various levels of abstraction using EA-based circuit approximation algorithms.

First of all, it is important to select representative applications to demonstrate the proposed approximation methods. The next stage is to develop the EA-based circuit approximation method for these applications. As indicated in Section 1.2, such an approximation method has to meet several requirements. It has to be able to design circuits of a sufficient complexity that can be employed in the real-world application. Therefore, the evaluation of candidate circuits has to be effective. To optimize the power dissipation, some power estimation method has to be included in the EA.

The main objective of the thesis can be divided to the following partial goals:

1. To identify real-world applications suitable for approximation.
2. To create fast and efficient power estimation methods for the selected levels of circuit description.
3. To implement an evolutionary algorithm for efficient approximation of digital circuits.
4. To tune and accelerate the evolutionary algorithm in order to speed up the circuit approximation process.
5. To experimentally evaluate the proposed method using the selected real-world application.
6. To compare obtained results with the state-of-the-art solutions.

1.4 Thesis outline

This thesis is composed as a collection of selected author's papers with an accompanying introductory part. The seven peer-reviewed papers encapsulate the contribution of this dissertation. All papers are attached in their original publication format.

The thesis is organized as follows. The first chapter provides the introduction to the research area and research objectives. Chapter 2 surveys the state-of-the-art already published in the literature. Chapter 3 summarizes the research contribution and introduces the selected papers. Chapter 4 summarizes the obtained results and proposes directions for the future research.

Chapter 2

Survey of the state of the art

This chapter introduces the background utilized in the presented work. It primarily addresses the power consumption of digital circuits and its estimation, approximate computing techniques and evolutionary methods for the digital circuit design.

2.1 Power consumption of digital circuits

Today it is widely accepted that power efficiency is a key goal of digital circuit design. Minimizing power consumption calls for conscious effort at each abstraction level and at each phase of the design process.

There are three sources of power dissipation in the standard complementary metal-oxide-semiconductor (CMOS) digital combinational circuits. The first one is the existence of logic transitions. As the "wires" switch between two logic (and so electrical) values, the parasitic capacitances are charged and discharged. This component of power dissipation, the so-called switching power P_{switch} is proportional to supply voltage V, wire voltage swing V_{swing}, switching activity coefficient α and switched capacitance C_{LOAD}. As the voltage swing is usually equal to the supply voltage, the dissipation varies with the square of the supply voltage. Short-circuit currents that directly flow from supply to ground when both complementary subnetworks conduct simultaneously are the second source of power dissipation. When inputs of the gate are stable, only one subnetwork is conducting, and no short circuit current can flow. When the output of the gate is changing its value in response to the changes in inputs, both subnetworks conduct simultaneously for a short interval. This power dissipation P_{short} varies with the average short-circuit current I_{SC} and supply voltage V. Both the above-mentioned sources of power dissipation in CMOS circuits are related to the transitions at gate outputs. They are collectively referred to as *dynamic power*. In contrast, the last source of power dissipation is leakage current I_{leak} originating mainly from subthreshold MOS conduction. Nowadays, this leakage power dissipation becomes more important because transistors are smaller and smaller [48].

$$P_{CMOS} = P_{switch} + P_{short} + P_{leakage} \tag{2.1}$$

$$P_{CMOS} = \frac{1}{2} \cdot C_{LOAD} \cdot \alpha \cdot f \cdot V^2 + I_{SC} \cdot V + I_{leak} \cdot V \tag{2.2}$$

This section gives a brief introduction to the main factors having a crucial impact on the power dissipation of digital circuits. In the first section, the impact of selected fabrication technology is discussed. Digital circuits can be described at various levels of abstraction.

In the next sections, two different levels are presented — the transistor level netlist and the gate level netlist. For the both levels the power estimation methods and power saving techniques are presented.

2.1.1 The technology impact

The overall power consumption strictly depends on the selected fabrication technology. The technology is usually specified using the so-called layout design rules. Since migration from one process to a more complex one is difficult, these rules were simplified using scalable rules. Mead and Conway [28] popularized the scalable design rules based on a single parameter λ that characterizes the process resolution. This parameter is generally half of the minimum drawn transistor channel length.

The decreasing transistor sizes lead to a reduction in capacitance of the gates. In addition to that, the supply voltage is decreasing too. For example, the 500 nm technology needs 5 V supply, 180 nm needs 1.8 V and 45 nm technology uses 1.0 V supply only. As the consequence of that, threshold voltages V_{tn} (for n-mos transistors) and V_{tp} (for p-mos transistors) grow up relatively to the supply voltage. The threshold voltages influence the leakage. Hence, the leakage power dissipation becomes more significant for recent technology as we can see in Figure 2.1.

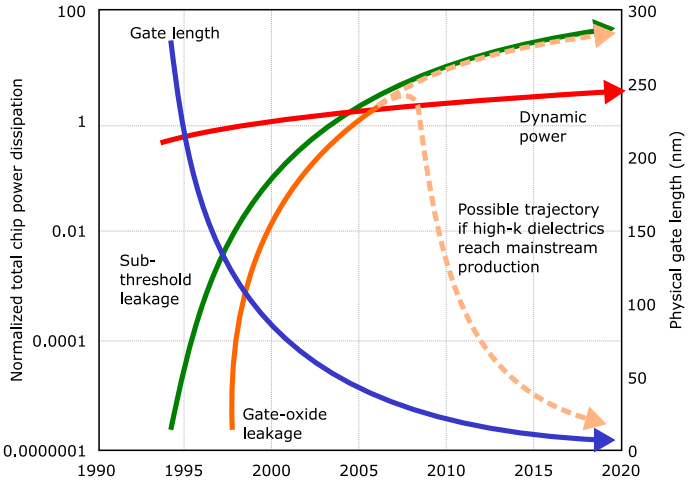

Figure 2.1: Dynamic and static power dissipation depending on technology design rules [18].

The target fabrication technology is usually chosen with respect to price and availability. For the automated design, it is important to consider that there is a trade-off between power consumption and performance (speed). In order to address this, technology libraries typically provide two or more versions of each cell, e.g. one fast, but dissipating, and one slow, but low-power. The designer can then select which implementation fits his/her needs.

2.1.2 Transistor-level digital circuits

As the circuit performance and power dissipation heavily depend on the chosen design style, the low-level description of digital circuits enables to choose different design styles. For example, fully complementary logic (CMOS), NMOS, differential cascade voltage switch or pass-transistor logic design style can be used [48]. The design styles differ in the power dissipation, performance and scalability of circuits that can be obtained. The transistor-

level circuits are usually described using a netlist consisting of transistors and additional devices such as capacitors or resistors.

One of the most accurate and straightforward method for power estimation of the circuits described at the level of transistors is to perform a circuit simulation by means of some SPICE-like[1] simulator. This family of simulators utilizes various numeric models of transistors. The most frequently used model in circuit simulation is the Berkeley Short-Channel IGFET_1 Model (BSIM) [11].

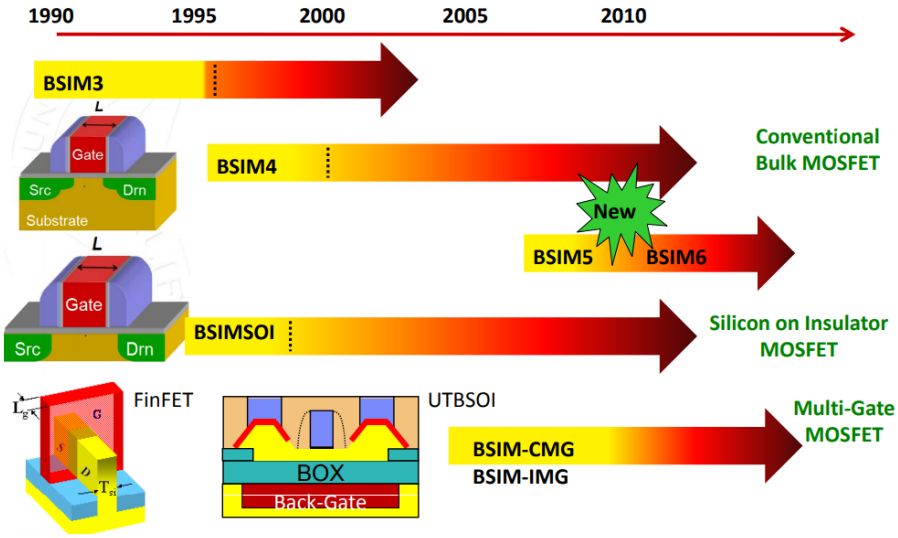

Figure 2.2: Evolution of BSIM family of Compact MOSFET Models [3].

There are several versions of BSIM models reflecting the evolution of fabrication process (see Fig 2.2). As the BSIM models are very complicated, it is impractical to derive closed-form equations for propagation delay, switching threshold and other parameters [71]. However, it is not difficult to find these circuit properties through circuit simulation.

It was shown that simulation results of the circuits utilizing these models are usually strongly pattern-dependent [20]. Hence, all the possible input transitions should be simulated in order to obtain trustworthy results. This can become computationally very expensive, especially for complex circuits, because the number of transitions grows exponentially.

2.1.3 Gate-level digital circuits

A gate level circuit is defined as a set of cells (gates) and their connections. To simulate circuits with millions of gates, the simulation model must be simple enough to handle this complexity, yet accurate enough to give trustworthy results.

The designers cannot change the design style in the same way as it was possible at the low-level (transistor) description, because the elementary gates are defined in the *technological library*. A typical technological library is developed and optimized for a certain fabrication process and contains various implementations and models of elementary gates and other basic components such as one-bit adders, and-or-invert structures, multiplexers etc. Corresponding to equation 2.2, four basic possibilities how to reduce power dissipation can be identified. The following list gives basic power optimization techniques applicable at the gate level:

[1] *Simulation Program with Integrated Circuit Emphasis*

- reduction of the switching activity α,
- reduction of load capacitance of gates C_{LOAD},
- frequency modification f,
- optimizing power domains V.

The load capacitance comes from the wires and transistors involved in a circuit. Good floorplaning and placement can minimize the wire capacitance. The load capacitance of gates can be reduced by choosing fewer stages of logic and smaller transistors. A common technique is to intentionally increase delay (up to a delay constraint) by choosing a slower, but more energy efficient, version of the gate from the technological library. Available implementations of the same gate (e.g. AND) differ in the channel sizes and threshold voltages.

One of the most powerful power optimization techniques is called *clock gating*. This approach masks (using AND gate) a clock signal with an enable signal to turn off the clock input of idle blocks. It is effective because the clock signal has a high activity factor. There is no clock signal in the combinational digital circuits, but we can select an arbitrary input with high switching activity (Fig. 2.3a).

Dynamic power is proportional to the frequency. Hence, a chip should not obviously run faster than it is necessary. Reducing the frequency also allows downsizing transistors as mentioned above. The performance can be recouped through parallelism, especially if the area is not as important as the power. The circuit may employ multiple frequency domains so that certain portions of the chip can run more slowly than others. However, the wires connecting the frequency domains must be synchronized.

Supply voltage has a quadratic effect on the dynamic power. Therefore, choosing a lower power supply significantly reduces power consumption. But V_{cc} value varies the performance of the circuit and the minimal value is typically specified by fabrication technology. However, the chip may be divided into multiple voltage domains affecting the performance of the circuits. In addition to that, the voltage domains can be turned off entirely to eliminate leakage power the during sleep mode. This technique is called *power gating*. Fig. 2.3b shows that the block is active, when the header switch transistors are connecting the supply voltage to the block. The outputs of the block may take on voltage levels in the forbidden zone. Hence, the output isolation gates must be used. This technique is required for leakage reduction of inactive parts of circuits [49].

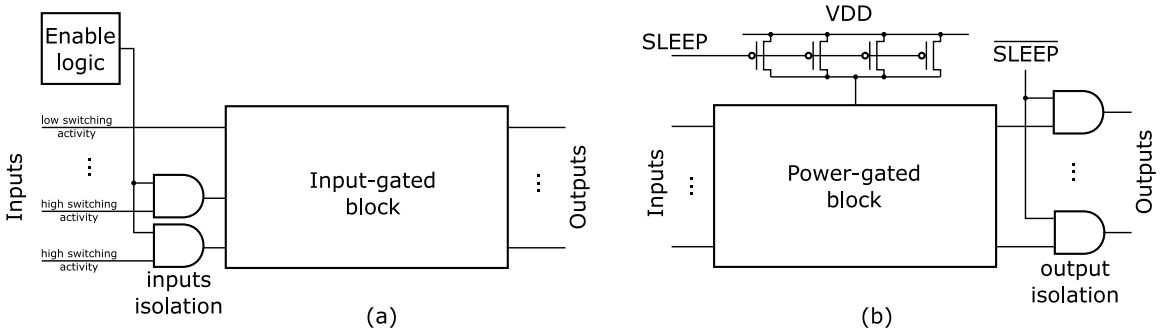

Figure 2.3: Two common low-power design techniques: (a) clock / signal gating, and (b) power gating [71].

It is infeasible to use an accurate SPICE simulator to determine power consumption of complex gate-level circuits. To address this problem, the power consumption of gate-level circuits is typically estimated using some suitable models. There are several models for the analysis of gate-level circuits that are much faster than numeric simulations. For each gate in the cell library the models (of physical behavior) are stored in a lib file satisfying Liberty standard[2]. These models used to be based on k-factor lookup tables, which gave the gate delays and the output signal transition times based on the gate input signal transition times and the gate capacitive load [17]. However, these lookup-table delay models did not contain enough information to characterize the parameters of multi-voltage circuits implemented in modern fabrication technologies. These limitations have motivated the development of the current source models. The current source models define the output DC current as a nonlinear function of the input and output voltages of the cell. Two different representations — *Composite Current Source Model* and *Effective Current Source Model* — were developed. These two representations are equivalent and can be transformed into a true current source model [6].

When the circuit parameters are determined using a suitable model, the switching activity factor α is estimated. This estimation is a non-trivial task, complicated by the fact that the switching activity depends not only on the circuit structure and functionality, but also on the applied input patterns. Two approaches have been proposed to address this problem – dynamic methods employing a variant of circuit simulator and static methods based on probabilistic techniques analyzed how the switching activity is propagated from inputs to outputs [44]. From the viewpoint of scalability, the probabilistic methods seem to be the preferred ones nowadays. Several variants of binary decision diagrams accompanied with partitioning have been proposed to determine the switching activity [26]. Some assumptions are usually introduced to reduce the computational complexity. When we neglect glitching, for example, we can use a zero-delay model to compute the switching activity [7].

2.2 Approximate arithmetic circuits

Despite of the development of design techniques for energy efficient circuits, the overall energy consumption of computer systems is still rapidly growing. As many important applications are inherently error resilient, precision of the involved computations can be traded for improved energy efficiency, performance, and/or chip area. Various approaches exploiting this fact have been developed in recent years and presented under the umbrella of the so-called approximate computing [5, 32].

In many scenarios, the use of approximate computing is unacceptable, while in others, the approximate computing paradigm can be proactively used to reduce power consumption. The motivations for introducing approximate computing can be: (i) satisfying the requirement for low power dissipations, (ii) some applications are error resilient and a strictly accurate output is not needed, or (iii) eliminating expensive fault-tolerant mechanisms that are needed to ensure reliability of circuits implemented with recent fabrication technology. According to [13, 53], applications suitable for approximate computing can be broadly classified to four classes:

1. Applications with analog inputs which operate on noisy real-world data.

2. Applications with analog output intended for human perception.

[2] http://opensourceliberty.org

3. Applications with no unique answer such as web search and machine learning.

4. Iterative and convergent applications that iteratively process large amounts of data and the equality of results depends on the number of iterations.

These approximations can be conducted at different system levels with circuit approximation being one of the most popular. Circuit approximation techniques can be classified into two main groups. (1) *Frequency/voltage over-scaling*, where timing-induced errors can appear as the circuit is operated on lower voltage than the nominal value [56]. (2) *Functional approximation*, where the original circuit is replaced by a less complex one which exhibits some output errors but improves non-functional circuit parameters such as power consumption or area on the chip.

In this work, we focus on functional approximation only. The principle of this approach is to implement a different function (circuit) with respect to the original one provided that the non-functional parameters such as power dissipation or area are improved, and the error is acceptable. Typically, the goal is to find a design showing a minimum area (or power consumption) and satisfying a given error constraint [72].

2.2.1 Approximation methods for arithmetic circuits

We focus on approximate arithmetic circuits because they are frequently used in the key applications relevant for approximate computing. The methods for functional approximations can be divided into two categories: (1) manual, and (2) automated.

The *manual (ad-hoc) methods* are developed for a specific circuit component. In this work, examples of manual approximation of two key arithmetic circuits — adders and multipliers — are described. These circuits are widely approximated because they realize key operations in applications requiring low power processing. However, other circuits such as dividers and multiply-and-accumulate (MAC) circuits were manually approximated too, but they are not widely employed. Hence, these circuits are not discussed in this chapter. On the other hand, the *automated methods* use some general-purpose circuit resynthesis and approximation techniques and enable us to approximate arbitrary circuits. These methods starts with an original (exact) circuit and, typically iteratively, modifies its structure.

2.2.2 Manual approximation methods

Adders

An adder performs the addition of two binary numbers. Two basic implementations are: (1) ripple-carry adder (RCA), where the carry of each full-adder is propagated to the next full-adder, and (2) carry lookahead adder (CLA), where several units working in parallel generate three signals ('sum', 'propagate' and 'generate') that are employed to quickly generate the carry-in signals. The CLA has significantly shorter delay than RCA. However, the area and power dissipation of CLA is larger than RCA. Many approximation principles for the adders implemented using one of these two schemes have been proposed in the literature [16]. The approximations can be classified into the following classes.

- *Speculative adders* were proposed by Lu [25]. In this architecture, CLA structure is approximated using prediction of the carry for each sum bit.

- *Segmented adders*, where the addition is divided to n smaller subadders operating in parallel. These subadders have a fixed carry and their delay is n-times shorter [33].

An advanced version divides the addition to the carry generation and sum generation, where each summarization utilizes the information from the previous carry generation [73].

- *Approximate Carry Select Adders* consist of several subadders. Each subadder is made of two speculative adders — one with carry-in "0" and another with carry-in "1". The carry-out of the first adder is connected to a multiplexor in the next block selecting the output of one from two speculative adders [9].

- *Approximate Full Adders* are implemented in LSBs of the adder. For example, the simple use of OR gates instead of full-adders and ignoring of carries in the LSB part can provide us enormous power and time savings [27].

Multipliers

Related to the addition, the multiplication is a more complex operation. Generally, it consists of stages of the partial product generation, accumulation and final addition. There are several accurate multiplier architectures. The manually approximated n-bit multipliers are usually derived from the in the one of the following schemes: (1) an array multiplier, where the sum and carry signals are generated by n-bit adders in each of n rows and they are passed to the adders in the next row, and (2) Wallace (or Dadda) tree multipliers dividing the multiplication into layers, where the adders work in parallel without any carry propagation within the layer. The array multiplier is smaller than the tree multiplier, but slower. The approximations can be implemented in the following parts of the multipliers [16]:

- *Approximation in generating partial products* modifies the submultipliers, which the multiplier is composed of. For example, Kulkarni et al. proposed an approximate 2×2-bit multiplier where only one single entry is altered ($3 \times 3 = 7$ and the remaining ones are correct) [19]. Larger multipliers are designed using this 2-bit multiplier as a building block.

- *Approximation in the partial product tree* modifies the structure of the multiplies. This approach is utilized in the broken-array multiplier [27]. This multiplier omits some rows and columns in the array multiplier. A straightforward truncation of LSBs in operands (e.g. the usage of accurate 6-bit multiplier instead of the 8-bit one) also modifies the partial product tree by omitting some partial product cells. The omitting approach can be done in an adaptive way. In the multiplier proposed by Kyaw et al. [21], the LSB cells function is controlled by the MSBs of operands.

- *Approximation in counters or compressors* in the partial product tree utilizes the tree structure of the multiplier. The key operations in each level are compressions, where 3 bits or 4 bits are compressed to 2 bits (3:2 or 4:2 compressors). These circuits can be approximated, for example, by a substitution of full-adders by approximate ones [34].

2.2.3 Automated approximation methods

SALSA

The Systematic methodology for Automatic Logic Synthesis of Approximate circuits (SALSA) is an automated approach that turns the approximation synthesis to the standard synthesis

task. A virtual circuit containing an accurate solution, a candidate circuit and decision circuit (with one output) is constructed. The output is active when the error bound constraint is violated. The *don't care states* are iteratively applied to the approximate solution. These states are accepted if the output of the virtual circuit remains zero for all input combinations. Thereafter, a traditional don't care based optimization technique is applied [69].

SASIMI

Another systematic approach, Substitute-And-SIMplIfy (SASIMI), tries to identify signal pairs in the circuit that show the same value with high probability, and substitutes one for the other. These substitutions are resulting in some logic to be eliminated from the circuit. In addition to that, the downsizing of gates on critical paths (simplification) may be enabled. Moreover, the connection of the signal pairs using a configurable substitution circuit provides a kind of quality configurable circuit that can dynamically operate at different accuracy levels depending on the application requirements [68].

ABACUS

In contrast with previous automated methods, Automated Behavioral Approximate CircUit Synthesis operates on HDL level. It automatically generates approximate circuits directly from the behavioral-level of description. In order to perform desired approximations, the method modifies the Abstract Synthesis Tree (AST) using the following operators: (1) simplification of datatypes, (2) substitution of arithmetic operations by approximate operations, (3) transformation of arithmetic expressions, (4) substitution of variables with constants, and (5) loop transformations. In each iteration of the algorithm, the operations are randomly applied to the accurate circuits while the error bound is checked after the application [46]. The search algorithm is based on a simple hill-climbing algorithm or multi-objective NSGA-II algorithm [45].

AIG-Rewriting

Another automatic synthesis approach uses And-Inverter Graphs (AIGs) based rewriting. The AIG is a widely employed representation in the logic synthesis. The algorithm identifies the longest paths in the circuit. Then so-called *cuts* are selected by performing the cut enumeration on the selected paths. In a logic circuit represented by an acyclic graph, a cut of node n is a set of nodes of the network, called leaves, such that each path from primary inputs to n passes through at least one leaf [31]. Each cut is replaced by an approximate cut (typically by zero constant) to generate a new candidate circuit. If the candidate meets the error constraints, it is accepted to the next iteration [2].

Evolutionary algorithm based methods

In the seminal paper on evolutionary design of approximate circuits [54], the problem of automated design of small approximate circuits consisting of elementary gates is addressed. This approach was extended with heuristic seeding to improve the scalability of the evolutionary method [64].

Except the group at Brno University of Technology, the evolutionary approach has been adopted only in few cases. For example, the evolutionary approximation was used in the context of FPGAs. GRATER tool [24] employs a genetic algorithm to determine the

precision of variables within an OpenCL kernel. By selectively reducing the precision, the number of parallel approximate kernels that can be mapped in the fixed area budget of an FPGA can be increased with respect to the original kernel implementations

A multi-objective evolutionary algorithm was employed to solve the so-called binding problem of High-Level Synthesis (HLS), in which suitable approximate components are assigned to nodes of the data flow graph describing a complex digital circuit [66].

ASLAN

Automatic methodology for Sequential Logic ApproximatioN (ASLAN) targets the task of the synthesis of approximate sequential circuits. The algorithm tries to identify combinational blocks in a sequential circuit that are amenable to approximations. Then, existing combinational approximation techniques are utilized to obtain a series of approximate versions having different quality levels. A gradient/descent approach is used to iteratively approximate the entire sequential circuit while the overall error bound is checked using a formal verification approach [47].

Summary of automated approximations

Typical arithmetic circuits that have been approximated with automated methods are 8- or 16-bit adders and 8-bit multipliers. As implementations of these methods are not available, it is hard to perform their fair comparison. An obvious question is whether these methods are suitable to approximate complex circuits such as 32-bit multipliers.

2.2.4 Quality of approximate circuits

The quality of approximate combinational circuits is typically expressed using one or several error metrics. In addition to the error rate, the average-case as well as the worst-case situation can be analyzed. Among others, the mean absolute error (MAE) and the mean relative error (MRE) are the most useful metrics that are based on the average-case analysis. Selection of the right metrics is a key step of the whole design. When an arithmetic circuit is approximated, for example, it is necessary to base the error quantification on an arithmetic error metric. For general logic circuits, where no additional knowledge is available and where there is not a well-accepted error model, Hamming distance or error rate is typically employed.

The following paragraphs summarize the error metrics that have been employed in literature to quantify the deviation between the outputs produced by a functionally correct design and an approximate design. These metrics are divided to two categories. The category of arithmetic errors consists of metrics that compare integer values of the circuit outputs. The Boolean error metrics are classified as general errors.

Arithmetic error metrics

Let $f : \mathbb{B}^n \rightarrow \mathbb{B}^m$ be an n-input m-output Boolean function that describes the correct functionality (the accurate function) and $f' : \mathbb{B}^n \rightarrow \mathbb{B}^m$ be an approximation of it, both implemented by two circuits, namely F and F'.

The *worst-case arithmetic error*, sometimes denoted as *error magnitude* or *error significance* [1], is defined as

$$e_{wce}(f, f') = \max_{\forall x \in \mathbb{B}^n} |int(f(x)) - int(f'(x))|, \tag{2.3}$$

where $int(x)$ represents a function $int : \mathbb{B}^m \to \mathbb{Z}$ returning an integer value of the m-bit binary vector x. Typically, a natural unsigned binary representation is considered, i.e. $int(x) = \sum_{i=1}^{m} 2^i \cdot x_i$. The worst-case error represents the fundamental metric that is useful to guarantee that the approximate output differs from the correct output by at most error bound e.

In the literature, the *relative worst-case error*

$$e_{wcre}(f, f') = \max_{\forall x \in \mathbb{B}^n} \frac{|int(f(x)) - int(f'(x))|}{int(f'(x))} \tag{2.4}$$

is frequently employed to constrain the approximate circuit to differ from the correct one by at most a certain margin. Note that a special care must be devoted to the cases for which the output value of the original circuit is equal to zero, i.e. the cases when the denominator approaches zero. This issue can be addressed by either omitting test cases when $int(f(x)) = 0$, or biasing the denominator by 1. The first approach is usually employed in the manual approximation methods where the zero results are accurate [16].

The *average-case arithmetic error* (also known as *mean absolute error*) is defined as the sum of absolute differences in magnitude between the original and approximate circuit, averaged over all inputs:

$$e_{mae}(f, f') = 2^{-n} \sum_{\forall x \in \mathbb{B}^n} |int(f(x)) - int(f'(x))|. \tag{2.5}$$

If we replace the expression in the sum by the equation for relative error distance, we can calculate the *mean relative error*:

$$e_{mre}(f, f') = 2^{-n} \sum_{\forall x \in \mathbb{B}^n} \frac{|int(f(x)) - int(f'(x))|}{int(f'(x))}. \tag{2.6}$$

Note that the values produced by absolute error metrics e_{mae} and e_{wce} can be very large. Hence, these values can be expressed as a part of the output range using multiplication by 2^{-m}. For example, the worst-case arithmetic error 64 for an 8-bit output circuit is equal to 25% error.

General error metrics

In addition to the arithmetic error metrics, there are metrics that are not related to the magnitude of the output of the correct or approximate circuit.

The *error rate*, referred to as the *error probability*, represents the basic measure that is defined as the ratio of inputs vectors for which the output value differs from the original one:

$$e_{prob}(f, f') = 2^{-n} \sum_{\forall x \in \mathbb{B}^n : f(x) \neq f'(x)} 1. \tag{2.7}$$

In many cases, it is also worth to consider the Hamming distance between $f(x)$ and $f'(x)$. The *worst-case Hamming distance*, denoted also as *bit-flip error* [4], is defined as

$$e_{bf}(f, f') = \max_{\forall x \in \mathbb{B}^n} \sum_{i=1}^{m} (f(x) \oplus f'(x))_i, \tag{2.8}$$

and gives the maximum number of output bits that simultaneously output a wrong value. The average number of changed output bits, denoted as the *average Hamming distance*, can be expressed as follows:

$$e_{mhd}(f, f') = 2^{-n} \sum_{\forall x \in \mathbb{B}^n} \sum_{i=0}^{m} (f(x) \oplus f'(x))_i. \tag{2.9}$$

2.2.5 Quality evaluation

In the error metric formulas, the enumeration of all possible input vectors is employed. For larger number of inputs n, it is not feasible to enumerate $\mathbb{B}^n$. We can deal with this issue by (a) enumerating a subset of $\mathbb{B}^n$, or (b) obtaining the exact value using a formal verification approach. Therefore, the evaluation methods can be divided to the following classes.

- *Exact (formal) evaluation* where the error is calculated using formal verification methods or obtained by a simulation tool performing the exhaustive simulation.
- *Inexact evaluation* using just a subset of $\mathbb{B}^n$ or a probabilistic error analysis.

This section introduces several approaches for the evaluation of the quality of approximation.

Simulation

This method can be used in the exact evaluation scenario. A circuit simulator that calculates responses for all input vectors is employed to provide the exact error metric. The simulation method is capable of determining all error metrics introduced in the previous section. In order to maximize the efficiency of the implementation, the simulator usually uses parallel acceleration techniques such as vectorization enabling bit-level parallel simulation. The idea is to utilize bitwise operators operating on multiple bits to perform more than one evaluation of a gate in a single step. This approach benefits from the fact that modern processors are equipped with specialized SIMD instructions. For example, widely available Advanced Vector Extension (AVX) instruction set allows us to operate with 256-bit operands. It means that every circuit with eight inputs can completely be simulated in one pass by applying a single 256-bit test vector at each input. Therefore, the obtained speedup is 256 against the sequential simulation [15]. As shown in [65], the performance of the simulator can be substantially improved if the interpreter is avoided and replaced by a native machine code that directly calculates the responses.

Simulation can be also accelerated using special hardware such as FPGAs. The concept of Virtual Reconfigurable Circuit (VRC) [52] or Dynamic Partial Reconfiguration (DPR) technique [50] can be used to quickly establish a circuit in the FPGA.

Since it is unfeasible to perform the exhaustive enumeration for a larger number of inputs due to the exponential complexity, the simulation frequently utilizes a subset of input vectors, typically selected by Monte-Carlo method [16]. To determine the required subset size, statistical models can be used [23].

BDD-based formal verification

To overcome time limitations of the simulation, various formal approaches can be employed for exact error evaluation [59]. In the classical logic synthesis, determining whether two

Boolean functions are functionally equivalent represents a fundamental problem solved using formal verification techniques. These widely explored techniques can be extended for the error metrics calculation, in the so-called *relaxed equivalence checking* scenario.

One of the state-of-the-art verification methodologies is based on Reduced Ordered Binary Decision Diagrams (ROBDD). ROBDD is a canonical rooted acyclic graph-based representation of Boolean function, where each node represents either an input variable or terminal node, and the edges represent the value assignment. ROBDDs have been traditionally used to solve the equivalence checking problem due to their canonical property. The ROBDD tools can exactly answer two questions: (1) whether a variable (signal) in the circuit is satisfiable, i.e. whether some input vector exists which generates true value of the signal; and (2) the probability that a selected variable is true. In the literature [55], algorithms determining the worst-case error and the average error of an approximate circuit were published.

Most formal verification approaches developed for testing exact equivalence are not directly extendable for relaxed equivalence checking. A common approach to the error analysis is to construct an auxiliary circuit referred to as an approximation miter. The structures of various approximation miters is shown in Figure 2.4. For arithmetic error metrics, a two's complement subtractor followed by a circuit which determines the absolute value is employed. Such a circuit can be used for formal evaluation of the mean and the worst-case arithmetic errors. For Hamming distance and the error rate, incorporating exclusive-or gates connecting the corresponding outputs is sufficient.

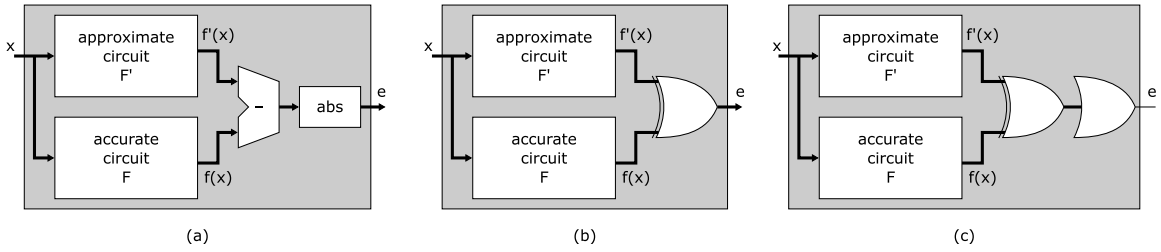

Figure 2.4: Approximation miter for the formal error analysis. Implementation of the error computation block for (a) arithmetic error, (b) Hamming distance and (c) error rate.

SAT-based formal verification

ROBDD based verification is inefficient in representing certain classes of functions. It is a well-known fact that the size of BDD is sensitive to the chosen variable ordering. In addition to that, there are functions, e.g. multipliers, for which the BDD size is always exponential, independent of variable ordering [59].

The advanced relaxed equivalence checking approach transforms the error computation to a satisfiability (SAT) problem which is solved by means of a SAT solver. Modern SAT solving algorithms are extremely effective at coping with large problem instances and large search spaces. The basic principle is to translate the problem of functional equivalence of two combinational circuits to the problem of deciding whether a Boolean formula given in conjunctive normal form (CNF) is satisfiable or not. This formula is constructed using miters shown in Figure 2.4. With respect to the BDD-based verification, SAT solving can effectively process large circuits, but the satisfiability (e.g. proving, that some signal can or cannot be set) is answered only. The worst-case upper-bound obtained by the miter and

a threshold circuit checking that difference e is below threshold τ (formally $e \leq \tau$) was usually determined [59].

Estimation

The errors of approximate circuits described on e.g. Abstract Syntax Tree or RTL levels can be determined using statistical analysis [22]. This approach is suitable when the circuit consists of approximate arithmetic modules (such as adders or multipliers) with known error distributions. For example, if the error of the module can be described using a Gaussian model, then mean μ and variance ν represent the error. At the circuit level, these error characteristics coming from the modules are combined to the resulting error.

2.3 Cartesian Genetic Programming

Cartesian genetic programming (CGP) grew from a method of evolving digital circuits developed by Miller et al. in 1998 [30]. CGP especially differs from other genetic programming branches in (i) the solution representation and (ii) the search mechanism. The key ingredients of CGP are briefly introduced in the following paragraphs. Then, the use of CGP in the synthesis of approximate circuits is discussed. Detailed description of CGP in the digital circuit design is available in [29].

2.3.1 Circuit representation

The key part of CGP is a representation of candidate circuits. A candidate circuit is represented as a special integer netlist describing the constant number of components (N). These components (nodes) are organized in a two-dimensional grid of n_c columns and n_r rows ($N = n_c \cdot n_r$). The number of primary inputs and outputs of the circuit is denoted as n_i and n_o. The function of the nodes depends on the level of abstraction used in modeling, where logic gates and more complex components from the technology library are naturally supported. Every component has up to n_a inputs and n_b outputs. For example, if standard logic gates are used as components, $n_a = 2$ and $n_b = 1$.

A unique address is assigned to all primary inputs and to the outputs of all components to define an addressing system enabling circuit topologies to be specified. The primary inputs are labeled $0, 1, \ldots, n_i - 1$ and the components' outputs are labeled $n_i, n_i + 1, \ldots, n_i + n_b \cdot n_c \cdot n_r - 1$. As no feedback connections are allowed in the basic version of CGP, only acyclic graphs can be encoded. Because of that, only combinational circuits can be created.

The main feature of this encoding is that while the size of the chromosome is constant (for given n_a, n_b, n_o, n_r and n_c), the size of circuits represented by this chromosome is variable (from 0 to $n_c \cdot n_r - 1$ components can be involved) as some components can remain disconnected (see Fig. 2.5). This redundancy has been identified as a crucial property of the efficient search in the space of digital circuits [29].

Figure 2.5 shows a gate level, four-input (a_0, a_1, b_0, b_1) and four-output (o_0, o_1, o_2, o_3), circuit consisting of eight gates implementing the two-bit multiplication. This circuit is represented in the CGP grid with $n_c = n_r = 3$ and the outputs of the components are labeled $4, 5, \ldots, 12$. One component (the exclusive-or gate with the output labeled 10) is inactive, i.e. no path from the output of the component to the primary output of the circuit exists.

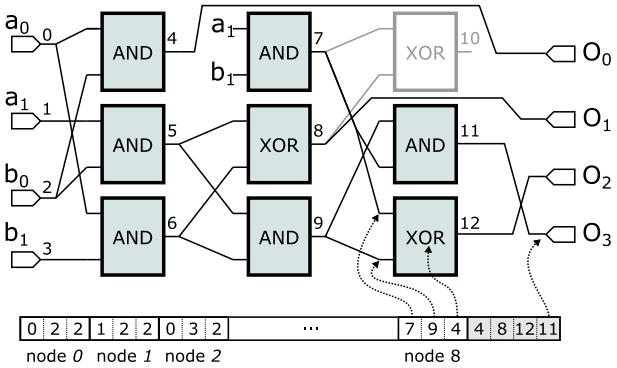

Figure 2.5: A two-bit multiplier represented in CGP with parameters: $n_i = n_o = 4, n_c = n_r = 3, n_a = 2, n_b = 1, \Gamma = \{0^{identity}, 1^{not}, 2^{and}, 3^{or}, 4^{xor}, 5^{nand}, 6^{nor}, 7^{xnor}, 8^{cont0}, 9^{const1}\}$.

In the integer netlist, each component is represented using $n_a + 1$ integers, where n_a integers specify destination addresses for its inputs and one integer refers to the set Γ containing all supported functions of the nodes. A component placed in the j-th column can obtain its input values v either from the primary inputs or from the components placed in previous columns, formally $v \leq n_i + (j-1) \cdot n_r \cdot n_b$.

The last part of the chromosome contains n_o integers specifying the connection of each primary output. The output can be connected either with the output of some node or with the primary input.

The whole circuit is then represented using the so-called chromosome (i.e. simplified integer netlist) whose size is

$$N_g = n_c \cdot n_r \cdot (n_a + 1) + n_o \quad \text{(integers)}. \tag{2.10}$$

2.3.2 Search algorithm

Every candidate circuit represents one design point in the design space. In CGP, new designs are created by introducing small random modifications to the chromosome. This operation is called the mutation and it typically modifies h integers of the chromosome. Note that all modifications must lead to valid circuits, i.e. only valid function codes and connections can be created.

Algorithm 1 presents the search method based on $(1 + \lambda)$ evolutionary strategy usually used for a single-objective circuit approximation by means of CGP [29]. The search algorithm can start either with a randomly generated initial population or with existing designs. The population size is $1 + \lambda$. After evaluating the initial population (i.e. scoring the circuit functionality and the cost while better circuits obtain higher scores), the following steps are repeated until the termination condition is not satisfied: (i) a new parent is selected; (ii) λ offspring circuits are created from the parent by means of mutation; (iii) the population is evaluated.

One mutation can affect either the component function, the component input connection, or the primary output connection. A mutation is called neutral if it does not affect the circuit's fitness. Neutral mutations are very important for the evolutionary design and optimization [29].

The fitness function is typically application-specific. If the initial population is initialized with circuits satisfying the functional requirements, the objective of the evolutionary

Algorithm 1: CGP optimization

Input: CGP parameters, fitness function, intial population P
Output: The highest scored individual and its fitness

```
1 EvaluatePopulation(P);
2 while ⟨terminating condition not satisfied⟩ do
3     α ← SelectHighest-scored-individual(P);
4     if fitness(α) ≥ fitness(p) then
5         p ← α;
6     P ← {p} ∪ {λ offspring of p created by mutation};
7     EvaluatePopulation(P);
8 return p, fitness(p);
```

synthesis formulated in the fitness function is to improve non-functional parameters and keep the functionality unchanged.

2.3.3 Circuit approximation using CGP

In the context of approximate computing, three evolutionary approximation strategies (shown in Fig. 2.6) were developed.

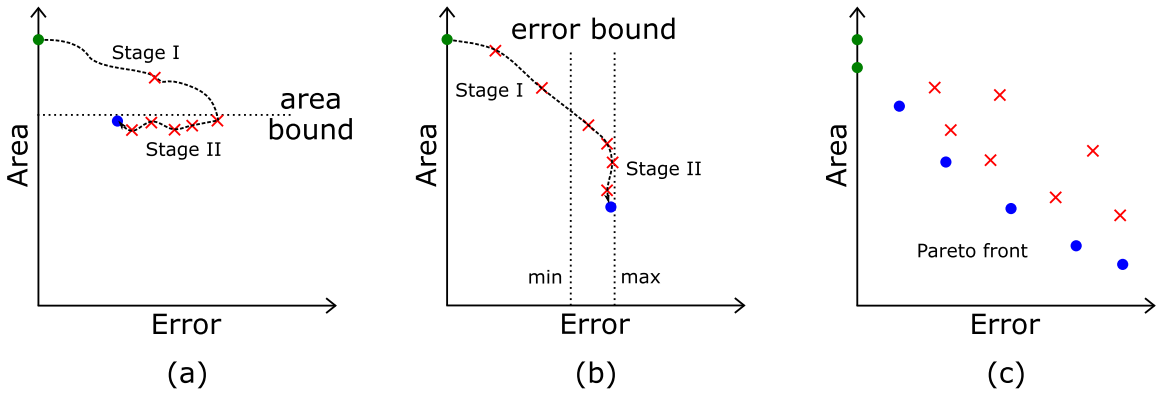

Figure 2.6: Evolutionary approximation strategies (a simplified situation for the area-error optimization): (a) resource-oriented, (b) error-oriented and (c) multi-objective. The white points denote the initial (accurate) circuit, the blue are the desired approximate circuits and the crosses denote the valid candidate solutions (design points).

Resources-oriented method

CGP is used to minimize the error criterion under the assumption that only m_i components (gates) are available and m_i is lower than the minimal number of components (gates) needed to implement the accurate function [64].

The strategy is divided to two stages. In the first stage, the accurate initial solution is reduced to contain less than m_i components. It can be done using a random component elimination algorithm or a suitable heuristic. In the second stage, the goal of CGP is to optimize the error while the number of components is kept lower than m_i. The main

advantage is that the user can control the used area (and power consumption) precisely by means of m_i.

Error-oriented method

In the error-oriented method, the target error (e.g. the worst-case error) range, determined by e_{min} and e_{max}, is specified by the user. The goal is to optimize the number of components (or area or power consumption) while the error of the circuits is kept between target errors e_{min} and e_{max} [62]. This strategy is divided to two stages as well. In the first stage, the design objective (area in Fig. 2.6b) is minimized since the error is lower than e_{max}. We can assume that the increasing error typically leads to smaller area. In the second stage, the algorithm optimizes the design objective while the error of circuits kept in the target error range. If various tradeoffs between the error and the number of components are requested, CGP is executed several times with e_{max} as the parameter.

Multi-objective CGP

Compared to previous methods that employ a single-objective optimization (one fitness function with constraints), the multi-objective method allows to optimize the error and other key circuit parameters (area, delay and power consumption) together in one CGP run [14]. We are primarily interested in approximate circuits belonging to the *Pareto set* which contains the so-called *nondominated solutions*. For example, consider two circuits C1 and C2. Circuit C1 dominates another circuit C2 if: (1) C1 is no worse than C2 in all objectives, and (2) C1 is strictly better than C2 in at least one objective.

The multi-objective CGP represents candidate circuits using the standard CGP encoding. The search algorithm uses a modified variant of a multi-objective genetic algorithm, such as Non-dominated Sorting Genetic Algorithm (NSGA-II) [8].

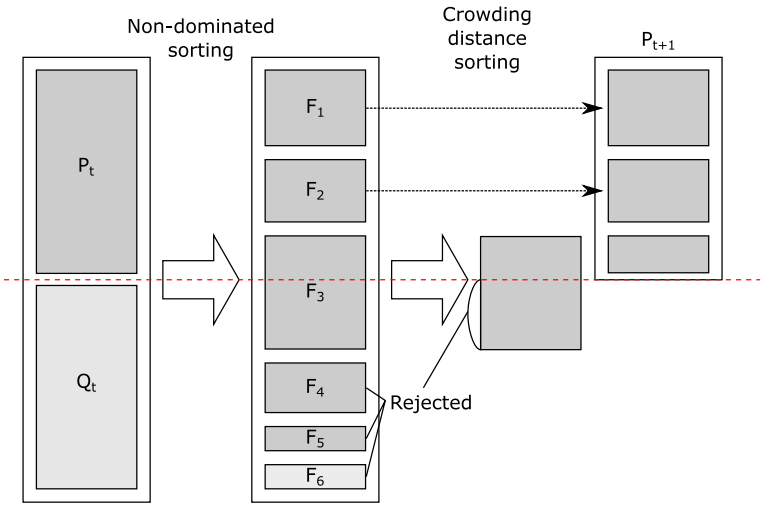

Figure 2.7: Creating of a new population in NSGA-II algorithm [8].

NSGA-II (Fig. 2.7) generates a set of offspring Q_t from the current population P_t. The individuals $P_t \cup Q_t$ are sorted according to the dominance relation into multiple fronts F_i. The first front F_1 contains all non-dominated solutions along the Pareto front. Each subsequent front $(F_2, F_3, \dots)$ is constructed by removing all the preceding fronts from the

population and finding a new Pareto front. The first fronts (F_1 and F_2 in Fig. 2.7) are copied to the next population P_{t+1}. If any front must be split (F_3 in Fig. 2.7), a crowding distance is used for the selection of individuals to P_{t+1}.

The advantage of multi-objective CGP is that it re-constructs the Pareto front in each CGP generation and tries to cover all possible compromise solutions. Several accurate implementations can be used as a seed. A set of non-dominated circuits is the output of the multi-objective algorithm.

Chapter 3

Research summary

This chapter summarizes the research conducted in the thesis. In the Section 3.1, overall view on the research is given. Section 3.2 describes motivations and contributions of seven selected papers where the most important results of the research were published. Section 3.3 lists the remaining papers of the author related to the research topic that are not included in this thesis. Sections 3.4 and 3.5 give additional information related to the work such as the list of research projects and awards.

3.1 Overview

The goal of the research conducted in this doctoral thesis is to improve the evolutionary circuit design methodology based on Cartesian genetic programming (CGP) in order to obtain an advanced method for the approximate circuit design. The method should produce approximate circuits that: (i) show excellent tradeoffs between quality and power-consumption, (ii) have reasonable complexity, (iii) can be defined at various levels of description, and (iv) are applicable in the real-world applications.

In order to demonstrate the capabilities of the proposed methods, the following representative real-world applications described on three basic levels of description were selected:

- The optimization and approximation of elementary digital circuits available in the technology library *(the transistor level)*
- Image classification conducted by a neural network *(the gate level)*
- Discrete cosine transformation (DCT) used in HEVC encoder *(the gate level)*
- Median filtering for image and signal filters *(the RTL modules level)*

In Paper I we presented an advanced simulation technique and a novel modification of CGP representation which is suitable for fast evaluation of transistor-level circuits. The proposed method was utilized in the evolutionary design and optimization of circuits at the transistor-level.

Due to the limited scalability of the transistor-level approach, the gate-level representation was used in CGP approximation of larger circuits. At the gate level, multi-objective CGP was employed for construction of a library of 8-bit approximate arithmetic circuits. This library contains hundreds of adders and multipliers, that are available online for download (Paper II). The approximate multipliers from the library helped us to investigate

the properties of approximation in neural image classifiers and introduce a new constraint for the forward path approximation in neural networks. This constraint was implemented to the CGP approximation method which was utilized to evolve approximate multipliers for neural networks. The neural networks employing the proposed approximate multipliers showed a good tradeoff between the power consumption and the overall classification accuracy (Paper III).

In the aforementioned papers, circuit simulators evaluating the exact approximation error were utilized. However, the exact circuit simulation is not feasible for complex circuits. To deal with this issue, formal verification techniques were adapted in the next work. We included several optimization approaches to the verification methodology in order to approximate complex arithmetic circuits (Paper IV). In particular, we utilized SAT solvers for analysis of the worst-case arithmetic error. An advanced BDD-based analysis was implemented in order to obtain the mean arithmetic error for complex adders. This algorithm was used in CGP to design approximate adders and subtractors that were later applied in discrete cosine transformation blocks of HEVC encoder (Paper V).

We investigated not only the possibilities on how to improve the scalability of the evolutionary approach, but also how to improve quality of the results. In the evolutionary approximation of multipliers, it was shown that the function set Γ covering all important components of the technology library (such as full adders and multiplexors) led to better results. As a side effect, the CGP with the function set including the more complex components led to better convergence compared to the CGP with the function set containing just one- and two-input logic gates (Paper VI).

The last presented work (Paper VII) shows that the evolutionary approximate synthesis conducted at RTL-level is capable of creating complex circuits. In particular, we evolved approximate median filters suitable for image and signal processing. In order to perform a data-independent and fast evaluation of candidate median filters, a new error metric was proposed. A significant improvement was obtained in terms of circuit complexity and quality with respect to CGP guided by the fitness function based on random simulation.

All results enabled us to deeper understand the relations among the error metrics, the level of circuit description and the search algorithm in the context of evolutionary approximations. We showed that evolutionary algorithms can be used for power-aware optimization of digital circuits described on various levels — from the lowest transistor netlist to the gate level description and the circuits utilizing complex building blocks.

3.2 Papers included in the thesis

This section presents details on the motivation and contributions for each paper together with the paper abstract.

3.2.1 Paper I

MRÁZEK Vojtěch and VAŠÍČEK Zdeněk. **Evolutionary Design of Transistor Level Digital Circuits using Discrete Simulation**. In: *Genetic Programming, 18th European Conference, EuroGP 2015*. Berlin: Springer, 2015, pp. 66-77. ISBN 978-3-319-16500-4.

Author participation: 50 %

Conference ranking: B (CORE[1]) / B1 (Qualis[2])

[1] `http://www.core.edu.au/conference-portal`

[2] `http://www.conferenceranks.com/`

Motivation and contributions

The transistor level description is useful for the design and optimization of small circuits suitable for e.g. technology libraries specialized for approximate computing. In this paper, we presented a fundamental modification of the CGP representation of circuits described on the transistor level. Instead of single-output and typically two-input nodes supporting single-directional data flow, into this representation, we included *nmos* and *pmos transistors* and *junctions*. The junctions are necessary for the description of transistor-level circuits, but their usage can lead to cyclic connections that are unwanted in combinational circuits.

The evaluation of candidate circuits was performed by means of the proposed discrete even-driven simulator. Six different discrete values are supported: "strong" as well as "degraded" zeros and ones, a high-impedance state and a forbidden state. The simulation engine stimulated candidate circuits with all possible input vectors and checked, whether the output values were identical with the specification, and no short-circuit connections or values oscillations (caused by cyclic connections) occurred during the simulation.

The proposed approach was employed in the automated design of accurate circuits that could be used as components of the technology library. In some cases, it allowed us to find the same solutions as those published in the literature. For example, an implementation of full-adder employing 14 transistors was rediscovered. In the contrast with genetic programming literature [70], where only small circuits (up to 10 transistors) were evolved, the proposed evolutionary design was able to handle circuits with up to 30 transistors due to the fast evaluation of candidate circuits.

Later, the work was extended with power consumption estimation of candidate circuits [38]. The numeric simulation models of transistors, typically used in power consumption calculation, are very complex and computationally expensive. Hence, we proposed an estimation algorithm that utilized precalculated power dissipations of pmos and nmos transistors for each input transition obtained by means of the SPICE simulator. In the candidate circuit evaluation process, all possible combinations of inputs were simulated using the proposed simulator. During the simulation, probabilities of transitions of all transistors were acquired. These probabilities and the precalculated table of power dissipations were used to estimate the total power consumption.

Since the discrete simulation ignores some facts (e.g. it does not care about a potential timing issues, especially for large circuits containing hundreds of transistors), the accurate SPICE simulator was used time to time to validate the functionality of candidate circuits [38]. If the validation did not pass, the evolutionary algorithm rolled back to the last valid solution.

In order to design approximate arithmetic circuits, we proposed to employ a two-stage design process [38]. In the first stage, the circuit was approximated on the gate level. Then, the resulting approximate circuit was optimized directly on the transistor level. Using this approach, we evolved optimized four-bit approximate multipliers that outperformed the gate-level approximations in terms of power consumption.

In this paper, a new approach suitable for the evolutionary design of digital circuits conducted directly on the transistor level was introduced. The novelty lies in a new circuit representation and a new simulation engine. The approach can be utilized to design or optimize low-power modules of technology libraries specialized for approximate computing.

Abstract

The objective of the paper is to introduce a new approach to the evolutionary design of digital circuits conducted directly at transistor level. In order to improve the time-consuming evaluation of candidate solutions, a discrete event-driven simulator was introduced. The proposed simulator operates on multiple logic levels to achieve reasonable trade-off between performance and precision. A suitable level of abstraction reflecting the behavior of real MOSFET transistors is utilized to minimize the production of incorrectly working circuits. The proposed approach is evaluated in the evolution of basic logic circuits having more than 20 transistors. The goal of the evolutionary algorithm is to design a circuit having the minimal number of transistors and exhibiting the minimal delay. In addition to that, various parameter settings are investigated to increase the success rate of the evolutionary design.

3.2.2 Paper II

MRÁZEK Vojtěch, HRBÁČEK Radek, VAŠÍČEK Zdeněk and SEKANINA Lukáš. **EvoApprox8b: Library of Approximate Adders and Multipliers for Circuit Design and Benchmarking of Approximation Methods**. In: *Proc. of the 2017 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. Lausanne: European Design and Automation Association, 2017, oo. 258-261. ISBN 978-3-9815370-9-3.

Author participation: 25 %
Conference ranking: B (CORE) / A1 (Qualis)

Motivation and contributions

Due to the limited scalability of transistor-level approach, CGP at the gate-level representation was employed for construction of larger circuits. In this paper, a large library of 8-bit approximate arithmetic circuits was introduced. The approximate circuits were evolved using a multi-objective implementation of CGP based on NSGA-II algorithm proposed in [14]. It considered power consumption, area and mean absolute error as the design objectives. An exhaustive simulation utilizing 256-bit AVX instructions was employed in the candidate circuit evaluation. The simulator calculated the mean absolute error of the candidate circuit and the probability of being in logic zero and one for each CGP node output. These probabilities combined with parameters of logic functions from the technology library were used to estimate the power consumption.

The evolved circuits outperformed the circuits published in the literature. Hence, we constructed hardware as well as software models of the circuits and made them online. This library contains hundreds of approximate adders and multipliers that are available. The library was well accepted by the community, for example, 47 unique users from 13 countries visited the library websites in April 2018.

This paper presented the first large library of approximate arithmetic circuits that are available online and thus everyone can download them and apply in various applications. In addition to that, researchers designing new approximate circuits can use these models for comparison and benchmarking.

Abstract

Approximate circuits and approximate circuit design methodologies attracted a significant attention of researchers as well as industry in recent years. In order to accelerate the

approximate circuit and system design process and to support a fair benchmarking of circuit approximation methods, we propose a library of approximate adders and multipliers called EvoApprox8b. This library contains 430 nondominated 8-bit approximate adders created from 13 conventional adders and 471 non-dominated 8-bit approximate multipliers created from 6 conventional multipliers. These implementations were evolved by a multi-objective Cartesian genetic programming. The EvoApprox8b library provides Verilog, Matlab and C models of all approximate circuits. In addition to standard circuit parameters, the error is given for seven different error metrics. The EvoApprox8b library is available at: `www.fit.vutbr.cz/research/groups/ehw/approxlib`.

3.2.3 Paper III

MRÁZEK Vojtěch, SARWAR Syed Shakib, SEKANINA Lukáš, VAŠÍČEK Zdeněk and ROY Kaushik. **Design of Power-Efficient Approximate Multipliers for Approximate Artificial Neural Networks**. In: *Proceedings of the 35th IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. Austin, TX: Association for Computing Machinery, 2016, pp. 811-817. ISBN 978-1-4503-4466-1.

Author participation: 55 %
Conference ranking: A (CORE) / A1 (Qualis)

Motivation and contributions

The image classification using neural networks (NN) is currently a very popular topic. Since the recognition task has been identified as error-resilient, the implementations of neural classifiers are after approximated [67, 51, 10].

This research started with applying approximate multipliers from the EvoApprox8b benchmark to the neural classifier in order to reduce its power consumption. We observed that the NN with approximate multipliers exhibits significantly worse classification accuracy than the original NN. We analyzed this problem and recognized that almost 80% of multiplications in NN should produce zero. We hypothesized that multiplication by zero must be accurate, but multiplication of non-zero operands can be approximated. We introduced this "accurate zero-multiplication" constraint into the error oriented evolutionary approximation and evolved 7-bit and 11-bit approximate unsigned multipliers. Finally, the evolved multipliers were extended using one-complement to 8- and 12-bit signed ones.

The evolved multipliers were applied in two NNs (a multilayer perceptron and convolutional LeNet-5 network) performing two different benchmark tasks — handwritten number recognition (MNIST dataset) and Street-View House Number classification (SVHN dataset). We recognized that (despite the NNs were trained) a re-training process is able to adapt the pre-trained NN to the inaccurate multipliers, and thus improve the classification accuracy of approximate NNs up to tens of percent. The resulting NNs employing approximate multipliers exhibited significant power consumption reduction and classification accuracy comparable with solutions presented in the literature.

In this application, a specific constraint related to approximate multiplication in NNs was discovered. This paper showed that the proposed methodology of evolutionary approximation can easily handle very specific constraints. Multipliers satisfying this constraint were evolved and applied in the neural networks. The resulting neural networks show a good tradeoff between the power consumption and the overall classification accuracy.

Abstract

Artificial neural networks (NN) have shown a significant promise in difficult tasks like image classification or speech recognition. Even well-optimized hardware implementations of digital NNs show significant power consumption. It is mainly due to non-uniform pipeline structures and inherent redundancy of numerous arithmetic operations that have to be performed to produce each single output vector. This paper provides a methodology for the design of well-optimized power-efficient NNs with a uniform structure suitable for hardware implementation. An error resilience analysis was performed in order to determine key constraints for the design of approximate multipliers that are employed in the resulting structure of NN. By means of a search-based approximation method, approximate multipliers showing desired tradeoffs between the accuracy and implementation cost were created. Resulting approximate NNs, containing the approximate multipliers, were evaluated using standard benchmarks (MNIST dataset) and a real-world classification problem of Street-View House Numbers. Significant improvement in power efficiency was obtained in both cases with respect to regular NNs. In some cases, 91% power reduction of multiplication led to classification accuracy degradation of less than 2.80%. Moreover, the paper showed the capability of the back-propagation learning algorithm to adapt with NNs containing the approximate multipliers.

3.2.4 Paper IV

ČEŠKA Milan, MATYÁŠ Jiří, MRÁZEK Vojtěch, SEKANINA Lukáš, VAŠÍČEK Zdeněk and VOJNAR Tomáš. **Approximating Complex Arithmetic Circuits with Formal Error Guarantees: 32-bit Multipliers Accomplished**. In: *Proceedings of 36th IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. Irvine, CA: Institute of Electrical and Electronics Engineers, 2017, pp. 416-423. ISBN 978-1-5386-3093-8.

Author participation: 17 %
Conference ranking: A (CORE) / A1 (Qualis)

Motivation and contributions

In Paper II and Paper III, the evolutionary approximation of arithmetic circuits such as multipliers and adders was presented. The error was determined exactly in both cases to give the potential users a guarantee regarding the worst-case error. Hence, we used a highly optimized parallel (vectorized) circuit simulator to evaluate the quality of candidate circuits. However, this method is not feasible for exact evaluating of complex circuits because the number of input combinations grow exponentially with increasing the number of inputs. In order to address this issue and approximate larger circuits, we proposed to employ a SAT solver in the fitness function.

We proposed two novel approaches to accelerate the relaxed equivalence checking. One of the key components of the method is the miter circuit that verifies, whether the difference d between the approximate output and the accurate output is below a threshold τ (formally $|d| \leq \tau$). We developed an advanced miter construction that replaced the standard absolute value calculation. The proposed formula $(d \geq 0 \wedge d \leq \tau) \vee (d < 0 \wedge d \geq -\tau)$ reduced the number of SAT clauses and thus the evaluation time. Another crucial improvement lies in the proposed verifiability-driven search strategy. The trick is to control the time which the

SAT solver has to decide whether a candidate multiplier satisfies a given error bound or not. This solution allowed us to skip candidate multipliers that are not easily verifiable.

These approaches enabled to significantly improve the scalability limits of the evolutionary circuit approximation. We constructed a large dataset of approximate multipliers (with up to 32-bit inputs) and approximate adders (with up to 128-bit inputs).

In this paper, formal relaxed equivalence checking was employed in the approximation of complex arithmetic circuits. Since the checking can be very complicated and time-consuming for specific circuits, we included several optimization techniques to the verification methodology. In particular, we utilized CGP with SAT solvers to approximate large arithmetic circuits. By means of this approach, we significantly improved the scalability of the evolutionary approximation process.

Abstract

We present a novel method allowing one to approximate complex arithmetic circuits with formal guarantees on the approximation error. The method integrates in a unique way formal techniques for approximate equivalence checking into a search-based circuit optimization algorithm. The key idea of our approach is to employ a novel search strategy that drives the search towards promptly verifiable approximate circuits. The method was implemented within the ABC tool and extensively evaluated on functional approximation of multipliers (with up to 32-bit operands) and adders (with up to 128-bit operands). Within a few hours, we constructed a high-quality Pareto set of 32-bit multipliers providing trade-offs between the circuit error and size. This is for the first time when such complex approximate circuits with formal error guarantees have been derived, which demonstrates an outstanding performance and scalability of our approach compared with existing methods that have either been applied to the approximation of multipliers limited to 8-bit operands or statistical testing has been used only. Our approach thus significantly improves capabilities of the existing methods and paves a way towards an automated design process of provably-correct circuit approximations.

3.2.5 Paper V

VAŠÍČEK Zdeněk, MRÁZEK Vojtěch and SEKANINA Lukáš. **Towards Low Power Approximate DCT Architecture for HEVC Standard**. In: *Proc. of the 2017 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. Lausanne: European Design and Automation Association, 2017, pp. 1576-1581. ISBN 978-3-9815370-9-3.

Author participation: 40 %

Conference ranking: B (CORE) / A1 (Qualis)

Motivation and contributions

Discrete cosine transformation (DCT) is a very important part of widely employed HEVC (H.265) encoders as well as decoders. The goal of this work was to reduce the power consumption of DCT blocks in HEVC coders and encoders while keeping reasonable PSNR of the video processing.

The transformation can be implemented as a multiplication of inputs by a constant matrix. The most effective approach is to utilize multiplier-less constant multipliers (MCMs). Since adders and subtractors are the key components of MCMs, these arithmetic circuits were approximated by means of the proposed CGP approximation algorithm. Three approx-

imate MCM blocks were proposed. Due to the increasing bitwidth of adders/subtractors in MCMs, 16 different instances of adders and subtractors were approximated.

In this work, formal relaxed-equivalence checking was employed in CGP. In contrast with the SAT-based solution, we were interested not only in the worst-case analysis but also in the average-case error. Unfortunately, the SAT-based approach cannot be used to determine the average error. Hence an advanced BDD-based analysis algorithm was developed and implemented in order to obtain the mean and the worst-case arithmetic errors for candidate circuits. The proposed BDD-based analysis was used in the error evaluation in the error-oriented CGP approximation algorithm. Apart from the error analysis, the BDDs were used to determine the switching activity of each gate because they provided the information about the probability that a gate output is in logic one. The switching activity and the information from technology library (liberty file) were used in power consumption estimation of candidate circuits. The resulting HEVC implementations with approximate circuits in DCT blocks showed better quality/power tradeoffs than relevant implementations available in the literature.

In this paper, we proposed advanced algorithms for the mean and the worst-case error analysis with BDDs. We implemented a fast power-consumption estimation based on BDDs which was employed in CGP. By means of the proposed approach, new designs of complex approximate adders and subtractors applicable in an innovative application were developed.

Abstract

Video processing performed directly on IoT nodes is one of the most performance as well as energy demanding applications for current IoT technology. In order to support real-time high-definition video, energy-reduction optimizations have to be introduced at all levels of the video processing chain. This paper deals with an efficient implementation of Discrete Cosine Transform (DCT) blocks employed in video compression based on the High Efficiency Video Coding (HEVC) standard. The proposed multiplier-less 4-input DCT implementations contain approximate adders and subtractors that were obtained using genetic programming. In order to manage the complexity of evolutionary approximation and provide formal guarantees in terms of errors of key circuit components, the worst and average errors were determined exactly by means of Binary decision diagrams. Under conditions of our experiments, approximate 4- input DCTs show better quality/power trade-offs than relevant implementations available in the literature. For example, 25% power reduction for the same error was obtained in comparison with a recent highly optimized implementation.

3.2.6 Paper VI

MRÁZEK Vojtěch, VAŠÍČEK Zdeněk and HRBÁČEK Radek. The Role of Circuit Representation in Evolutionary Design of Energy-Efficient Approximate Circuits. *IET Computers & Digital Techniques*. Stevenage: The Institution of Engineering and Technology, (to appear), p. 11. ISSN 1751-8601.

Author participation: 40 %
Impact factor: 0.515 (Q3)

Motivation and contributions

The fitness function is not the only component that determines the quality of the obtained results. The representation of circuits, especially the set of possible node functions (Γ) has a great impact not only on the scalability but also on the quality of obtained approximate circuits and the convergence of the evolutionary design process (i.e. the number of generations needed to find an acceptable solution).

In this paper, we investigate the role of the function set Γ and thereby the number of inputs n_a and outputs n_b of CGP nodes. Two settings were compared: (i) CGP nodes that can act as standard 1- or 2-input gates (*gate-based representation*) and (ii) CGP nodes that have 3 inputs and 2 outputs and can act as standard gates as well as more complex blocks such as full-adders (*cell-based representation*). The circuits were approximated by the error-oriented CGP guided by estimated power consumption.

The evolved circuits were subsequently synthetized (technology mapped) by means of a commercial tool. Then, the quality of the synthetized circuits (power, area and delay) was evaluated. The circuits showed better quality for the same target error when we employed cell-based representation in the CGP. In addition to that, the convergence speed of the approximate EA-based logic synthesis was analyzed. Although the cell-based representation led to a larger search-space, in the case of approximate multipliers, better results were obtained with respect to the gate-based representation.

In the evolutionary approximation of multipliers, it was shown that the cell-based representation including the efficient implementations of various components such as full adders and multiplexors leads to better results than the representation employing the function set containing just one- and two-input logic gates only. This work illustrated the importance of choosing the right circuit representation in the approximate multiplier synthesis task.

Abstract

Circuit approximation has been introduced in recent years as a viable method for constructing energy efficient electronic systems. An open problem is how to effectively obtain approximate circuits showing good compromises between key circuit parameters – the error, power consumption, area and delay. The use of evolutionary algorithms in the task of circuit approximation has led to promising results. Unfortunately, only relatively small circuit instances have been tackled because of the scalability problems of the evolutionary design method. This paper demonstrates how to push the limits of the evolutionary design by choosing a more suitable representation on the one hand and a more efficient fitness function on the other hand. In particular, we show that employing full adders as building blocks leads to more efficient approximate circuits. We focused on the approximation of key arithmetic circuits such as adders and multipliers. While the evolutionary design of adders represents a rather easy benchmark problem, the design of multipliers is known to be one of the hardest problems. We evolved a comprehensive library of energy-efficient 12-bit multipliers with a guaranteed worst-case error. The library consists of 65 Pareto dominant solutions considering power, delay, area and error as design objectives.

3.2.7 Paper VII

VAŠÍČEK Zdeněk and MRÁZEK Vojtěch. **Trading between Quality and Non-functional Properties of Median Filter in Embedded Systems**. *Genetic Programming and Evolvable Machines*. Berlin: Springer, 2017, vol. 18, no. 1, pp. 45-82. ISSN 1389-2576.

Author participation: 50 %
Impact factor: 1.514 (Q2)

Motivation and contributions

The last paper is devoted to the approximation of circuits represented using a high-level representation. In particular, we approximated median filters implemented as median networks described at the level of so-called *compare-and-swap* components. The approximate median filters were already investigated in the literature [64], but the approximation error was calculated using a randomly generated set of input data. However, the calculated error depends on the input dataset. We proposed a new approach enabling to evaluate the quality of median networks. Firstly, we proposed an error metric that is data independent. Secondly, we introduced a formal method how to determine the error. The proposed metric, the so-called *distance error*, is calculated using the *permutation principle* introduced in this paper.

This new metric enabled to evaluate approximate median filters in $n!$ steps, where n is the number of inputs. The proposed distance error metric was employed in the resource-oriented CGP. The goal of the algorithm was to optimize the average distance error while keeping the number of active CGP nodes below a certain value. In contrast with the gate-level approximations, all operations were conducted with k-bit components. The function set Γ covered all functions that are important in the filtering: k-bit minimum, k-bit maximum and a junction connecting one of the inputs directly to the output of the node.

The evolved filters were applied in two tasks — image filtering of salt-and-pepper noise and processing of the data generated by an accelerator sensor. In addition to that, proposed approximated median structures were transferred to the C-code and implemented as approximate software for microcontrollers.

This paper demonstrated how to approaximate circuit described on a higher level of abstraction. Since the evolved filters were evaluated on microcontrollers, the paper also showed that the proposed approach is applicable for software approximations as well.

Abstract

Genetic improvement has been used to improve functional and non-functional properties of software. In this paper, we propose a new approach that applies a genetic programming (GP)-based genetic improvement to trade between functional and non-functional properties of existing software. The paper investigates possibilities and opportunities for improving non-functional parameters such as execution time, code size, or power consumption of median functions implemented using comparator networks. In general, it is impossible to improve non-functional parameters of the median function without accepting occasional errors in results because optimal implementations are available. In order to address this issue, we proposed a method providing suitable compromises between accuracy, execution time and power consumption. Traditionally, a randomly generated set of test vectors is employed so as to assess the quality of GP individuals. We demonstrated that such an approach may produce biased solutions if the test vectors are generated inappropriately. In order to measure the accuracy of determining a median value and avoid such a bias, we propose and formally analyze new quality metrics which are based on the positional error calculated using the permutation principle introduced in this paper. It is shown that the proposed method enables the discovery of solutions which show a significant improvement in execution time, power consumption, or size with respect to the accurate median

function while keeping errors at a moderate level. Non-functional properties of the discovered solutions are estimated using data sets and validated by physical measurements on physical microcontrollers. The benefits of the evolved implementations are demonstrated on two real-world problems—sensor data processing and image processing. It is concluded that data processing software modules offer a great opportunity for genetic improvement. The results revealed that it is not even necessary to determine the median value exactly in many cases which helps to reduce power consumption or increase performance. The discovered implementations of accurate, as well as approximate median functions, are available as C functions for download and can be employed in a custom application (`http://www.fit.vutbr.cz/research/groups/ehw/median`).

3.2.8 Author's contributions to selected papers

The papers presented in this thesis were created in collaboration with the Evolvable Hardware Group led by my supervisor prof. Sekanina. Preliminary results of Paper III were investigated during my visit to Purdue University, IN, USA. Although all co-authors contributed to the papers, this section explicitly summarizes author's contribution to selected papers.

- **Paper I** — development of discrete simulator, novel circuit representation, implementation of the evolutionary algorithm and experimental evaluation.
- **Paper II** — development of power estimation method, software models of circuits and the library front-end.
- **Paper III** — discovering the zero multiplication constraint, implementation of the evolutionary algorithm and experimental evaluation.
- **Paper IV** — development of the error-oriented method, miter simplification and experimental evaluation.
- **Paper V** — development of BDD-based error evaluation, power-estimation method and implementation of evolutionary algorithm.
- **Paper VI** — development of power estimation functions and experimental evaluation.
- **Paper VII** — proposing the error metric, implementation of the evolutionary algorithm and experimental evaluation.

3.3 List of other publications

2018

- ČEŠKA Milan, MATYÁŠ Jiří, MRÁZEK Vojtěch, SEKANINA Lukáš, VAŠÍČEK Zdeněk a VOJNAR Tomáš. ADAC: Automated Design of Approximate Circuits. In: *Proceedings of 30th International Conference on Computer Aided Verification (CAV'18).*

  Author participation: 17 %

  Conference ranking: A* (CORE) / A1 (Qualis)

- MRÁZEK Vojtěch and VAŠÍČEK Zdeněk. Evolutionary Design of Large Approximate Adders Optimized for Various Error Criteria. In: *GECCO Companion '18 Proceedings of the Companion Publication of the 2018 on Genetic and Evolutionary Computation Conference*. Kyoto: Association for Computing Machinery, 2018, p. 2. ISBN 978-1-4503-5764-7.

  Author participation: 50 %

- MRÁZEK Vojtěch, SÝS Marek, VAŠÍČEK Zdeněk, SEKANINA Lukáš and MATYÁŠ Václav. Evolving Boolean Functions for Fast and Efficient Randomness Testing. In: *GECCO '18 Proceedings of the 2018 on Genetic and Evolutionary Computation Conference*. Kyoto: Association for Computing Machinery, 2018, p. 8. ISBN 978-1-4503-5618-3.

  Author participation: 40 %

  Conference ranking: A (CORE) / A1 (Qualis)

2017

- MRÁZEK Vojtěch and VAŠÍČEK Zdeněk. Parallel Optimization of Transistor Level Circuits using Cartesian Genetic Programming. In: *GECCO Companion '17 Proceedings of the Companion Publication of the 2017 on Genetic and Evolutionary Computation Conference*. Berlin: Association for Computing Machinery, 2017, pp. 1849-1856. ISBN 978-1-4503-4939-0.

  Author participation: 90 %

- SEKANINA Lukáš, VAŠÍČEK Zdeněk and MRÁZEK Vojtěch. Approximate Circuits in Low-Power Image and Video Processing: The Approximate Median Filter.*Radioengineering*. 2017, vol. 26, no. 3, pp. 623-632. ISSN 1210-2512.

  Author participation: 30 %

  Impact factor: 0.945 (Q3)

- SHAFIQUE Muhammad, HAFIZ Rehan, JAVED Muhammad Usama, ABBAS Sarmad, SEKANINA Lukáš, VAŠÍČEK Zdeněk and MRÁZEK Vojtěch. Adaptive and Energy-Efficient Architectures for Machine Learning: Challenges, Opportunities, and Research Roadmap. In: *2017 IEEE Computer Society Annual Symposium on VLSI*. Los Alamitos: IEEE Computer Society Press, 2017, pp. 627-632. ISBN 978-1-5090-6762-6.

  Author participation: 14 %

  Conference ranking: B1 (Qualis)

2016

- HRBÁČEK Radek, MRÁZEK Vojtěch and VAŠÍČEK Zdeněk. Automatic Design of Approximate Circuits by Means of Multi-Objective Evolutionary Algorithms. In: *Proceedings of the 11th International Conference on Design & Technology of Integrated Systems in Nanoscale Era*. Istanbul: Istanbul Sehir University, 2016, pp. 239-244. ISBN 978-1-5090-0335-8.

  Author participation: 25 %

- MRÁZEK Vojtěch and VAŠÍČEK Zdeněk. Automatic Design of Arbitrary-Size Approximate Sorting Networks with Error Guarantee. In: *Power and Timing Modeling, Optimization and Simulation (PATMOS), 2016 26rd International Workshop on*. Bremen: Institute of Electrical and Electronics Engineers, 2016, pp. 221-228. ISBN 978-1-5090-0733-2.

  Author participation: 50 %

  Conference ranking: B2 (Qualis)

- MRÁZEK Vojtěch. Evoluční snižování příkonu: Od obvodů na úrovni tranzistorů po neuronové sítě na čipu. In: *Počítačové architektury a diagnostika PAD 2016*. Bořetice: Faculty of Information Technology BUT, 2016, pp. 61-64. ISBN 978-80-214-5376-0.

  Author participation: 100 %

- NEVORAL Jan, RŮŽIČKA Richard and MRÁZEK Vojtěch. Evolutionary Design of Polymorphic Gates Using Ambipolar Transistors. In: *2016 IEEE Symposium Series on Computational Intelligence*. Athens: Institute of Electrical and Electronics Engineers, 2016, pp. 1-8. ISBN 978-1-5090-4240-1.

  Author participation: 20 %

  Conference ranking: B5 (Qualis)

- VAŠÍČEK Zdeněk, MRÁZEK Vojtěch and SEKANINA Lukáš. Evolutionary Functional Approximation of Circuits Implemented into FPGAs. In: *2016 IEEE Symposium Series on Computational Intelligence*. Athens: Institute of Electrical and Electronics Engineers, 2016, pp. 1-8. ISBN 978-1-5090-4240-1.

  Author participation: 20 %

  Conference ranking: B5 (Qualis)

2015

- MRÁZEK Vojtěch and VAŠÍČEK Zdeněk. Automatic Design of Low-Power VLSI Circuits: Accurate and Approximate Multipliers. In: *Proceedings of 13th IEEE/IFIP International Conference on Embedded and Ubiquitous Computing*. Porto: Institute of Electrical and Electronics Engineers, 2015, pp. 106-113. ISBN 978-1-4673-8299-1.

  Author participation: 50 %

  Conference ranking: C (CORE) / B2 (Qualis)

- MRÁZEK Vojtěch, VAŠÍČEK Zdeněk and SEKANINA Lukáš. Evolutionary Approximation of Software for Embedded Systems: Median Function. In: *GECCO Companion '15 Proceedings of the Companion Publication of the 2015 on Genetic and Evolutionary Computation Conference*. New York: Association for Computing Machinery, 2015, pp. 795-801. ISBN 978-1-4503-3488-4.

  Author participation: 40 %

- MRÁZEK Vojtěch. Evoluční návrh nízkopříkonových obvodů. In: *Počítačové architektury a diagnostika PAD 2015*. Zlín: Faculty of Applied Informatics, Tomas Bata University in Zlín, 2015, pp. 1-6. ISBN 978-80-7454-522-1.

  Author participation: 100 %

2014

- MRÁZEK Vojtěch and VAŠÍČEK Zdeněk. Acceleration of Transistor-Level Evolution using Xilinx Zynq Platform. In: *2014 IEEE International Conference on Evolvable Systems Proceedings*. Piscataway: Institute of Electrical and Electronics Engineers, 2014, pp. 9-16. ISBN 978-1-4799-4480-4.

  Author participation: 20 %

  Conference ranking: B5 (Qualis)

- MRÁZEK Vojtěch. Akcelerace evolučního návrhu digitálních obvodů na úrovni tranzistorů s využitím platformy Zynq. In: *Proceedings of the 20th Student Conference, EEICT 2014*. Brno: Brno University of Technology, 2014, pp. 229-231. ISBN 978-80-214-4923-7.

Author participation: 100 %

3.4 Research projects and grants

- FIT-S-17-3994 — *Advanced parallel and embedded computer systems, Brno Univesity of Technology.* Team member.
- FIT/FSI-J-17-4294 — *Enhancement of genetic optimization methods for computer engineering, Brno Univesity of Technology.* Investigator.
- GA16-08565S — *Advancing cryptanalytic methods through evolutionary computing, Czech Science Foundation.* Team member.
- LQ1602 — *IT4Innovations excellence in science, Ministry of Education, Youth and Sports of Czech Republic.* Team member.
- GA16-17538S — *Relaxed equivalence checking for approximate computing, Czech Science Foundation.* Team member.
- GA14-04197S — *Advanced Methods for Evolutionary Design of Complex Digital Circuits, Czech Science Foundation.* Team member.
- FIT-S-14-2297 — *Architecture of parallel and embedded computer systems, Brno Univesity of Technology.* Team member.

3.5 Awards

- The Best IP Award at the Design, Automation and Test in Europe (DATE) conference 2017.
- Prof. Ing. Jan Hlavička, DrSc. award for the excellent results in the PhD study at Computer Architectures & Diagnostics workshop for PhD students 2015.
- 1st prize in the Student EEICT 2014 competition awarded for the paper *Acceleration of Transistor-Level Evolutionary Design of Digital Circuits Using Zynq.*
- 3rd prize in the Student EEICT 2012 competition awarded for the paper *Intelligent energy measurement device.*

Chapter 4

Conclusions

This chapter summarizes the challenges and the contributions of the thesis. Then the outcomes for the research community are discussed. Finally, directions of the future research are proposed.

4.1 Challenges and Methodology

The biggest challenge of the work was to combine two, clearly independent, topics: the design of VLSI circuits and the evolutionary design. Although the evolutionary design has been successfully employed in the digital circuit design, only a few papers had been published on leading hardware conferences. During this work, the gap between hardware design and evolutionary optimization seems to get narrowed, because some of our papers have been accepted on the top conferences such as ICCAD or DATE. Two main approaches helped us in this direction. The first was the considering of real hardware parameters such as area and power consumption in the design process. The second was that we focused on approximate circuits and their applications.

In addition to that, we had to deal with the scalability of evolutionary design, mostly with the scalability of evaluation. We performed several crucial optimizations that led to the fast circuit parameters determination. There were two contradictory requirements in the evaluation of the circuits — the speed and the accuracy. Although various accurate approaches for the power and area estimation had been implemented in synthesis tools, we had to propose and implement own algorithms showing good tradeoffs between the accuracy and the computational intensity. Similarly, the standard equivalence checking algorithms are not sufficient for approximate computing and relaxed equivalence checking algorithms had to be implemented and optimized.

In this work, an automated approximation methodology was proposed for digital circuits. This methodology can work on various level of description and outperforms the manual and systematic approaches. This work brought new ideas for the hardware as well as the evolutionary community. The main contribution for the hardware community is that search-based algorithms can provide better results than the state-of-the-art methods for approximation of digital circuits and various application-specific error metrics or constraints can easily be considered in the design. The new contributions for the evolutionary computing community are: (i) the advanced fitness evaluation can significantly improve the scalability of evolutionary circuit design, and (ii) considering the state-of-the-art solutions

as starting points allows to optimize more complex circuits than the evolutionary design from scratch.

4.2 Contributions

This section summarizes the contributions of my research to automated approximate synthesis on the selected levels of circuit description.

Contributions to the automated synthesis of transistor-level circuits We improved the CGP representation to easily handle the circuits described on the transistor-level. For circuit evaluation, a fast multi-valued discrete simulation was developed, implemented and evaluated. This approach allowed us to design a human-competitive one-bit full-adder and other competitive digital primitives [39]. The discrete simulator was also accelerated on an FPGA [37]. Since the proposed discrete simulator does not capture all relevant physical phenomena, timing issues occurred during the simulation of large circuits. A novel combination of the less accurate, but fast discrete simulation with the accurate, but slow numeric SPICE simulation was introduced [38]. Finally, a two-stage algorithm combining gate-level approximation and transistor-level optimization was introduced [38, 41].

Contributions to the automated synthesis of gate-level circuits In the synthesis of circuits described on the gate-level, we focused on approximation of arithmetic circuits. The objective was to provide circuits showing better tradeoffs between power consumption and error than existing methodologies. We adopted the error-oriented CGP and evolved complex approximate adders and multipliers. These circuits were applied in two real-world applications. We showed, that image classification using neural networks [36] as well as DCT transformation in HEVC encoder [61] can be implemented with significantly reduced power consumption while the error is only slightly worsened.

As a part of these case studies, we utilized and improved three main approaches for the error evaluation — parallel simulation (vectorization) [35, 14, 36], BDD-based formal verification [61, 42] and SAT-based formal verification [12]. We compared the performance of these approaches in the approximate adders synthesis task (Tab. 4.1).

Table 4.1: The worst recorded time needed to perform the error analysis of candidate unsigned adders. The average time is reported from $5 \cdot 10^6$ evaluation of candidate circuits

	16-bit adder			**20-bit adder**			**32-bit adder**		
approach	e_{prob}	e_{wce}	e_{mae}	e_{prob}	e_{wce}	e_{mae}	e_{prob}	e_{wce}	e_{mae}
vectorization	40 s	41 s	42 s	386 s	390 s	392 s	n/a	n/a	n/a
BDD	1 us	1 ms	1 ms	17 us	7 ms	9 ms	0.2 ms	9 ms	18 ms
SAT	n/a	2 ms	n/a	n/a	4 ms	n/a	n/a	7 ms	n/a

The parallel simulation represents a universal method which can calculate virtually every error metric, but its bad scalability is the main limiting factor. The BDD-based error analysis is a very efficient approach which scales very well on some cases of circuits, but it cannot be applied to multipliers or dividers (and other structurally similar circuits). It is not applicable to calculate some error metrics, e.g. the relative error which requires

division. The SAT-based method also scales very well but it is applicable to the worst-case error analysis only.

In order to obtain the worst-case error of a complex arithmetic circuit, the problem can be formulated as a satisfiability problem and solved with a SAT solver. However, this approach does not scale for multipliers. Hence, we introduced a verifiability driven CGP in which the SAT solver can use only a very short time to prove or disprove the submitted formula representing a candidate solution. Our method enables to generate and evaluate more candidate designs in a single evolutionary run than if there is no time limit for SAT solver [12].

Contributions to the automated synthesis of RTL circuits We selected the design of approximate median filters as an example of the approximate synthesis of circuits described using complex RTL modules. We introduced a new error metric called *distance error* whose main property is that it is a deterministic, input-data independent metric [60]. This metric can be utilized in the approximate sorting-networks characterization [40].

Moreover, it was possible to transform the evolved median filters described on the RTL level to a program for embedded systems (by transformation of RTL modules to software functions). We thus proposed fast (and therefore power-efficient) approximate median filters for microprocessors. We also showed that the EA-based approximations define a new direction in the promising topic called *genetic improvement of software* [43].

Summary of contributions In Chapter 1, we defined the hypothesis, that *it is possible to reduce power dissipation of digital combinational circuits described on various levels of abstraction using EA-based circuit approximation algorithms.* We selected four different real-world applications described on three levels of abstraction. For each target application, the following steps were taken. At first, an effective power consumption estimation method was developed. This method was employed in the EA-based search algorithm. In order to speed up the circuit synthesis, the evaluation of candidate circuits was accelerated. Table 4.2 summarizes the power-estimation method, the modification of EA-based algorithm and the acceleration technique used in selected applications.

Table 4.2: Overview of approaches taken for selected real-world applications

Target application	Circuit representation	Power estimation	Search algorithm	Fitness acceleration
Tech. library	transistors	switching activity	CGP (two stages)	discrete simulation
Neural networks	gates	area-correlation	error-oriented CGP	vectorization
DCT for HEVC	gates	dynamic power	error-oriented CGP	BDD verification
Median filters	RTL modules	area-correlation	resources-oriented CGP	permutation principle

Finally, the obtained results were compared with conventional solutions. We found the transistor-level circuits realizing approximate multiplication more effectively than circuits obtained using a simple CMOS gate substitution. The neural network image classifiers employing the proposed approximate multipliers traded the power consumption for the overall classification accuracy better than a manual approximation. DCT blocks as well as median filter approximations had lower power dissipation, but acceptable approximation error in comparison with the approximate multiplier-less multipliers. Approximate median filters showed significant power savings with respect to the optimal accurate solutions.

Hereby, based on these results, we can confirm, that EA-based approximation algorithms can create circuits with lower power consumption than conventional approaches.

4.3 Developed libraries available online

As a part of the research, we have developed the *EvoApprox8B* library. This library contains approximate hardware (HDL) and software (C/C++) models of arithmetic circuits[1]. Fig. 4.1 shows the user interface which provides all important parameters. The EvoApprox8B library currently contains hundreds of approximate 8-bit adders and 8-, 12-, 16- and 32-bit multipliers. These circuits can be used either for benchmarking of circuit approximation tools or directly in user applications such as image filters, neural network accelerators or special big-data processing systems.

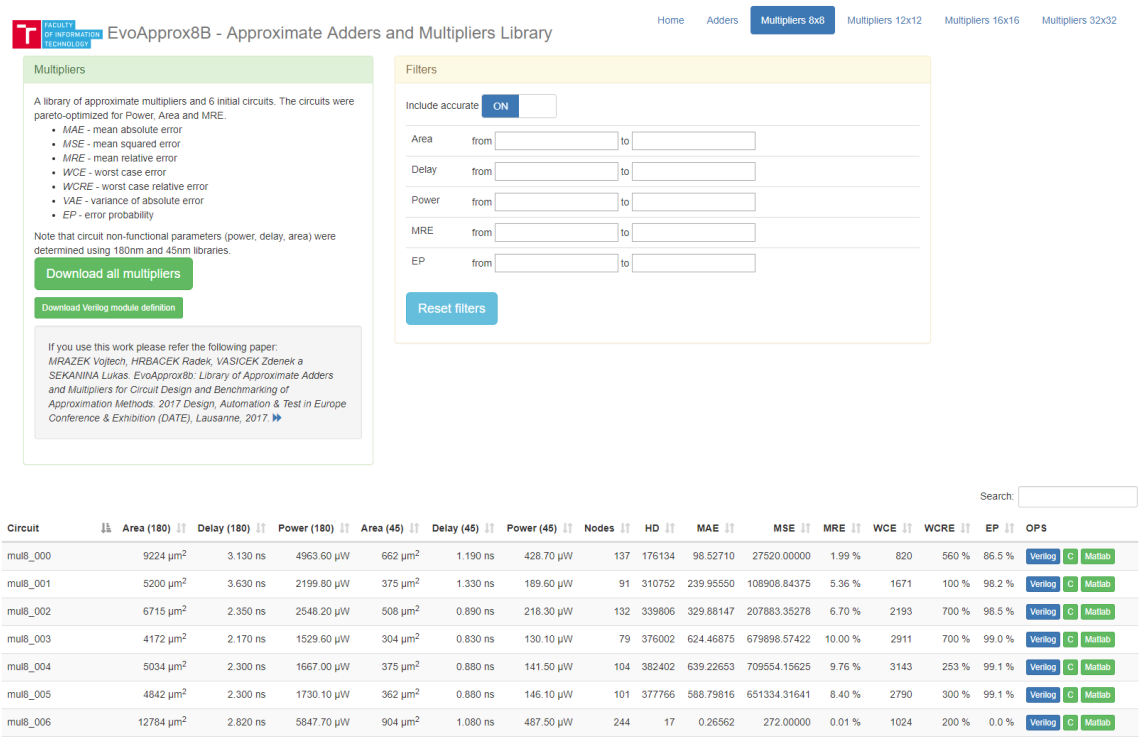

Figure 4.1: Screenshot of the EvoApprox8B website

We also made online the SPICE netlists of twelve 4-bit approximate multipliers that were obtained using CGP operating on the transistor level description[2]. The software implementations realizing evolved approximate median filters were also published online[3].

[1] `https://www.fit.vutbr.cz/research/groups/ehw/approxlib`
[2] `http://www.fit.vutbr.cz/~imrazek/euc2015/`
[3] `http://www.fit.vutbr.cz/~imrazek/median2015/`

4.4 Future research directions

This thesis dealt with application-oriented approximation of digital circuits. There are many different ways presented in the following list how to further develop the research results.

- The evolutionary optimization of transistor-level circuits could be used for non-conventional technologies such as ambipolar transistors.
- A comprehensive library of arithmetic circuits has been developed. These circuits could be employed in other real-world applications and enable to determine the most significant functional criteria influencing the accuracy of the target application.
- The SAT-based verification with termination conditions for SAT solver was very successful for evolutionary approximation of arithmetic circuits. Additional improvement could be obtained by introducing an adaptive termination strategy.
- Error-calculation algorithms employing BDD structures could be further improved by an advanced miter construction without absolute values calculators.
- New applications of approximate sorting networks and median filters such as statistic indicators in big-data datasets could be proposed and evaluated.

Bibliography

[1] Chan, W. T. J.; Kahng, A. B.; Kang, S.; et al.: Statistical analysis and modeling for error composition in approximate computation circuits. In *2013 IEEE 31st International Conference on Computer Design (ICCD)*. Oct 2013. ISSN 1063-6404. pp. 47–53.

[2] Chandrasekharan, A.; Soeken, M.; Große, D.; et al.: Approximation-aware Rewriting of AIGs for Error Tolerant Applications. In *Proceedings of the 35th International Conference on Computer-Aided Design*. ICCAD '16. New York, NY, USA: ACM. 2016. ISBN 978-1-4503-4466-1. pp. 83:1–83:8.

[3] Chauhan, Y. S.; Venugopalan, S.; Karim, M. A.; et al.: BSIM — Industry standard compact MOSFET models. In *2012 Proceedings of the European Solid-State Device Research Conference (ESSDERC)*. Sept 2012. ISSN 1930-8876. pp. 46–49.

[4] Chen, T.-H.; Alaghi, A.; Hayes, J. P.: Behavior of stochastic circuits under severe error conditions. *it - Information Technology*. vol. 56, no. 4. jan 2014.

[5] Chippa, V. K.; Chakradhar, S. T.; Roy, K.; et al.: Analysis and characterization of inherent application resilience for approximate computing. In *The 50th Annual Design Automation Conference 2013, DAC'13*. ACM. 2013. pp. 1–9.

[6] Chopra, K.; Kashyap, C.; Su, H.; et al.: Current source driver model synthesis and worst-case alignment for accurate timing and noise analysis. In *ACM/IEEE International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems*. 2006. pp. 45–50.

[7] Coudert, O.: Gate sizing for constrained delay/power/area optimization. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*. vol. 5, no. 4. Dec 1997: pp. 465–472. ISSN 1063-8210.

[8] Deb, K.; Pratap, A.; Agarwal, S.; et al.: A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*. vol. 6, no. 2. Apr 2002: pp. 182–197. ISSN 1089-778X.

[9] Du, K.; Varman, P.; Mohanram, K.: High Performance Reliable Variable Latency Carry Select Addition. In *Proceedings of the Conference on Design, Automation and Test in Europe*. DATE '12. San Jose, CA, USA: EDA Consortium. 2012. ISBN 978-3-9810801-8-6. pp. 1257–1262.

[10] Du, Z.; Palem, K.; Lingamneni, A.; et al.: Leveraging the error resilience of machine-learning applications for designing highly energy efficient accelerators. In

2014 19th Asia and South Pacific Design Automation Conference (ASP-DAC). Jan 2014. ISSN 2153-6961. pp. 201–206. doi:10.1109/ASPDAC.2014.6742890.

[11] Dunga, M. V.; Yang, W.; Xi, X.; et al.: *BSIM 4.6.1 MOSFET Model User's Manual*. Department of Electrical Engineering and Computer Sciences, UC Berkeley. 2007.

[12] Češka, M.; Matyaš, J.; Mrazek, V.; et al.: Approximating complex arithmetic circuits with formal error guarantees: 32-bit multipliers accomplished. In *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. Nov 2017. pp. 416–423.

[13] Esmaeilzadeh, H.; Sampson, A.; Ceze, L.; et al.: Neural Acceleration for General-Purpose Approximate Programs. In *Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*. MICRO-45. Washington, DC, USA: IEEE Computer Society. 2012. ISBN 978-0-7695-4924-8. pp. 449–460.

[14] Hrbacek, R.; Mrazek, V.; Vasicek, Z.: Automatic design of approximate circuits by means of multi-objective evolutionary algorithms. In *2016 International Conference on Design and Technology of Integrated Systems in Nanoscale Era (DTIS)*. April 2016. pp. 1–6.

[15] Hrbacek, R.; Sekanina, L.: Towards Highly Optimized Cartesian Genetic Programming: From Sequential via SIMD and Thread to Massive Parallel Implementation. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*. GECCO '14. New York, NY, USA: ACM. 2014. ISBN 978-1-4503-2662-9. pp. 1015–1022.

[16] Jiang, H.; Liu, C.; Liu, L.; et al.: A Review, Classification, and Comparative Evaluation of Approximate Arithmetic Circuits. *J. Emerg. Technol. Comput. Syst.*. vol. 13, no. 4. August 2017: pp. 60:1–60:34. ISSN 1550-4832.

[17] Kahng, A. B.; Liu, B.; Xu, X.: Constructing Current-Based Gate Models Based on Existing Timing Library. In *Proceedings of the 7th International Symposium on Quality Electronic Design*. ISQED '06. Washington, DC, USA: IEEE Computer Society. 2006. ISBN 0-7695-2523-7. pp. 37–42.

[18] Kim, N. S.; Austin, T.; Baauw, D.; et al.: Leakage current: Moore's law meets static power. *Computer*. vol. 36, no. 12. Dec 2003: pp. 68–75. ISSN 0018-9162.

[19] Kulkarni, P.; Gupta, P.; Ercegovac, M.: Trading Accuracy for Power with an Underdesigned Multiplier Architecture. In *2011 24th Internatioal Conference on VLSI Design*. Jan 2011. ISSN 1063-9667. pp. 346–351.

[20] Kumar, R.; Liu, Z.; Kursun, V.: TECHNIQUE FOR ACCURATE POWER AND ENERGY MEASUREMENT WITH THE COMPUTER-AIDED DESIGN TOOLS. *Journal of Circuits, Systems and Computers*. vol. 17, no. 03. 2008: pp. 399–421.

[21] Kyaw, K. Y.; Goh, W. L.; Yeo, K. S.: Low-power high-speed multiplier for error-tolerant application. In *2010 IEEE International Conference of Electron Devices and Solid-State Circuits (EDSSC)*. Dec 2010. pp. 1–4.

[22] Li, C.; Luo, W.; Sapatnekar, S. S.; et al.: Joint Precision Optimization and High Level Synthesis for Approximate Computing. In *Proceedings of the 52Nd Annual Design Automation Conference*. DAC '15. New York, NY, USA: ACM. 2015. ISBN 978-1-4503-3520-1. pp. 104:1–104:6.

[23] Liu, G.; Zhang, Z.: Statistically certified approximate logic synthesis. In *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. Nov 2017. pp. 344–351.

[24] Lotfi, A.; Rahimi, A.; Yazdanbakhsh, A.; et al.: Grater: An approximation workflow for exploiting data-level parallelism in FPGA acceleration. In *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*. March 2016. pp. 1279–1284.

[25] Lu, S.-L.: Speeding up processing with approximation circuits. *Computer*. vol. 37, no. 3. Mar 2004: pp. 67–73. ISSN 0018-9162.

[26] Machado, F.; Riesgo, T.; Torroja, Y.: A Method for Switching Activity Analysis of VHDL-RTL Combinatorial Circuits. In *Proceedings of the 16th International Conference on Integrated Circuit and System Design: Power and Timing Modeling, Optimization and Simulation*. PATMOS'06. Berlin, Heidelberg: Springer-Verlag. 2006. ISBN 3-540-39094-4, 978-3-540-39094-7. pp. 645–657.

[27] Mahdiani, H. R.; Ahmadi, A.; Fakhraie, S. M.; et al.: Bio-Inspired Imprecise Computational Blocks for Efficient VLSI Implementation of Soft-Computing Applications. *IEEE Transactions on Circuits and Systems I: Regular Papers*. vol. 57, no. 4. April 2010: pp. 850–862. ISSN 1549-8328.

[28] Mead, C.; Conway, L.: *Introduction to VLSI Systems*. Philiphines: Addison-Wesley. 1980. ISBN 978-0201043587.

[29] Miller, J. F.: *Cartesian Genetic Programming*. Springer-Verlag. 2011.

[30] Miller, J. F.; Thomson, P.; Fogarty, T.: *Designing Electronic Circuits Using Evolutionary Algorithms. Arithmetic Circuits: A Case Study*. Wiley. 1998. pp. 105–131.

[31] Mishchenko, A.; Chatterjee, S.; Brayton, R.: DAG-aware AIG rewriting: a fresh look at combinational logic synthesis. In *2006 43rd ACM/IEEE Design Automation Conference*. July 2006. ISSN 0738-100X. pp. 532–535.

[32] Mittal, S.: A Survey of Techniques for Approximate Computing. *ACM Comput. Surv.*. vol. 48, no. 4. March 2016: pp. 62:1–62:33. ISSN 0360-0300.

[33] Mohapatra, D.; Chippa, V. K.; Raghunathan, A.; et al.: Design of voltage-scalable meta-functions for approximate computing. In *2011 Design, Automation Test in Europe*. March 2011. ISSN 1530-1591. pp. 1–6.

[34] Momeni, A.; Han, J.; Montuschi, P.; et al.: Design and Analysis of Approximate Compressors for Multiplication. *IEEE Transactions on Computers*. vol. 64, no. 4. April 2015: pp. 984–994. ISSN 0018-9340.

[35] Mrazek, V.; Hrbacek, R.; Vasicek, Z.; et al.: EvoApprox8b: Library of Approximate Adders and Multipliers for Circuit Design and Benchmarking of Approximation Methods. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*. March 2017. pp. 258–261.

[36] Mrazek, V.; Sarwar, S. S.; Sekanina, L.; et al.: Design of Power-efficient Approximate Multipliers for Approximate Artificial Neural Networks. In *Proceedings of the 35th International Conference on Computer-Aided Design*. ICCAD '16. New York, NY, USA: ACM. 2016. ISBN 978-1-4503-4466-1. pp. 81:1–81:7.

[37] Mrazek, V.; Vasicek, Z.: Acceleration of transistor-level evolution using Xilinx Zynq Platform. In *2014 IEEE International Conference on Evolvable Systems*. Dec 2014. pp. 9–16.

[38] Mrazek, V.; Vasicek, Z.: Automatic Design of Low-Power VLSI Circuits: Accurate and Approximate Multipliers. In *2015 IEEE 13th International Conference on Embedded and Ubiquitous Computing*. Oct 2015. pp. 106–113.

[39] Mrazek, V.; Vasicek, Z.: Evolutionary Design of Transistor Level Digital Circuits Using Discrete Simulation. In *Genetic Programming*, edited by P. Machado; M. I. Heywood; J. McDermott; M. Castelli; P. García-Sánchez; P. Burelli; S. Risi; K. Sim. Cham: Springer International Publishing. 2015. ISBN 978-3-319-16501-1. pp. 66–77.

[40] Mrazek, V.; Vasicek, Z.: Automatic design of arbitrary-size approximate sorting networks with error guarantee. In *2016 26th International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS)*. Sept 2016. pp. 221–228.

[41] Mrazek, V.; Vasicek, Z.: Parallel Optimization of Transistor Level Circuits Using Cartesian Genetic Programming. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. GECCO '17. New York, NY, USA: ACM. 2017. ISBN 978-1-4503-4939-0. pp. 1849–1856.

[42] Mrazek, V.; Vasicek, Z.: Evolutionary Design of Large Approximate Adders Optimized for Various Error Criteria. In *GECCO Companion '18 Proceedings of the Companion Publication of the 2018 on Genetic and Evolutionary Computation Conference*. Association for Computing Machinery. 2018. ISBN 978-1-4503-5764-7. pp. 1–2.

[43] Mrazek, V.; Vasicek, Z.; Sekanina, L.: Evolutionary Approximation of Software for Embedded Systems: Median Function. In *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*. GECCO Companion '15. New York, NY, USA: ACM. 2015. ISBN 978-1-4503-3488-4. pp. 795–801.

[44] Najm, F. N.: A survey of power estimation techniques in VLSI circuits. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*. vol. 2, no. 4. Dec 1994: pp. 446–455. ISSN 1063-8210.

[45] Nepal, K.; Hashemi, S.; Tann, H.; et al.: Automated High-Level Generation of Low-Power Approximate Computing Circuits. *IEEE Transactions on Emerging Topics in Computing*. 2017: pp. 1–1. ISSN 2168-6750.

[46] Nepal, K.; Li, Y.; Bahar, R. I.; et al.: ABACUS: A technique for automated behavioral synthesis of approximate computing circuits. In *2014 Design, Automation Test in Europe Conference Exhibition (DATE)*. March 2014. ISSN 1530-1591. pp. 1–6.

[47] Ranjan, A.; Raha, A.; Venkataramani, S.; et al.: ASLAN: Synthesis of approximate sequential circuits. In *2014 Design, Automation Test in Europe Conference Exhibition (DATE)*. March 2014. ISSN 1530-1591. pp. 1–6.

[48] Roy, K.; Prasad, S. C.: *Low-power CMOS VLSI circuit design*. New York: Wiley. 2000. ISBN 0-471-11488-x.

[49] Rusu, S.; Tam, S.; Muljono, H.; et al.: A 45 nm 8-Core Enterprise Xeon Processor. *IEEE Journal of Solid-State Circuits*. vol. 45, no. 1. Jan 2010: pp. 7–14. ISSN 0018-9200.

[50] Salvador, R.; Otero, A.; Mora, J.; et al.: Implementation techniques for evolvable HW systems: virtual VS. dynamic reconfiguration. In *22nd International Conference on Field Programmable Logic and Applications (FPL)*. Aug 2012. ISSN 1946-147X. pp. 547–550.

[51] Sarwar, S. S.; Venkataramani, S.; Raghunathan, A.; et al.: Multiplier-less Artificial Neurons exploiting error resiliency for energy-efficient neural computing. In *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*. March 2016. pp. 145–150.

[52] Sekanina, L.: *Evolvable Components - From Theory to Hardware Implementations*. Springer Berlin Heidelberg. 2003. ISBN 3-540-40377-9. 194 pp.

[53] Sekanina, L.: Introduction to Approximate Computing: Embedded Tutorial. In *19th IEEE International Symposium on Design and Diagnostics of Electronic Circuits and Systems*. Institute of Electrical and Electronics Engineers. 2016. ISBN 978-1-5090-2467-4. pp. 90–95.

[54] Sekanina, L.; Vašíček, Z.: Approximate Circuit Design by Means of Evolvable Hardware. In *2013 IEEE International Conference on Evolvable Systems (ICES)*. Proceedings of the 2013 IEEE Symposium Series on Computational Intelligence (SSCI). IEEE Computer Society. 2013. ISBN 978-1-4673-5847-7. pp. 21–28.

[55] Soeken, M.; Große, D.; Chandrasekharan, A.; et al.: BDD minimization for approximate computing. In *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*. Jan 2016. pp. 474–479.

[56] Srinivasan, G.; Wijesinghe, P.; Sarwar, S. S.; et al.: Significance driven hybrid 8T-6T SRAM for energy-efficient synaptic storage in artificial neural networks. In *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*. March 2016. pp. 151–156.

[57] Vasicek, Z.: Cartesian GP in Optimization of Combinational Circuits with Hundreds of Inputs and Thousands of Gates. In *Genetic Programming*. Cham: Springer International Publishing. 2015. ISBN 978-3-319-16501-1. pp. 139–150.

[58] Vasicek, Z.: *New Methods for Synthesis and Approximation of Logic Circuits*. Habilitation thesis. Faculty of Information Technology, Brno University of Technology. 2017.

[59] Vasicek, Z.: Relaxed equivalence checking: a new challenge in logic synthesis. In *2017 IEEE 20th International Symposium on Design and Diagnostics of Electronic Circuits Systems (DDECS)*. April 2017. pp. 1–6.

[60] Vasicek, Z.; Mrazek, V.: Trading between quality and non-functional properties of median filter in embedded systems. *Genetic Programming and Evolvable Machines*. vol. 18, no. 1. Mar 2017: pp. 45–82. ISSN 1573-7632.

[61] Vasicek, Z.; Mrazek, V.; Sekanina, L.: Towards low power approximate DCT architecture for HEVC standard. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*. March 2017. pp. 1576–1581.

[62] Vasicek, Z.; Sekanina, L.: Evolutionary design of approximate multipliers under different error metrics. In *17th International Symposium on Design and Diagnostics of Electronic Circuits Systems*. April 2014. pp. 135–140.

[63] Vasicek, Z.; Sekanina, L.: How to evolve complex combinational circuits from scratch? In *2014 IEEE International Conference on Evolvable Systems*. Dec 2014. pp. 133–140.

[64] Vasicek, Z.; Sekanina, L.: Evolutionary Approach to Approximate Digital Circuits Design. *IEEE Transactions on Evolutionary Computation*. vol. 19, no. 3. June 2015: pp. 432–444. ISSN 1089-778X.

[65] Vašíček, Z.; Slaný, K.: Efficient Phenotype Evaluation in Cartesian Genetic Programming. In *Genetic Programming*. Berlin, Heidelberg: Springer Berlin Heidelberg. 2012. ISBN 978-3-642-29139-5. pp. 266–278.

[66] Vaverka, F.; Hrbacek, R.; Sekanina, L.: Evolving component library for approximate high level synthesis. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*. Dec 2016. pp. 1–8.

[67] Venkataramani, S.; Ranjan, A.; Roy, K.; et al.: AxNN: Energy-efficient neuromorphic systems using approximate computing. In *2014 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*. Aug 2014. pp. 27–32.

[68] Venkataramani, S.; Roy, K.; Raghunathan, A.: Substitute-and-simplify: A unified design paradigm for approximate and quality configurable circuits. In *2013 Design, Automation Test in Europe Conference Exhibition (DATE)*. March 2013. ISSN 1530-1591. pp. 1367–1372.

[69] Venkataramani, S.; Sabne, A.; Kozhikkottu, V.; et al.: SALSA: Systematic logic synthesis of approximate circuits. In *DAC Design Automation Conference 2012*. June 2012. ISSN 0738-100X. pp. 796–801.

[70] Walker, J. A.; Hilder, J. A.; Tyrrell, A. M.: Evolving Variability-Tolerant CMOS Designs. In *Evolvable Systems: From Biology to Hardware*. Berlin, Heidelberg: Springer Berlin Heidelberg. 2008. ISBN 978-3-540-85857-7. pp. 308–319.

[71] Weste, N.; Harris, D.: *CMOS VLSI Design: A Circuits and Systems Perspective*. USA: Addison-Wesley Publishing Company. fourth edition. 2010. ISBN 0321547748, 9780321547743.

[72] Xu, Q.; Mytkowicz, T.; Kim, N. S.: Approximate Computing: A Survey. *IEEE Design Test*. vol. 33, no. 1. Feb 2016: pp. 8–22. ISSN 2168-2356.

[73] Zhu, N.; Goh, W. L.; Yeo, K. S.: An enhanced low-power high-speed Adder For Error-Tolerant application. In *Proceedings of the 2009 12th International Symposium on Integrated Circuits*. Dec 2009. ISSN 2325-0631. pp. 69–72.

Related papers

Paper I

Evolutionary Design of Transistor Level Digital Circuits using Discrete Simulation

MRÁZEK Vojtěch and VAŠÍČEK Zdeněk

In: *Genetic Programming, 18th European Conference, EuroGP 2015.* Berlin: Springer International Publishing, 2015, pp. 66-77. ISBN 978-3-319-16500-4.

Evolutionary Design of Transistor Level Digital Circuits using Discrete Simulation

Vojtech Mrazek and Zdenek Vasicek

Brno University of Technology, Faculty of Information Technology,
Božetěchova 2, 612 66 Brno, Czech Republic
{imrazek,vasicek}@fit.vutbr.cz

Abstract. The objective of the paper is to introduce a new approach to the evolutionary design of digital circuits conducted directly at transistor level. In order to improve the time consuming evaluation of candidate solutions, a discrete event-driven simulator was introduced. The proposed simulator operates on multiple logic levels to achieve reasonable trade-off between performance and precision. A suitable level of abstraction reflecting the behavior of real MOSFET transistors is utilized to minimize the production of incorrectly working circuits. The proposed approach is evaluated in evolution of basic logic circuits having more than 20 transistors. The goal of an evolutionary algorithm is to design a circuit having the minimal number of transistors and exhibiting the minimal delay. In addition to that, various parameter settings are investigated to increase the success rate of the evolutionary design.

1 Introduction

In recent years, a lot of papers showing the merits of evolutionary design techniques in the field of digital circuits design have been published. Implementation of various combinational circuits competitive to the circuits designed using conventional approaches have been obtained by using cartesian genetic programming (CGP) which is considered to be the most efficient technique to perform the gate-level evolutionary design [2–4, 7].

However, while the gate-level evolutionary design represents an intensively studied research area, the synthesis of transistor-level digital circuits remains, in contrast with design of transistor-level analog circuits, on a peripheral concern of the researchers despite the fact that even some basic logic expressions can be implemented much effectively at transistor level. Only few papers were devoted to evolution of digital circuits directly at transistor level. Zaloudek et al. published an approach based on a simple simulator which was designed to quickly evaluate the candidate solutions [11]. Unfortunately, a rough approximation of transistor behavior caused that this approach produced many incorrectly working circuits. Trefzer used another technique to evolve some basic logic gates [6]. Instead of using a time consuming analog circuit simulator, a reconfigurable analog transistor array was employed. However, it was shown that many of the discovered solutions relied on some properties of the utilized reconfigurable array. About

50% of the evolved circuits failed in the analog simulation. Walker et al. used a different technique to evolve transistor-level circuits [8]. In order to speed up the time consuming evaluation of candidate solutions, a cluster of SPICE-based simulators was utilized. Even if it was possible to evolve correct solutions, only small problem instances could be investigated due to the overhead of SPICE simulators.

A new approach to the evolutionary design of digital circuits is introduced in this paper. In this work, the evolutionary approach operates directly at transistor level. Since the evolutionary-based approach requires generating a large number of candidate solutions, it is necessary to minimize the time needed to evaluate the candidate circuits in order to obtain a satisfactory success rate. However, a reasonable level of abstraction must be applied to avoid production of incorrectly working circuits. In order to address this issue, a discrete simulator which operates on multiple logic levels is proposed. It is expected that this approach enables to achieve reasonable trade-off between performance and precision.

The paper is organized as follows. Section 2 discusses behavior of real unipolar transistors. Section 3 introduces the proposed method. Section 4 summarizes and analyses the obtained results. The analysis of the discovered circuits is performed using a SPICE simulator. Finally, concluding remarks are given in Section 5.

2 Behavior of MOSFET transistors

Behavior of the MOSFET transistors can be described at various levels of abstraction.

At the most accurate level, transistor circuits are modeled using a complex system of equations having tens of parameters that are derived from the underlying device physics. In order to accurately simulate the transistor level circuits, SPICE-based simulators are usually used. Apart from the commercial simulators such as HSPICE or PSPICE, there exist also academic tools such as ngSPICE. Even if the SPICE-based simulators provide a wide variety of MOS transistor models with various trade-offs between complexity and accuracy, the runtime grows rapidly with the increasing size of the simulated circuits. To reduce the time of simulation, a multithreaded version of SPICE simulator or an FPGA-based hardware acceleration can be utilized [1].

On the other hand, so-called switch-level model can be used [10]. A switch-level simulator models MOS circuits using a network of transistors acting as switches. Each transistor can be in one of three discrete states – open, closed or unknown. Compared to the SPICE-based simulators, the speed of the simulation is improved in orders of a magnitude. This model can acquire aspects that cannot be expressed at gate model, however, the accuracy is naturally lower compared to the approach mentioned in the previous paragraph. For example, the value of threshold voltage influencing state of the transistors is completely ignored. Moreover, the accuracy of simulation decreases as the transistors shrinks.

2.1 Discrete model suitable for evolutionary design

As it was discussed in the Section 1, a fast simulator is needed to enable the evolutionary design to sufficiently explore the search space. Simultaneously, reasonable accuracy is required to evolve the correct circuits that will work in real environment. In order to meet these requirements, we propose to utilize a discrete simulator which exhibits speed of the switch-level simulators and accuracy of the SPICE-based simulators. We propose to use a model (abstracted from dynamic parameters such as power consumption or delay) based on the switch-level transistor model extended to a threshold drop degradation effect.

Threshold voltage, commonly abbreviated as V_{tp} (V_{tn}), is the minimum gate-to-source voltage differential that is needed to create a conducting path between the source and drain terminals. As a consequence of the threshold voltage, degraded voltage values can be presented in MOS circuits. An open n-MOS transistor is known to pass logic 0 (i.e. V_{ss}) well but logic 1 (i.e. V_{dd}) poorly. This loss is known as threshold drop. An attempt to pass logic 1 never gives value above $V_{dd} - V_{tn}$. Similarly, p-MOS transistor is known to pass logic 0 poorly. The reduction in voltage swing can be beneficial to the power consumption. The designer has to be careful, however, because the degraded output may cause circuit malfunction.

As a target technology, TSMC with feature size equal to 0.25 μm is chosen. The following parameters of p-MOS and n-MOS transistors will be utilized in MOS circuits. The length of the n-MOS transistor is $L_N = 0.25\ \mu m$, width is $W_N = 0.5\ \mu m$. p-MOS transistors have $L_P = 0.25\ \mu m$ and $W_P = 2W_N = 1\ \mu m$. According to the simulation, $V_{tn} = 0.987\ V$ and $V_{tp} = 0.717\ V$. In order to

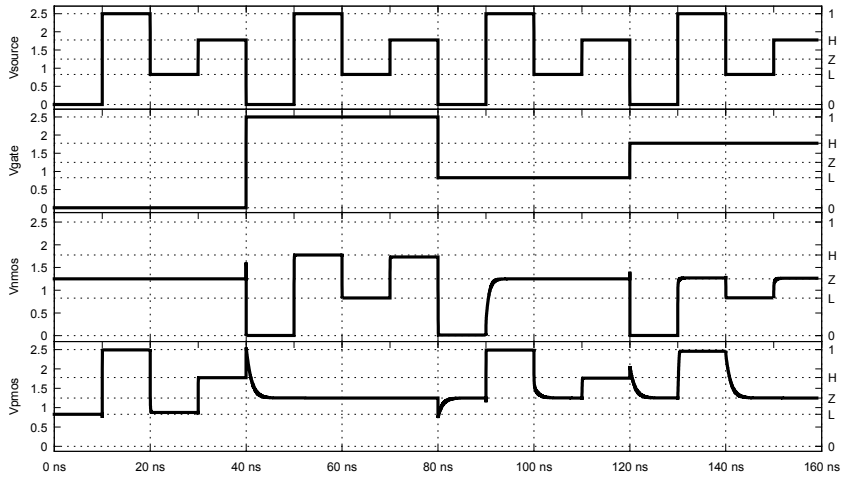

Fig. 1. Output waveforms for p-MOS and n-MOS transistors for various voltage applied to the source and gate terminals. The waveform was obtained using an analog SPICE simulator, a TSMC 0.25 μm technology and 2.5V power supply. The corresponding discrete values are shown on the right side.

Table 1. Behavior of n-MOS and p-MOS transistors modeled using six discrete values.

n-MOS							*p-MOS*						
gate	source						gate	source					
	1	H	L	0	Z	X		1	H	L	0	Z	X
1	H	X	L	0	Z	X	1	Z	Z	Z	Z	Z	X
H	X	X	L	0	Z	X	H	Z	Z	Z	Z	Z	X
L	Z	Z	Z	Z	Z	X	L	1	H	X	X	Z	X
0	Z	Z	Z	Z	Z	X	0	1	H	X	L	Z	X
Z	Z	Z	Z	Z	Z	X	Z	Z	Z	Z	Z	Z	X
X	X	X	X	X	X	X	X	X	X	X	X	X	X

support various implementations of digital circuits, we will distinguish among six voltage levels: logic 0 (denoted as '0'), logic 1 ('1'), degraded 0 (V_{tp}, 'L'), degraded 1 ($V_{dd} - V_{tn}$, 'H'), high impedance ('Z') and undefined value ('X'). A SPICE-based simulator was used to derive the discrete model. The results of simulation are given in Figure 1. The fourth terminal of p-MOS (n-MOS) is connected to V_{dd} (V_{ss}). In order to detect high impedance state, outputs of p-MOS and n-MOS transistors are connected to a voltage divider.

Let us discuss behavior of n-MOS transistor (p-MOS works analogically). If logic 0 is applied to the gate, the transistor is closed and its output is in a high impedance state. The similar situation occurs if 'L' is used. However, if $V_{gate} =$ 'L' and $V_{source} =$ '0', the transistor is not completely closed. As we do not want to model strength of the signal values, we need to suppose that the output is in a high impedance state. This little inaccuracy does not constitute any serious problem due to the presence of stronger values within a circuit. If logic 1 or 'H' is applied to the gate, the transistor is open. Logic 0 as well as 'L' connected to the source are fully transferred to output, but logic 1 and 'H' are degraded. As we can see, the double degraded value can not be recognized from high impedance state. Hence, we have to avoid the double degradations that may cause malfunctions.

The behaviour of n-MOS and p-MOS transistors which follows the results obtained from the SPICE-based simulation valid for the chosen technology and power supply is summarized in Table 1.

3 The proposed method

3.1 Circuit representation

In order to evolve complex digital circuits at the transistor level a suitable representation enabling to encode bidirectional graph structures containing junctions is needed. To address this problem, we proposed an encoding inspired by CGP[2].

Each digital circuit having n_i primary inputs and n_o primary outputs (i.e. a candidate solution) is represented using an array of nodes arranged in n_c columns and n_r rows. Each node consists of two source terminals and one output

terminal. Each node can act as p-MOS transistor, n-MOS transistor, or junction. The utilized nodes are shown in Figure 2. Source terminals of each node can independently be connected to the output terminal of a node placed in previous l columns. In addition to that, source terminals of any transistor node can be connected to one of the primary circuit inputs.

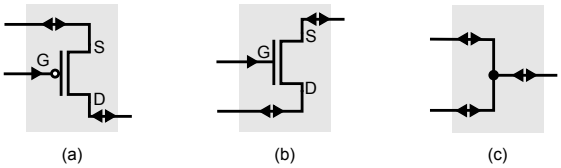

Fig. 2. Basic building blocks of transistor-level circuits: (a) p-MOS transistor, (b) n-MOS transistor, and (c) junction that combines two signals together. If a proper voltage is applied on the gate electrode denoted as G (V_{ss} for p-mos, V_{dd} for n-mos), transistor connects its source electrode (denoted as S) with drain (D). Possible directions of signal flow which have to be considered during the evaluation are shown.

Presence of the junction node represents the main feature of the proposed technique. This node is able to combine two input signals and one output signal together. As a consequence of that, loops and multiple connections are natively supported.

The following encoding scheme is utilized. The primary inputs and node outputs are labeled from 0 to $n_i + n_c \cdot n_r - 1$. A candidate solution is represented in the chromosome by $n_c \cdot n_r$ triplets (x_1, x_2, f) determining for each node its function f, and label of nodes x_1 and x_2 connected to the source terminals.

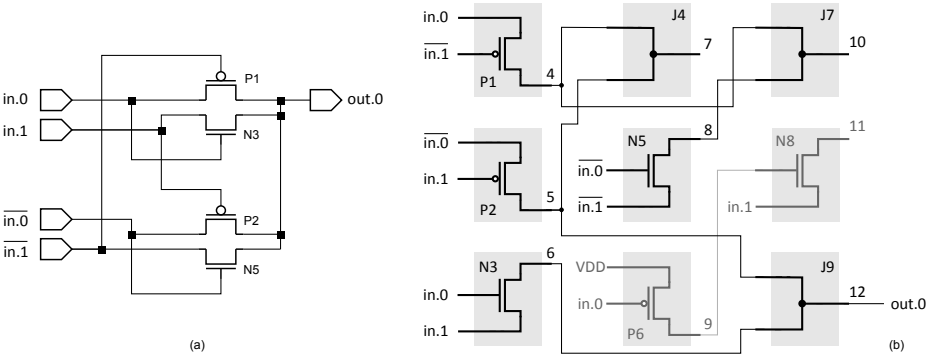

Fig. 3. Example of a candidate circuit implementing function XNOR using eight transistors (four transistors are used to implement inverted variables $\overline{in.0}$ and $\overline{in.1}$). Parameters are as follows: n_i=4 (0,V_{dd},in.0,in.1), n_o=1 (out.0), n_c=3, n_r=3, l=2. Chromosome: *(2,-3,pmos)(-2,3,pmos)(3,2,nmos)(4,5,junction)(-3,-2,nmos)(1,2,pmos)(4,8,junction)(9,3,nmos)(5,6,junction)(12).*

Apart from that, negative indices $-2 - n_i < x_i \leq -2$ are allowed in case of x_i. The negative value indicates that the inverted primary variable labeled as $|x_i|$ is required. The last part of the chromosome contains n_o integers specifying the labels of nodes where the n_o primary outputs are connected to. The first two primary inputs are reserved for power supply rails.

Figure 3 demonstrates the principle of utilized encoding on a XNOR circuit implemented using pass-transistor logic. The shown chromosome encodes a candidate circuit using eight nodes, however, only some of them contribute to the phenotype and are active.

3.2 Evaluation of the candidate solutions

Evaluation of the candidate solutions encoded using the proposed representation consists of two steps.

Firstly, set of active nodes is determined. Only the active nodes are considered during the evaluation. The inactive nodes are ignored. Potentially unwanted nodes causing short-circuits can be removed in this step. A node is active if either (a) its output is connected to any of the primary outputs, or (b) it is a transistor node and its output is connected to the source of an active node, or (c) it is a junction node whose source terminal is connected to an active node. The detection of active nodes can be performed in linear time complexity.

Then, multi-level discrete event-driven simulator is utilized to determine response for each input combination. The advantage of this approach is that only necessary nodes are updated if there is a change of a value. The following steps are used to determine output value of for a given input combination. Firstly, outputs of all nodes are initialized to the value 'Z'. Then, value 0 and 1 are assigned to the first two primary inputs. This change triggers re-evaluation of all the nodes connected directly to the power supply rails. Each node determines its new output value and propagates it to all related nodes. As an open transistor connect source with drain, bidirectional data-flow have to be utilized. It means that the new value must be propagated to the nodes connected not only to the drain but also to source terminal. Similarly, junctions have to propagate the new value to all terminals. The new value of a junction node is calculated as the strongest value presented on all the terminals. The new value of a transistor node is determined according to the value connected to the source as well as drain. During the evaluation of a new output value of a transistor node, the new calculated value is compared with current value at drain terminal. If the values are not compatible, short circuit exception is raised. Otherwise, the stronger value is propagated to all related nodes. The relation between the discrete values is as follows: 'Z' $\prec$ 'L' $\prec$ '0' $\prec$ 'X'; 'Z' $\prec$ 'H' $\prec$ '1' $\prec$ 'X'. It means that if at least one of the values is equal to 'X', 'X' is propagated to all related nodes.

Each transistor has associated a state which determines whether the transistor is in direct or reverse mode. The current flows from drain to source in reverse mode. It happens when 'Z' is assigned to source terminal and a value different from 'Z' is connected to the drain. This state helps to avoid situation in which a double degradation could happen.

In order to avoid malfunction circuits, final test is performed at the end of the simulation. If there is at least a single transistor with 'Z' state assigned to its gate terminal, short circuit exception is raised.

The principle of discrete simulation will be demonstrated for in_0=1 and in_1=0 using the candidate circuit shown in Figure 3. The primary inputs are successively initialized to the following values: V_{dd} (i.e., the primary input with index 0) ← '1', $V_{ss}(1)$ ← '0', $in_0(2)$ ← '1', $in_1(3)$ ← '0'. Then the inverted values are assigned $\overline{in_0}$(-2) ←'0', $\overline{in_1}$(-3)←'1' As no power rail is used in the example, the first two assignments do not trigger any reevaluation. However, assignment of value '1' to in_0 causes that P1 and N3 are evaluated. Nor P1 nor N3 have fully specified inputs, thus these changes do not generate any new event. In the next step, in_1 connected to P2 and N3 is assigned. Now, the node N3 has fully specified inputs and the new calculated value '0' is propagated through drain to the node J9. Then, the value of $\overline{in_0}$ is changed to '0'. As a consequence of that, P2 is evaluated to 'L' and propagated through J4 to J9. In addition to that, N5 is refreshed. Because there is a stronger value, '0', assigned to the other pin of J9, the '0' is propagated back to the output terminal of transistor P2 and junction J4. The, $\overline{in_1}$ becomes to be logical '1'. Transistor P1 is closed, so the drain is in high impedance state. This value is propagated to J4, however '0' presented at the second terminal is stronger and it is propagated back to P1 and then to J7. The last transistor which has to be evaluated is the closed transistor N5 with 'Z' at its output. High impedance state is delivered to J7, but J7 already contains a stronger value '0'. Primary output is connected to the node J9 which has value '0' on its output. This value corresponds with the XNOR specification, so the circuit produces a valid output for the used input vector.

3.3 Search strategy

As a search algorithm, $(1 + \lambda)$ evolutionary strategy is utilized [2]. The initial population is randomly generated. Every new population consists of the best individual and λ offspring created using a point mutation operator which modifies h randomly selected genes. In the case when two or more individuals have received the same fitness score in the previous population, the individual which did not serve as a parent in the previous population will be selected as a new parent. This strategy is used to ensure the diversity of population. The evolution is terminated when a predefined number of generations is exhausted or a required solution is found.

The search is guided by the fitness function which determines how good the current candidate circuit is. For evolution of logic circuits, all possible input combinations have to be applied at the candidate circuit inputs. The output values are collected and the goal is to minimize the difference between obtained responses and required Truth table. In order to smooth the search space, the fitness value is constructed as follows. If an obtained output value equals to the expected one, 5 points are added to the fitness value. If the calculated value exhibits the same polarity but represents degraded voltage, 2 points are used.

Otherwise, no point is added because the response is invalid. Additional penalties may be applied. If there is a short-circuit exception asserted during the simulation, the simulation is terminated and penalty is applied to the total fitness value. Similarly, if the simulator exceeds the predefined number of steps (i.e. node outputs are not in stable state), the simulation is terminated and the fitness value is penalized. As soon as a fully working solution is found, the number of utilized transistors is reduced. Two points are added for each unused node and one point for node which acts as junction. Note that the transistors required to implement inverted input of the utilized variables are considered.

4 Experimental results

The proposed method was evaluated in the evolution of basic logic circuits as well as some benchmark circuits whose conventional solutions consist of up to 30 transistors. In particular, we tried to evolve XOR and XNOR gate, 3 bit majority, 1 bit full adder and benchmark circuits b1, c17, newtag, mc, daio and lion from *LGSynth benchmarks*. The goal of the experiments was to evolve fully functional implementations exhibiting full voltage swing on the outputs.

In order to investigate the effect of array size, three arrangements are used for each benchmark circuit. The first two configurations utilize a single row of nodes, while the third uses an array consisting of two rows. The total number of nodes was chosen according to the number of transistors required to implement a given function using a conventional design approach.

In addition to that, the impact of various connection possibilities was investigated. Firstly, the presence of inverted input variables introduced in Section 3.1 and its impact on the success-rate was studied. Then, additional restriction to the connection of source terminal of p-MOS and drain terminal of n-MOS was applied. We prevent to connect this electrode directly to the primary inputs. As a consequence of that, implementations with higher operating frequency can be evolved. This setup is denoted as 'S/D←N', while the unrestricted setup is denoted as 'S/D←I+N'.

The results were obtained from 20 independent runs using the following experimental setup: $\lambda = 4, l = n_c, h = 5$. The evolution is terminated after 8 hours or when no improvement was achieved within the last hour. All the successfully evolved solutions were validated using a SPICE simulator.

The results were compared with a reference implementation described at gate-level and implemented using standard cells.

The impact of the introduced restriction and the presence of implicit inverters is evaluated by means of a *success proportion* [9]. Success proportion is the cumulative probability of success calculated by the number of runs that have found a solution at or before generation i divided by the total number of runs in the experiment. A successful run is such a run in which a fully working solution was discovered. The results for two chosen benchmark circuits are given in Figure 4. As it can be seen, the usage of implicit inverters significantly increased the performance of the evolutionary design. On the other hand, the restriction

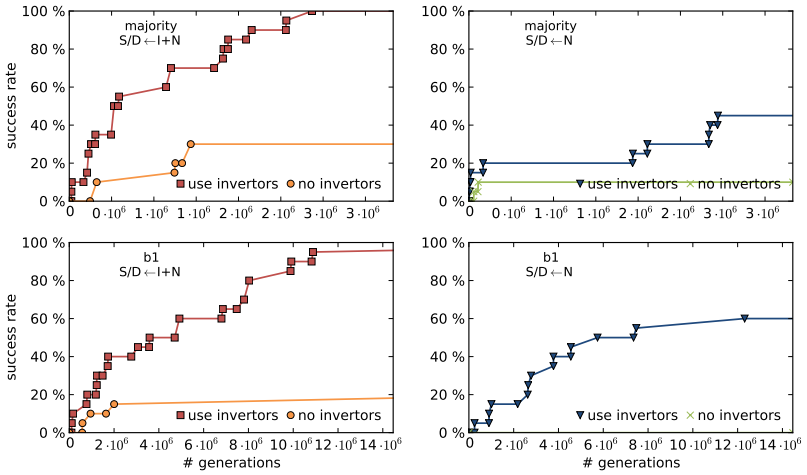

Fig. 4. Success proportion of the evolutionary design of ‘majority‘ and ‘b1‘ benchmark circuits. The array consisting of a single row and 30 columns for ‘majority‘ and 60 columns for ‘b1‘ are used.

applied to the source (drain) terminals of p-MOS (n-MOS) nodes reduce the performance of the evolution. Substantially higher number of generations are needed to achieve the same success rate.

The success rate of the evolutionary design for the chosen digital circuits is summarized in Table 2. In addition to that, we analyzed the evolved solutions and determined the number of utilized transistors (see the last two columns). Similarly to the previous findings, the usage of implicit inverters as well as the unrestricted possibilities of S/D terminal connections improved the performance of the evolutionary approach in all cases. Another parameter which can have a great impact on the success rate is the size of array. Too small array on the one hand and too large array on the other hand have a negative impact on the success rate. While the small array may prevent to find a valid solution because there is not a space to represent a target circuit, large array increases substantially the search space. Fortunately, it seems that increasing of the number of available nodes does not increase the size of the evolved circuit.

The discovered circuits were verified and characterized using a SPICE simulator with an accurate transistor model. Except of a single evolved implementation of ‘b1‘ circuit, all the circuits were valid and operated correctly. Thus we can conclude that the proposed discrete abstraction is successful.

Table 3 summarizes the basic parameters of the evolved solutions and the conventional implementations. Apart from the number of utilized transistors, delay and maximum operating frequency is given. If we compare the maximum operating frequency of the evolved circuits with the conventional circuits, we can see a significant improvement in all cases except the circuit ‘c17‘. This result is

Table 2. Success rate for the benchmark circuits for various array sizes, connection possibilities and availability of inverted primary inputs.

	$n_r \times n_c$	S/D←N+I		S/D←N		# transistors	
		with inv.	w/o inv.	with inv.	w/o inv.	min	max
xnor	1×10	100%	65%	0%	0%	6	8
	1×15	100%	100%	100%	5%	6	12
	2×15	100%	100%	100%	45%	6	12
xor	1×10	100%	75%	0%	0%	6	8
	1×15	100%	100%	100%	5%	6	12
	2×20	100%	100%	100%	5%	6	12
majority	1×20	100%	25%	0%	5%	10	14
	1×30	100%	30%	45%	10%	10	16
	2×30	80%	35%	60%	15%	10	17
adder-1	1×30	30 %	5%	0%	0%	14	20
	1×40	65%	0%	0%	0%	18	20
	2×40	50%	0%	5%	0%	18	25
b1	1×40	100%	15%	40%	0%	12	19
	1×60	100%	20%	60%	0%	12	20
	2×60	75%	5%	25%	0%	12	23
c17	1×40	5%	0%	0%	0%	22	24
	1×60	5%	0%	0%	0%	25	26
	2×60	0%	0%	5%	0%	25	28

very encouraging, because the delay was not optimized explicitly. We analyzed the circuits and determined that this improvement was achieved by replacing traditional gates implemented as CMOS logic with much effective implementation which utilized so-called transmission-gates. The usage of transmission-gates increases the speed but simultaneously reduces the number of utilized transistor.

A lot of different implementations were discovered. Example of an evolved circuit of one bit adder is shown in Figure 5. The discovered circuit is similar to low-power full adder consisting of 14 transistors which was introduced in [5]. The evolution was able to discover an implementation which belongs to the family of pass-transistor logic. The evolved solution utilizes three transmission gates to provide fast and compact solution and exhibits approx. 27% reduction in power consumption compared to the common CMOS implementation. Carry is represented by output labeled as out_0 and sum is available at out_1. Input in_2 corresponds to the input carry.

5 Conclusion

A new approach suitable to the evolutionary design of digital circuits conducted directly at transistor level was introduced in this paper. A discrete event-driven

Table 3. Parameters of the conventional as well as evolved digital circuits. The first part of the table contains the number of inputs, number of outputs and time and number of generations required to evolve the solution. Then, the parameters of conventional implementation are given. (a) Contains parameters of the fastest discovered solution, while (b) contains parameters of the most compact evolved solution.

		xor	xnor	majority	adder-1	b1	c17
	Inputs	2	2	3	3	3	5
	Outputs	1	1	1	2	2	2
	Time of evolution (min)	10	10	10	120	60	480
	Max. # generations	$14 \cdot 10^6$	$14 \cdot 10^6$	$5 \cdot 10^6$	$45 \cdot 10^6$	$30 \cdot 10^6$	$80 \cdot 10^6$
	Delay (ps)	208.3	180.9	335.2	422.7	360.1	324.0
	Frequency (GHz)	4.80	5.53	2.98	2.37	2.78	3.09
	Transistors	8	8	22	48	30	28
	Delay (ps)	87.5	87.8	271.4	291.4	173.2	355.4
(a)	Frequency (GHz)	11.43	11.39	3.68	3.43	5.77	2.81
	Transistors	6	8	16	14	16	24
	Delay (ps)	87.5	142.4	599.3	291.4	401.5	573.8
(b)	Frequency (GHz)	11.43	7.02	1.67	3.43	2.49	1.74
	Transistors	6	6	10	14	12	22

simulator operating on multiple logic levels was utilized to achieve reasonable trade-off between performance and precision. The proposed method was evaluated on a set of benchmark circuits.In order to improve the success rate, implicit inverters were introduced to the encoding.

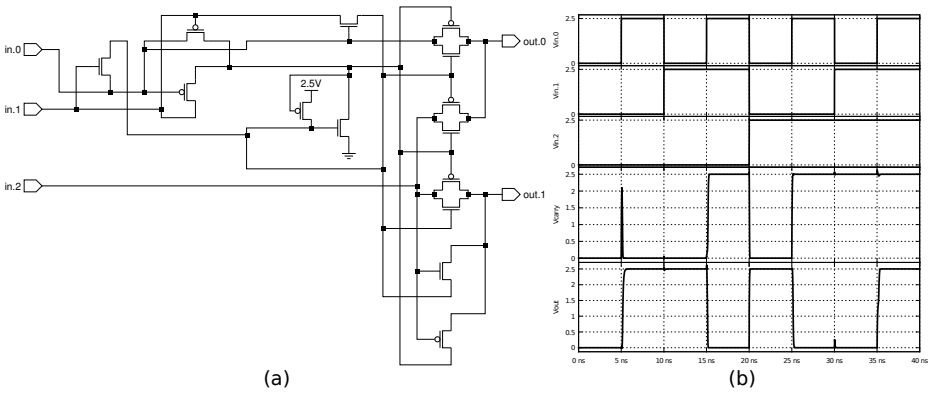

Fig. 5. (a) The most compact and simultaneously the fastest circuit consisting of 14 transistors implementing one-bit full adder. (b) Output waveform obtained using a SPICE simulator.

It was demonstrated that the proposed method is able to produce valid solutions despite the fact that a relative simple discrete model of MOS transistors (compared to the complex models used in SPICE-based simulators) was utilized. According to the analysis of the obtained results, we can confirm, that the evolution was able to discover solutions that are based not only on complementary logic but also on pass-transistor logic.

However, future work has to be conducted to improve the scalability of the proposed method. One of the possible directions is to introduce more complex building blocks such as transmission gate.

Acknowlegement This work was supported by the Czech science foundation project 14-04197S.

References

1. Kapre, N., DeHon, A.: Accelerating spice model-evaluation using fpgas. In: Field Programmable Custom Computing Machines, 2009. FCCM '09. 17th IEEE Symposium on. pp. 37–44 (April 2009)
2. Miller, J.F. (ed.): Cartesian genetic programming. Natural Computing Series, Springer, Berlin, 22. edn. (2011)
3. Miller, J.F., Job, D., Vassilev, V.K.: Principles in the Evolutionary Design of Digital Circuits – Part I. Genetic Programming and Evolvable Machines 1(1), 8–35 (2000)
4. Miller, J.F., Job, D., Vassilev, V.K.: Principles in the Evolutionary Design of Digital Circuits – Part II. Genetic Programming and Evolvable Machines 1(3), 259–288 (2000)
5. Shams, A., Bayoumi, M.: A novel high-performance cmos 1-bit full-adder cell. Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on 47(5), 478–481 (May 2000)
6. Trefzer, M.: Evolution of Transistor Circuits. Ph.D. thesis, Ruprecht-Karls-Universitt Heidelberg (2006)
7. Vassilev, V., Job, D., Miller, J.: Towards the Automatic Design of More Efficient Digital Circuits. In: Proc. of the 2nd NASA/DoD Workshop on Evolvable Hardware. pp. 151–160. IEEE Computer Society, Los Alamitos, CA, USA (2000)
8. Walker, J., Hilder, J., Tyrrell, A.: Evolving variability-tolerant cmos designs. In: Hornby, G., Sekanina, L., Haddow, P. (eds.) Evolvable Systems: From Biology to Hardware. LNCS, vol. 5216, pp. 308–319. Springer Berlin Heidelberg (2008)
9. Walker, M., Edwards, H., Messom, C.H.: Success effort and other statistics for performance comparisons in genetic programming. In: IEEE Congress on Evolutionary Computation. pp. 4631–4638 (2007)
10. Weste, N.H., Harris, D.: CMOS VLSI design: a circuits and systemes perspective. Addison-Wesley, Boston, USA, 3. edn. (2005)
11. Zaloudek, L., Sekanina, L.: Transistor-level evolution of digital circuits using a special circuit simulator. In: Evolvable Systems: From Biology to Hardware. LNCS, vol. 5216, pp. 320–331. Springer Verlag (2008)

Paper II

EvoApprox8b: Library of Approximate Adders and Multipliers for Circuit Design and Benchmarking of Approximation Methods

MRÁZEK Vojtěch, HRBÁČEK Radek, VAŠÍČEK Zdeněk and SEKANINA Lukáš

In: *Proc. of the 2017 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. Lausanne: European Design and Automation Association, 2017, pp. 258-261. ISBN 978-3-9815370-9-3.

EvoApprox8b: Library of Approximate Adders and Multipliers for Circuit Design and Benchmarking of Approximation Methods

Vojtech Mrazek, Radek Hrbacek, Zdenek Vasicek and Lukas Sekanina
Brno University of Technology, Faculty of Information Technology, IT4Innovations Centre of Excellence
Brno, Czech Republic
Email: {imrazek, ihrbacek, vasicek, sekanina}@fit.vutbr.cz

***Abstract*—Approximate circuits and approximate circuit design methodologies attracted a significant attention of researchers as well as industry in recent years. In order to accelerate the approximate circuit and system design process and to support a fair benchmarking of circuit approximation methods, we propose a library of approximate adders and multipliers called EvoApprox8b. This library contains 430 non-dominated 8-bit approximate adders created from 13 conventional adders and 471 non-dominated 8-bit approximate multipliers created from 6 conventional multipliers. These implementations were evolved by a multi-objective Cartesian genetic programming. The EvoApprox8b library provides Verilog, Matlab and C models of all approximate circuits. In addition to standard circuit parameters, the error is given for seven different error metrics. The EvoApprox8b library is available at: www.fit.vutbr.cz/research/groups/ehw/approxlib**

I. Introduction

Approximate circuits are becoming a viable alternative to conventional accurately operating circuits if energy efficiency is crucial and target application is error resilient [1]. This is the case of many, for example, image and video processing circuits that are predominantly composed of adders and multipliers. In order to approximate circuits such as adders and multipliers for a particular application, a designer can either perform a single purpose (ad hoc) approximation or apply some of the circuit approximation methods available in the literature. We will only deal with functional approximation in which logic function implemented by the original circuits is simplified. Other approximation techniques enabling power reduction such as voltage over scaling are not considered in this paper.

Unfortunately, almost all papers dealing with circuit approximation show some of the following features that are undesirable from a practical point of view: (1) The approximation method is described, but a corresponding software implementation is not available. (2) An implementation of the original (accurate) circuit is not available. (3) The quality of approximation and other parameters of approximate circuits are expressed relatively to parameters of the original circuit. If the original circuit is not available, it is hard or even impossible to obtain real parameters of the approximate circuits and reproduce the results. (4) Implementations of the resulting approximate circuits are not available. (5) Only a few approximate versions created from the original circuit are reported, forming thus a sparsely occupied Pareto front. (6) It is unclear if a given number of test vectors used to evaluate approximate circuits is sufficient for obtaining a trustworthy error quantification if the error is determined using simulation. (7) A given approximation method is only rarely compared against competitive approximation methods.

To the best of our knowledge, paper [2] and related software is the only one which does not suffer from the aforementioned problems. However, the paper is solely devoted to 16 bit adders and still only a few approximate adders can be downloaded.

The undesirable properties (1) – (7) are resulting in a situation in which the user does not know which approximation method is the most suitable one for his/her problem because the quality of available approximation methods cannot fairly be compared. If the user does not want to apply any of the approximation methods, its intention could be to use just a resulting approximate circuit showing desired parameter values. However, the chance of direct downloading a circuit with desired parameters is very low, because only a few approximate versions of the original circuit exist in the corresponding repository. On the other hand, it has to be emphasized that a fair comparison of approximation methods is difficult as they should be compared under the same conditions (under the same error metrics, fabrication technology, synthesis tools, available run-time etc.) and start with the same original circuit, which is difficult to ensure in the progressively developing field of approximate computing.

In order to address at least some of the aforementioned challenges, the objective of this paper is to introduce a rich and well-focused library of approximate components (called EvoApprox8b) that can be downloaded and immediately used in various applications or for benchmarking of circuit approximation methods. The library consists of approximate 8-bit adders and 8-bit multipliers that are crucial components of, for example, approximate image and video processing applications. In addition to circuit parameters (error, area, delay, power etc.), the library contains Matlab, C and Verilog implementations for all components which allows the user not only to immediately use the components, but also to calculate the error for a new target error metric or obtain desired parameters by performing a synthesis for a fabrication technology different to the reported one. In addition to that

these circuits can be utilized as building blocks of more complex approximate arithmetic circuits (as shown e.g. in [3]).

There are 430 different approximate 8-bit adders and 471 different approximate 8-bit multipliers forming a Pareto front in a three-dimensional space defined using the error, delay, and power metrics. These implementations were obtained by a multi-objective Cartesian genetic programming (CGP) according to [4]. As a starting point for the CGP-based approximation process, 13 different adder and 6 different multiplier accurately-operating architectures were employed. The adders include Ripple-Carry Adder (RCA), Carry-Select Adder (CSA), Carry-Look-ahead Adder (CLA), multiple Tree Adder (TA) and Higher Valency Tree Adder (HVTA) architectures. The multipliers include Ripple-Carry Array, multiple Carry-Save Array and Wallace Tree architectures. These accurate implementations are also included in the library.

CGP has been adopted as our design method because papers [5], [6], [7] revealed that it can provide much better implementations of accurate as well as approximate circuits than common circuit design and optimization tools. The EvoApprox8b library is provided as an open source project, see the EvoApprox8b web site [8].

The rest of the paper is organized as follows. Section II surveys the methods developed for approximation of adders and multipliers. Section III describes the approximation method used to create the EvoApprox8b library. The library is presented in Section IV. Conclusions are given in Section V.

II. Approximate adders and multipliers

Software-oriented benchmark sets such as AxBench [9] were developed for evaluation of approximate software and corresponding approximation methods. Regarding circuit approximation, several approximate circuits can be downloaded from KIT CES web site, including Generic Accuracy Configurable Adders (GeAR) [2]. The remaining papers dealing with circuit approximation methods suffer from problems discussed in Section I and, hence, performing a fair comparison of approximation methods or resulting approximate circuits is difficult. In this section, we will briefly survey approaches developed for adders and multipliers approximation.

Adders and multipliers are approximated by either general-purpose approximation methods or problem-specific methods. In the case of general-purpose methods (e.g. SALSA [10] and SASIMI [11]), adders and multipliers serve as one of many circuit classes that can be approximated. Problem-specific methods exploit the structure of conventional adders and multipliers. Another class of circuits are quality configurable circuits (e.g. GeAR adders) which allow for a dynamic approximation according the expected quality of result [2].

Four types of approximate adders are considered in [12]: (1) Speculative adders in which it is presumed that the probability that the carry propagation chain is longer than k bits ($k << n$) is very low for n-bit adders. Hence, according to k bits the carry is speculated for each sum bit. (2) Segmented adders, where an n-bit adder is divided into k-bit sub-adders and the carry is then generated by using different methods. (3) Carry-select adders in which multiple sub-modules are used to compute the sum for different carry values, and the result is determined according to the carry of a sub-module. (4) Approximate full adders where the full adder (an elementary adder's component) is approximated.

Approximate multipliers are based on the following principles [13]: (1) Approximation in generating partial products utilizing a simpler structure to generate partial products [3]. (2) Approximation in the partial product tree by ignoring some partial products or dividing partial products into several modules and applying approximation in the less significant modules. (3) Using approximate counters or compressors in the partial product tree. (4) Using approximate Booth multipliers. (5) Composing complex approximate multipliers by means of simple approximate multipliers as shown in [3].

Recently, an evolutionary approach was applied in the task of approximate circuits design with respect to multiple objectives [6], [7], [4]. Conventional circuits were used as an initial population. The circuit approximation problem is formulated as a multi-objective search problem and solved using the state-of-the art CGP method [14] combined with the NSGA-II algorithm [15]. In many cases, CGP-based approach is capable of optimizing accurate circuits in such a way that they remain accurate, but they show better parameters (e.g. area) than approximate circuits [4].

There are many error metrics developed to evaluate the quality of approximate arithmetic circuits [16]. While most methods apply test vectors to estimate the error (e.g. 10^8 test vectors were used to evaluate 16 bit adders in [12]), the exact error calculation based on formal models such as binary decision diagrams is rarely used. The accuracy of simulation-based error calculation (which depends on the number and quality of test vectors) can significantly influence the whole approximation process.

III. Circuit Approximation Method

The method used to obtain the library follows the methodology introduced in [4]. It is a general-purpose approximation method for combinational circuits based on a multi-objective CGP. It represents candidate circuits as directed acyclic graphs and tries to simultaneously minimize delay, power consumption and error to discover a set of approximate circuits along a Pareto front. Moreover, various constraints can be formulated to reduce the search space.

A. Circuit representation

In CGP, candidate solutions are represented in a two-dimensional array of programmable nodes [14]. An n_i-input and n_o-output combinational circuit is modelled using an array of $n_c \cdot n_r$ programmable nodes forming a Cartesian grid. A set of available n_a-input/n_b-output node functions is denoted Γ. The primary inputs and programmable node outputs are uniquely numbered. For each node the chromosome contains (n_a+1) values that represent the node function and n_a addresses specifying the input connections. The chromosome also contains n_o values specifying the gates (node outputs)

connected to the primary outputs. The chromosome size is $n_c n_r (n_a + 1) + n_o$ integers.

In our case, Γ contains functions implemented by standard components (such as gates and adders) of technology library. We used a subset of componets from a TSMC 180 nm technology library. A complete list of technology components including their area and leakage power can be found in [4]. There are multiple implementations of the same component differing in the implementation cost. During the circuit approximation, a proper size was selected for each gate depending on the output load of the gate.

B. Search method

New candidate circuits are created by means of a point mutation operator which modifies a given number of integers of the chromosome. The multi-objective search is conducted by the NSGA-II algorithm which is based on the idea of *Pareto dominance relation*. The individuals in each generation are sorted according to the dominance relation into multiple fronts. The solutions within the individual fronts are sorted according to the *crowding distance* metric, which helps to preserve a reasonable diversity along the fronts [15]. The crowding distance is the average distance of two solutions on either side along each of the objectives. A new population is then created by exploiting appropriate Pareto fronts as defined in [17]. The result of a single evolutionary run is not just one solution, but a set of non-dominated solutions occupying the Pareto front. The search method is implemented as a parallel evolutionary algorithm operating with multiple populations distributed on several islands and the best individuals are allowed to migrate among the islands.

C. Fitness functions

A selection of the error metric significantly influences obtained results [6]. As the error metric is usually an application-specific function there is no reason to prefer one over another in our library. We decided to optimize according to the mean relative error, but we also quantified the errors according to other commonly used error metrics for all evolved circuits (Section IV). The mean relative error is calculated accurately, i.e. for all possible 2^{n_i} input vectors, as:

$$f_{\mathrm{MRE}} := \frac{\sum_{\forall i} \frac{\left|O^{(i)}_{\mathrm{orig}} - O^{(i)}_{\mathrm{approx}}\right|}{\max(1, O^{(i)}_{\mathrm{orig}})}}{2^{n_i}}, \qquad (1)$$

where $O^{(i)}_{\mathrm{orig}}$ is the decimal representation of the i-th circuit correct output, $O^{(i)}_{\mathrm{approx}}$ is the individual's i-th output, and n_i is the number of primary inputs (i.e. the operand's width is $n_i/2$ bits). In addition to that, we constrained the worst absolute and relative errors to reduce the search space.

The circuit area is a sum of the areas of components used in the circuit. Power consumption is estimated according to the algorithm given in [4]. The delay of a candidate circuit is calculated using the parameters defined in the liberty timing file available for the utilized semiconductor technology. The delay t_{d} of a cell c_i is modeled as a function of its input transition time t_{s} and capacitive load C_{l} on the output of the cell, i.e. $t_{\mathrm{d}}(c_i) = f(t_{\mathrm{s}}^{c_i}, C_{\mathrm{l}}^{c_i})$. The delay of the circuit C is determined as the delay along the longest path.

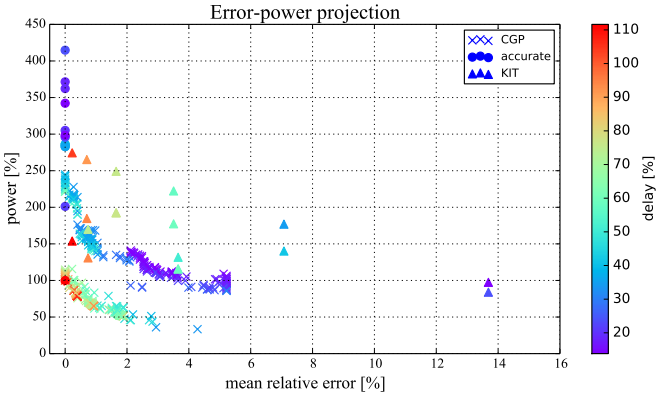

Fig. 1. Pareto fronts with parameters of evolved approximate 8-bit adders, conventional accurate adders and approximate adders according to KIT's paper [2].

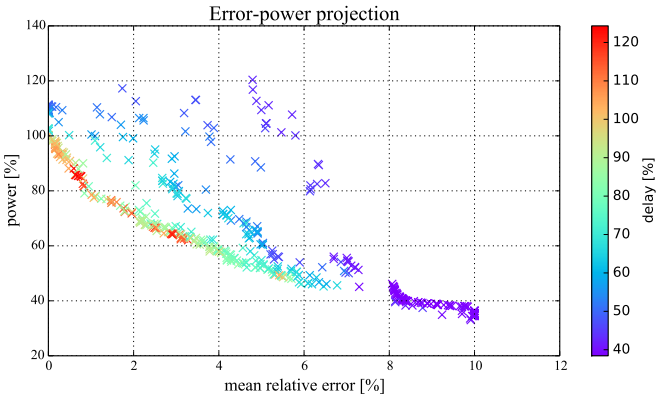

Fig. 2. Pareto fronts with parameters of evolved approximate 8-bit multipliers.

D. Parameters setting

The CGP/NSGA-II parameters were set as follows: 500 individuals in the population, 100,000 generations, 10 islands, mutation rate 5 %, number of rows $n_{\mathrm{r}} = 1$. The number of columns was $n_{\mathrm{c}} = 200$ in the case of the adders and $n_{\mathrm{c}} = 1000$ in the case of the multipliers.

The circuits were designed with respect to three objectives – the mean relative error (MRE), power consumption and delay. The MRE was constrained to be at most 10 %, the worst case error was constrained to be at most 5 % of the output range and the worst case relative error was limited to 1000 %. All candidate solutions violating these requirements were discarded.

IV. EvoApprox8b Library

The EvoApprox8b library contains 430 non-dominated 8-bit approximate adders evolved from the initial population of 13 conventional adders and 471 non-dominated 8-bit approximate multipliers that were evolved from 6 conventional implementations. In addition to the conventional implementations, the

library also contains 43 exact adders and 28 exact multipliers that were obtained by CGP and that are not dominated by any accurate implementation. All Pareto fronts are shown in Figure 1 and 2. All parameters are related to the Ripple-Carry Adder and Ripple-Carry Array Multiplier architectures (considered as 100% in the figures). Figure 1 also provides parameters of 8-bit approximate adders created according to [2]. Evolved adders show quite competitive properties under the metrics used in the figure.

All approximate circuits and their various parameters can be found at the EvoApprox8b web site [8]. It contains circuit models for Verilog, Matlab and C. This enables the user to integrate the circuits to hardware as well as software projects and design tools. All circuits can thus be simulated in order to obtain their other parameters that are not listed on the web site (e.g. errors under different error metrics or power consumption for another fabrication technology). The circuits can be sorted according to a chosen parameter which is useful when the user is looking for a circuit satisfying particular constraints.

The following list gives parameters that are provided for all circuits in the library: Area, delay, power (all estimated according to [4] which was validated against Cadence Encounter RTL Compiler and TSMC 180 nm library), nodes (the number of nodes, where a gate, a half adder and a full adder are counted as one node), HD (Hamming distance), EP (error probability), MAE (mean absolute error), MSE (mean squared error), MRE (mean relative error), WCE (worst case error), and WCRE (worst case relative error). Note that as n_i is the number of primary inputs, the operand's width is $n_i/2$ bits.

$$\mathrm{HD} = \sum_{\forall i} OnesCount(O^{(i)}_{\mathrm{approx}} \oplus O^{(i)}_{\mathrm{orig}}), \tag{2}$$

$$\mathrm{EP} = \frac{\sum_{\forall i: O^{(i)}_{\mathrm{approx}} \neq O^{(i)}_{\mathrm{orig}}} 1}{2^{n_\mathrm{i}}}, \tag{3}$$

$$\mathrm{MAE} = \frac{\sum_{\forall i} \left| O^{(i)}_{\mathrm{approx}} - O^{(i)}_{\mathrm{orig}} \right|}{2^{n_\mathrm{i}}}, \tag{4}$$

$$\mathrm{MSE} = \frac{\sum_{\forall i} \left| O^{(i)}_{\mathrm{approx}} - O^{(i)}_{\mathrm{orig}} \right|^2}{2^{n_\mathrm{i}}}, \tag{5}$$

$$\mathrm{MRE} = \frac{\sum_{\forall i} \frac{\left| O^{(i)}_{\mathrm{approx}} - O^{(i)}_{\mathrm{orig}} \right|}{\max(1, O^{(i)}_{\mathrm{orig}})}}{2^{n_\mathrm{i}}}, \tag{6}$$

$$\mathrm{WCE} = \max_{\forall i} \left| O^{(i)}_{\mathrm{approx}} - O^{(i)}_{\mathrm{orig}} \right|, \tag{7}$$

$$\mathrm{WCRE} = \max_{\forall i} \frac{\left| O^{(i)}_{\mathrm{approx}} - O^{(i)}_{\mathrm{orig}} \right|}{\max(1, O^{(i)}_{\mathrm{orig}})}. \tag{8}$$

V. Conclusions

In this paper we presented a rich library of approximate 8 bit adders and multipliers which is primarily intended for creating approximate blocks of image and video processing circuits. The Matlab, C and Verilog models that are available from [8] can allow software as well as hardware developers to integrate the approximate adders and multipliers to their designs. As we are aware of, this is the first open-source library containing hundreds of approximate components that allows for reproducible comparisons across various layers of design abstraction. These components can be utilized as building blocks of more complex approximate adders and multipliers as demonstrated for multipliers in [3].

Acknowledgments

This work was supported by Czech science foundation project GA16-17538S and the Ministry of Education, Youth and Sports of the Czech Republic from the National Programme of Sustainability (NPU II) project IT4Innovations excellence in science – LQ1602.

References

[1] S. Mittal, "A survey of techniques for approximate computing," *ACM Comput. Surv.*, vol. 48, no. 4, p. 62:162:33, 2016.

[2] M. Shafique, W. Ahmad, R. Hafiz, and J. Henkel, "A low latency generic accuracy configurable adder," in *Proceedings of the 52nd Annual Design Automation Conference*. ACM, 2015, pp. 86:1–86:6.

[3] P. Kulkarni, P. Gupta, and M. D. Ercegovac, "Trading accuracy for power in a multiplier architecture," *J. Low Power Electronics*, vol. 7, no. 4, pp. 490–501, 2011.

[4] R. Hrbacek, V. Mrazek, and Z. Vasicek, "Automatic design of approximate circuits by means of multi-objective evolutionary algorithms," in *Proc.of the 11th Int. Conf. on Design and Technology of Integrated Systems in Nanoscale Era*. IEEE, 2016, pp. 239–244.

[5] Z. Vasicek and L. Sekanina, "A global postsynthesis optimization method for combinational circuits," in *Proc. of the Design, Automation and Test in Europe, DATE*. EDA Consortium, 2011, pp. 1525–1528.

[6] ——, "Evolutionary design of approximate multipliers under different error metrics," in *IEEE International Symposium on Design and Diagnostics of Electronic Circuits and Systems*. IEEE, 2014, pp. 135–140.

[7] ——, "Evolutionary approach to approximate digital circuits design," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 3, pp. 432–444, 2015.

[8] V. Mrazek, R. Hrbacek, Z. Vasicek, and L. Sekanina. (2016) Approximate 8-bit Adders and Multipliers. [Online]. Available: www.fit.vutbr.cz/research/groups/ehw/approxlib

[9] A. Yazdanbakhsh, D. Mahajan, P. Lotfi-Kamran, and H. Esmaeilzadeh, "Axbench: A multi-platform benchmark suite for approximate computing," *IEEE Design and Test*, 2016.

[10] S. Venkataramani, A. Sabne, V. J. Kozhikkottu, K. Roy, and A. Raghunathan, "SALSA: systematic logic synthesis of approximate circuits," in *The 49th Annual Design Automation Conference 2012, DAC'12*. ACM, 2012, pp. 796–801.

[11] S. Venkataramani, K. Roy, and A. Raghunathan, "Substitute-and-simplify: a unified design paradigm for approximate and quality configurable circuits," in *Design, Automation and Test in Europe, DATE'13*. EDA Consortium, 2013, pp. 1367–1372.

[12] H. Jiang, J. Han, and F. Lombardi, "A comparative review and evaluation of approximate adders," in *Proceedings of the 25th Edition on Great Lakes Symposium on VLSI*. ACM, 2015, pp. 343–348.

[13] H. Jiang, C. Liu, N. Maheshwari, F. Lombardi, and J. Han, "A comparative evaluation of approximate multipliers," in *IEEE/ACM International Symposium on Nanoscale Architectures*. IEEE, 2016, pp. 191–196.

[14] J. F. Miller, *Cartesian Genetic Programming*. Springer-Verlag, 2011.

[15] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[16] J. Liang, J. Han, and F. Lombardi, "New metrics for the reliability of approximate and probabilistic adders," *IEEE Trans. Computers*, vol. 62, no. 9, pp. 1760–1771, 2013.

[17] R. Hrbacek, "Parallel multi-objective evolutionary design of approximate circuits," in *GECCO '15 Proceedings of the 2015 conference on Genetic and evolutionary computation*. ACM, 2015, pp. 687–694.

Paper III

Design of Power-Efficient Approximate Multipliers for Approximate Artificial Neural Networks

MRÁZEK Vojtěch, SARWAR Syed Shakib, SEKANINA Lukáš, VAŠÍČEK Zdeněk and ROY Kaushik

In: *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design.* Austin, TX: Association for Computing Machinery, 2016, pp. 811-817. ISBN 978-1-4503-4466-1.

Design of Power-Efficient Approximate Multipliers for Approximate Artificial Neural Networks

Vojtech Mrazek[1]
imrazek@fit.vutbr.cz

Syed Shakib Sarwar[2]
sarwar@purdue.edu

Lukas Sekanina[1]
sekanina@fit.vutbr.cz

Zdenek Vasicek[1]
vasicek@fit.vutbr.cz

Kaushik Roy[2]
kaushik@purdue.edu

[1]Faculty of Information Technology, Centre of Excellence IT4Innovations
Brno University of Technology
Brno, Czech Republic

[2]School of Electrical and Computer Engineering
Purdue University
West Lafayette, IN, USA

ABSTRACT

Artificial neural networks (NN) have shown a significant promise in difficult tasks like image classification or speech recognition. Even well-optimized hardware implementations of digital NNs show significant power consumption. It is mainly due to non-uniform pipeline structures and inherent redundancy of numerous arithmetic operations that have to be performed to produce each single output vector. This paper provides a methodology for the design of well-optimized power-efficient NNs with a uniform structure suitable for hardware implementation. An error resilience analysis was performed in order to determine key constraints for the design of approximate multipliers that are employed in the resulting structure of NN. By means of a search based approximation method, approximate multipliers showing desired tradeoffs between the accuracy and implementation cost were created. Resulting approximate NNs, containing the approximate multipliers, were evaluated using standard benchmarks (MNIST dataset) and a real-world classification problem of Street-View House Numbers. Significant improvement in power efficiency was obtained in both cases with respect to regular NNs. In some cases, 91% power reduction of multiplication led to classification accuracy degradation of less than 2.80%. Moreover, the paper showed the capability of the back propagation learning algorithm to adapt with NNs containing the approximate multipliers.

Categories and Subject Descriptors

B.6.3 [**Hardware**]: Logic Design—*Automatic synthesis*; I.2.6 [**Computing Methodologies**]: Artificial Intelligence

ICCAD '16, November 07-10, 2016, Austin, TX, USA

DOI: http://dx.doi.org/10.1145/2966986.2967021

1. INTRODUCTION

Recent advances in artificial intelligence methods and a huge amount of computing resources available on a single chip have led to a renewed interest in efficient implementations of complex neuromorphic systems based on artificial neural networks (NNs). Implementing complex NNs in low power embedded systems requires careful optimization strategies at various levels including neurons, interconnects, learning algorithms, data storage and memory access. This work is focused on reducing power consumption of computations performed in neurons, which is of the same importance as optimizing the data storage and memory access [7].

Inexact or approximate computing has been adopted in recent years as a viable approach to reduce power consumption and improve the overall efficiency of computers. In approximate computing, circuits are not implemented exactly according to the specification, but they are simplified in order to reduce power consumption or increase operation frequency. It is assumed that the errors occurring in simplified circuits are acceptable, which is typical for error resilient application domains such as multimedia, classification and data mining. Applications based on NNs have proven to be highly error resilient [2].

This paper provides a methodology for the design of well-optimized power-efficient NNs that have a uniform structure (i.e. all nodes are identical in all layers) which is thus suitable for hardware implementation. An error resilience analysis is performed in order to determine key constraints for the design of approximate multipliers that are employed in the resulting structure of NN. In order to avoid a manual approximation of accurate multipliers, systematic methods capable of performing approximations have been introduced recently [21, 20, 14]. These methods typically start with a gate-level description of the accurate circuit and an error constraint that specifies the type of error that can be accepted. The approximation algorithm is typically constructed as a design space exploration algorithm directly approximating some parts of the circuit [11] or the whole circuit [18]. The search is guided by an error metric such as the average error magnitude or maximum arithmetic error.

In addition to developing highly-optimized power efficient

NNs, an automated design space exploration method is proposed. The method is capable to design approximate multipliers in such a way that the resulting multipliers satisfy not only a given error, but also a set of other application-specific constraints.

2. ARTIFICIAL NEURAL NETWORKS

In machine learning, artificial neural networks are a family of models inspired by biological neural networks. A typical artificial neural network consists of an input layer of neurons, several hidden layers of neurons and an output layer of neurons.

2.1 Artificial Neuron

A typical structure of neuron is as follows [4, 22]. The output h_i of neuron i is defined as $h_i = \sigma(\sum_{j=1}^{N} w_{ij}x_j - \theta)$, where $\sigma(\cdot)$ is an activation function, N is the number of inputs of the neuron, w_{ij} is weight of the link, x_j is the j-th input and θ is a threshold or bias. The purpose of the activation function is (in addition to introducing non-linearity into NN) to map the resulting values into the interval $(-1, 1)$ or $(0, 1)$. The activation can be a threshold function, semi-linear or non-linear function. A common example of the non-linear function, which is used in this work, is sigmoid function $\sigma(x) = (1 + e^{-x})^{-1}$.

2.2 Architecture and learning

The NNs are classified into feed-forward neural networks (FNNs), recurrent neural networks (RNN) and their combination. In RNNs, there is at least one feedback connection. The earliest and the simplest architecture is the perception model which utilizes just one layer of output neurons that are connected with all the inputs. The extended version, the multilayer perception model (MLP), uses one or more layers (a.k.a. hidden layers) of neurons between the input and output layers. In the hidden layer, each neuron is directly linked to the outputs of the previous layer. An important contribution to the state of the art in NNs has been the development of large-scale NNs such as the convolutional NNs introduced by LeCun [9], where more types of layers (e.g. convolutional layers) are employed. Another type of layers is the average pooling layer which is used for weighted subsampling. Nowadays there are many different application-specific layers intended for, e.g., image classification [8], segmentation [1], speech processing [6] etc.

Learning is the most important capability of neural networks. It is performed by an algorithm that iteratively updates the synapses (weights) and other parameters of neural network. Determining the most suitable parameters and weights of NN can be viewed as a complex nonlinear optimization problem. Learning methods are usually divided into supervised, unsupervised, reinforcement, and evolutionary methods [4]. The most popular algorithms for supervised learning, which we will employ, are the least mean squares method and back propagation algorithm [4].

2.3 Approximations in NNs

As neural networks are inherently error-resilient, various approaches have been proposed to approximate them [13].

Venkataramani et al. [19] proposed a methodology of identifying error-resilient neurons based on the backpropagation gradients. For the error-resilient neurons, an approximation using precision modification and piecewise-linear approximation of activation function was applied to create an approximate neural network. Since training is by itself an error-healing process, after creating the approximate version, the NN is retrained. They also proposed a neuromorphic processing engine platform to determine the best tradeoff between the precision and energy.

Zhang et al. [24] used a different approach for critical neuron identification. A neuron is considered as critical, if small jitters on the neuron's computation introduce large output quality degradation; otherwise, the neuron is resilient. They presented a theoretical approach for finding the critical neurons. The least critical neurons are candidates for approximation. Due to the tight interconnection between the neurons, the ranking of candidate neurons is updated after approximation of each neuron. Hence, an iterative algorithm for the criticality ranking and approximation was developed. Three approximation strategies were used – precision scaling, memory access skipping and approximating the multiplier circuits.

Du et al. [5] proposed an inexact Neural Network accelerator showing that it is possible to use inexact multipliers in NNs. The multipliers were designed using an inexact logic minimization algorithm [11]. For small fully connected neural networks, their strategies were able to find good configurations. They exploited the fact that the output layer has a small number of neurons and since there is no synaptic weight after these neurons, lowering the errors through retraining is difficult [13].

Judd et al. [7] showed that computations and memory accesses significantly contribute to power consumption. Hence they used bit-precise weights reduction in standard multiplication and reduced memory accesses of standard memories in their implementation for GPUs and ASIC.

Power consumption of the synaptic weight memory was optimized by Srinivasan et al. [17] who applied a conventional 6T SRAM that is known to be susceptible to bit-cell failures due to voltage over-scaling. A significance driven hybrid 8T-6T SRAM was proposed wherein the sensitive MSBs are stored in robust 8T bit-cells. The memory access power reduction was exchanged for a small loss in the classification accuracy.

Sarwar et al. [16] introduced approximate multipliers based on alphabet-set multiplication. The weights were divided into parts having 4 bits. Multiplication by each 4-bit part of the weight was implemented by shifting a precomputed input value and followed by summation. Authors showed that reducing the set of precomputed values has a significant impact on power consumption and a small impact on the total accuracy of neural network. This architecture is suitable for an efficient hardware implementation because the resulting NN shows a uniform structure and each neuron has the same architecture.

In summary, the first four approaches presented in this section have shown that it is possible to approximate some neurons. The resulting NNs can be characterized as non-uniform NNs. However, for an efficient VLSI implementation and for implementing a general-purpose NN (not an application specific one), all (or almost all) neurons have to be uniform. Moreover, the selected components were approximated manually and independently of a target NN. It was also shown that not only multiplication but also the

memory access has a significant impact on the total power consumption.

2.4 Approximate multipliers in NNs

Since NN contains hundreds of thousands multiplications, it seems to be useful to introduce approximate multipliers to reduce power consumption. In order to determine the impact of inexact multiplication on NNs' accuracy, the following sensitivity analysis has been carried out.

A non-trivial MLP network (1 hidden layer, 100 hidden neurons) trained for recognizing handwritten numbers of MNIST dataset (described in Section 4.2.1) was chosen as our benchmark problem and evaluated using *DeepLearnToolbox*.[1]. Its accurate implementation shows the classification accuracy 94.16% when precise 8-bit multipliers are used.

To emulate imprecise multiplication, a jitter function $\Delta : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ is introduced. Let the output of inexact multiplier m be defined as $m(a,b) = a \cdot b + \Delta(a,b)$. To ensure that the relative worst-case error of 8-bit multiplier m is 5.2%, the range of the jitter function Δ is bounded by ± 852, calculated as $5.2\% \cdot 2^{2 \cdot 7}$. Note that this worst-case error was chosen according to approximate multipliers proposed in [18].

When function m is used instead of accurate multiplication and no retraining is applied, the classification accuracy of the network decreased to 10.77%. A detailed analysis revealed that there are more than 80% cases where one of the input operands of multiplication is zero. The random jitter then provides a non-zero output value and this error is accumulated. Hence we hypothesized that the multiplication must be accurate if at least one of the operands is zero.

To investigate this hypothesis, we re-defined the approximate multiplier m to m', where:

$$m'(a,b) = \begin{cases} m(a,b) & \text{if } a \cdot b \neq 0 \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

Now the original NN which employs approximate multipliers m' exhibits the classification accuracy of 94.20%. Although the impact of approximate multipliers on the accuracy is application-specific, this benchmark showed that it is necessary to have the accurate multiplication by 0. Figure 1 shows the absolute difference between outputs of the same neurons in the case that approximate multipliers provide (a) inexact and (b) exact multiplication by 0.

3. PROPOSED DESIGN METHOD

The proposed method is based on uniform NNs that utilize approximate multipliers. In this section, we will define feasible approximate multipliers, describe a design space exploration search method for obtaining the feasible multipliers, and introduce the overall methodology for NN approximation.

3.1 Constraints and cost function

A digital combinational circuit with n inputs and m outputs computes a completely-specified Boolean function $\mathcal{F} : B^n \to B^m$, $B = \{0,1\}$, that maps n-input Boolean vector $x = \langle x_1, x_2, \ldots x_n \rangle$ to an m-output Boolean vector $y = \langle y_1, y_2, \ldots y_m \rangle$ with associated hardware cost. Let n-bit accurate multiplier be represented by a function $\mathcal{M} : B^n \times B^n \to B^{n+n}$ and let $\delta : B^m \to \mathbb{N}$ assign a natural number to an m-bit Boolean vector.

[1] https://github.com/rasmusbergpalm/DeepLearnToolbox

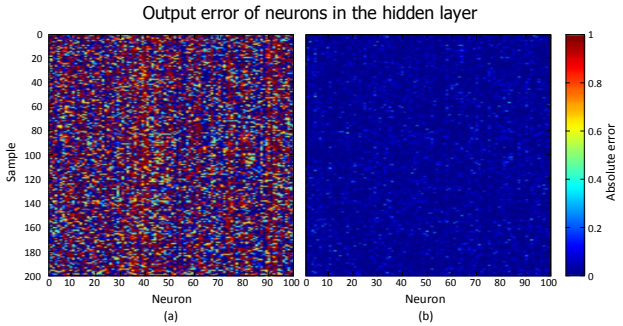

Figure 1: The error of the output neurons in the approximate NN in comparison with the original NN. The approximate NN utilizes approximate multipliers showing a 5.2% error and (a) inaccurate and (b) accurate multiplication by zero

The error metric is defined as maximal relative error ε, i.e. it is requested that the maximal arithmetic error of multiplication for each combination of operands is lower than ε on the whole output range (which is $0 \ldots (2^{2n}-1)$). This error ε will be one of the input parameters of the algorithm designing approximate multiplies.

A candidate approximate multiplier $\mathcal{M}'$ is a *feasible* solution is two conditions hold. (i) The error is acceptable:

$$\forall_{(a,b) \in B^n \times B^n} : |\delta(\mathcal{M}(a,b)) - \delta(\mathcal{M}'(a,b))| \leq \varepsilon \cdot (2^{2n}-1). \quad (2)$$

and (ii) multiplication by 0 is accurate:

$$\forall_{a \in B^n} : \quad \mathcal{M}(a,\{0\}^n) = \mathcal{M}'(a,\{0\}^n) \quad \wedge \quad \mathcal{M}(\{0\}^n,a) = \mathcal{M}'(\{0\}^n,a). \quad (3)$$

In the approximation process, the implementation cost of multiplier $S_{\mathcal{M}'}$ will be estimated as the number of used gates. The number of two-input gates is a sufficient metric because the circuits are relatively simple (as it will be seen in Section 5). The number of used gates $S_{\mathcal{M}'}$ is determined recursively as follows: (1) the gate is used if its output is connected to output of the circuit; (2) the gate is connected if its output is connected to an input of any used gate.

The cost function for the approximation process is defined as

$$\mathbf{C}_{\mathcal{M}'} = \begin{cases} S_{\mathcal{M}'} & \text{if constraints (2) and (3) are met} \\ \infty & \text{otherwise} \end{cases}. \quad (4)$$

3.2 Approximate multiplier design

In order to approximate an accurate multiplier, various approaches have been proposed. In this work, we employ Cartesian Genetic Programming (CGP) [12] because it can easily handle constraints given on candidate circuits, the method is naturally multi-objective and high-quality approximate circuits have already been obtained with it [18].

The standard CGP is a branch of genetic programming which represents candidate designs using directed acyclic graphs [12]. A candidate circuit is modeled using a 2D array of programmable nodes with n_c columns and n_r rows. In our case, the nodes will be 2-input Boolean functions, where Γ is the set of available functions. The circuit utilizes n_i primary

inputs and n_o primary outputs. Feedback connections are not enabled.

The primary inputs and the outputs of the nodes are labeled $0, 1 \ldots n_c \cdot n_r + n_i - 1$ and considered as addresses which the node inputs can be connected to. A candidate solution is represented in the so-called chromosome (which is, in fact, a netlist) by $n_r \cdot n_c$ triplets (x_1, x_2, ψ) determining for each node its function ψ ($\psi \in \Gamma$) and input connections. The last part of the chromosome contains n_o integers specifying the nodes where the primary outputs are connected to. While the chromosome size s is constant $s = n_c n_r (n_a + 1) + n_o$, the circuit size is variable and measured as the number of active (i.e. used) nodes. See an example in Fig. 2. The set of valid chromosomes (netlists) represents the whole search space.

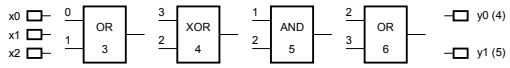

Figure 2: Example of a circuit in CGP with parameters: $n_i = 3$, $n_o = 2$, $n_c = 4$, $n_r = 1$, $\Gamma = \{0^{and}, 1^{or}, 2^{xor}\}$. Chromosome: 0, 1, 1; 3, 2, 2; 1, 2, 0; 2, 3, 1; 4, 5. Gate 6 is not used. Logic behavior of the circuit is:
$y_0 = ((x_0 \text{ or } x_1) \text{ xor } x_2)$; $y_1 = x_1$ and x_2.

CGP employs a simple search method . In our case, the initial population P of CGP contains one of various implementations of accurate multipliers and a few circuits generated using mutation of the accurate multiplier. Creating the accurate multiplier in the initial population is trivial as there is a one-to-one mapping between multiplier netlists and CGP chromosomes. The next step consists in the evaluation of candidate circuits using the fitness function. Each member of P then receives the so-called fitness score and the highest-scored individual becomes a new parent of the next population. From this parent, λ candidate solutions are generated using mutation. The termination criterion is given by the number of iterations.

Despite many attempts to propose a suitable crossover operator to CGP, the mutation is still used as the crucial genetic operator. The mutation operator modifies up to h randomly chosen genes (integers) of the chromosome. Their new values are generated randomly, but it is checked whether the new values are valid. One mutation can affect either the gate function, gate input connection, or primary output connection.

In order to approximate multipliers, the fitness is defined as $fitness(\mathcal{M}') = -\mathbb{C}_{\mathcal{M}'}$ and Γ = {NAND, NOR, XNOR, AND,OR, XOR, NOT, identity}.

3.3 Evaluation platform

This section describes the evaluation platform used for simulations of the proposed approximate NNs. The software framework is based on C++ project *tiny-cnn*[2]. We have implemented two new types of layers to NNs: the approximate fully connected layer and approximate convolution layer. In the software simulation, the approximate multiplication was realized using a lookup table. The framework uses weights and inputs with double floating point precision. We rounded them to the fixed point representation in the range $\langle -1, 1 \rangle$. All numbers are unsigned, the sign is determined after the computation.

[2]https://github.com/nyanp/tiny-cnn

We have also synthetised multipliers for neural network. The multipliers were implemented at the Register-Transfer Level (RTL) in Verilog and mapped to the IBM 45nm technology using Synopsys Design Compiler Ultra. The hardware multiplication unit utilizes a combinational approximate unsigned multiplier circuit and logic for the sign extension. We have utilized the one's complement method which is easy to calculate ($4n$ XOR gates), but provides lower accuracy w.r.t. the two's complement method (extra three one-subtractors) used in standard applications. The framework can estimate energy consumption and area under iso-speed conditions. The clock frequency for 8 bit neurons is 3 GHz and 2.5 GHz for 12 bit neurons.

There are equal count of multiplications and additions and one activation function in the neuron computational model. Since the count of operations is big (tens or hundreds) and the multiplication consumes significantly more energy than addition, the multiplication is the most consuming part and power reduction of this part significantly contributes to the overall power consumption reduction.

3.4 Overall design methodology

Finally, the overall methodology for design of approximate multipliers that will be used in approximate NNs is presented in Figure 3. The inputs to the methodology are the accurate neural network (with accuracy J), training and testing data, quality constraint Q, accurate multiplier and initial error ε. The whole procedure is as follows. The CGP algorithm is utilized for creating a set of approximate multipliers from the accurate one. The approximate multipliers are used in the pretrained network. In order to achieve the best quality results, the network is retrained. The NN implementation showing the best accuracy K is selected. The accuracy K is checked if it meets the quality constraint Q w.r.t. accurate neural network with accuracy J. If the constraint is not met, the relative maximal approximation error ε is decreased and next iteration is performed. Due to non-deterministic generation of approximate multipliers by CGP, it is necessary to generate several approximate multipliers and then re-evaluate the accuracy of NN.

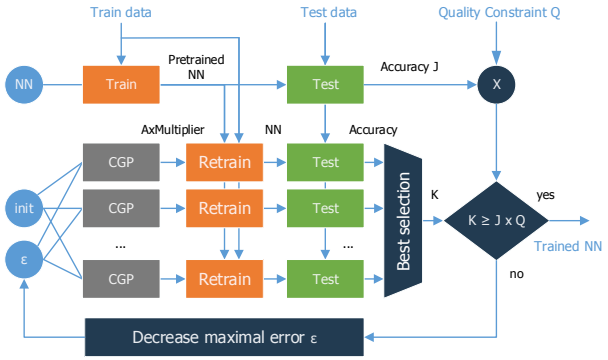

Figure 3: Overview of approximate multiplier design for approximate NNs

4. EXPERIMENTAL SETUP

The goal of the experiments is to investigate the impact of proposed approximation methods on the accuracy and power

consumption of NNs. This section provides the experimental setup and benchmark problems description.

4.1 CGP configuration

CGP will be used to design 7 bit and 11 bit unsigned multipliers. The sign extension, i.e. 8 bit and 12 bit-width multipliers, will be designed manually using the one's complement method. The maximum target arithmetic error ε of approximate multipliers is taken from the set {0.5%, 1%, 2%, 5%, 10%, 15%, 20%}. We did not employ arithmetic error beyond 20% for approximate multipliers since the classification accuracy drops significantly. The approximation process starts with accurate multipliers (Ripple Carry Array, Carry Save Array with RCA and CSA adders, Wallace Tree with RCA, CSA and CLA adders [23]) which constitute the so-called initial population. Considering two bit widths, 7 target errors and 6 types of initial multiplier architectures, there are 84 initial configurations in total.

4.2 NN Accuracy analysis

The accurate multipliers are replaced with candidate approximate multipliers in the NN which is then retrained in the supervised learning scenario. Two types of NN and two classification datasets are utilized for the accuracy analysis.

4.2.1 Handwritten numbers

The first dataset is MNIST (Mixed National Institute of Standards and Technology) database of handwritten numbers [10] which consists of two sets of data. The first one is the training data set containing 60,000 28×28 images and their labels. The second one contains 10,000 test pairs. The digits are normalized and centered in fixed-sized images. The dataset is very popular for quantifying the accuracy of classification methods. It was shown that neural networks are able to provide the error rate as low as 0.27% using convolutional networks [3]. In this case, we used a MLP network with 28×28 input neurons, 300 neurons in the hidden layer and 10 output neurons whose outputs are interpreted as the probability of each of 10 target classes $(0 - 9)$.

4.2.2 House numbers

The second dataset is SVHN (Street View House Numbers) which is obtained from house numbers in Google Street View images [15]. The images come from a significantly harder, unsolved, real-world environment. The dataset contains 73,257 digits for training and 26,032 digits for testing. Each digit is represented as a pair of 32×32 RGB image and label. While MLP does not provide good accuracy in this case, LeNet-6 (a 6 layer NN in Figure 4 [9]) is able to classify the images with a very small error. The network consumes a 32×32 grayscale image as an input. In order to reduce the complexity, we transformed original RGB images to grayscale using an equation $Y = 0.299R + 0.587G + 0.114B$.

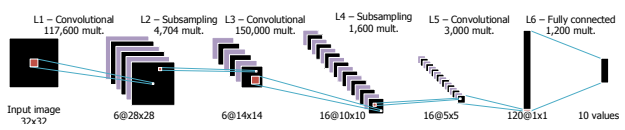

Figure 4: LeNet structure, where L1, L3, L5 and L6 contain approximate multipliers, i.e. 98 % of multiplications are approximated.

Layers L1, L3 and L5 perform the convolution. The L3 employs a special table that indicates which feature map from 6 previous maps is used for generating each of 16 output feature maps. Last layer (L6) connects all 120 values with each neuron of the output layer. Convolutional and fully connected layers represent 98 % of all multiplications performed in the network. Hence, the approximation was applied only for this layers. Layers L2 and L4 perform a subsampling by weighted average, but this process was not approximated because it has a small impact on power consumption.

5. RESULTS

The first part of this section is devoted to the results of the proposed CGP-based approximation of multipliers. The second part deals with approximate NNs. We also report detailed parameters of approximate multipliers.

5.1 Multiplier approximation with CGP

CGP is used with settings given in Section 3.2. Five circuits ($\lambda = 5$) are evaluated in each iteration and new circuits are created by modifying just 1 integer in the chromosome ($h = 1$) of the parent circuit. CGP operates with $n_c \times n_r = 900$ and $n_c \times n_r = 300$ (respectively) nodes for 11-bit and 7-bit multiplier (respectively). The evaluation of a candidate 11 bit approximate multiplier requires evaluation of 256x more test vectors than for the 7 bit multiplier (2^{22} vs. 2^{14}). Hence, the maximal time for CGP was set to 120 minutes for 11 bit multipliers and 30 minutes for 7 bit multipliers. CGP performed 1,343 (and 122,773) iterations on average for 11 (and 7) bit multiplier. The setting of CGP corresponds with typical values used in the literature [12, 18].

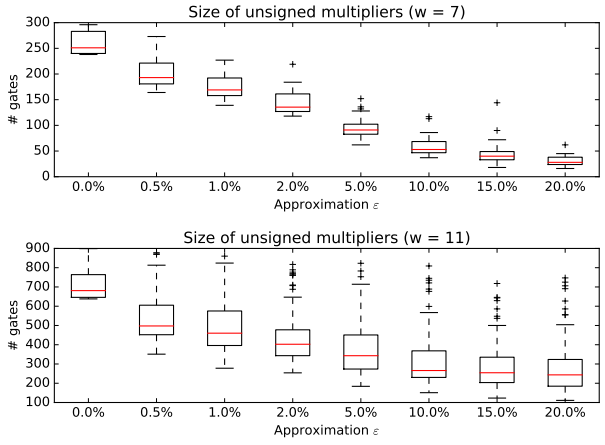

Figure 5: The number of gates in approximate multipliers

Figure 5 gives the number of gates in approximate multipliers as boxplots showing the results from 60 independent runs for a given error ε. If the error is zero only 6 values are presented which corresponds with gate counts in our accurate multipliers. In addition to obtaining many different tradeoffs between the error and the number of gates, the proposed method guarantees the exact multiplication by zero in all approximate multipliers. The spread in obtained gate counts is high especially for the 11-bit multipliers. Please

note that the approximate multipliers do not prolong delay of the original accurate multipliers.

5.2 Approximate NNs

For constructing the approximate NNs, each of designed approximate multipliers was utilized. In total, we thus obtained 2×852 NNs (LeNet6 and MLP with (28×28)-100-10 layers) using 852 approximate multipliers which were subsequently extended to signed versions using the one's complement. The accuracy of approximate NNs is presented in Figure 6 for a pretrained network (column *initial*) and then for 5 and 10 retrains, respectively, using the backpropagation algorithm. Each boxplot represents 60 multipliers, e.g. in MNIST $w = 8, \varepsilon = 15\%$, there is one multiplier leading to the accuracy 20% and another to 97% in the initial placement in pretrained neural network.

Because it is infeasible to estimate power consumption of each of 852 circuits, we did a precise power analysis in the following way. We have selected circuits for each error ε and bit-width w that have one of top three best accuracies for SVHN. Then we selected a circuits from the top three sets that provide the best tradeoff between MNIST accuracy and the number of used gates. The accuracy of NNs utilizing the selected approximate multipliers is shown in Figure 7. The accuracy is normalized w.r.t. a circuit with the same bit-width and $\varepsilon = 0\%$. We have followed the alphabet-reduced

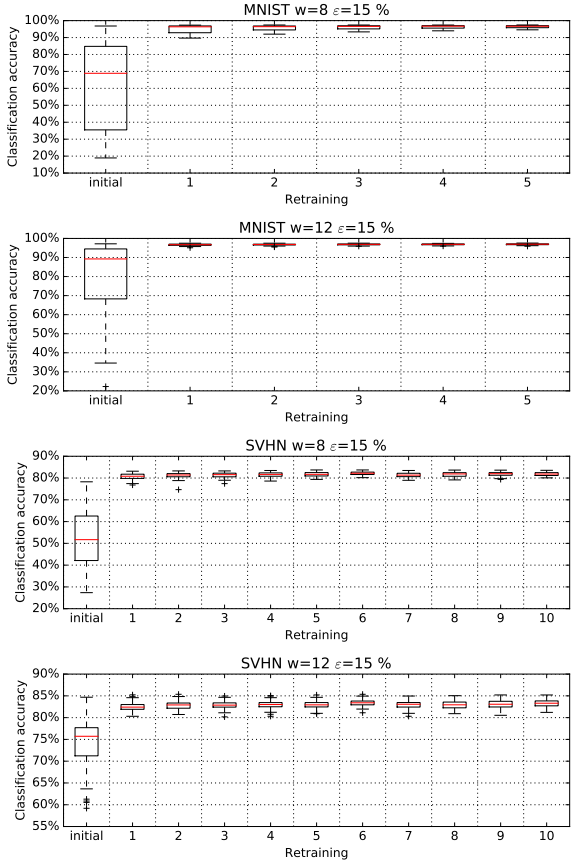

Figure 6: Accuracy of NNs for several configurations during the retraining process. The data shows statistical information for all designed multipliers with selected w and ε.

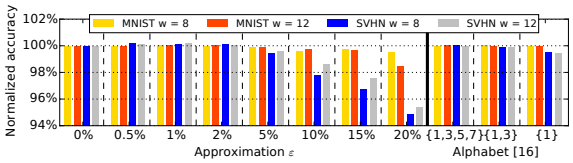

Figure 7: Normalized accuracy of NNs utilizing the best approximate multipliers developed by the proposed method for a given ε and its comparison with [16]. For each configuration, the accuracy is normalized w.r.t. NN employing accurate multipliers ($\varepsilon = 0$).

approach proposed in [16] and perform the simulation. The reduced alphabet $\{1\}$ enables to employ just 40 out of 256 weights for $w = 8$ and 200 out of 4,096 weights for $w = 12$. It can be seen that results from [16] are very similar for the proposed approach when $\varepsilon = 5\%$.

Error ε	**Power** μW	**Area** μm	**Accuracy SVHN**	**Accuracy MNIST**
0 %	250.0	440.0	87.00	97.67
0.5 %	201.0	367.7	87.15	97.66
1 %	175.0	316.6	87.08	97.68
2 %	107.0	218.3	87.07	97.65
5 %	58.6	129.9	86.54	97.58
10 %	45.2	109.5	85.11	97.31
15 %	22.3	63.2	84.20	97.42
20 %	22.9	65.8	82.52	97.22

(a)

Error ε	**Power** μW	**Area** μm	**Accuracy SVHN**	**Accuracy MNIST**
0 %	831.0	1175.0	87.04	97.70
0.5 %	417.0	664.9	87.15	97.69
1 %	475.0	720.8	87.22	97.71
2 %	284.0	523.8	87.06	97.71
5 %	247.0	483.0	86.68	97.61
10 %	125.0	285.0	85.81	97.48
15 %	115.0	262.4	84.95	97.38
20 %	111.0	252.5	83.06	96.18

(b)

Table 1: Power consumption and area of (a) 8-bit and (b) 12-bit sign-extended approximate multipliers and the absolute accuracy of NNs utilizing these multipliers.

Table 1 gives power consumption of selected approximate multipliers (in IBM 45nm process) and the accuracy of NNs that are utilizing these multipliers in two classification tasks. In comparison with the original NNs (which utilize the accurate multiplication), one can observe that approximate NN ($w = 8, \varepsilon = 10\%$) provides 81.9% power reduction of multiplication process while its accuracy decreases by 1.89% for SVHN and 0.36% for MNIST. If the error of multiplication remains below 20% the accuracy is only slightly decreased (in some cases it is even improved, but the improvement is negligible) for the MNIST problem. The NN trained for the SVHN dataset is more sensitive to approximations. The reason is that SVHN is a significantly harder classification problem than MNIST, because SVHN contains natural scene images with a high variability. However, the accuracy degradation of NN is around 1% if $\varepsilon \leq 5\%$. And finally, for example, 91% multiplier power reduction ($w = 8, \varepsilon = 15\%$) corresponds with the accuracy degradation of NN less than 2.80%.

To summarise the results, it was shown in Section 3.3 that the multiplication has a significant impact on total power consumption of calculation. When the calculation of LeNet consumes approximately 44% [7], 91% reduction of multiplication power leads to a significant total power consumption reduction of the NN.

6. CONCLUSION

This paper provided a methodology for the design of power-efficient NNs with approximate multipliers. An analysis of error resiliency of neural networks showed the feasibility of using the proposed multipliers to achieve trade-off between classification accuracy versus energy consumption. By means of CGP, approximate multipliers were designed to achieve the desired tradeoffs between the accuracy and implementation cost. Resulting approximate NNs, containing the approximate multipliers, were evaluated using standard benchmarks (MNIST dataset) and a real-world classification problem of Street-View House Numbers (SVHN). A significant improvement in power efficiency was obtained compared to the exact (or original) NNs. In some cases, 91% power reduction of multiplication was obtained with classification accuracy degradation less than 2.80% for SVHN dataset.

7. ACKNOWLEDGMENTS

This work was supported by the Czech science foundation project 14-04197S and by The Ministry of Education, Youth and Sports of the Czech Republic from the National Programme of Sustainability (NPU II); project IT4Innovations excellence in science - LQ1602. Syed Shakib Sarwar and Kaushik Roy's research were funded in part by National Science Foundation.

8. REFERENCES

[1] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected CRFs. *CoRR*, abs/1412.7062, 2014.

[2] V. K. Chippa, S. T. Chakradhar, K. Roy, and A. Raghunathan. Analysis and characterization of inherent application resilience for approximate computing. In *DAC '13*, pages 113:1–113:9, 2013.

[3] D. C. Ciresan, U. Meier, L. M. Gambardella, and J. Schmidhuber. Convolutional neural network committees for handwritten character classification. In *ICDAR*, 2011.

[4] K.-L. Du and M. Swamy. *Neural Networks in a Softcomputing Framework*. Springer London, 2006.

[5] Z. Du, K. Palem, A. Lingamneni, O. Temam, Y. Chen, and C. Wu. Leveraging the error resilience of machine-learning applications for designing highly energy efficient accelerators. In *ASP-DAC '14*, 2014.

[6] G. Hinton, L. Deng, D. Yu, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, Nov 2012.

[7] P. Judd, J. Albericio, T. H. Hetherington, T. M. Aamodt, N. D. E. Jerger, R. Urtasun, and A. Moshovos. Reduced-precision strategies for bounded memory in deep neural nets. In *HiPEAC WAPCO '16*, 2016.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[9] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *the IEEE*, 86(11):2278–2324, Nov 1998.

[10] Y. LeCun, C. Cortes, and C. J. Burges. The MNIST database of handwritten digits. http://yann.lecun.com/exdb/mnist/.

[11] A. Lingamneni, A. Basu, C. Enz, K. V. Palem, and C. Piguet. Improving energy gains of inexact dsp hardware through reciprocative error compensation. In *DAC '13*, 2013.

[12] J. F. Miller. *Cartesian Genetic Programming*. Springer-Verlag, 2011.

[13] S. Mittal. A survey of techniques for approximate computing. *ACM Comput. Surv.*, 48(4), Mar. 2016.

[14] K. Nepal, Y. Li, R. I. Bahar, and S. Reda. Abacus: A technique for automated behavioral synthesis of approximate computing circuits. In *DATE '14*, 2014.

[15] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop 2011*, 2011.

[16] S. S. Sarwar, S. Venkataramani, A. Raghunathan, and K. Roy. Multiplier-less artificial neurons exploiting error resiliency for energy-efficient neural computing. In *DATE '16*, pages 145–150, 2016.

[17] G. Srinivasan, P. Wijesinghe, S. S. Sarwar, A. Jaiswal, and K. Roy. Significance driven hybrid 8T-6T SRAM for energy-efficient synaptic storage in artificial neural networks. In *DATE '16*, pages 151–156, 2016.

[18] Z. Vasicek and L. Sekanina. Evolutionary approach to approximate digital circuits design. *IEEE Tr. on Evolutionary Computation*, 19(3):432–444, 2015.

[19] S. Venkataramani, A. Ranjan, K. Roy, and A. Raghunathan. Axnn: Energy-efficient neuromorphic systems using approximate computing. In *ISLPED '15*, 2014.

[20] S. Venkataramani, K. Roy, and A. Raghunathan. Substitute-and-simplify: a unified design paradigm for approximate and quality configurable circuits. In *DATE'13*, 2013.

[21] S. Venkataramani, A. Sabne, V. J. Kozhikkottu, K. Roy, and A. Raghunathan. Salsa: systematic logic synthesis of approximate circuits. In *DAC '12*, pages 796–801.

[22] S.-C. Wang. *Interdisciplinary Computing in Java Programming*, chapter Artificial Neural Network, pages 81–100. Springer US, Boston, MA, 2003.

[23] N. Weste and D. Harris. *CMOS VLSI Design: A Circuits and Systems Perspective*. Addison-Wesley Publishing Company, USA, 4th edition, 2010.

[24] Q. Zhang, T. Wang, Y. Tian, F. Yuan, and Q. Xu. Approxann: An approximate computing framework for artificial neural network. In *DATE '15*, 2015.

Paper IV

Approximating Complex Arithmetic Circuits with Formal Error Guarantees: 32-bit Multipliers Accomplished

ČEŠKA Milan, MATYÁŠ Jiří, MRÁZEK Vojtěch, SEKANINA Lukáš, VAŠÍČEK Zdeněk and VOJNAR Tomáš

In: *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design.* Irvine, CA: Institute of Electrical and Electronics Engineers, 2017, pp. 416-423. ISBN 978-1-5386-3093-8.

Approximating Complex Arithmetic Circuits with Formal Error Guarantees: 32-bit Multipliers Accomplished

Milan Češka, Jiří Matyáš, Vojtech Mrazek, Lukas Sekanina, Zdenek Vasicek and Tomas Vojnar
Faculty of Information Technology, Centre of Excellence IT4Innovations
Brno University of Technology, Brno, Czech Republic
{ceskam, imatyas, imrazek, sekanina, vasicek, vojnar}@fit.vutbr.cz

***Abstract*—We present a novel method allowing one to approximate complex arithmetic circuits with formal guarantees on the approximation error. The method integrates in a unique way formal techniques for approximate equivalence checking into a search-based circuit optimisation algorithm. The key idea of our approach is to employ a novel search strategy that drives the search towards promptly verifiable approximate circuits. The method was implemented within the ABC tool and extensively evaluated on functional approximation of multipliers (with up to 32-bit operands) and adders (with up to 128-bit operands). Within a few hours, we constructed a high-quality Pareto set of 32-bit multipliers providing trade-offs between the circuit error and size. This is for the first time when such complex approximate circuits with formal error guarantees have been derived, which demonstrates an outstanding performance and scalability of our approach compared with existing methods that have either been applied to the approximation of multipliers limited to 8-bit operands or statistical testing has been used only. Our approach thus significantly improves capabilities of the existing methods and paves a way towards an automated design process of provably-correct circuit approximations.**

***Index Terms*—approximate computing, logical synthesis, genetic programming, formal methods**

I. Introduction

As many important applications are inherently error resilient, precision of the involved computations can be traded for improved energy efficiency, performance, and/or chip area. Various approaches exploiting this fact have been developed in recent years and presented under the umbrella of the so-called *approximate computing* [1]. These approximations can be conducted at different system levels with circuit approximation being one of the most popular.

Circuit approximation techniques can be classified into two main groups: (1) *Frequency/voltage over-scaling* where timing-induced errors can appear as the circuit is operated on a higher frequency or lower voltage than the nominal value. (2) *Functional approximation* where the original circuit is replaced by a less complex one which exhibits some errors but improves non-functional circuit parameters such as power consumption or chip area. We only deal with the latter approach in this paper. Circuit approximation can be formulated as a multi-objective optimization problem where the error and non-functional circuit parameters are conflicting design objectives. Since the resulting approximate circuits are common circuits, they can be implemented using the standard circuit design flow.

We focus on *approximate arithmetic circuits* (AACs) because they are frequently used in key applications relevant for approximate computing. Prominent examples are signal, image, and video processing circuits (such as filters, discrete transforms, and motion estimation blocks [2]), or the multiply-accumulate-transform structures of artificial neurons in neural networks (consuming about 50% of the total power in neural network accelerators [3]).

Various *error metrics*, such as the worst-case relative error or the mean absolute error, for evaluating approximate circuits have been proposed (cf. Sect. III). A crucial question is then how the error of a given approximation is derived. For that, as discussed in more details in the related work section, methods based on *simulating* the circuit on given inputs are often used. However, such approaches suffer from low scalability (exhaustive simulation), lack of strong guarantees (when simulating the circuit for a random subset of the possible inputs only), and/or specialization to certain circuits only (statistical models). Alternatively, as in our case, the error can be derived using *formal verification*. The main advantages of this approach lie in that (1) formal error bounds can be given as a part of the input and (2) the approach is more scalable than exhaustive circuit simulation.

While formal methods of (exact) equivalence checking have been studied for decades, only a few formal approximate checking methods have been used in circuit approximation tools. Depending on the particular error metric, the error calculation is transformed to a decision problem and solved by means of SAT solving or binary decision diagrams (BDDs). Despite of enormous progress in the area of SAT solvers and BDD libraries, approximation of arithmetic circuits with formal error guarantees was so far limited to circuits no more complex than 16-bit adders and 8-bit multipliers [4], [5], [6].

In this paper, we present a new method for designing complex approximate arithmetic circuits with formal bounds on the approximation error. The method uniquely integrates new formal techniques for approximate equivalence checking into search-based circuit optimization by means of Cartesian genetic programming (CGP). The key idea is to employ a novel search strategy driving the search towards promptly verifiable approximate circuits. We have implemented the strategy within the ABC tool and extended the underlying equivalence checking algorithm to support queries on the worst-case error. This extension builds on a new effective construction of miters, i.e. auxiliary circuits interconnecting the original correct circuit and its approximation such that their approximate equivalence can be checked.

We decided to optimize for the *worst-case error* since its exact value can be important in time-critical and dependable

systems (e.g., inverse kinematics in robot control [7]) or when complex approximate arithmetic circuits are constructed using less complex approximate building (circuit) blocks. The final error then depends on how the worst case error is propagated from low-level blocks to the result. Moreover, even in not so critical applications such as image processing, low average error but excessive worst-case error can produce unacceptable results [8]. Finally, our results suggest that there is also a high correlation between the worst-case error and the mean absolute error (Sect. V).

While our primary motivation is to automatically approximate complex multipliers, our method is directly applicable to other arithmetic circuits too. The method is capable of providing Pareto fronts showing high-quality compromises between the circuit error and non-functional circuit parameters. Results are presented for approximate multipliers (with up to 32-bit operands) and adders (with up to 128-bit operands) and compared with several approximate circuits available in literature. This is for the first time when such complex approximate arithmetic circuits with formally guaranteed error bounds have been presented.

Contributions: We propose a new miter construction allowing for efficient approximate equivalence checking tailored to search-based approximation of complex arithmetic circuits. We design a novel search strategy for synthesis of approximate circuits with formal error guarantees that integrates Cartesian genetic programming and the proposed approximate equivalence checking. Using a resource-limited verifier, the strategy drives the search towards promptly verifiable candidates and thus provides scalable approximation of complex circuits. We develop an implementation of the miter construction and the search strategy within the ABC tool and perform extensive experimental evaluation of our approach on large circuits including approximation of 128-bit adders and 32-bit multipliers. Within several hours, we are able to construct high-quality Pareto sets of 128-bit adders and 32-bit multipliers that represent the trade-offs between the circuit error and non-functional circuit parameters.

II. Related Work

This section presents a brief survey of the most important approaches developed for functional approximation of multipliers and adders. We restrict our attention to these two arithmetic operations because they represent the key components of more complex circuits and thus their approximation has been intensively studied. Moreover, multipliers—due to their complex structure—represent one of the most difficult arithmetic circuits from the perspective of both approximation as well as verification.

A. Approximation Methods

The approximation process usually starts with a fully functional circuit and a target error. *Circuit-dependent approximation methods* then take the structure of the arithmetic circuit at the input and (manually or algorithmically) introduce modifications to carefully preselected parts of the circuit. In the case of adders, it is possible to approximate elementary 1-bit adders, modify the carry propagation chain, or introduce segments of adders and generate the carry using different methods [9]. In the case of multipliers, generation of partial products, the summation tree, counters, or compressors are approximated [10]. In addition to that, the simple bit-width reduction belongs to this category of methods too.

More complex approximate circuits can be constructed by a smart *composition* of approximate elementary blocks. For example, a 2-bit multiplier was approximated in [11] and then used as a building block of more complex multipliers. This strategy can be improved, e.g., by configurable lossy compression of the partial product rows based on their progressive bit significance [12].

The concept of *quality configurable circuits* uses elementary circuits composed in such a way that their error can be modified online using several configuration bits in order to dynamically reduce the power consumption. The configuration bits can (dis)connect some preselected parts of the circuit. As the source codes of quality configurable adders [13] and multipliers [2] are available online, we compare them with approximate circuits obtained using our approach.

General-purpose methods, such as SALSA [14] or SASIMI [15], aim at automatically approximating circuits independently of their structure. These methods operate with different circuit representations and employ various heuristics to identify circuit parts suitable for approximation. Evolutionary algorithms have been recently applied to accomplish desired approximations in a holistic scenario [16], [17]. A comprehensive library of 8-bit adders and multipliers was built using multi-objective CGP [18].

B. Simulation-Based Error Computation

Conceptually, the simplest approach to obtain precise error bounds of an AAC is to simulate its function on all possible inputs. However, even on state-of-the-art computer architectures, this approach has principal scalability limitations causing that it cannot be used to synthesize approximate circuits with more than 12-bit operands [19].

Due to that, the error is commonly estimated using a subset of input vectors only, e.g. 10^8 inputs were used to evaluate 16-bit adders in [9]. Of course, the main drawback of this approach is that no formal guarantees on the error bound can be provided. Alternatively, the circuit error can be calculated using a statistical model constructed for elementary circuit components and their compositions [20], [21]. However, reliable and general statistical models can only be constructed in some specific situations.

C. Formal Error Computation

Recently, various applications of formal methods have been intensively studied in order to improve the scalability of the design process of correct as well as approximate circuits. For designing correct circuits (where one insists on preserving the original functionality but tries to optimize non-functional parameters), one can consider combinational

equivalence checking based on modern SAT solvers, efficient BDD representations of circuits, or algebraic computation techniques combining polynomial representation of circuits with logic reductions [22], [23]. For designing AACs, a more challenging notion of *relaxed* or *approximate equivalence checking* is needed. This notion requires to quantify the approximation error or, alternatively, prove whether the error is below a certain threshold.

To quantify the approximation error using formal verification techniques, a use of auxiliary circuits, called *miters*, combining the original circuit and the approximate circuit was proposed in [24]. In order to check whether a predefined worst-case error is violated by the candidate approximate circuit, a pseudo-Boolean SAT solver combining a SAT solver with integer linear programming was then employed.

The number of inputs for which an approximate circuit returns an incorrect result can be quantified using *SAT counting methods* (so-called #SAT solvers). However, despite the recent progress in the area of #SAT solvers (see, e.g., [25]), our preliminary experiments indicate that #SAT problems encoding the error quantification are currently beyond the capabilities of state-of-the-art #SAT tools even for 12-bit multipliers.

An efficient BDD-based approach allowing one to guarantee the worst-case and the average-case arithmetic error of approximate adders up to 16-bit operands was proposed in [5]. An alternative approach that uses BDDs representing characteristic functions was employed in [4]. Compared to our approach, this approximation method lags behind in scalability, which is demonstrated by the fact that it has been applied to the approximation of multipliers limited to 8-bit operands and adders limited to 16-bit operands only.

III. Error Metrics for AACs

Various metrics describing the error of AACs have been proposed and shown suitable for different application domains. The most popular error metrics relevant especially to arithmetic circuits are the *worst-case absolute error* (WCAE) and the *mean absolute error* (MAE). For a correct circuit G, further denoted as the *golden circuit*, which computes a function f_G, and its approximation C, computing a function f_C, where $f_G, f_C : \{0,1\}^n \to \{0,1\}^m$, these metrics, relativized by the range of the output, are defined as follows:

$$\mathrm{WCAE}(G,C) = \frac{\max_{x\in\{0,1\}^n} |\mathrm{int}(f_G(x)) - \mathrm{int}(f_C(x))|}{2^m},$$

$$\mathrm{MAE}(G,C) = \frac{\sum_{x\in\{0,1\}^n} |\mathrm{int}(f_G(x)) - \mathrm{int}(f_C(x))|}{2^m},$$

where $\mathrm{int}(x)$ denotes the integer representation of a bit vector x and $|i|$ denotes the absolute value of an integer i.

A. Checking Worst Case Errors

To compute whether the WCAE is violated, we can adopt the concept of approximation miter introduced in [24]. The general configuration of the approximation miter is shown in Fig. 1. The miter consists of the inspected approximate

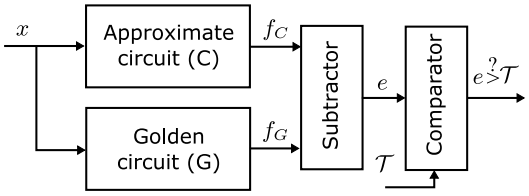

Fig. 1. Approximation miter for the worst-case error analysis, typically $e(x) = |f_G(x) - f_C(x)|$.

circuit C, the golden circuit G which serves as the specification, a subtractor, and a comparator which checks whether the error introduced by the approximation is greater than a given threshold $\mathcal{T}$. The output of the miter is a single bit which evaluates to 1 if and only if the error is violated, i.e. $\mathrm{WCAE}(G,C) > \mathcal{T}$.

For a given input vector x, the subtractor calculates the difference between the output of the golden circuit, i.e. $f_G(x)$, and the output of the approximate circuit, i.e. $f_C(x)$. Let $d = \mathrm{int}(f_G(x)) - \mathrm{int}(f_C(x))$ be the error magnitude. A direct computation of the WCAE according to its definition leads to evaluating the expression $e = |d|$, i.e. the absolute difference of the error magnitude. The absolute difference is typically calculated by means of a common two's complement subtractor (implemented using m full-adders with the first carry-in set to 1 and inverting each bit of the subtrahend) followed by a circuit determining the absolute value (computed using m half-adders and m XOR gates).

B. The Proposed Miter Construction

Miters used in the literature compute the absolute value of the difference between f_G and f_C. The computation is usually performed in two steps. Firstly, a subtractor with a signed output evaluates $f_G - f_C$. Secondly, the absolute value has to be computed. The circuit performing such a task contains XOR chains which are a known cause of poor performance of the state-of-the-art SAT solvers [26]. The main reasons are that unlike AND/OR gates, the Boolean constraint propagation over XOR gates is limited, and the XOR operations cause the CNF form of the formulae to grow rapidly.

In order to avoid long XOR chains at the output of the miter which slowdown the decision process, we propose to employ a different approach. The key idea is to compare the result of the subtractor with both the positive and negative value of the threshold and thus avoid the expensive evaluation of the absolute value. For a given threshold $\mathcal{T}$ on the worst-case absolute error WCAE, it holds that $e > \mathcal{T}$ is satisfied iff d is positive and $d > \mathcal{T}$, or d is negative and $-d > \mathcal{T}$. As we typically deal with numbers in the two's complement, the second condition is equal to $\neg d > (\mathcal{T} - 1)$. Hence, we can use the two's complement representation and examine the positive and negative values separately to avoid usage of the absolute difference of the output.

Since the threshold $\mathcal{T}$ is fixed during the design process, we can easily avoid the standard comparator consisting of a long chain of XOR gates. This helps us to further simplify the miter and improve the performance of the decision procedure.

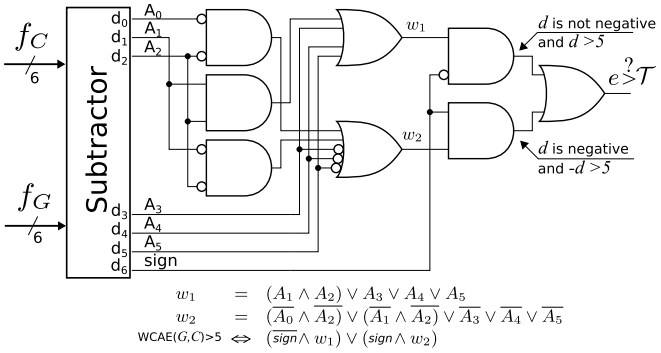

Fig. 2. The proposed approximation miter for the worst-case error analysis: an example for $\mathcal{T} = 5$, $N = 6$.

In particular, we replace the sequential comparison of the particular bits of the operands implemented as

$$A > B \equiv \bigvee_{0 \le i \le N-1} \left(A_i \wedge \neg B_i \bigwedge_{i < j \le N-1} \overline{A_j \oplus B_j} \right),$$

for B being a constant bit vector representing the threshold $\mathcal{T}$, by a simpler procedure implemented as

$$A > B \equiv \bigvee_{0 \le i \le N-1 \ \wedge \ B_i = 0} \left(A_i \bigwedge_{i < j \le N-1 \ \wedge \ B_j = 1} A_j \right).$$

As is evident, the resulting formula does not contain any XOR gate. Note that d is represented as an $m+1$ bit number in the two's complement—hence, A corresponds to the N least significant bits of d where $N = m$. The $(m+1)$-th bit is reserved for the sign and employed for determining whether d encodes a positive or negative number. The miter for $\mathcal{T} = 5$, f_C and f_G with 6-bit outputs is illustrated in Fig. 2.

The proposed construction, compared to the construction using the absolute value and full comparators, allows us to obtain smaller and structurally less complex miters. Such miters can be efficiently used in the SAT-based CEC procedures, resulting in a significant acceleration of the candidate circuit evaluation. Our experiments show that, in the case of arithmetic circuits having 64 output bits (e.g. 32-bit multipliers), the proposed construction improves the size of the miters (in terms of the number of And-Inverter Graph (AIG) nodes representing the circuit) by about 25–35% depending on the value of $\mathcal{T}$, where $\mathcal{T}$ ranged from 0.0001% to 0.5% of the maximal value at the output (i.e. 2^{64}) in our experiment.

IV. Search-Based Design of AACs

In this section, we present our novel approach to the search-based design of AACs combining principles of CGP with a verifiability-driven search strategy that employs a fitness function based on the approximate equivalence checking.

A. Cartesian Genetic Programming

CGP is a form of genetic programming where the candidate solutions are represented as a string of integers of a fixed length that is mapped to a directed acyclic graph [27]. This integer representation is called a *chromosome*. CGP can efficiently represent common computational structures including mathematical equations, computer programs, neural networks, and digital circuits. The candidate circuits are typically represented in a two-dimensional array of programmable two-input nodes. Every node is encoded by three integers in the chromosome representation where the first two numbers denote the node's inputs, the third represents the node's function (see the illustration in Fig. 3).

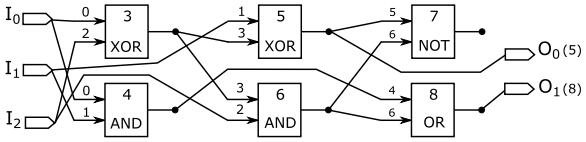

Fig. 3. Full adder represented by CGP. Chromosome: (0, 2, 2) (0, 1, 0) (1, 3, 2) (3, 2, 0) (5, 6, 3) (4, 6, 1) (5, 8), node functions: AND (0), OR (1), XOR (2), NOT (3).

In circuit approximation, the evolution loop starts with a *parent* representing a correctly working circuit. New candidate circuits are obtained from the parent using a *mutation operator* which performs random changes in the candidate's chromosome in order to obtain a new, possibly better candidate solution. In the next step, the algorithm evaluates the quality of each solution using a specified metric, called the *fitness function*. This function assesses important correctness and performance aspects of circuits. The candidate with the best fitness value is chosen as the parent of the next generation, the other solutions are removed and the evolution continues with generating new candidate circuits. The whole loop is repeated until a termination criterion is met. For details of CGP, see [27].

The most critical and time consuming part of the CGP loop is the fitness evaluation, which principally limits the scalability of the search-based design. To alleviate this problem, we propose below a novel search strategy.

B. Verifiability-Driven Search Strategy

The verifiability-driven search strategy can be seen as a general concept improving the scalability of evolutionary design methods. We demonstrate its key idea on the below problem.

Problem: *For a given golden circuit G and a threshold $\mathcal{T}$, our goal is to find a circuit C^* with the minimal size such that the error WCAE$(G, C^*) \le \mathcal{T}$.*

This problem formulation allows us to define the fitness function f in the following way:

$$f(C) = \begin{cases} \text{size}(C) & \text{if WCAE}(G, C) \le \mathcal{T}, \\ \infty & \text{otherwise} \end{cases}$$

where $\text{size}(C)$ denotes the size of the circuit C. Since the procedure deciding whether $\text{WCAE}(G, C) \le \mathcal{T}$ (further denoted as SAT solver) represents the most time consuming part of the design loop, we avoid calling the procedure as much as possible. Therefore, we only call SAT solver for circuits C satisfying $\text{size}(C) < \text{size}(B)$ where B is the best solution with an acceptable error (i.e., $\text{WCAE}(G, B) \le \mathcal{T}$) that we have found so far. Our experiments show that, during the evolution process, a significant set of candidate designs C does not satisfy the condition $\text{size}(C) < \text{size}(B)$ and thus their fitness can be easily assessed without SAT solver.

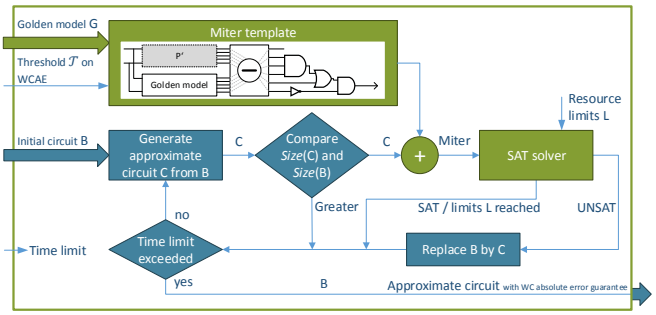

Fig. 4. The main steps of the proposed verifiability-driven search scheme.

Our experiments further indicate that a long sequence of candidate circuits B_i improving the size and having an acceptable error has to be typically explored to obtain a solution that is sufficiently close to C^*. Therefore, both the SAT and the UNSAT queries to SAT solver have to be short. To this end, we use an additional criterion for the evaluation of the circuit C, namely, the ability of SAT solver to prove that $\text{WCAE}(G, C) \leq \mathcal{T}$ with a given limit L on the resources available to the underlying decision procedure. If the procedure fails to prove $\text{WCAE}(G, C) \leq \mathcal{T}$ within the limit L, we set $f(C) = \infty$ and generate a new candidate. The design loop using the verifiability-driven search is illustrated in Fig. 4.

The inputs of the design process include: (1) the golden model G, (2) the threshold on the worst case absolute error $\mathcal{T}$, (3) the initial circuit B having an acceptable error (it can be either the golden model or a suitable approximation we want to start with), and (4) the time limit on the overall design process. The loop exploits the CGP principles; namely, it uses mutations to generate new candidate circuits C from the candidate circuit B representing the best approximation of the circuit C^* that we have found so far. The circuit C is then evaluated using the fitness function f as described above. If the candidate C belongs to the improving sequence (i.e., $\text{size}(C) < \text{size}(B)$ and $\text{WCAE}(G, C) \leq \mathcal{T}$), we replace B by C. The design loop terminates if the time limit is reached and B is returned as the output of the design process.

In our verifiability-driven search scheme, we use the resource limit L (as a parameter of the design loop) to drive the search towards candidates that can be promptly evaluated. We intentionally throw away improving candidates B_i that require greater resources and thus longer, but still feasible, verification time. The reason for this is the fact that by mutating these candidates we would most likely obtain solutions that would require the same or even longer verification times and thus finding the whole improving sequence would become time-infeasible. Instead, we require that every improving candidate B_i has to be verifiable using the resource limit L and thus drive the search towards candidates B_i that, for a given time limit on the overall design process, lead to longer improving sequences. Our experiments indicate that these sequences lead to candidate circuits that are closer to C^*. Since we are able to evaluate a much larger set of candidate circuits, we have a better chance to find a long improving sequence within the given time provided that it exists for the limit L.

The obvious disadvantage is that we possibly cut improving sequences that would lead to good solutions within the given design time. It can also happen that, for the limit L, no improving sequence exists, while it exists for a slightly greater resource limit. Despite of this limitation, our results clearly show that the proposed verifiability-driven search strategy allows us to utilise the given design time in a more efficient way compared to the standard evolution schemes.

C. Integration to the ABC Tool

The proposed approach performs the approximation at the level of the CGP problem representation (i.e., on acyclic oriented graphs with arbitrary two-input logic functions in the nodes). The green part of Fig. 4 shows the position of ABC in our methodology. ABC is primarily used to construct the miter and decide whether the maximal arithmetic error of the candidate circuit is not above $\mathcal{T}$. The proposed miter construction allows us to reduce the problem of approximate equivalence checking to the Boolean satisfiability (SAT) problem. In order to evaluate a candidate circuit, (1) a candidate chromosome is used to construct a corresponding AIG, (2) another AIG, representing the golden circuit, is constructed (just once at the beginning of the evolution), and (3) the miter is built. The state-of-the-art techniques used for CEC in the ABC tool—the `iprove` engine—are then applied to decide the equivalence. An important feature of the mix of techniques used in `iprove` is that one can control the time needed for one query, which is the key feature we exploit in our verifiability-driven search strategy. In particular, the satisfiability checking can be controlled by fine-tuning various resource limits for the different techniques used, such as the number of simulations performed to prove non-equivalence, the number of conflicts in structural hashing, or the number of logic-reduction steps. We so far used solely a limit on the maximal number of conflicts in which a single variable (representing an AIG node) can be involved during the backtracking process. Our experiments show that this resource limit allows us to effectively control the time needed for particular `iprove` queries and thus to drive the search towards promptly verifiable circuits.

A similar approach has recently been used in circuit approximation exploiting the approximate-aware rewriting of an AIG representation of circuits [4]. Principally, our approach differs in the candidate circuit representation (the gate-level CGP encoding), its evaluation, and in using the verifiability-driven evolution instead of a simple greedy algorithm for AIG pruning. The gate-level representation is an important feature of our approach which allows us to efficiently capture XOR-intensive structures existing in arithmetic circuits.

V. Results

To evaluate the proposed method, we primarily focused on complex approximate multipliers as they are the most challenging benchmark problems. Since only 8-bit multipliers with guaranteed error bounds were presented in the literature so far, there are no solutions available for a direct comparison in the case of 16-bit and more complex approximate multipliers.

Hence, (1) we compare the 16-bit approximate multipliers that we generated using our method with 16-bit multipliers (available in the literature) whose error was determined using simulation, and then (2) we present Pareto fronts (the error and key circuit parameters) for 20-bit, 24-bit, 28-bit, and 32-bit approximate multipliers and up to 128-bit approximate adders to demonstrate the scalability of the proposed method.

A. Experimental Setup

We implemented our approach, including the miter construction and verifiability-driven evolution, within the ABC tool [28]. Array multipliers and ripple carry adders composed of 2-input gates were employed as the initial (golden) circuits for CGP. The number of nodes in the CGP's grid is equal to the number of gates of the initial circuit. The set of functions consists of the common two-input logic gates, the buffer, and the inverter. We used 2 circuits in the population and 5 integers were modified by the mutation operator.

For each target WCAE, we performed 30 independent runs of CGP to obtain statistically significant results. Each CGP was executed for 2 hours on an Intel Xeon X5670 2.4 GHz processor using a single core. The individual CGP runs are independent and thus we executed them in parallel using a cluster of these processors to accelerate the design process.

For purposes of the fitness evaluation, the circuit size is estimated as the sum of the relative area of the two-input gates used, where the sizes of each gate are taken from the technology library. At the end of the evolution, the 5 most fitting circuits for each WCAE were synthetized using the Synopsys Design Compiler (high-effort compiling for a better quality of the results) for a 45 nm technology library in order to obtain non-functional parameters like the area and power-delay product (PDP). The accurate implementations were created by means of Verilog $*$ and $+$ operators and synthesized in the same way as approximate circuits.

B. 16-bit Approximate Multipliers

An evaluation of the verifiability-driven search: In the first experiment, we approximated the golden 16-bit multiplier for 9 target values of WCAE from the set $\{0.1, 0.2, 0.5, 1, 2, 5, 10, 15$ and $20\%\}$ and evaluated the proposed method with three different settings of the resource limit L controlling the maximal number of conflicts for one AIG node: (1) no limits, i.e., L=∞, (2) L=$160K$, and (3) L=$20K$. The limits L=160K and L=20K roughly correspond to the time limit of 120 sec. and 3 sec., respectively, on 16-bit multipliers.

Fig. 5 shows that, for WCAE $\geq 2\%$, the resource limit L has a marginal impact on the PDP and area. However, with a decreasing target WCAE, the limit L=20K provides significantly better results. For example, if WCAE = 0.1% and L=20K, 22,050 SAT calls were produced and 11% of them were terminated on average because of the termination condition. In the case of L=160K, 856 SAT calls were produced only (15% terminated). The average number of SAT calls (across all target errors) that were forced to terminate is 6.28% (for L=160K) and 8.84% (for L=20K). If L=∞, 170 SAT

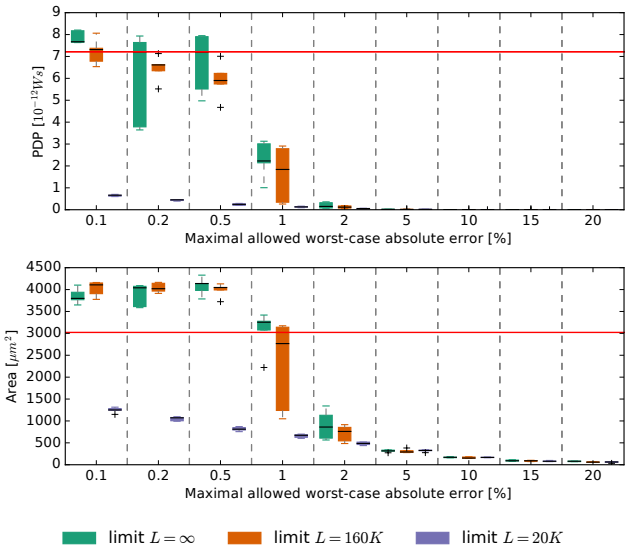

Fig. 5. PDP and area of approximate 16-bit multipliers for 9 target errors obtained using 3 different resource limits L on the SAT solver. The red line shows the PDP and area of the accurate multiplier.

calls were evaluated for WCAE = 0.1% only. Despite the fact that some potentially good candidate circuits are quickly rejected, the aggressive resource limits allowed us to generate and evaluate significantly more candidate circuits and thus to substantially improve the quality of results. Box plots in Fig. 5 also show that independent runs with L=20K lead to circuits having very similar parameters (low inter-quartile distances) and thus this limit is be used in the following experiments.

Note that the parameters of some approximate multipliers shown in Fig. 5 are worse than for the accurate multiplier. The reason is that the relative area is the only non-functional circuit parameter optimized by CGP while the PDP and area are computed at the end of the optimization using the Synopsys Design Compiler. We have never observed this discrepancy for the limit L=20K.

A comparison with other multipliers: Next, we generated 16-bit approximate multipliers using the setup described in the previous section and compared them with approximate multipliers available in the literature. In order to perform a fair comparison (the error of the published multipliers was originally estimated using simulation), we modified our method and applied a binary search strategy to determine the WCAE exactly. In addition to WCAE, we also provide MAE obtained using simulation (10^9 vectors).

We considered the following 16-bit approximate multipliers:

M1 Approximate configurable multipliers from the lpACLib library [13], where the multiplication is recursively simplified using two different variants (denoted as *Lit* and *V1*) of an elementary block representing a 2-bit multiplier. The partial results are summed using accurate adders. We implemented 32 different architectures consisting of four 8-bit multipliers where each of these multipliers is configurable as exact/approximate (2^4 configurations) and can be built using either *Lit* (M1Lit) or *V1* (M1V1) blocks.

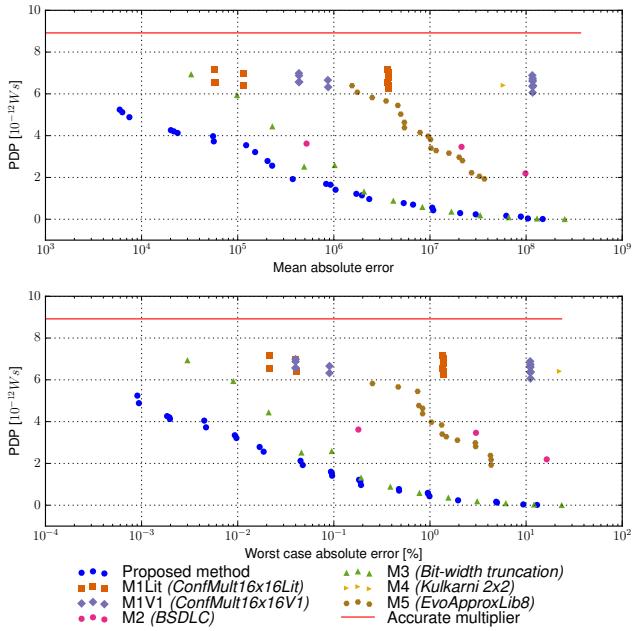

Fig. 6. Parameters of 16-bit approximate multipliers considered in our study.

M2 The approximate multiplier employing the bit-significance-driven logic compression as introduced in [12].

M3 Approximate multipliers obtained from exact multipliers using the bit-width reduction. The reduction replaces 16-bit multipliers by accurate x-bit multipliers (for $x < 16$). It ignores the LSBs of the operands and leaves the LSBs of the result zero.

M4 The approximate multiplier composed of approximate 2-bit multipliers as proposed in [11].

M5 Approximate multipliers composed of 8-bit multipliers that are available in the EvoApproxLib library [18]. The construction principle is taken from [11].

For all considered multipliers, the value of PDP is plotted against WCAE and MAE in Fig. 6 (only Pareto fronts are visualized). While the bit-width reduction provides the same quality of results as our method for large target errors (up to 20% WCAE), it is significantly outperformed by our approach for small target errors. Despite that the existing approximate multipliers typically exhibit good tradeoffs between the error and PDP in specific applications (as demonstrated in the relevant literature), Fig. 6 clearly shows that these multipliers are considerably Pareto-dominated by the multipliers obtained using our approach. These results were, in fact, expected as the proposed method is based on a global holistic optimization approach while the other approximate multipliers were composed of smaller ones and the composition procedure always introduces some overhead. Finally, it is an interesting observation that MAE follows the trend of WCAE. It seems that WCAE can be used as a good indicator of MAE.

C. Complex Multipliers

The aim of our further experiments is to show that the proposed method is scalable and can approximate complex multipliers. We present the results of the approximation process on 12-bit, 16-bit, 20-bit, 24-bit, 28-bit, and 32-bit multipliers. The target WCAEs were adapted accordingly to respect the range of values in the different considered bit widths. We used the same setup as in the previous sections but increased the time of optimization to 4 hours for the 24-bit multiplier and 6 hours for larger multipliers. The reason is that the search space becomes much bigger. While the exact 12-bit multiplier contains 850 two-input gates, the 32-bit exact multiplier requires over 6,300 gates. We obtained (as the result of evolution) over 1190 unique multipliers. Because of this huge number and for the sake of clarity, Fig. 7 shows parameters of approximate multipliers occupying the Pareto fronts only.

In the experiments, we observed that, in the case of 12-bit multipliers, 2.4% of SAT calls were terminated on average due to the resource limit L=20K only. However, this number increased to 36.9% in the case of approximate 32-bit multipliers. For all bit widths, the MAE is around 30% of the worst-case error, which again demonstrates that WCAE is a good indicator of MAE. Fig. 7 also shows that the obtained approximations cover the whole range (up to 100%) of the Area axis. However, this is not the case for PDP. The reason is that we optimize the relative area and PDP is computed after the synthesis.

Since Pareto fronts shown in Fig. 7 follow the trend of the highly competitive fronts for the 16-bit multipliers presented before, we believe that the tradeoffs between the circuit error and size obtained for more complex multipliers are also very good and thus the corresponding circuits represent the cutting edge of approximate multipliers and can serve as a new benchmark set for approximate computing.

D. Approximate Adders

In order to demonstrate that the proposed method is applicable for other complex arithmetic circuits, we constructed Pareto fronts for approximate adders with 20-bit to 128-bit operands. Approximation of adders is much easier than approximation of multipliers since adders are structurally less complicated and the number of outputs is lower. For example, the exact 20-bit adder requires 140 two-input gates and the 128-bit adder consists of 1,000 gates.

The approximate adders were constructed using the same setup as in the previous section. A single CGP run took 2 hours (for all bit widths). Fig. 8 shows parameters of approximate adders occupying the corresponding Pareto fronts. We report 16 to 18 non-dominated implementations of 24-bit, 28-bit, and 32-bit adders in terms of PDP and WCAE. For 64-bit and 128-bit adders, 12 tradeoffs are reported only because we have restricted the number of target error levels. Similarly to the evolved multipliers, the proposed approximate adders are also good candidates for including into a new benchmark suite.

VI. Conclusion

Automated design of approximate circuits with formal error guarantees is a landmark of provably-correct construction of energy-efficient systems. We present a solution to this problem, introducing a novel verifiability-driven search strategy

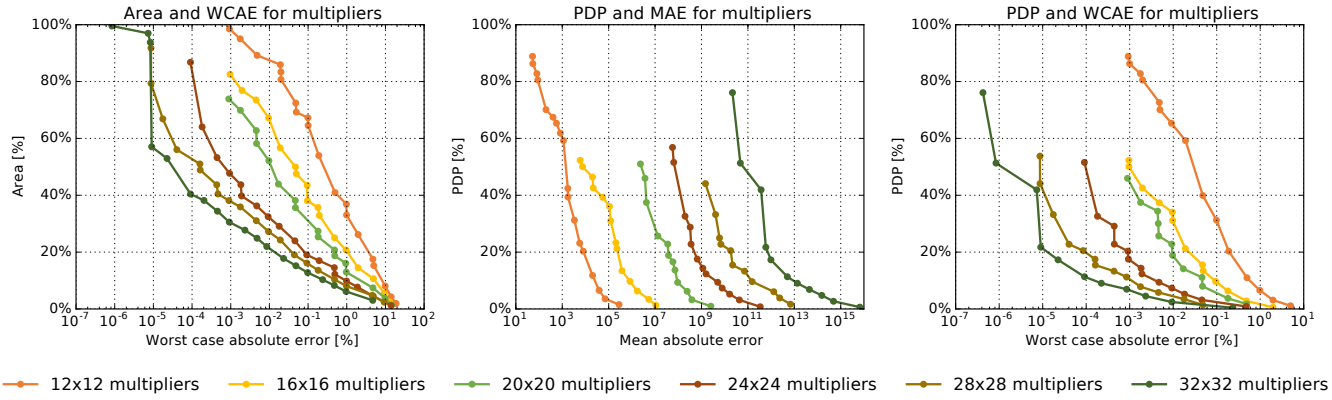

Fig. 7. Pareto fronts showing parameters of evolved approximate multipliers. 100% refers to parameters of the accurate multiplier for a given bit width.

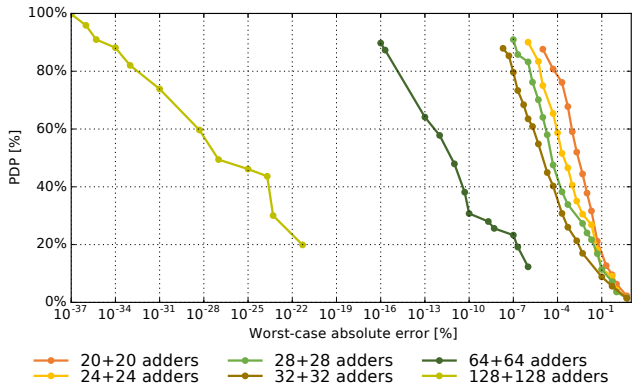

Fig. 8. Pareto fronts showing parameters of evolved approximate adders. 100% refers to parameters of the accurate adder for a given bit width.

that uniquely integrates approximate equivalence checking into a search-based circuit optimisation algorithm. Able to construct high-quality Pareto sets of 32-bit multipliers and 128-bit adders, our method shows excellent scalability and paves the way for design automation of complex approximate circuits.

In the future, we will thoroughly explore relationships between resource limits on the underlying SAT solvers and the structure of the resulting circuits. This will allow us to further improve the performance of our method and thus to go beyond the approximation of 32-bit multipliers. We will also integrate the constructed circuits into real-world energy-aware systems to demonstrate practical impacts of our work.

Acknowledgments: This work has been supported by the Czech Science Foundation grant No. GA16-17538S.

References

[1] S. Mittal, "A survey of techniques for approximate computing," *ACM Comput. Surv.*, vol. 48, no. 4, pp. 62:1–62:33, 2016.

[2] M. Shafique, R. Hafiz *et al.*, "Invited: Cross-layer approximate computing: From logic to architectures," in *Proc. of DAC'16*, 2016, pp. 1–6.

[3] P. Judd, J. Albericio *et al.*, "Proteus: Exploiting numerical precision variability in deep neural networks," in *ICS'16*, 2016, pp. 1–12.

[4] A. Chandrasekharan, M. Soeken *et al.*, "Approximation-aware rewriting of AIGs for error tolerant applications," in *Proc. of ICCAD'16*, 2016, pp. 83:1–83:8.

[5] Z. Vasicek, V. Mrazek, and L. Sekanina, "Towards low power approximate DCT architecture for HEVC standard," in *Proc. of DATE'17*, 2017, pp. 1576–1581.

[6] C. Yu and M. Ciesielski, "Analyzing imprecise adders using BDDs – a case study," in *Proc. of ISVLSI'16*, 2016, pp. 152–157.

[7] B. Grigorian and G. Reinman, "Dynamically adaptive and reliable approximate computing using light-weight error analysis," in *Proc. of AHS'14*, 2014, pp. 248–255.

[8] D. S. Khudia, B. Zamirai *et al.*, "Rumba: An online quality management system for approximate computing," in *ISCA'15*, 2015, pp. 554–566.

[9] H. Jiang, J. Han, and F. Lombardi, "A comparative review and evaluation of approximate adders," in *Proc. of GLVLSI'15*, 2015, pp. 343–348.

[10] H. Jiang, C. Liu *et al.*, "A comparative evaluation of approximate multipliers," in *Int. Symp. Nanoscale Architectures*, 2016, pp. 191–196.

[11] P. Kulkarni, P. Gupta, and M. D. Ercegovac, "Trading accuracy for power in a multiplier architecture," *J. Low Power Electronics*, vol. 7, no. 4, pp. 490–501, 2011.

[12] I. Qiqieh, R. Shafik *et al.*, "Energy-efficient approximate multiplier design using bit significance-driven logic compression," in *Proc. of DATE'17*, 2017, pp. 7–12.

[13] M. Shafique, W. Ahmad *et al.*, "A low latency generic accuracy configurable adder," in *Proc. of DAC'15*, 2015, pp. 86:1–86:6.

[14] S. Venkataramani, A. Sabne *et al.*, "SALSA: systematic logic synthesis of approximate circuits," in *Proc. of DAC'12*, 2012, pp. 796–801.

[15] S. Venkataramani, K. Roy, and A. Raghunathan, "Substitute-and-simplify: a unified design paradigm for approximate and quality configurable circuits," in *Proc. of DATE'13*, 2013, pp. 1367–1372.

[16] Z. Vasicek and L. Sekanina, "Evolutionary approach to approximate digital circuits design," *IEEE Trans. Evol. Comput.*, vol. 19, no. 3, pp. 432–444, 2015.

[17] K. Nepal, S. Hashemi *et al.*, "Automated high-level generation of low-power approximate computing circuits," *IEEE Trans. Emerg. Topics Comput.*, pp. 1–13, 2017.

[18] V. Mrazek, R. Hrbacek *et al.*, "Evoapprox8b: Library of approximate adders and multipliers for circuit design and benchmarking of approximation methods," in *Proc. of DATE'17*, 2017, pp. 258–261.

[19] V. Mrazek, S. S. Sarwar *et al.*, "Design of power-efficient approximate multipliers for approximate artificial neural networks," in *Proc. of ICCAD'16*, 2016, pp. 81:1–81:7.

[20] C. Li, W. Luo *et al.*, "Joint precision optimization and high level synthesis for approximate computing," in *DAC'15*, 2015, pp. 1–6.

[21] S. Mazahir, O. Hasan *et al.*, "Probabilistic error modeling for approximate adders," *IEEE Trans. Comput.*, vol. 66, no. 3, pp. 515–530, 2017.

[22] M. Ciesielski, C. Yu *et al.*, "Verification of gate-level arithmetic circuits by function extraction," in *Proc. of DAC '15*, 2015, pp. 52:1–52:6.

[23] A. Sayed-Ahmed, D. Große *et al.*, "Formal verification of integer multipliers by combining Gröbner basis with logic reduction," in *Proc. of DATE'16*, 2016, pp. 1048–1053.

[24] R. Venkatesan, A. Agarwal, K. Roy, and A. Raghunathan, "Macaco: Modeling and analysis of circuits for approximate computing," in *Proc. of ICCAD'11*, 2011, pp. 667–673.

[25] S. Chakraborty, K. S. Meel *et al.*, "Approximate probabilistic inference via word-level counting," in *Proc. of AAAI'16*, 2016, pp. 3218–3224.

[26] C.-S. Han and J.-H. R. Jiang, "When boolean satisfiability meets gaussian elimination in a simplex way," in *CAV'12*, 2012, pp. 410–426.

[27] J. F. Miller, *Cartesian Genetic Programming*, 2011.

[28] R. Brayton and A. Mishchenko, "ABC: An academic industrial-strength verification tool," in *Proc. of CAV'10*, ser. LNCS, 2010, pp. 24–40.

Paper V

Towards Low Power Approximate DCT Architecture for HEVC Standard

VAŠÍČEK Zdeněk, MRÁZEK Vojtěch and SEKANINA Lukáš

In: *Proc. of the 2017 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. Lausanne: European Design and Automation Association, 2017, pp. 1576-1581. ISBN 978-3-9815370-9-3.

Towards Low Power Approximate DCT Architecture for HEVC Standard

Zdenek Vasicek, Vojtech Mrazek and Lukas Sekanina
Brno University of Technology, Faculty of Information Technology, IT4Innovations Centre of Excellence
Brno, Czech Republic
Email: {vasicek, imrazek, sekanina}@fit.vutbr.cz

***Abstract*—Video processing performed directly on IoT nodes is one of the most performance as well as energy demanding applications for current IoT technology. In order to support real-time high-definition video, energy-reduction optimizations have to be introduced at all levels of the video processing chain. This paper deals with an efficient implementation of Discrete Cosine Transform (DCT) blocks employed in video compression based on the High Efficiency Video Coding (HEVC) standard. The proposed multiplierless 4-input DCT implementations contain approximate adders and subtractors that were obtained using genetic programming. In order to manage the complexity of evolutionary approximation and provide formal guarantees in terms of errors of key circuit components, the worst and average errors were determined exactly by means of Binary decision diagrams. Under conditions of our experiments, approximate 4-input DCTs show better quality/power trade-offs than relevant implementations available in the literature. For example, 25% power reduction for the same error was obtained in comparison with a recent highly optimized implementation.**

I. Introduction

Small embedded systems connected to Internet of Things (IoT) are expected to be a new infrastructure of the information society. These systems range from simple smart sensors to advanced embedded applications requiring considerable computing resources. Video processing performed directly on IoT nodes is one of the most performance as well as energy demanding applications. In order to support real-time coding and decoding of high definition video on low cost devices, energy consumption optimization has to be introduced at all levels of the video processing system.

One of the most frequently performed and thus energy demanding operations is the Discrete Cosine Transform (DCT) block. In commonly optimized DCT blocks, bit width of all operations is reduced as much as possible, multipliers are replaced with additions, subtractions and shifts, and less important logic is pruned in order to reduce power consumption [1]. As video processing is, in principle, an error resilient application, DCT can further be approximated. Various approaches to the DCT approximation will be reported in Section II.

This paper deals with a deep optimization and approximation of the DCT block employed in the state of the art High Efficiency Video Coding (HEVC) standard [2]. In addition to the common optimization techniques, we propose to approximate adders and subtractors of a multiplierless DCT in such a way that the resulting error is kept under a predefined threshold. As the error of addition/subtraction is computed formally, without applying circuit simulation, it is always guaranteed that the target error is never exceeded. The error calculation procedure is based on relaxed equivalence checking using Binary Decision Diagrams (BDDs). BDDs are also used to analyze power consumption during the approximation process.

The approximate adders and subtractors are generated using a genetic programming-based approximation method. The so-called functional gate-level approximation is, in fact, performed in which accurate logic circuits are gradually simplified while an acceptable error is tolerated. Various approximate implementations of DCT showing good trade-offs between the quality of video processing and power consumption were obtained. These implementations were employed in reference HEVC implementation in order to compare them with available relevant implementations and determine their impact on the quality of video processing on selected benchmark video sequences. For example, 25% power reduction for the same error was obtained in comparison with a recent highly optimized implementation [1].

The rest of the paper is organized as follows. Section II briefly surveys relevant approximate circuit design approaches. Section III is devoted to the principles of DCT and its efficient implementation. The proposed approximation method for adders and subtractors is presented in Section IV. After presenting the experimental setup in Section V, results are reported and discussed in Section VI. Conclusions are given in Section VII.

II. Related Work

Efficient implementations of signal processing blocks in general and DCT in particular have been developed for decades (see the introduction to DCT in Section III). This effort has recently led to establishing a new High Efficiency Video Coding standard. Profiling results of HEVC given in [3] indicate that there is a good potential for improving energy and implementation efficiency especially of forward as well as inverse DCT blocks. These improvements can, in fact, be considered as approximations introduced at various levels of the algorithm and its implementation. They primarily include employing the integer data representation, bit width reduction, multiplierless multiplication and coefficient approximation [3], [1].

A. DCT as a Test Problem for Approximate Computing

However, with the development of the approximate computing paradigm, more radical approximations have been proposed. DCT blocks of JPEG and MPEG often serve as one of several test circuits used to evaluate the quality of general purpose approximation methods which usually perform the optimization and approximation at the gate level; see, for example, SALSA [4], ASLAN [5] and the logic isolation based approximation method [6]. In paper [7], several simplified transistor-level implementations of a 1 bit full adder were introduced and used in approximate adders that are employed in DCT of the JPEG implementation. Raha et al. proposed a power efficient video encoder in which all the adders and subtractors of the motion estimation and DCT blocks were replaced by their approximate configurable versions. The method enabled to automatically adjust a degree of hardware approximation dynamically based on the video characteristics [8]. A very specific optimization approach exploiting the sensitivity error analysis and error variance propagation in the DCT graph was developed in [9]. With respect to the available budget, the optimal number of 1 bit adders that should be approximated in all adders and multipliers was computed using the mixed integer nonlinear problem solver. The determined 1 bit adders were then replaced by their approximate versions taken from [7].

With respect to the approach developed in this paper, the aforementioned methods show two main drawbacks. Firstly, they do not deal with DCT intended for HEVC. Secondly, the papers show that DCT can be approximated, but the results are not compared against other DCT approximation methods.

B. Approximate Adders

As the proposed approach is based on approximate adders, this subsection briefly surveys this subarea of approximate computing. Adders are approximated by either general-purpose approximation methods or problem-specific methods. In the former case, the adders serve as one of many circuit classes that can be approximated by methods such as [4], [6] or multiobjective genetic programming-based methods [10]. Problem-specific methods exploit the structure of conventional adders. Another class of circuits are quality configurable adders (e.g. GeAR adders) which allow for a dynamic control of the error-power trade-off [11].

Four types of approximate adders are considered and evaluated in [12]: (1) Speculative adders in which the carry is speculated for each sum bit using only one or several bits. (2) Segmented adders, where an n-bit adder is divided into k-bit sub-adders and the carry is then generated by using different methods. (3) Carry-select adders in which multiple sub-modules are used to compute the sum for different carry values, and the result is determined according to the carry of a sub-module. (4) Approximate 1 bit full adders where the full adder is approximated at the transistor level and used as a building block of more complex adders [7].

C. Error Calculation

Almost all circuit approximation methods compute the error by circuit simulation. An open question is how many test vectors have to be applied in order to obtain trustworthy results. Only a few papers deal with formal analysis of candidate approximate circuits to establish the error and provide formal guarantees on the error bound. Checking the worst error can be based on satisfiability (SAT) solving as demonstrated in [13]. Binary decision diagrams (BDDs) were used to obtain the average arithmetic error, worst error, error rate [14] and average Hamming distance [15]. However, the formal methods are currently inapplicable for determining some types of errors for certain classes of circuits, e.g. the average error for non-trivial multipliers.

III. DCT AND ITS APPROXIMATION

A. Principle of DCT

For a given input vector $\mathbf{X} = [\mathbf{x_0}, \mathbf{x_1}, \ldots, \mathbf{x_{N-1}}]^\mathbf{T}$, its corresponding DCT output $\mathbf{Y} = [\mathbf{y_0}, \mathbf{y_1}, \ldots, \mathbf{y_{N-1}}]^\mathbf{T}$ is given by $\mathbf{Y} = \mathbf{C_N X}$, where $\mathbf{C_N}$ denotes the N-point DCT kernel. In general, $\mathbf{C_N}$ consists of real numbers. The elements $c_{ij} \in \mathbf{C_N}$ are defined as $c_{ij} = \frac{A}{\sqrt{N}} \cos[\frac{\pi}{N}(j + \frac{1}{2})i]$, where $i, j = 0, \ldots, N-1$. A is equal to 1 and $\sqrt{2}$ for $i = 0$ and $i > 0$ respectively.

HEVC standard utilizes 4-point, 8-point, 16-point and 32-point DCT in its forward (video coder) as well as inverse (video decoder) form. In order to simplify the complexity of hardware circuits computing the output of DCT, HEVC scales the coefficients by a power of two and rounds them to the integer value. The scaling and rounding was optimized to preserve the main advantages and properties of DCT such as the orthogonality of basis vectors, equal norm of basis vectors and good compression efficiency on the one hand as well as the low complexity of hardware circuits on the other hand [16].

Let us restrict ourself to a 4-point 1D forward transform whose output is defined as follows:

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} = \mathbf{C_4} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 64 & 64 & 64 & 64 \\ 83 & 36 & -36 & -83 \\ 64 & -64 & -64 & 64 \\ 36 & -83 & 83 & -36 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad (1)$$

The output of DCT could be determined by means of straightforward matrix multiplication. This approach, however, requires to perform N^2 multiplications and $N(N-1)$ additions. The costly matrix multiplication could be avoided by utilizing the symmetry properties of basis vector. A decomposition technique known as Even-Odd decomposition [16] could be employed to reduce the computational complexity of DCT. The 4-point DCT given by Eq. 1 can then be rewritten to

$$\begin{bmatrix} y_0 \\ y_2 \end{bmatrix} = \mathbf{C_2^o} \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} 64 & 64 \\ 64 & -64 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \end{bmatrix}$$
$$\begin{bmatrix} y_1 \\ y_3 \end{bmatrix} = \mathbf{C_2^e} \begin{bmatrix} b_0 \\ b_1 \end{bmatrix} = \begin{bmatrix} 83 & 36 \\ 36 & -83 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \end{bmatrix} \quad (2)$$

where $a_0 = x_0 + x_3$, $a_1 = x_1 + x_2$, $b_0 = x_0 - x_3$, $b_1 = x_1 - x_2$. This gives us a hardware architecture shown in Fig. 1. In the first part (I.), the partial sums a_i and b_i are calculated using two w-bit adders and two w-bit subtractors, both producing a $(w+1)$-bit signed integer value. In the second part (II.), multiplication is performed. Finally, the multiplied values are summed up by two adders and two subtractors (part III.).

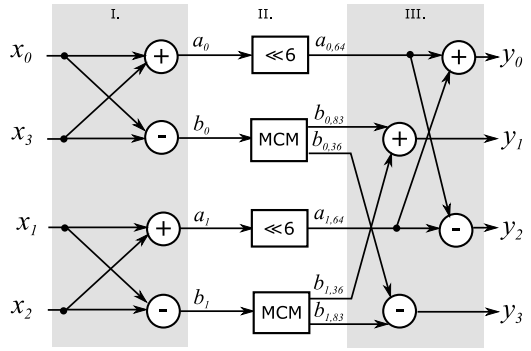

Fig. 1: Architecture of 4-point integer DCT.

In order to reduce the complexity of adder (subtractor) computing the output y_0 (y_2), the multiplication by factor 64 which is performed by means of a simple logic shift may be postponed and executed after summing (subtracting) the partial sums a_0 and a_1. This simple modification helps to reduce the bitwidth of adder (subtractor) by six. Both components operate with two $(w+1)$-bit integers.

As the coefficients utilized in Eq. 2 are constant, we can avoid the usage of costly multipliers and employ a much efficient approach – multiplierless multiple constant multiplier (MCM) which determines the value of $36 \cdot b_i$ and $83 \cdot b_i$ efficiently. MCM is a hardware block that exclusively consists of additions, subtractions, and shifts.

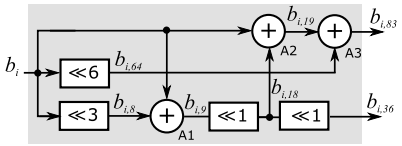

Fig. 2: Multiplierless multiple constant multiplier (83, 36)

The most compact implementation of MCM for 83 and 36 is shown in Fig. 2. The MCM has a single input denoted b_i and two outputs $b_{i,83}$ and $b_{i,36}$. According to Fig. 1, b_i is an $(w+1)$-bit signed integer. The MCM consists of three adders arranged in such a way that they have a minimal possible bitwidth. One $(w+4)$-bit signed adder (A1) which adds $(w+1)$-bit and $(w+4)$-bit signed values is shared between the subcircuit outputting $83 \cdot b_i$ and the subcircuit outputting $36 \cdot b_i$. The remaining two adders (i.e. $(w+6)$-bit A2 and $(w+7)$-bit A3) are employed to determine $83 \cdot b_i$.

B. The Proposed Approximation of DCT

DCT can be approximated by reducing the number of utilized adders (subtractors) and/or employing approximate

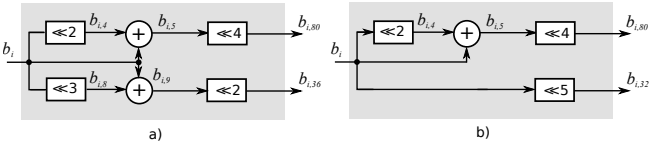

Fig. 3: Two variants of approximate MCM (83, 36)

adders (subtractors). In this paper, we investigate the impact of both approaches.

The number of components can be reduced, for example, by approximating the coefficients of the matrix $\mathbf{C_2^e}$ shown in Eq. 2. It means, in fact, that we replace the MCM blocks employed in the architecture shown in Fig. 1 with different ones. The selection of the proper coefficients is nontrivial as it is necessary to consider not only the distance between the original and approximate coefficients, but also the number of components and the bit-width of the components. In this case, we exhaustively enumerated all possible MCM implementations with two coefficients that are within the range of +-16. We obtained three architectures representing the best trade-off among the considered parameters. Two of them are shown in Fig. 3. In the first case (Figure 3a), coefficient 83 is replaced with coefficient 80. This small difference helps to reduce the number of adders by one. In addition to that, the bit-width of the adder and subtractor combining the outputs of MCM block in DCT can be reduced by 2 because the shift by 2 can be postponed. When the coefficient 32 is used instead of the coefficient 36, we can remove another adder (see Figure 3b). This helps to reduce the bit-width by 4.

The third architecture, as proposed in [1], utilizes MCM with coefficient 64 and 32. i.e. the multiplication is replaced by the shifts. The bit-width is reduced by 5.

IV. Design of approximate adders and subtractors

Several approaches have been proposed to approximate arithmetic circuits. In this work, we employed Cartesian Genetic Programming (CGP) [17] because it can easily handle constraints given on candidate circuits, the method is naturally multi-objective and high-quality approximate circuits have already been obtained with it [18].

The standard CGP is a branch of genetic programming which represents candidate designs using directed acyclic graphs. A candidate circuit is modeled using an array of programmable nodes. In our case, 2-input Boolean functions are considered. The circuit utilizes n_i primary inputs and n_o primary outputs. Feedback connections are not enabled.

The primary inputs and the outputs of the nodes are labeled $0, 1 \ldots N$ and considered as addresses which the node inputs can be connected to. A candidate solution is represented in the so-called chromosome (which is, in fact, a netlist) by $N - n_i$ triplets (x_1, x_2, ψ) determining for each node its function ψ ($\psi \in \Gamma$ where Γ is the set of available functions) and input connections. The last part of the chromosome contains n_o integers specifying the nodes where the primary outputs are connected to.

CGP employs a simple search method in which the initial population P contains at least one implementation of the accurate circuit (adder or subtractor) and a few circuits generated using mutation of the accurate circuit. The next step consists of the evaluation of candidate circuits using the fitness function. Each member of P then receives the so-called fitness score and the highest-scored individual becomes a new parent of the next population. From this parent, λ candidate solutions are generated using mutation. The termination criterion is given by the maximum number of iterations.

A. Error metrics

In order to design signed approximate adders and subtractors exhibiting suitable parameters, the worst-case arithmetic error and average-case arithmetic error need to be kept under some level. In our approach, we employed the worst-case arithmetic error as a design constrain and average-case arithmetic error as a design objective.

The error metrics are determined using BDDs as follows. For each candidate solution, we create a virtual circuit that is subsequently represented using BDDs. The virtual circuit (shown in Fig. 4) takes a $2n$-bit input vector x and produces absolute difference between the $(n+1)$-bit output of accurate adder (subtractor) $f(x)$ and approximate adder (subtractor) $f'(x)$. Subtraction in virtual circuit is calculated using $m = n+2$ full-adders with first carry-in set to 1 and inverting each bit of the subtrahend. The absolute value is computed using $m-1$ half-adders and $m-1$ XOR gates.

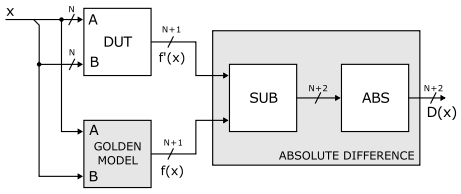

Fig. 4: Virtual circuit used for arithmetic error analysis

The difference $D(x)$ is represented by m-bit binary vector which corresponds with a natural number. It holds that $D(x) = \sum_{0 \leq i < m} d_i(x) \cdot 2^i$. Then, Algorithm 1 is used for the worst case error analysis.

Algorithm 1: worst-case error analysis

Input: BDD representation of the virtual circuit (d)
Output: The maximum arithmetic error (ε_{max})

$\varepsilon_{max} \leftarrow 0$, $\mu \leftarrow true$;
for $i \in \{m-1, m-2, \ldots, 0\}$ **do**
  if *satisfiable*($\mu \wedge d_i$) **then**
    $\mu \leftarrow \mu \wedge d_i$; $\varepsilon_{max} \leftarrow \varepsilon_{max} + 2^i$;
return ε_{max};

The average-case arithmetic error can be obtained by m calls of SATcount operation (one per each bit of D) which determines the number of input assignments that evaluates d_i to one.

Algorithm 2: average-case error analysis

Input: BDD representation of the virtual circuit (d)
Output: The average arithmetic error (ε_{avg})

$\varepsilon_{avg} \leftarrow 0$;
for $i \in \{m-1, m-2, \ldots, 0\}$ **do**
  $\varepsilon_{avg} \leftarrow \varepsilon_{avg} + 2^{i-2n} \cdot satcount(d_i)$;
return ε_{avg};

B. Fitness function

In this paper, the following fitness function is utilized:

$$fitness(A) = -\begin{cases} Pwr(A) & \text{if } \varepsilon_{max}(A) \leq E \\ \infty & \text{otherwise} \end{cases},$$

where A represents a candidate circuit, E the maximum acceptable error level and $Pwr(A)$ is the power consumption of A. In order to estimate the power consumption of a candidate circuit, we adopted a method based on the switching activity estimation introduced in [10].

For each candidate circuit, a virtual circuit is constructed and represented using BDDs. The construction of BDD starts with a candidate circuit denoted as DUT in Fig. 4. At the end of this step, BDD for each output bit of $f'(x)$ is available. At this point, transition probabilities are determined for each gate of DUT. These probabilities are employed to determine the switching activity and subsequently the power consumption. Then, the rest of the virtual circuit is created and worst-case error ε_{max} is analyzed using the Algorithm 1.

V. Experimental setup

We will evaluate the impact of the 4-point DCT (with bit-width-optimized adders) approximations on the quality of video processing and power consumption. The accurate implementation of 4-point DCT (denoted as A1) requires five different adders and two different subtractors (see the discussion in Section III-A). In addition to A1, three approximate architectures are considered: architecture A2 which utilizes MCM from Fig. 3a, A3 employing MCM from Fig. 3b and finaly A4 which replaces MCM with logic shifts. As the MCMs presented in Fig. 3 contain shifts that can be moved after the adders and subtractors situated in the third part of DCT architecture, six additional adders and three subtractors needs to be designed. In total, we need to implement 16 different circuits (11 adders and 5 subtractors) to construct four considered DCT architectures.

The goal of CGP is to design various approximate implementations of each circuit. We considered seven error levels defining the maximum acceptable worst-case error, in particular $E \in \{0.1\%, 0.2\%, 0.5\%, 1\%, 2\%, 5\%, 10\%\}$. Note that 100% corresponds with $\varepsilon_{max} = 2^{w+1}$, where w is the circuit's bit-width. For each E, 20 independent CGP runs were executed with parameters: $\lambda = 5$, $\Gamma = \{BUF, AND, OR, XOR, NAND, NOR, XNOR\}$. CGP is

TABLE I: Average time needed to perform error analysis for w-bit adders/subtractors

	$w=4$	$w=8$	$w=12$	$w=16$
simulation	4.5 μs	1.9 ms	682.4 ms	140.90 s
BDD ε_{max}	10.3 μs	3.5 ms	127.9 ms	1.38 s
Speedup	0.43×	0.54×	5.33×	102.30×
BDD ε_{avg}	14.0 μs	4.6 ms	312.7 ms	2.93 s
Speedup	0.32×	0.42×	2.18×	48.09×

terminated when 10^6 generations are exhausted or when its duration exceeded 120 minutes.

We obtained 140 different solutions for each circuit and 2240 solutions in total. For each E and each circuit, we identified a single solution exhibiting the best trade-off between the power consumption and average-case error (the exact average-case error was calculated using Algorithm 2). Then, the chosen circuits were utilized to create 7 approximate variants for each of four considered DCT architectures.

We obtained 32 different implementations of DCT whose parameters were subsequently evaluated. Firstly, we implemented these architectures in Verilog, synthesized in ABC and measured their power consumption. Secondly, we implemented them in C language and employed in reference implementation of HEVC[1] where we replaced the code performing the 4-point DCT with our implementations of DCT.

VI. Results

The results of performance analysis of the BDD-based method calculating the error of approximate adders (subtractors) are summarized in Table I. We measured the average time needed to perform the whole error analysis when employed in evolutionary loop. We run CGP for 10 minutes to obtain statistically significant results.

The proposed BDD-based technique is compared with an optimized approach based on exhaustive simulation. As evident, the BDD-based method performs better for more complex circuits (adders having at least 12-bits, i.e. 24 inputs). This result was expected as the simulation algorithm is able to evaluate circuit response for up to 256 different input vectors in one pass using modern CPUs. For example, for a 16-bit adder, the achieved speedup is greater than 100 in the case of the worst-case error analysis. The average-case error analysis is approximately two times slower compared to the worst-case error analysis. Despite this, the speedup greater than 48 was achieved for 16-bit adder when compared with the exhaustive simulation.

Figure 6 shows results of evolutionary approximation of one circuit instance – 16-bit signed adder. According to the boxplots size, the proposed evolutionary method provides relative stable results independently of the number of executed evolutionary runs. It is evident that the power consumption

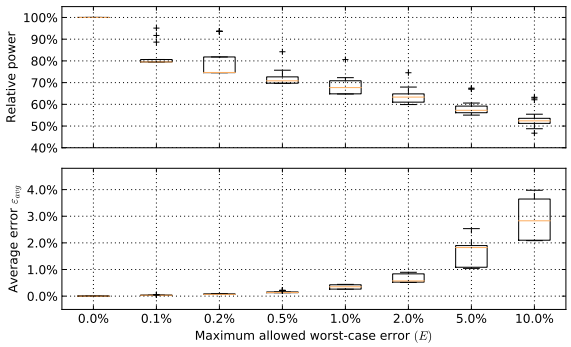

Fig. 6: Parameters of evolved approximate 16-bit adders for various error levels. For each E, 20 solutions are considered.

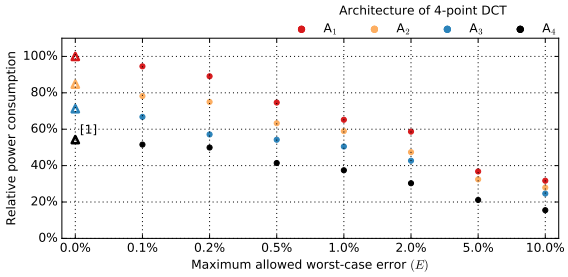

Fig. 7: Parameters of various DCT architectures constructed using the evolved adders and subtractors. The triangles represent the implementations employing the accurate components.

decreases when error E is increasing. Naturally, the average-case error follows the opposite trend.

Parameters of all (i.e. 32) obtained implementations of 4-point DCT are summarized in Figure 7. The architectures utilizing accurate adders and subtractors (symbol A1, $E=0$) exhibit the highest power consumption. The power consumption decreases with increasing number of adders removed from MCM as well as decreasing bit-width of adders and subtractors placed in the third part of DCT architecture. As a consequence of that, architecture A4 shows a 45% reduction in power consumption compared to A1. When we compare the power consumption of seven approximate versions of each architecture, we can see that power consumption decreases with increasing E. For $E=10\%$, the architectures exhibit very similar power consumption.

The impact of the approximate DCT on the quality of obtained video sequences was evaluated on six common test video sequences[2]. Results for three of them are shown in Figure 5. For each video sequence, we measured the peak signal-to-noise ratio (PSNR) between the original frames and frames encoded and subsequently decoded by HEVC codec. To evaluate PSNR, 100 video frames were used. When we compare parameters of the architectures utilizing the accurate adders and subtractors (see triangles in Figure 5), it is evident

[1]Reference software for ITU-T H.265 high efficiency video coding available at https://www.itu.int/rec/T-REC-H.265.2

[2]YUV video sequences available at http://trace.eas.asu.edu/yuv/

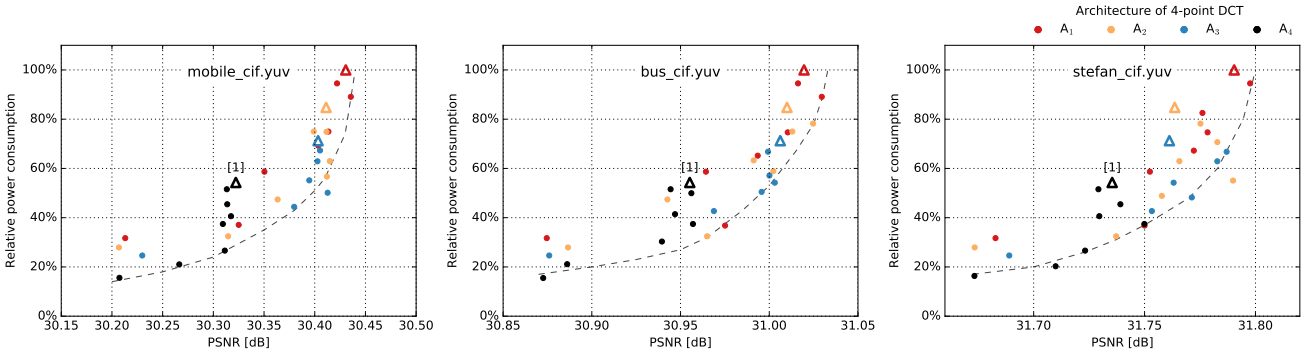

Fig. 5: The quality expressed in terms of average PSNR between original and encoded video sequences and relative power consumption evaluated for three common video sequences. Each point corresponds with one implementation of 4-point DCT. The triangles represent the implementations employing the accurate components.

that A4 proposed in [1] provides the worst results. In all cases, A2 provides better results than A3 and A3 provides better results than A4. The difference in quality between A2 and A3 is very small, however, A3 is able to achieve more than 30% power reduction. When approximate adders and subtractors are introduced to DCT, the dependency between the power consumption and quality is more complex. It happens, for example, that A1 with approximate adders/subtractors exhibiting 0.1% worst-case error provides better PSNR then the exact A1. We hypothesize that this phenomenon is related to the construction of DCT coefficient values in the reference HEVC implementation.

VII. Conclusions

We proposed a new method for optimization and approximation of the DCT block employed in the HEVC standard. The proposed multiplierless DCT implementations contain approximate adders and subtractors that were obtained using CGP. In order to manage the complexity of evolutionary approximation, the worst and average errors were determined by means of BDDs. Under conditions of our experiments, evolved implementations show better quality/power trade-offs than relevant implementations available in the literature. Our future work will be devoted to applying the proposed approximation method on other blocks of HEVC in order to find even better quality/power trade-offs.

Acknowledgments

This work was supported by Czech science foundation project GA16-17538S.

References

[1] M. Jridi and P. Meher, "A scalable approximate dct architectures for efficient HEVC compliant video coding," *IEEE Trans. Circuits Syst. Video Technol*, pp. 1–10, 2016.

[2] G. J. Sullivan, J. R. Ohm *et al.*, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Trans. Circuits Syst. Video Technol*, vol. 22, no. 12, pp. 1649–1668, 2012.

[3] F. Bossen, B. Bross *et al.*, "HEVC complexity and implementation analysis," *IEEE Trans. Circuits Syst. Video Technol*, vol. 22, no. 12, pp. 1685–1696, 2012.

[4] S. Venkataramani, A. Sabne, V. J. Kozhikkottu, K. Roy, and A. Raghunathan, "SALSA: systematic logic synthesis of approximate circuits," in *The 49th Annual Design Automation Conference 2012, DAC'12*. ACM, 2012, pp. 796–801.

[5] A. Ranjan, A. Raha, S. Venkataramani, K. Roy, and A. Raghunathan, "ASLAN: Synthesis of approximate sequential circuits," in *Proc. Conference on Design, Automation and Test in Europe*, ser. DATE'14. EDA Consortium, 2014, pp. 1–6.

[6] S. Jain, S. Venkataramani, and A. Raghunathan, "Approximation through logic isolation for the design of quality configurable circuits," in *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2016, pp. 612–617.

[7] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, "Low-power digital signal processing using approximate adders," *IEEE Trans. on CAD of Integr. Circuits and Systems*, vol. 32, no. 1, pp. 124–137, 2013.

[8] A. Raha, H. Jayakumar, and V. Raghunathan, "A power efficient video encoder using reconfigurable approximate arithmetic units," in *2014 27th International Conference on VLSI Design and 2014 13th International Conference on Embedded Systems*, 2014, pp. 324–329.

[9] F. S. Snigdha, D. Sengupta, J. Hu, and S. S. Sapatnekar, "Optimal design of JPEG hardware under the approximate computing paradigm," in *2016 53nd ACM/EDAC/IEEE Design Automation Conf. (DAC)*, 2016, pp. 1–6.

[10] R. Hrbacek, V. Mrazek, and Z. Vasicek, "Automatic design of approximate circuits by means of multi-objective evolutionary algorithms," in *Proc.of the 11th Int. Conf. on Design and Technology of Integrated Systems in Nanoscale Era*. IEEE, 2016, pp. 239–244.

[11] M. Shafique, W. Ahmad, R. Hafiz, and J. Henkel, "A low latency generic accuracy configurable adder," in *Proc. 52nd Annual Design Automation Conference*. ACM, 2015, pp. 86:1–86:6.

[12] H. Jiang, J. Han, and F. Lombardi, "A comparative review and evaluation of approximate adders," in *Proc. 25th Edition on Great Lakes Symposium on VLSI*. ACM, 2015, pp. 343–348.

[13] R. Venkatesan, A. Agarwal, K. Roy, and A. Raghunathan, "MACACO: Modeling and analysis of circuits for approximate computing," in *2011 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2011, pp. 667–673.

[14] M. Soeken, D. Grosse, A. Chandrasekharan, and R. Drechsler, "BDD minimization for approximate computing," in *21st Asia and South Pacific Design Automation Conf. ASP-DAC 2016*. IEEE, 2016, pp. 474–479.

[15] Z. Vasicek and L. Sekanina, "Evolutionary design of complex approximate combinational circuits," *Genetic Programming and Evolvable Machines*, vol. 17, no. 2, pp. 1–24, 2016.

[16] M. Budagavi, A. Fuldseth, G. Bjontegaard, V. Sze, and M. Sadafale, "Core transform design in the high efficiency video coding (hevc) standard," *IEEE J. Sel. Topics Signal Process.*, vol. 7, no. 6, pp. 1029–1041, Dec 2013.

[17] J. F. Miller, *Cartesian Genetic Programming*. Springer-Verlag, 2011.

[18] Z. Vasicek and L. Sekanina, "Evolutionary approach to approximate digital circuits design," *IEEE Trans. Evol. Comput.*, vol. 19, no. 3, pp. 432–444, 2015.

Paper VI

The Role of Circuit Representation in Evolutionary Design of Energy-Efficient Approximate Circuits

MRÁZEK Vojtěch, VAŠÍČEK Zdeněk and HRBÁČEK Radek

IET Computers & Digital Techniques. Stevenage: The Institution of Engineering and Technology, 2018. ISSN 1751-8601.

IET Computers & Digital Techniques

Research Article

IET Journals
The Institution of Engineering and Technology

Role of circuit representation in evolutionary design of energy-efficient approximate circuits

ISSN 1751-8601
Received on 13th September 2017
Revised 20th March 2018
Accepted on 24th April 2018
doi: 10.1049/iet-cdt.2017.0188
www.ietdl.org

Vojtech Mrazek[1] ✉, *Zdenek Vasicek*[1], *Radek Hrbacek*[1]

[1]*Faculty of Information Technology, Brno University of Technology, Centre of Excellence IT4Innovations, Czech Republic*

✉ *E-mail: imrazek@fit.vutbr.cz*

Abstract: Circuit approximation has been introduced in recent years as a viable method for constructing energy-efficient electronic systems. An open problem is how to effectively obtain approximate circuits showing good compromises between key circuit parameters – the error, power consumption, area and delay. The use of evolutionary algorithms in the task of circuit approximation has led to promising results. Unfortunately, only relatively small circuit instances have been tackled because of the scalability problems of the evolutionary design method. This study demonstrates how to push the limits of the evolutionary design by choosing a more suitable representation on the one hand and a more efficient fitness function on the other hand. In particular, the authors show that employing full adders as building blocks leads to more efficient approximate circuits. The authors focused on the approximation of key arithmetic circuits such as adders and multipliers. While the evolutionary design of adders represents a rather easy benchmark problem, the design of multipliers is known to be one of the hardest problems. The authors evolved a comprehensive library of energy-efficient 12-bit multipliers with a guaranteed worst-case error. The library consists of 65 Pareto dominant solutions considering power, delay, area and error as design objectives.

1 Introduction

In recent years, a new research field was established to investigate how computer systems can be made more energy efficient, faster and less complex by relaxing the requirement that they are correct. This field, denoted as *approximate computing*, exploits the fact that many applications are error resilient and the errors in computing are thus either invisible or acceptable [1]. The concept of approximation has intensively been studied, developed and applied not only in computer science, but also in mathematics and engineering disciplines. However, it has never been applied in the areas in which only accurate implementations have traditionally been accepted. Nowadays, the designers intentionally introduce errors into computation to satisfy the never-ending requirement for lowering of power consumption.

As one of the most promising energy-efficient computing paradigms that is able to cope with current challenges of computer engineering, approximate computing has gained a lot of research attention in the past few years. We can identify two main directions in approximate computing: energy-efficient computing with unreliable components and approximation of systems implemented on common platforms [1]. In the first case, the problem is that the exact computation utilising nanometre transistors provided by recent technology nodes is extremely expensive in terms of energy requirements and reliable behaviour. An open question is how to effectively and reliably compute with a huge number of unreliable components. The second research direction is motivated by the fact that many applications (typically in the areas of multimedia, graphics, data mining and big data processing) are inherently *error resilient*. This resilience can be exploited in such a way that the error is exchanged for improvements in power consumption, throughput or implementation cost. After analysing many applications, Chippa *et al.* [2] reported that about 83% of the runtime is spent in computations that can be approximated.

Various approximation techniques have been proposed recently. A good survey of the proposed approaches can be found, for instance, in [1, 3]. According to the level of the computer stack where the approximations are conducted, the approaches could be roughly divided into software level and hardware level. At the software level, for example, we could selectively ignore certain computations and/or memory accesses that are not critical for obtaining the desired quality of the result. At the hardware layer, we could either use a less accurate yet more energy-efficient circuit for computation or purposely reduce the supply voltage for certain hardware components to trade-off energy and accuracy.

As the complexity of today's computer systems grows, the manual approximation is not an efficient design method. Hence, several automated approximate design methods have been introduced. The design of approximate circuits is typically based on modifying fully functional circuits. Venkataramani *et al.* [4], for example, uses a quality function which decides whether a predefined quality constraint is met or not. The algorithm is allowed to modify the circuit as long as the quality constraint is not violated. Among others, 32-bit adders, 8-bit multipliers, finite impulse filter (FIR) filters and discrete cosine transform (DCT) blocks were approximated. Another approach looks for signal pairs having similar values with a high probability. By substituting one signal for the other, a part of the circuit can be removed resulting in area and power savings at the cost of an error introduced to the output [5]. Unlike the aforementioned methods, Nepal *et al.* [6] proposed an approach operating directly on the behavioural descriptions of circuits. His method generates approximate circuits from input behavioural descriptions by performing global transformations on an abstract synthesis tree created from the behavioural description. The outcome approximate circuits are still expressed in behavioural code and can be synthesised by means of standard synthesis tools. The main weakness of these design methods, however, is that they are typically able to produce only a few design points.

Several papers dealing with the evolutionary design of approximate circuits have been published. One of the seminal work on this topic is the paper of Sekanina *et al.* who addressed the problem of evolutionary design of small approximate circuits consisting of elementary gates [7]. Later, Vasicek and Sekanina [8] introduced heuristic seeding and demonstrated how to improve the scalability of the evolutionary design of approximate circuits. The proposed method was applied to the evolution of 4-bit multipliers and 25-input median circuits. Recently, Mrazek *et al.* [9] utilised evolutionary approach to evolve energy-efficient 8-bit approximate multipliers optimised for the usage in artificial neural networks. In addition to that, multi-objective design of 8-bit approximate multipliers was addressed by Hrbacek *et al.* [10]. Apart from the approximate design of median circuits and work of Hrbacek, the

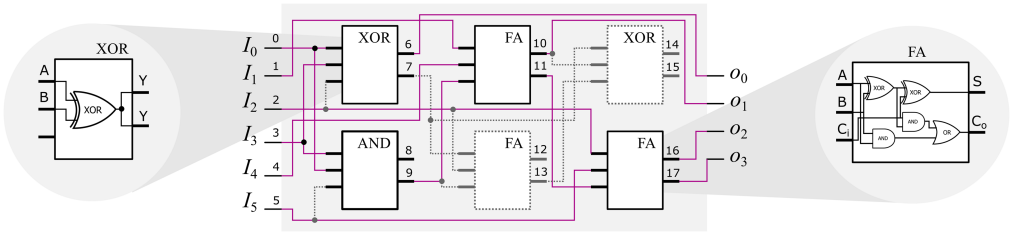

Fig. 1 *Example of a circuit (3-bit ripple-carry adder) encoded using CGP with parameters:* $n_i = 6$, $n_o = 4$, $n_a = 3$, $n_b = 2$, $n_c = 3$, $n_r = 2$, $\Gamma = \{0^{and}, 1^{or}, 2^{xor}, 3^{full-adder}\}$. *Given these parameters, the considered circuit can be represented using a chromosome consisting of the following sequence of 28 integers: 0, 3, 2, 2; 3, 0, 5, 0; 1, 4, 9, 3; 7, 2, 9, 3; 7, 10, 13, 2; 2, 5, 11, 3; 6, 10, 16, 17. Two nodes with outputs 12, 13, 14 and 15 are not used (top-right XOR gate and full adder). These redundant and inactive nodes as well as the redundant wires are greyed out. The common two-input logic gates are implemented as shown on the left – only two inputs are employed, the third input is ignored, both outputs implement the same logic function. More complex blocks (full adder in this example) may utilise all inputs and may capture more Boolean functions (two functions are implemented in case of the full adder). The principle of the CGP encoding is as follows. The first 24 integers are divided into six quadruples that define input connections and function of each of six CGP nodes. The first quadruple (0, 3, 2, 2) is associated with the top-left node. The first three numbers of each quadruple determine indexes where the node's inputs are connected. The last (underlined) number defines the function of a node. Considering the first quadruple, the node represents XOR gate connected to* I_0, I_3 *and* I_2. *However, the third connection, i.e.* I_2, *is in fact redundant because XOR is a two-input logic gate. The last four numbers of the chromosome, i.e. the numbers 6, 10, 16 and 17, define the connection of the primary outputs. The first primary output* O_0 *associated with the first number is connected to the first output of the top-left node because this node output has index 6*

authors represent the circuits by means of basic logic gates. There is no work that investigates whether there is a better representation that may improve the performance of the evolutionary algorithm or the quality of the obtained approximate circuits. In addition to that, the power consumption is typically optimised indirectly as the number of gates or the area on the chip (see e.g. [8, 9]).

In this paper, we present a comprehensive analysis which compares two different representations – the mainstream *gate-level representation* where we represent the circuits using common logic gates, and *cell-based representation* that utilises more complex building blocks such as full adders. Interestingly, the gate-level representation represents a routinely adopted approach since the late 1990s [11]. We hypothesise that evolution at the level of more complex cells could produce solutions of higher quality because the standard cells available in every technology library exhibit substantially better design parameters (area, power, delay) compared to the equivalent circuits implemented using standard gates. In order to confirm the validity of this claim, we applied evolutionary methods to the design of key arithmetic circuits such as adders and multipliers. In particular, 8-bit and 12-bit approximate circuits were considered. In order to support the evolution of 12-bit circuits, we implemented a state-of-the-art circuit simulator operating on 256 bits in parallel. The simulator, first introduced in [10], is employed to determine the quality of approximate circuits. In addition to that the switching activity is simultaneously calculated. A robust power estimation engine based on known switching activity represents a key idea how to ensure that the evolutionary approach produces energy-efficient solutions.

The contributions of this paper are as follows. This is the first time a detailed analysis of various representations of digital circuits is evaluated and discussed. We analyse also the impact of the chosen representation on results produced by a synthesis tool utilised to obtain a physical implementation of a given circuit. Finally, this is the first paper that presents an automatic approach that is able to produce high-quality approximate 12-bit multipliers with guaranteed error parameters. We obtained >60 Pareto dominant implementations that are available for download (http://www.fit.vutbr.cz/research/groups/ehw/approxlib). This result is encouraging for evolutionary computation community on the one hand and practically useful for hardware community on the other hand. The 8-bit and 12-bit approximate multipliers can be employed directly to improve the power efficiency of deep neural networks [1, 9] or as building blocks of complex circuits. Four 8-bit multipliers, for example, can be employed to construct a 16-bit approximate multiplier using the common approach of constructing larger multipliers from smaller ones [12].

The rest of this paper is organised as follows. The proposed method is introduced in Section 2. The design methodology followed by the analysis of obtained results is presented in Section 3. Finally, the conclusions are given in Section 4.

2 Proposed method

In order to approximate digital circuits, various approaches have been proposed [4–7]. In this work, we employ Cartesian genetic programming (CGP) [11]. CGP can easily handle constraints given on candidate circuits, the method is naturally multi-objective and high-quality approximate circuits have already been obtained with CGP [8, 10].

This section introduces the overall idea of the proposed method, the utilised evolutionary algorithm and the construction of the fitness function for the design of energy-efficient approximate circuits.

2.1 Representation of digital circuits

Standard CGP is a branch of genetic programming which represents candidate designs using directed acyclic graphs [11]. A candidate circuit is modelled using a two-dimensional (2D) array of programmable nodes with n_c columns and n_r rows. Originally, CGP with single-output programmable nodes was introduced by Miller in 1998 [13]. Simple nodes with two inputs and single output were considered in the evolution of digital circuits [13]. This approach, however, can be generalised to support nodes with arbitrary number of inputs and outputs. In this work, we use the extended version of CGP that supports nodes with n_a inputs and outputs [10]. This arrangement allows one to have a node evaluating up to n_b Boolean functions defined over n_a variables. The function of a node, however, cannot be arbitrary. Each node can implement one of n_G functions defined by Γ. The node parameters (i.e. n_a, n_b) are fixed during the evolution. In order to fully specify the behaviour of each node, we use the following principle. In case that a node implements a Boolean function, which utilises less than n_a operands, the redundant input connections are ignored. In case that a node implements less than n_b Boolean functions, the unused outputs are internally connected to the output of the last Boolean function. Let us suppose, for example, that a single output Boolean function is chosen from Γ. Then, all n_b outputs produce the same value (see e.g. schematics of XOR gate shown in Fig. 1 implemented using three-input two-output CGP node). This mechanism helps us to avoid the necessity to use chromosome validation and repair mechanisms.

The circuit utilises n_i primary inputs and n_o primary outputs. Feedback connections are not enabled. The primary inputs are labelled $0, 1, \ldots, (n_i - 1)$. The first output of the first CGP node is assigned index n_i, the second output $n_i + 1$ and the last output of

this node is associated with $n_i + n_b - 1$. The remaining outputs of the CGP nodes are successively labelled $(n_i + n_b), (n_i + n_b + 1), \ldots, (n_c \cdot n_r \cdot n_b + n_i - 1)$. All the labels are considered as addresses where the node inputs and primary outputs can be connected to. A candidate solution is represented by means of the so-called *chromosome* (which is, in fact, a netlist) by $n_r \cdot n_c$ tuples consisting of $n_a + 1$ items $(x_1, x_2, \ldots, x_{n_a}, \psi)$ determining for each node its function ψ ($\psi \in \Gamma$) and input connections x_i ($0 \leq x_i < n_c \cdot n_r \cdot n_b + n_i$). The last part of the chromosome contains n_o integers specifying the nodes where the primary outputs are connected to. While the chromosome size s is constant $s = n_c n_r (n_a + n_b) + n_o$, the circuit size is variable and measured as the number of active (i.e. used) nodes. The set of valid chromosomes (netlists) represents the whole search space.

The encoding used in CGP is highly redundant as many nodes can be disconnected and deactivated during evolutionary optimisation. In order to deactivate a node, it is sufficient to reconnect all active references to that node. Moreover, there are usually many ways to implement a given logic function in each CGP instance. This redundancy together with a relatively powerful mutation operator is considered as a key feature of CGP allowing for an efficient circuit evolution [11].

An example of a circuit (common 3-bit ripple carry adder) represented using CGP is shown in Fig. 1. Despite the fact that the adder can be represented using three three-input/two-output CGP nodes (three full adder cells), we employ six three-input/two-output CGP nodes arranged into three columns and two rows. This arrangement helps us to demonstrate the redundancy of CGP representation (only four out of six nodes are active) and the flexibility of the proposed encoding (a half adder is implemented using two logic gates, full adders are implemented as standard cells). In addition to that, the example shows that the chromosome captures a valid netlist even though the second output of the AND node (output with label 9) is connected to another node. As stated earlier, all the outputs of a node representing a single-output logic gate are equivalent. The corresponding chromosome is shown in Fig. 1.

2.2 Search strategy

CGP employs a simple search method. In our case, the initial population P of CGP contains one of various implementations of the accurate circuit (e.g. multiplier) and a few circuits generated using mutation of the accurate circuit. Creating the accurate multiplier required by the initial population is trivial as there is a one-to-one mapping between circuit netlists and CGP chromosomes. The next step consists in the evaluation of candidate circuits using the fitness function. Each member of P then receives the so-called fitness score and the highest-scored individual becomes the new parent of the next population. From this parent, λ candidate solutions are generated using mutation. The termination criterion is given by the number of iterations or maximum acceptable runtime.

Despite several attempts to propose a suitable crossover operator to CGP [14, 15], the mutation is still used as the crucial genetic operator. The mutation operator modifies up to h randomly chosen genes (integers) of the chromosome. Their new values are generated randomly, but it is checked whether the new values are valid. One mutation can affect either the node function, node input connection, or primary output connection. As the mutation operator is able to disconnect gates (by changing either primary output connection, node input connection or node function), it can be employed to reduce redundancy of the initial circuit.

2.3 Fitness function

Since the goal is to design energy-efficient approximate circuits, it is necessary to integrate this requirement into the fitness. The power consumption of the candidate circuits can be optimised directly or indirectly. In the context of evolutionary computation, only the latter approach has been applied in the literature because it does not require to implement complex simulation engines or employ time-demanding analogue simulations. In [8], for example, the authors demonstrated that it is sufficient to reduce the number of gates because the power consumption of arithmetic circuits is highly correlated with the area. For recent technology nodes, however, this simplification may lead to unsatisfactory results especially for circuits consisting of few gates exhibiting high switching activity. Hence we propose to optimise the power directly.

The power consumption of digital circuits can be divided into the dynamic (P_{dynamic}) and the static (P_{static}) power components. The first one occurs every time the output of a gate changes its logic value. Static power consumption is caused mainly by the leakage current which exists even when the circuit is in a stable state, i.e. not switching. Even though the static power component has always been present, it has gained importance in sub-micrometre and nanometre devices [16]. As a consequence of that, the total power consumption has to be optimised by reducing static as well as dynamic part of the power consumption.

Since the static part of the power consumption depends only on a function of a logic gate, the total static power consumption can be obtained by summing static leakage P_{leak} for all gates of the candidate circuits, i.e. $P_{\text{static}} = \sum_{\forall i} P_{\text{leak}}^{(i)}$. The situation gets complicated when we want to precisely determine dynamic power. In order to simplify this process and avoid running a costly analogue simulator, we propose to exploit the knowledge of the switching activity of each gate. The dynamic power consumption of a single gate P_{dyn} can be defined as follows:

$$P_{\text{dyn}} = \frac{1}{2} \times C_{\text{load}} \times V_{\text{dd}}^2 \cdot f \cdot E(\text{transitions}), \tag{1}$$

where C_{load} is the total load capacitance of the output (i.e. the sum of all input capacitances of the connected gates defined in the liberty file), V_{dd} is the supply voltage, f is the target frequency and E(transitions) is the expected value of the output transitions per global clock cycle (switching activity) [17]. The total dynamic power is equal to $P_{\text{dynamic}} = \sum_{\forall i} P_{\text{dyn}}^{(i)}$ and the total power consumption of a candidate circuit is calculated as

$$P_{\text{total}} = P_{\text{static}} + P_{\text{dynamic}}. \tag{2}$$

To simplify the problem, glitches are not typically considered. This decision enables to use the zero-delay model. As a consequence of that, the switching activity can be obtained using common circuit simulator which evaluates the response for all (or some for complex problems) input vectors. Total switching activity of a gate is calculated as follows:

$$E(\text{transitions}) = 2 \cdot (p_0 \cdot p_1) = 2 \cdot p_1 \cdot (1 - p_1), \tag{3}$$

where p_0 is the probability that the output of a considered gate is equal to logical zero, similarly p_1 is the probability that the output is equal to logical one. There are more ways to determine the transition probabilities. The simplest approach is to use the simulation and count the number of cases for which the output value was equal to 1. The advantage of this approach is that this calculation can be done during the function verification which represents an inevitable step of the fitness evaluation.

Various error criteria can be utilised to evaluate the quality of approximate arithmetic circuits. The average-case and worst-case arithmetic errors represent the most common metrics considered in the context of design of approximate arithmetic circuits [10, 18]. The worst-case error $e_{\text{wst}}(C)$ is employed in this paper. This metric is defined as the maximum absolute difference in magnitude between the original and approximate circuit computed over all inputs:

$$e_{\text{wst}}(C) = \max_{\forall \boldsymbol{i}} |O(C_{\text{orig}}, \boldsymbol{i}) - O(C, \boldsymbol{i})|, \tag{4}$$

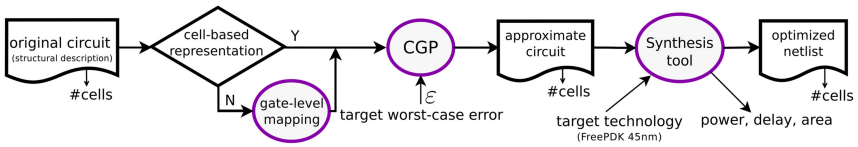

Fig. 2 *Proposed experimental setup for evolutionary design of approximate circuits represented using either standard gates or standard cells*

where $O(C_{\text{orig}}, \boldsymbol{i})$ denotes the output value of the fully functional circuit for the input vector $\boldsymbol{i}$ and $O(C, \boldsymbol{i})$ denotes the output value of approximate circuit C. For a circuit having n_{i} inputs, the input vectors are the numbers $0 \leq \boldsymbol{i} < 2^{n_{\text{i}}}$.

Let ε be the maximum acceptable worst-case error. Then, the fitness value of a candidate circuit C is calculated as follows:

$$\text{fitness}(C) = -\begin{cases} P_{\text{norm}}(C) & \text{if } e_{\text{wst}}(C) < \varepsilon \\ \infty & \text{otherwise,} \end{cases} \tag{5}$$

where $P_{\text{norm}}(C) = P_{\text{total}}(C)/P_{\text{total}}(C_{\text{orig}})$ is the normalised power consumption of C and $P_{\text{total}}(C_{\text{orig}})$ is the energy consumption of the original reference circuit C_{orig} and represents a constant value. The goal of the evolutionary strategy is to maximise the fitness and thus minimise the cost metric P_{norm}. It means that solutions with the error greater than ε are infeasible. Depending on design objectives, CGP has to be executed multiple times with different target errors ε_{i} if a Pareto front is requested.

3 Experimental results

The proposed method was evaluated in the task of evolutionary design of approximate adders and multipliers. These arithmetic circuits were chosen because they represent key components of many real-world applications in signal processing and machine learning [1]. In addition to that, the evolution of multipliers is considered as a very difficult benchmark in the evolutionary community. Apart from common 8-bit instances typically addressed in the literature (see e.g. a survey paper [19] which lists and compares different 8-bit approximate multipliers), 12-bit instances were chosen to validate our hypothesis. In case of 12-bit multipliers, we have to deal with complex netlists consisting of hundreds of gates. As our goal is to evolve approximate circuits with known worst-case error parameters, we need to evaluate the response for all input vectors for every candidate solution. It means that 2^{24} input combinations have to be evaluated for a 12-bit multiplier. This is the main reason, why we did not consider 16-bit instances. As it is infeasible to evaluate 2^{32} responses in a reasonable time, it would be necessary to employ a random simulation. Using a subset of all possible input combinations, however, may lead to a bias in evaluation since we cannot guarantee whether the worst acceptable error is met or not. Due to the limited space, only results for 12-bit multipliers will be presented in detail.

The experimental setup depicted in Fig. 2 is as follows. The goal of the evolution is to design an approximate multiplier showing the lowest possible power consumption for a given worst-case error. The power consumption is estimated as described in Section 2.3 and represents the only criterion reflected in the fitness function. The evolutionary algorithm starts with a common conventional multiplier whose fitness (i.e. power) is gradually optimised while keeping the worst-case error within the required bound. The worst-case error is used as a constraint ε_{i}. For each circuit, 11 error levels ranging from $\varepsilon_1 = 0.02\,\%$ to $\varepsilon_{11} = 20\,\%$ were considered. This range covers the values that are typically employed in the literature (see e.g. [9, 18, 19]). The CGP parameters were initialised as follows. We employed $\lambda = 24$ individuals in the population, mutation rate was $h = 5\,\%$, number of rows $n_{\text{r}} = 1$. The number of columns equals to the number of components of the initial conventional multiplier used to seed the evolution. The setting of the CGP parameters is based on the experiments conducted in our previous research [8]. Two different sets of experiments were executed. In the first scenario, only common two-input gates were considered. Each CGP node had two inputs and one output (i.e. $n_{\text{a}} = 2$, $n_{\text{b}} = 1$) and could implement one of the following eight functions: $\Gamma_1 = \{\text{BUF (buffer), INV (inverter), AND, OR, XOR, NAND, NOR, XNOR}\}$. In the second scenario, the set of functions was extended to 15 functions that correspond with common standard cells available in the chosen target technology: $\Gamma_2 = \Gamma_1 \cup \{\text{NAND3 (3-input NAND), NOR3 (3-input NOR), MUX2 (2-to-1 multiplexer), AOI21 (3-input AND/NOR), OAI21 (3-input OR/NAND), FA (full adder), HA (half adder)}\}$. The advantage of complex cells is that they are highly optimised for each target technology considering the area on a chip and performance. The full adder, for example, available as a technology cell occupies typically lower area, consumes less power and has lower delay compared to a full adder implemented using common gates. In order to support these functions, a CGP node with three inputs and two outputs, which corresponds with $n_{\text{a}} = 3$ and $n_{\text{b}} = 2$, was employed. Due to practical reasons, runtime was chosen as the only terminating criterion. This decision helps us to easily predict the end of evolution.

The evaluation of the fitness consists of two steps. In the beginning, active nodes are identified. Then, only active nodes are evaluated by means of a circuit simulator for all input combinations. In both scenarios, we utilised exactly the same circuit simulator based on a machine-code translation introduced in [20] and further extended in [21]. During the detection of the active nodes, each complex block such as full adder is replaced by an equivalent circuit consisting of common two-input gates. This simple preprocessing helps one to improve the performance of the simulator. Since there are only basic operations, SIMD instructions from AVX extension allowing processing 256-bit vectors can directly be exploited. As a consequence of that, we can simulate the response for 256 input vectors in parallel.

In order to avoid a possible bias preventing discovering of some implementations caused by seeding, we will seed the evolution with several conventional architectures of multipliers. The multipliers include ripple-carry array multiplier (denoted as RCAM), two carry-save array multipliers (CSAM1 and CSAM2) and three Wallace tree architectures (WTM1, WTM2 and WTM3). The array multiplier (RCAM) offers the lowest speed but occupies the smallest area on a chip compared to the other variants and especially the most expensive Wallace tree multiplier. The carry-save array multiplier is implemented as a set of 1-bit full adders without any carry-chaining that is finally reduced using a single n-bit adder. This arrangement helps one to slightly improve the delay compared to RCAM without introducing a noticeable area overhead. Two variants of carry-save multiplier are considered depending on the adder employed in the final stage – CSAM1 utilising ripple-carry adder (RCA) and CSAM2 with carry-save adder (CSA). Wallace-tree adder multiplier is the fastest known architecture which sums the partial products using multiple levels of CSA. Similarly to the carry-save array multiplier, the products are finally summed up using a single n-bit adder. In our case, three adders are considered in the final stage – RCA (denoted as WTM1), CSA (WTM2) and carry-look-ahead adder (WTM3). The latter variant offers the lowest delay but occupies a substantial area on a chip.

In total, six different multiplier architectures were described in Verilog language and synthesised using Synopsys DC and 45 nm technology. For each architecture, two Verilog netlists were generated. In the first case, the gate-level description was employed where all high-level building blocks such as half adders, full adders and multiplexers were implemented using generic two-input gates. In the second case, the structural description was

utilised. In this case, all the cells available in the chosen technology could be utilised in the description of the multipliers.

The parameters of the synthesised multipliers such as the power consumption, area on a chip or delay are summarised in Table 1. In addition to that, parameters of the initial netlists (i.e. specification) such as the number of cells (i.e. the number of components, the verilog netlist consists of), the number of full adders (column FAs), estimated power consumption P_{total} calculated using (2) and normalised power consumption P_{norm} are provided.

The results of the synthesis suggest that the Synopsys DC integrates a powerful optimisation engine that is able to significantly improve the initial design parameters. This is noticeable especially if we compare the estimated power consumption with the power consumption of the final implementation. In all cases, the power consumption was substantially improved. Despite a visible difference between the estimated and real power consumption and considering the fact that the inaccuracy of the power estimation tool is believed to be around 10%, the power consumption increases proportionally with the increasing fitness value which is important for our evolutionary optimisation process. Let us notice that 12-bit CSAM1 occupies a smaller area on a chip than RCAM at 45 nm which is quite surprising since RCAM is generally known as the most compact multiplier architecture.

Interestingly, the synthesis tool is quite capable in discovering and recovering the full adders from the gate-level netlists. The number of full adders in the implemented circuits is nearly the same independently on the chosen representation. This suggests that it does not matter whether we conduct the evolution at the level of gates or at the level of more complex building blocks such as full and half adders as the synthesis tool produces quite similar results in both cases.

3.1 Impact of the chosen representation on the efficiency of evolutionary design process

As evident from parameters in Table 1, the netlists containing only standard gates require a substantially higher number of components compared to the netlists composed of cells. For example, 768 gates are required to represent RCAM by means of standard gates. The same design can be implemented using 276 components only provided that half and full adders can be employed. Since the netlists are used to create the initial population, the higher number of components implies long chromosomes and more complex search spaces. In this particular case, the chromosome size increased by >64% when going down to the gate-level representation.

The chromosome length does not impact only the size of the search space but also the process of evaluation and consequently the scalability of the evaluation. The problem is that more nodes may be active because more nodes are available. As a consequence of that, more time is needed to evaluate fitness and quality of a candidate circuit.

In order to evaluate the impact of the chosen representation to the efficiency of the evolutionary design process, we will investigate two different parameters – the average size of a candidate solution and the convergence rate. The average number of active nodes impacts the speed of the fitness evaluation. In addition to that and when investigated at the end of a long-term evolutionary run, it also reflects the quality of discovered approximations because it is natural to expect that compact solutions typically result in a better power consumption.

The average size of a candidate solution can be measured directly as the number of active nodes or indirectly as the average time required to evaluate a single candidate solution. The problem of the first approach is that the average number of active CGP nodes does not reflect varying simulation complexity of each node. Hence, we adopted the latter approach that enables not only to fairly compare both representations but also to estimate practical limits of the evolutionary design process. In fact, it means that we consider the average size of candidate circuits expressed in terms of two-input equivalent gates. This is caused by the construction of the circuit simulator that is used to determine the fitness (the complex cells are replaced with two-input logic gates).

The average time required to evaluate a candidate solution is shown in Fig. 3. Since the size of a candidate solution varies with the error level (the approximate circuits with higher error level typically consist of a lower number of components), the results are reported for each error level separately. Sixty independent evolutionary runs (10 per each seed) were executed for each error level and each representation. The results of these runs are summarised by means of a single boxplot item. Each run was executed for 16,200 s to guarantee statistical significance. As evident, the results confirm our expectation related to the size of candidate solutions and error level. The average size of candidate solutions decreases nearly linearly with the increasing error level independently on the used representation. This trend is noticeable especially in the middle of the range because the error levels are on a logarithmic scale. On average, 224 ms (167 ms) is required to evaluate a candidate solution represented using standard gates (cells). The evaluation time decreased >2.3 times (1.9 for standard cells) when we increased the allowed error from 0.02 to 20%. It suggests that solutions consisting of approximately half of the equivalent gates were produced for 20% error.

If we compare the results for both representations, we can see that the representation based on standard cells leads to a more efficient design process. The time of the evaluation was reduced noticeably. As a consequence of that, more generations can be executed, and more compact solutions can be discovered. Considering the average results, more than four (standard gates) and nearly six (standard cells) candidate solutions can be evaluated per second. It means that we can improve the scalability by a factor ranging from 1.1 (higher errors) to 1.5 (lower errors) just by

Table 1 Parameters of six different 12-bit multipliers described at the level of gates (upper part) and standard cells (bottom part). The estimated power P_{total} as well as power of the implemented circuit is given in mW, area on a chip in μm^2 and delay in ns

Multiplier arch.	Specification					Implemented circuit				
	Repr.	No. instances	No. FAs	P_{total}	P_{norm}	No. cells	No. FAs	Area	Delay	Power
RCAM	gates	768	0	1.60	1.00	433	110	1679	3.22	1.23
CSAM1	gates	768	0	1.61	1.01	369	118	1595	2.28	1.18
CSAM2	gates	831	0	1.73	1.08	377	118	1608	2.33	1.19
WTM1	gates	809	0	1.66	1.04	394	115	1655	2.07	1.22
WTM2	gates	966	0	1.90	1.19	424	113	1708	2.39	1.23
WTM3	gates	1269	0	1.88	1.17	562	103	1961	1.68	1.39
RCAM	cells	276	120	1.27	1.00	424	110	1666	3.40	1.20
CSAM1	cells	276	120	1.28	1.01	374	118	1605	2.28	1.16
CSAM2	cells	296	123	1.37	1.08	393	117	1639	2.25	1.19
WTM1	cells	313	119	1.31	1.04	414	113	1687	2.12	1.21
WTM2	cells	362	129	1.48	1.17	463	111	1767	2.06	1.28
WTM3	cells	824	102	1.52	1.20	532	104	1889	1.77	1.33

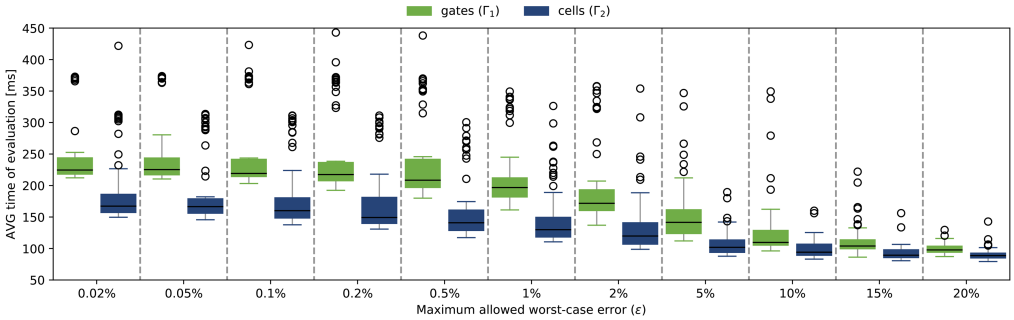

Fig. 3 *Average time required to evaluate a candidate solution representing an approximate 12-bit multiplier*

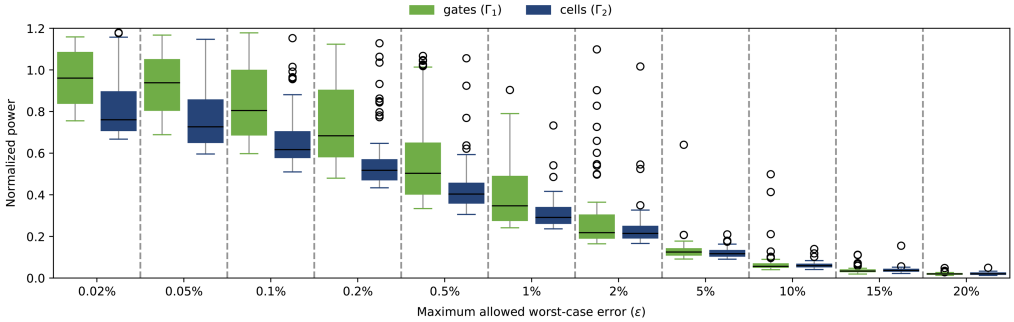

Fig. 4 *Normalised power consumption P_{norm} (the lower value the better result) of discovered 12-bit approximate multipliers*

changing the representation to standard cells. Looking and the boxplots, we can see that there is only negligible difference in the spread between both representations.

The representation impacts the speed of the fitness evaluation but does not necessarily mean that the evolution will produce better (i.e. more power efficient) solutions. Hence we analysed the power consumption of the discovered solutions. The results are shown in Fig. 4 where we plot the boxplots for normalised power (fitness value). We can observe the same trend as in Fig. 3. The power consumption decreases with the increasing error level independently on the chosen representation. In addition to that, a large improvement in the power efficiency can be seen in case of standard cell representation. This is noticeable especially for lower error levels. The cell-based representation is able to discover more efficient implementations. For example in the case of 0.2% error, the average power of a multiplier represented by standard cells equal to the power of best multiplier represented by standard gates. Starting at 5% error, there is no significant difference between parameters of the discovered results. This is caused mainly by the fact that the discovered approximate multipliers consist of an extremely small number of components. For example, approximate multipliers exhibiting 5% error consist of 6–14 times fewer components on average compared to original implementation. More than 64% of the components constitute simple gates. If we compare the boxplots, we can see that not only the extreme but even the spread is improved in the case of the cell-based representation.

The positive effect of the cell-based representation on the evolutionary design process is also evident on the convergence curves that are shown in Fig. 5. The evolutionary design process converges faster to the desired solutions not only from the point of view of absolute time but also when the number of generations is considered. The difference in convergence rate is largest for 0.2% error and gradually disappears with increasing error. This behaviour corresponds with our conclusions stated in the previous paragraph. For 2% error, the cell-based representation reaches the inflection point around a 1000th generation. When we employ a standard gate-level representation, we require more than three times higher number of generations to achieve the same results. This number corresponds to more than four times longer runtime. Nearly, the same convergence is achieved when we increase the error by one order. On contrary, when we decrease error to 0.2%, the difference is substantial. Not only that the gate-level representation converges slowly but also the time of evaluation is larger. Expressed in terms of runtime it means that after >270 min of evolution we obtained solutions whose quality is comparable with candidate solutions represented using cells that were generated in 20th minute of evolution.

A more detailed analysis is provided in Table 2 where we calculated the computational effort required to design an approximate multiplier exhibiting a required power reduction. The table shows the mean number of generations that have to be evaluated to obtain a multiplier satisfying the required criteria. In addition to that, the percentage of the evolutionary runs discovering the required circuit is given. The averages are determined from 120 independent evolutionary runs (60 for each representation). Note that not all combinations are viable as the maximum possible power reduction decreases with decreasing error. The power reduction is determined using P_{norm}.

As evident, the cell-based representation leads to much lower computational complexity than the gate-level representation. In average, less than half generations are required to obtain the same results. To give one example, we need to evaluate at least 455 generations in average to obtain an approximate multiplier with error below 0.2% that consumes 35% less power compared to the accurate RCAM multiplier. For this particular case, 50 out of 60 evolutionary runs successfully discovered such a multiplier. This corresponds to success rate 83%. For error levels above 10%, we can see that practically all runs were able to discover a multiplier having power consumption reduced by at least 95%. The mean number of generations for 95% power reduction suggests that evolution of the multipliers with a high error rate is a relatively easy task. Even the evolution of sub-optimal multipliers is easy. Every evolutionary run was able to discover an approximate multiplier exhibiting at least 20% power reduction for $\varepsilon \geq 1\%$. Less than one hundred generations were required in average. On the other hand, we can see that the computational complexity increases as the required power reduction is approaching the maximum possible reduction for a given error. More than one thousand (2000 for gate-level representation) generations are typically required to achieve the best possible power reduction.

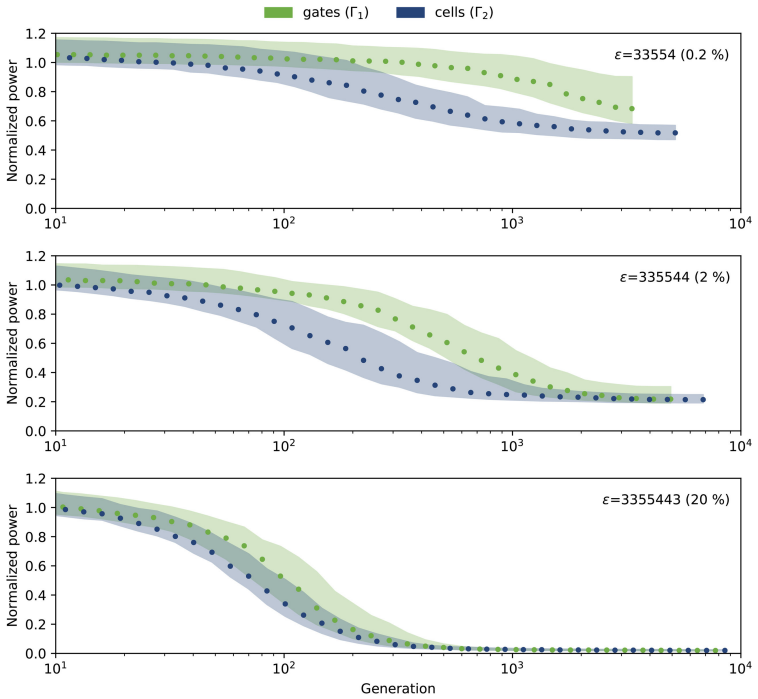

Fig. 5 *Convergence curves for evolutionary approximation of 12-bit multipliers for three error levels and both representations. For each combination, the median value (dotted line), the lower bound and upper bound of the interquartile range are calculated using 60 evolutionary runs*

Table 2 Performance of gate-level and cell-based representation expressed as the mean number of generations that have to be evaluated to obtain a 12-bit approximate multiplier exhibiting a required power reduction. The percentage of evolutionary runs that discovered a circuit satisfying the given parameters is shown after slash symbol (100% means that all 60 runs executed for each target error and each representation successfully produced such a circuit). The better results are printed in bold. In each generation 24 candidate solutions were generated and evaluated

ε	Repr.	Required power reduction					
		20%	35%	50%	65%	80%	95%
0.02	gates	2516/17	—	—	—	—	—
	cells	**935/78**	—	—	—	—	—
0.05	gates	2232/38	—	—	—	—	—
	cells	**688/75**	**2079/35**	—	—	—	—
0.1	gates	1576/58	2697/13	—	—	—	—
	cells	**533/92**	**1073/75**	—	—	—	—
0.2	gates	1045/72	1934/53	2948/8	—	—	—
	cells	**340/95**	**455/83**	**1566/62**	—	—	—
0.5	gates	622/82	1159/78	1932/62	2902/7	—	—
	cells	**207/98**	**382/97**	**669/88**	**1881/27**	—	—
1	gates	509/100	884/97	1283/87	1803/60	—	—
	cells	**138/100**	**271/100**	**403/98**	**809/93**	—	—
2	gates	346/98	552/95	743/90	1107/83	2046/43	—
	cells	**107/98**	**194/98**	**336/98**	**501/95**	**1443/52**	—
5	gates	156/100	273/100	353/98	513/98	890/98	—
	cells	**52/100**	**93/100**	**143/100**	**216/100**	**371/100**	—
10	gates	110/100	185/100	259/100	288/97	447/97	**2810/38**
	cells	**42/100**	**75/100**	**111/100**	**163/100**	**271/100**	2875/25
15	gates	87/100	142/100	206/100	282/100	398/100	824/93
	cells	**36/100**	**63/100**	**93/100**	**134/100**	**210/100**	**768/97**
20	gates	51/100	85/100	117/100	162/100	231/100	536/100
	cells	**42/100**	**73/100**	**103/100**	**139/100**	**202/100**	**473/100**

3.2 Results of synthesis

For each evolved solution, we created a Verilog netlist consisting of the same components as encoded in the corresponding chromosome. The netlists were then implemented in 45 nm technology (https://www.eda.ncsu.edu/) using Synopsys Design Compiler Ultra. The synthesis effort was set to high. Apart from an optimised netlist, the synthesis tool reports energy consumption, delay and area on a chip under iso-speed conditions.

The power consumption of the evolved approximate circuits synthesised using synthesis tool is shown in Fig. 6. When we compare the results with the estimated power consumption shown

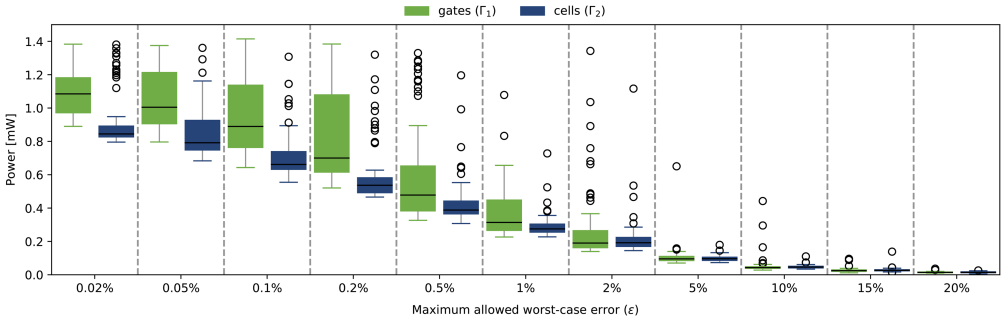

Fig. 6 *Power consumption of the synthesised 12-bit evolved approximate multipliers*

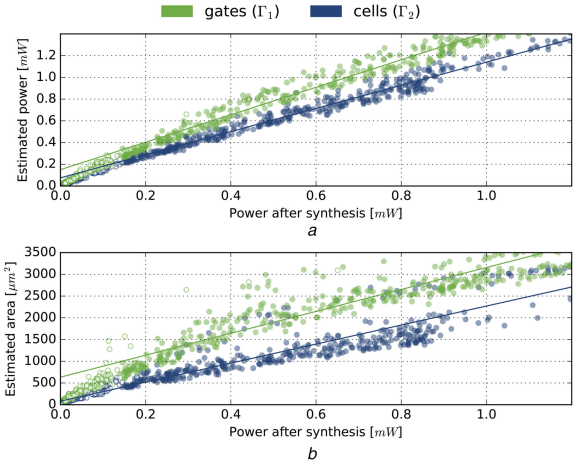

Fig. 7 *Estimated power consumption and area of 12-bit approximate multipliers represented in evolution using*
(a) Standards gates and ***(b)*** Standard cells vs. power after synthesis

in Fig. 4, we can see that both boxplots contain very similar data at least when we consider relative relations and spread. Solutions represented using the standard cells exhibit better energy efficiency compared to the solutions described using standard gates. The absolute values are different, but this was expected as discussed earlier because the discovered netlists may be modified during synthesis by the optimisation engine. In fact, the power consumption of the synthesised and implemented multipliers is typically lower than the power consumption before synthesis and the difference increases with the increasing circuit complexity. This is evident from the plots shown in Fig. 7 where we analysed the degree of correlation between estimated power and power after synthesis. There is a nearly linear relationship. In order to measure the linear relationship, we employed the Pearson correlation coefficient whose value varies between −1 and +1. The correlations of −1 or +1 imply an exact linear relationship, while zero implies no correlation. The correlation coefficient is slightly better than 0.99, in particular it equals 0.994 for standard cells and 0.992 for standard gates. The high positive value implies that as estimated power increases, so does the power of the synthesised multiplier. If we look at the output of simple linear regressions, we can see that the points concentrate around two trend lines having their slope slightly larger than 1. It means that the synthesis tool was able to improve the power in all cases and we actually overestimated the final value. For standard cells, the power of the multipliers was reduced during synthesis by a factor of 1.1 on average. This factor increases to 1.4 when standard gates are employed.

It is necessary to realise, however, that the ability to quantify the relative dependencies of the design power consumption is much more important than the ability to capture the absolute values since the search is driven by the comparison of parental and candidate solution's fitnesses. For comparison of the values it is only needed to achieve a high-fidelity value which can be calculated as follows. Let $R = \{R_0, \ldots, R_n\}$ be a set of n reference values and $E = \{E_0, \ldots, E_n\}$ be a set of n estimated values. Let T be a set of n circuits that were used to calculate R_i and E_i. The fidelity describing the quality of the estimation with respect to its ability to quantify relative dependencies of the tuples reference/estimation values is defined as

$$\text{Fidelity}_{\%} = 100\frac{2}{n(n-1)}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\mu_{ij}, \tag{6}$$

where μ_{ij} is determined as

$$\mu_{ij} = \begin{cases} 1 & \text{if } R_i > R_j \wedge E_i > E_j \text{ or } R_i = R_j \wedge E_i = E_j \\ & \quad \text{or } R_i < R_j \wedge E_i < E_j \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

According to the analysis, the fidelity calculated using the set of 660 evolved approximate multipliers represented using standard cells is 97.21%. For the second set of 660 multipliers encoded using standard gates, the fidelity is nearly similar – 97.25%. It means that the estimated values correlate with the results after synthesis in 422,814 (422,960 for the second set) out of 434,940 inspected cases.

The second plot in Fig. 7 shows the dependency between estimated area and power of the synthesised multipliers. The obtained results suggest that worse results will be obtained by considering the area in the fitness function only and demonstrates the superiority of the fitness function based on power estimation. The fidelity dropped to 93.7% (95.3% for the second set), the Pearson correlation coefficient decreased to 0.95 (0.97) and even the slope and distance of the regression lines increased.

In order to better understand the behaviour of both investigated representations for errors higher than 2%, we separately analysed the evolved netlists and corresponding netlists after synthesis and determined the number of cells and the corresponding area on a chip. We divided the cells utilised in the netlists to seven categories – buffers (BUF), inverters (INV), common two-input gates (F2X1), more complex cells with three and four inputs (F3X1, F4X1), half adders (HA) and full adders (FA). The area on a chip occupied by these cells before and after synthesis is shown in Fig. 8. For each error level, we plot the area of ten multipliers exhibiting the best power as reported by the synthesis tool.

Let us discuss the results for the gate-level representation. The evolved netlists consist mainly of two-input gates. Few buffers and inverters are employed only. The evolution is not directly forced to use the buffers and inverters but we assume that the evolution introduced them to improve the power since they exhibit a better output capacity and we consider the capacity of each gate in (2). If we look at the netlists optimised by the chosen synthesis tool we can see that the synthesis tool is able to identify a large number of complex cells such as full adders and three-input cells. Few instances of half adders and four-input cells are utilised only. Interestingly, the number of inverters increased substantially after synthesis. We inspected the netlists and identified that buffers and inverters are used mainly to improve the timing.

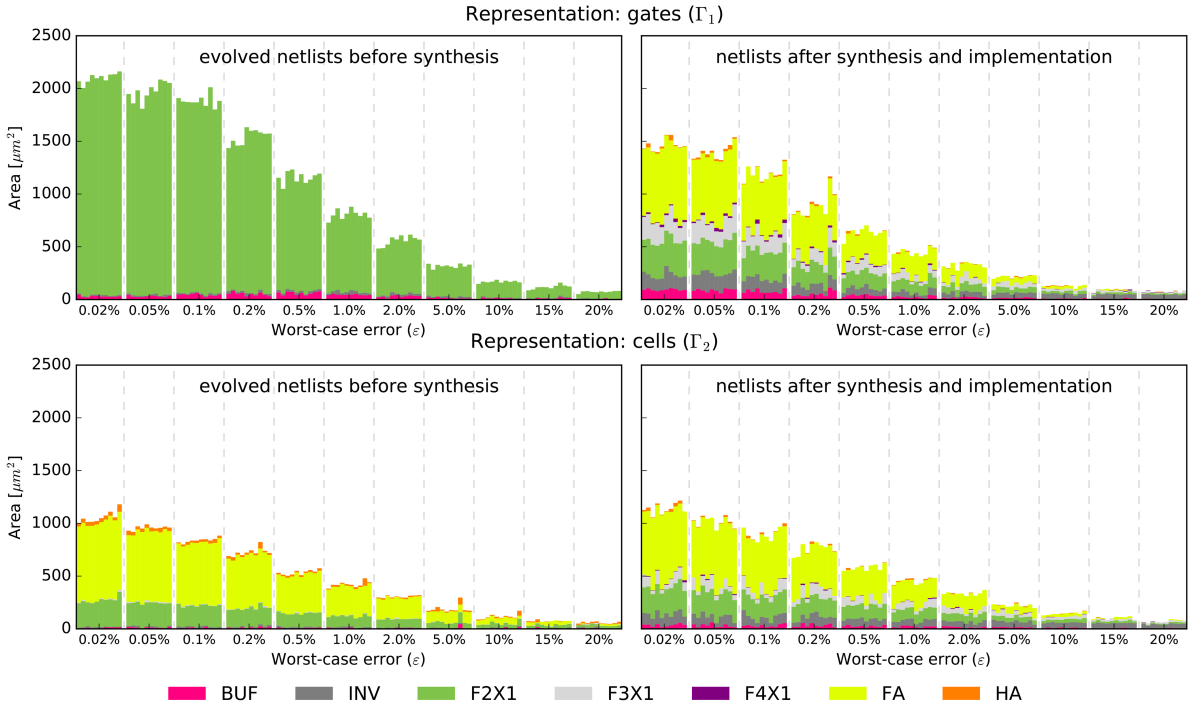

Fig. 8 *Area on a chip of evolved 12-bit approximate multipliers occupied by various types of standard cells before and after synthesis. For each error level, area of ten discovered fittest solutions is analysed separately for gate-level (top row) and cell-based (bottom row) representation*

As evident, full adder represents a key component for the construction of power-efficient multipliers. On the other hand, we can see that the number of full adders decreases with increasing error. Only one or two instances are utilised in approximate multipliers with an error higher than 5%. This observation is valid even for cell-based representation which explains why evolution at the level of gates and cells provide the same results.

As we already discussed, the synthesis tool was able to improve the energy consumption of the initial netlists. In addition to that, we can see that even area on a chip was reduced substantially. The area of approximate multipliers exhibiting 0.02% error, for example, was reduced from >2000 to 1500 μm^2. On the average, the area was improved by 32%.

In the case of the cell-based representation, the area on a chip after synthesis remains practically the same as before synthesis. We can also observe that the number of full adders in the evolved multipliers remains preserved. It suggests that the evolution produces highly optimised solutions that are hard to improve by the synthesis. This is also evident if we compare the results after synthesis for both representations. Multipliers produced from the cell-based netlists are more compact.

There are only a few half adders in the evolved solutions. Surprisingly, they practically disappeared after the synthesis. In comparison with the gate-level evolution, the buffers and inverters are utilised only rarely. This leads to a conclusion that it does not offer any advantage to consider complex three-input and four-input building blocks during the evolution because they are not utilised at all. Even the half adder seems to be not important. The essential thing is to include a full adder in the set of possible functions.

3.3 Evaluation of the proposed multipliers

The approximate circuit design problem is naturally a multi-objective optimisation problem in which the accuracy and other circuit parameters are conflicting design objectives. In our approach, only the power consumption was intentionally considered in the fitness function. It is thus fair to evaluate also the other circuit parameters. Hence, we took all synthesised approximate multipliers, determined the circuit parameters (worst-case error, power, area, delay) and identified the points on the Pareto front. In order to determine the Pareto dominant solutions, the Pareto-dominance relation was employed [22]. The resulting Pareto set projected into three 2D plots is shown in the upper part of Fig. 9. The bottom part of Fig. 9 shows the parameters of the chosen solutions as a function of mean error distance which was not directly optimised. As evident, the cell-based representation lead to substantially better solutions compared to the gate-level representation. A rich library of various implementations was obtained. When we ignored the solutions that exhibit nearly similar parameters, we obtained 17 Pareto-dominant solutions for the gate-level approach and 27 solutions for the cell-based approach. Note that two parameters were considered similar when their difference was <0.1 for the error (in log scale), power consumption and delay, and <100 for the area.

Fig. 9 includes also a comparison with the state-of-the-art approaches. For this purpose, we implemented two approximate architectures that are believed to provide the best results according to the latest review [19]. In particular, we implemented the truncated CSA array multiplier and the broken-array multiplier. Truncation, sometimes also referred to as a bit-width reduction, represents a straightforward approach to perform approximation. The key idea is to remove the least significant bits of the input operands and use a smaller multiplier instead of an accurate one. Similarly to the truncated multiplier, the broken-array multiplier removes some of the carry-save adders in an array multiplier, however the removed adders are determined by two parameters. In addition to that, we reimplemented one of the first approximate multipliers published by Kulkarni *et al.* in 2011 [23]. Kulkarni's multiplier employs a common modular approach and constructs multipliers of higher bit-widths using small manually designed 2-bit approximate multiplier. For more details related to the implemented architectures, we kindly refer the reader to the review. Surprisingly, the multipliers constructed using our method provide the best results if we consider cell-based representation only. Despite the fact that only the worst-case error and power were considered during the evolution, the obtained multipliers perform very well even under the mean error distance. There are only some minor discrepancies in the area– e_{wst} and area– e_{mae} plots. On the other hand, Kulkarni's modular approach results in an extremely inefficient 12-bit approximate multiplier. More than eight times better power consumption can be achieved for the same mean error.

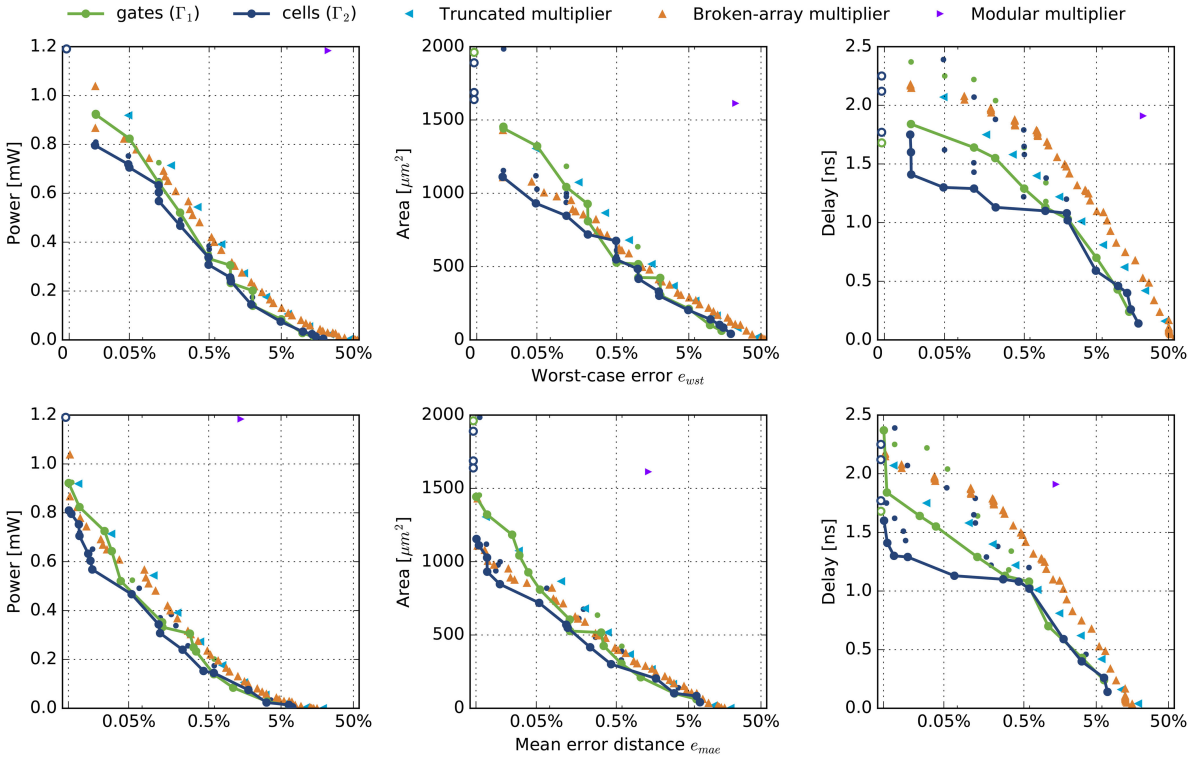

Fig. 9 *Parameters of the evolved approximate 12-bit multipliers occupying the global Pareto front (worst-case error, power, area, and delay considered) as well as the state-of-the-art approximate multipliers. Parameters of the accurate multipliers are shown using empty circles. Note that the x-axis has logarithmic scale. Solid line shows local Pareto fronts in which the remaining two objectives are ignored*

The situation is even worse for the worst-case error. Kulkarni's multiplier exhibits >50 times worse power consumption compared to the other considered multipliers having the same worst-case error.

4 Conclusion

In this paper, we introduced and evaluated a CGP-based evolutionary method for the design of energy-efficient approximate circuits. In contrast to evolutionary circuit design approaches available in the literature, we proposed to conduct the evolution at a higher level. In particular, we employed standard cells routinely used in design automation as building blocks for implementing digital circuits. This decision required to extend common CGP to support nodes with multiple outputs. In addition to that, a more powerful 256-bit simulator and power-estimation engine was employed in the fitness function. These changes gave rise to a more efficient design method which was able to handle more complex problem instances.

It was demonstrated that the proposed cell-based representation enabled not only to reduce the time of the evaluation but also to improve the convergence of the evolutionary design process. Interestingly, the detailed analysis suggests that including of a full adder in the set of possible functions is the key feature. The experiment related to the evolutionary design of 12-bit approximate multipliers revealed that the candidate circuits produced in the 20th minute of the evolution exhibit practically the same quality as the candidate solutions generated in the 270th minute of the evolution conducted at the level of common gates. Considering the fact that the results were achieved on a common 2.4 GHz Xeon CPU using a single thread application, the proposed method can be easily integrated to a standard design flow. As today's CPUs typically contain many cores, we can run several independent evolutionary runs and obtain more different results at the same time.

According to the literature, 8-bit multiplier represents the most complex instance with precisely determined error parameters that have been evolutionarily approximated. In this paper, we were able to push the limits and discover several non-dominated 12-bit multipliers exhibiting different circuit parameters. As shown, the discovered multipliers outperform the state-of-the-art multipliers available in the literature.

Despite this optimism, it is clear that exhaustive simulation cannot be applicable for 16-bit multipliers, where 2^{32} input test vectors have to be evaluated to guarantee the worst-case error. Similarly to the methods originating from the hardware community, however, we can cope with this issue by relaxing the requirement for strict worst-case error guarantee. It means that only a subset of all possible input combinations is used to determine the error parameters. By relaxing this requirement, we can approximate even larger problem instances. Even though worst-case error was employed only, the proposed approach can be applied for any error criteria.

5 Acknowledgments

This work was supported by the Czech Science Foundation grant no. GA16-17538S and by the Brno University of Technology project FIT/FSI-J-17-4294.

6 References

[1] Mittal, S.: 'A survey of techniques for approximate computing', *ACM Comput. Surv.*, 2016, **48**, (4), pp. 62:1–62:33

[2] Chippa, V.K., Chakradhar, S.T., Roy, K., *et al.*: 'Analysis and characterization of inherent application resilience for approximate computing'. The 50th Annual Design Automation Conf., (DAC'13, 2013), Austin, TX, USA, 2013, pp. 1–9

[3] Xu, Q., Mytkowicz, T., Kim, N.S.: 'Approximate computing: a survey', *IEEE Des. Test*, 2016, **33**, (1), pp. 8–22

[4] Venkataramani, S., Sabne, A., Kozhikkottu, K., *et al.*: 'SALSA: systematic logic synthesis of approximate circuits'. Proc. of DAC'12, 2012, pp. 796–801

[5] Venkataramani, S., Roy, K., Raghunathan, A.: 'Substitute-and-simplify: a unified design paradigm for approximate and quality configurable circuits'. Proc. of DATE'13 (European Design Automation Association), Grenoble, France, 2013, pp. 1367–1372

[6] Nepal, K., Li, Y., Bahar, R.I., *et al.*: 'Abacus: a technique for automated behavioral synthesis of approximate computing circuits'. Proc. of the Conf.

on Design, Automation and Test in Europe (DATE '14), Dresden, Germany, 2014, pp. 1–6
[7] Sekanina, L., Vasicek, Z.: 'Approximate circuit design by means of evolvable hardware'. IEEE Int. Conf. on Evolvable Systems (ICES), Singapore, 2013, pp. 21–28
[8] Vasicek, Z., Sekanina, L.: 'Evolutionary approach to approximate digital circuits design', *IEEE Trans. Evol. Comput.*, 2015, **19**, (3), pp. 432–444
[9] Mrazek, V., Sarwar, S.S., Sekanina, L.*, et al.*: 'Design of power-efficient approximate multipliers for approximate artificial neural networks'. Proc. of ICCAD'16, Austin, TX, USA, 2016, pp. 81:1–81:7
[10] Hrbacek, R., Mrazek, V., Vasicek, Z.: 'Automatic design of approximate circuits by means of multi-objective evolutionary algorithms'. Proc. of DTIS'16, Istanbul, Turkey, 2016, pp. 239–244
[11] Miller, J.F.: '*Cartesian genetic programming*' (Springer-Verlag, New York, NY, USA, 2011)
[12] Weste, N.H., Harris, D.: '*CMOS VLSI design: a circuits and systems perspective*' (Addison-Wesley, Boston, USA, 2005, 3rd edn.)
[13] Miller, J.F., Thomson, P., Fogarty, T.: '*Designing electronic circuits using evolutionary algorithms. Arithmetic circuits: a case study*' (Wiley, 1998), pp. 105–131
[14] Clegg, J., Walker, J.A., Miller, J.F.: 'A new crossover technique for Cartesian genetic programming'. Proc. of The Genetic and Evolutionary Computation Conf. (GECCO), London, 2007
[15] Slany, K., Sekanina, L.: 'Fitness landscape analysis and image filter evolution using functional-level CGP'. Proc. of European Conf. on Genetic Programming, Valencia, Spain, 2007 (LNCS, **4445**), pp. 311–320
[16] Wiltgen, A., Escobar, K., Reis, A.*, et al.*: 'Power consumption analysis in static CMOS gates'. 2013 26th Symp. on Integrated Circuits and Systems Design (SBCCI), Curitiba, Brazil, 2013, pp. 1–6
[17] Monteiro, J., Devadas, S., Ghosh, A.*, et al.*: 'Estimation of average switching activity in combinational logic circuits using symbolic simulation', *IEEE Trans. Computer-Aided Des. Integr. Circuits Syst.*, 1997, **16**, (1), pp. 121–127
[18] Jiang, H., Han, J., Lombardi, F.: 'A comparative review and evaluation of approximate adders'. Proc. of GLVLSI'15, Pittsburgh, PA, USA, 2015, pp. 343–348
[19] Jiang, H., Liu, C., Maheshwari, N.*, et al.*: 'A comparative evaluation of approximate multipliers'. Int. Symp. Nanoscale Architectures, Beijing, China, 2016, pp. 191–196
[20] Vasicek, Z., Slany, K.: 'Efficient phenotype evaluation in cartesian genetic programming'. Proc. of the 15th European Conf. on Genetic Programming, Malaga, Spain, 2012 (LNCS, **7244**), pp. 266–278
[21] Hrbacek, R.: 'Parallel multi-objective evolutionary design of approximate circuits'. GECCO '15 Proc. of the 2015 Conf. on Genetic and Evolutionary Computation, Madrid, Spain, 2015, pp. 687–694
[22] Deb, K.: '*Multi-objective optimization using evolutionary algorithms*' (Wiley, New York, NY, USA, 2001)
[23] Kulkarni, P., Gupta, P., Ercegovac, M.: 'Trading accuracy for power with an underdesigned multiplier architecture'. 2011 24th Int. Conf. on Very-Large-Scale Integration (VLSI) Design, Chennai, India, 2011, pp. 346–351

Paper VII

Trading between Quality and Non-functional Properties of Median Filter in Embedded Systems

VAŠÍČEK Zdeněk and MRÁZEK Vojtěch

Genetic Programming and Evolvable Machines. Berlin: Springer Verlag, 2017, vol. 18, no. 1, pp. 45-82. ISSN 1389-2576.

Genet Program Evolvable Mach (2017) 18:45–82
DOI 10.1007/s10710-016-9275-7

Trading between quality and non-functional properties of median filter in embedded systems

Zdenek Vasicek[1] · **Vojtech Mrazek**[1]

Received: 20 December 2015 / Revised: 6 July 2016 / Published online: 19 July 2016

Abstract Genetic improvement has been used to improve functional and non-functional properties of software. In this paper, we propose a new approach that applies a genetic programming (GP)-based genetic improvement to trade between functional and non-functional properties of existing software. The paper investigates possibilities and opportunities for improving non-functional parameters such as execution time, code size, or power consumption of median functions implemented using comparator networks. In general, it is impossible to improve non-functional parameters of the median function without accepting occasional errors in results because optimal implementations are available. In order to address this issue, we proposed a method providing suitable compromises between accuracy, execution time and power consumption. Traditionally, a randomly generated set of test vectors is employed so as to assess the quality of GP individuals. We demonstrated that such an approach may produce biased solutions if the test vectors are generated inappropriately. In order to measure the accuracy of determining a median value and avoid such a bias, we propose and formally analyze new quality metrics which are based on the positional error calculated using the permutation principle introduced in this paper. It is shown that the proposed method enables the discovery of solutions which show a significant improvement in execution time, power consumption, or size with respect to the accurate median function while keeping errors at a moderate level. Non-functional properties of the discovered solutions are estimated using data sets and validated by physical measurements on physical microcontrollers. The benefits of the evolved implementations are demonstrated on two real-

✉ Zdenek Vasicek
vasicek@fit.vutbr.cz

Vojtech Mrazek
imrazek@fit.vutbr.cz

[1] Faculty of Information Technology, IT4Innovations Centre of Excellence, Brno University of Technology, Brno, Czech Republic

world problems—sensor data processing and image processing. It is concluded that data processing software modules offer a great opportunity for genetic improvement. The results revealed that it is not even necessary to determine the median value exactly in many cases which helps to reduce power consumption or increase performance. The discovered implementations of accurate, as well as approximate median functions, are available as C functions for download and can be employed in a custom application (http://www.fit.vutbr.cz/research/groups/ehw/median).

Keywords Genetic programming · Genetic improvement · Cartesian genetic programming · Median function · Comparison network · Permutation principle · Median filter

1 Introduction

Genetic programming (GP) has traditionally been used to evolve entirely new expressions or functions to solve a particular problem which is usually specified by a training data set [23]. With the development of search based software engineering, GP has been applied to repair errors in software and assist in numerous tasks of software engineering [10]. The successful applications of GP in search based software engineering have attracted more and more researchers which has resulted in the establishment of a new research direction called *Genetic improvement* (GI). The Genetic improvement of software is defined as the application of evolutionary and search-based optimization methods with the aim of improving *functional* and/or *non-functional* properties of existing software [38].

The number of lines of code, execution time, memory usage, or power consumption represent the typical *non-functional* properties of software. These properties can be improved, for example, by replacing the existing code fragments by newly evolved code fragments that are semantically equivalent. The improvement of existing software by optimizing its non-functional properties was first addressed by White et al. [38]. Eight different target functions were considered in the paper. The authors showed that GP was able to discover code optimization tricks that are probably unreachable by current compilers. These tricks enabled slight improvements in the execution time of the chosen functions. Recently, Cody-Kenny et al. [3] demonstrated that GP was able to reduce the number of instructions for various manually constructed off-the-shelf implementations of a sort and prefix-code programs written in Java. Genetic Improvement has also been used to evolve an improved version of C++ code using automated code transplantation [22]. The authors evolved a faster version of Boolean satisfiability solver MiniSAT which is specialized for solving a particular problem known as Combinatorial interaction testing. Finally, the reduction of energy consumption of non-trivial programs was addressed in [26]. The authors introduced a system which can further optimize the low level Intel X86 code generated by optimizing compilers. While the previous examples have dealt with non-functional improvements, Langdon and Harman showed that GP can, in addition to non-functional parameters, improve the functionality of existing code [15]. The authors demonstrated that GP was able to

automatically improve behaviour (i.e. accuracy) of a widely-used DNA sequencing system consisting of 50,000 lines of C++ code.

A similar research direction has been explored in the field of approximate computing which is a promising approach to obtain energy-efficient computer systems. In constrast to Genetic improvements that preserve the code functionality, approximate computing exploits the fact that many applications are error resilient and do not require a perfect output to be produced. Hence a suitable compromise is sought between the error (quality), power consumption and performance. The approximations can be introduced at the level of hardware as well as software. An approximate solution is typically obtained by a heuristic procedure that modifies the original implementation. For instance, artificial neural networks were used to approximate software modules [7] in order to accelerate computations and reduce power consumption. In addition, search-based methods were allowed to approximate hardware components [21, 36]. Using GP in the context of approximate computing has been reported for digital circuit approximation [29, 34].

In this paper, we deal with GP-based improvements of non-functional properties of programs (C functions) that are intended for low-cost microcontrollers. As we seek significant improvements mainly of power consumption and execution time, we consider the approximate computing scenario and accept some errors in the outputs. The function to be approximated is the median filter which is crucial in signal processing, image processing and sensor data processing. The goal of the GP-based search strategy is to improve the existing, in most cases even functionaly optimal, median implementations and find programs showing suitable compromises between the accuracy, execution time and power consumption for various median functions when implemented on a microcontroller. This paper develops our previous results that were presented at the first GI workshop [20]. In contrast to our previously published work, a more efficient GI method based on a two-stage optimization process is introduced. The quality of the candidate solutions is measured as a distance between the candidate program and a fully-working median implementation. A generalized version of the zero–one principle [14] denoted as *permutation principle* is used to determine this distance. The permutation principle, first introduced in this paper, is formally proven in Sect. 4.2. Compared to the previous results, the quality of the obtained solutions is improved significantly. In addition, the benefit of GI approach is demonstrated using two real-world problems that are typically handled by embedded systems—sensor data processing and image processing. The obtained results are evaluated using data sets and by physical measurements on physical microcontrollers.

In the context of this work, one can observe that the evolutionary design and/or optimization of an (accurate) median outputting program has been carried out by GP only rarely [27]. However, a considerable number of research papers were devoted to the design and optimization of sorting algorithms (e.g., [1, 38]) and sorting networks (e.g. [11, 12, 28, 33]), which are useful structures when the median value has to be obtained. As checking whether a specification (i.e. an original code) and a candidate solution are semantically equivalent is time consuming, exact equivalence checking is not performed in the fitness function. The fitness is usually based on evaluating candidate solutions using a training data set and subsequent testing at the

end of evolution using other data sets. According to the zero–one principle, the training data are typically restricted to binary input vectors. A genetic improvement of two different implementations of Bubble-Sort algorithm was demonstrated in [38], where GP enabled the discovery of code optimization tricks probably unreachable by current compilers. These tricks enabled slightly improved execution time of the chosen sorting functions.

The rest of the paper is organized as follows. Section 2 briefly surveys genetic improvement and its relation to the approximate computing. Then an overview of the key areas related to this paper is given in Sect. 3. In particular, the median function and possibilities for the improvement are discussed. In Sect. 4, the permutation principle is introduced. The proposed method is described in Sect. 5. Section 6 introduces the experimental setup. The results are presented and analyzed in Sect. 7. Then, the obtained medians are applied to solve real-world problems. A detailed discussion is given in Sect. 8. Finally, Sect. 9 concludes the paper.

2 From genetic improvement to approximate computing

In contrast to approximate computing that has been developed to improve energy efficiency and performance for the cost of accuracy, GI has always kept the code functionality identical with the original software. In approximate computing, software and hardware is approximated (i.e. simplified with respect to fully accurate implementations) in order to reduce power consumption or increase performance. As a consequence, errors can emerge during computations. In many cases errors can be tolerated because human perception capabilities are limited, no golden solution is available for validation of results, or users are willing to accept some inaccuracies. Therefore, the error (accuracy of computations) can be used as a design metric and traded for area on a chip, delay, throughput, or power consumption.

One way to reduce energy consumption is by allowing timing errors by voltage over scaling or frequency over clocking. Another approximation technique, which is relevant for this paper, is *functional approximation*. The idea of functional approximation is to implement a slightly different function to the original one, provided that the error is acceptable and the non-functional parameters are improved adequately. Functional approximation can be conducted at the level of software as well as hardware.

After introducing several approximate circuits that were created manually [9], researchers started to develop more efficient systematic semi-automatic and fully-automatic methods. EnerJ [24], an extension of Java that adds approximate data types, represents one of the semi-automatic methods. Using these types, the system automatically maps approximate variables to low-power storage, uses low-power operations, and even applies more energy-efficient algorithms provided by the programmer. Axilog is a set of language annotations that provide the necessary syntax and semantics for an approximate hardware design and reuse in Verilog [39]. Axilog enables the designer to relax the accuracy requirements in certain parts of the design, while keeping the critical parts strictly precise. In contrast to fully-automatic methods, an approximate solution is typically obtained by a heuristic

procedure that modifies the original implementation. For example, artificial neural networks were proposed in [7] to learn to behave like a general-purpose code written in an imperative language. The trained network then replaced the original code. There are also general search-based methods that allows us to approximate hardware components [21, 36].

While the approximate problem has already been addressed by GP community [30, 34], GI has been used to improve functional and non-functional properties of software so far. However, by having the fitness function of GP-based GI permit errors, one can easily obtain approximate solutions. Applying the GI methodology for approximate computing (particularly for approximate software) seems to be straightforward. The main outcomes would be to obtain better trade-offs among key system parameters (note that the search-based methods are not constrained by various assumptions of mathematically rigorous methodologies) and reducing the optimization time with respect to commonly used solvers such as ILP. The key advantage is that the GI systems can be constructed as multi-objective (i.e. they provide a Pareto front showing the best trade-offs among the error, speed, memory usage, energy consumption, network loading, etc.) at the end of each run.

3 Background

In this section, we give an overview of the key areas related to this paper, especially the median function, construction of median networks and possibilities for the improvement of median functions. The section is concluded with problem formulation.

3.1 Median of a data set

Given a finite sequence of data samples, the median is defined as a value separating the higher half of data samples from the lower half. The median is of central importance in robust statistics [16], as it is the statistic that is the most resistant to outliers that could be presented in a given sequence. Contrasted to the mean, the median is a robust measure of central tendency. The main feature of the robust methods is their high efficiency in a neighbourhood of the assumed statistical model which is widely exploited in signal processing where the median is usually employed to filter the measured data.

There exists two basic approaches to determine the median of a given sequence. A straightforward and naïve approach is to employ a generic sorting algorithm, for example, the most popular and efficient quicksort algorithm. Implementations of sorting algorithms are very compact and robust, however, the execution time needed to determine the median value may vary with the values of the elements in a particular input sequence. This kind of nondeterminism may be problematic in real-time applications intended for microcontrollers having limited computing power. In addition, the sorting of the whole input sequence generates an substantial overhead. In order to eliminate the overhead, a more efficient in-place algorithm known as Quick select can be applied [14].

An alternative way of calculating the median value is to use a *median network*. The median network is a kind of sorting network whose concept is deeply elaborated in [14]. A sorting network is defined as a sequence of elementary compare-swap operations that sorts all input sequences. The sequence of comparators is fixed and depends only on the number of elements to be sorted, not on the values of the elements. Similarly, a median network is a sequence of elementary compare-swap operations that calculates the median for all input sequences. A compare-swap operation of two elements (a, b) compares a and b and exchanges (if it is necessary) the elements in order to obtain a sorted sequence.

3.2 Construction of median networks

A sorting network with n inputs and n outputs can be constructed using an instance of the sorting algorithm which is operating over a sequence of n items. The only condition is that the algorithm must be data independent. Bitonic-sorting and Batcher's odd-even merge sorting are examples of such algorithms.

A median network can be constructed from a sorting network by removing the useless compare-swap operations (i.e. operations that do not contribute to the output value). Aside from this, an optimal sequence of compare-swap operations is known for some median networks [4, 31]. Generally, the direct design of the median and sorting networks is a nontrivial task, especially for larger values of N.

In order to illustrate the difference among the results produced by various algorithms, let us suppose that we need to construct a 9-median network (i.e. a median network for $n = 9$ inputs). When Bitonic-sorting algoritm is used, we obtain a sequence of 23 operations. The Batcher's odd-even merge sorting produces the median network consisting of 22 operations (see Fig. 1a). The optimal 9-median

```
dtype median9_22(dtype *din)
{
  CS(0,1); CS(3,4); CS(5,6);
  CS(7,8); CS(0,2); CS(5,7);
  CS(6,8); CS(0,3); CS(1,2);
  CS(6,7); CS(0,5); CS(1,4);
  CS(2,3); CS(1,2); CS(3,4);
  CS(1,6); CS(2,7); CS(3,8);
  CS(4,5); CS(2,4); CS(3,6);
  CS(3,4)
  return din[4];
}
```

(a)

```
dtype median9_19(dtype *din)
{
  CS(1,2); CS(4,5); CS(7,8);
  CS(0,1); CS(3,4); CS(6,7);
  CS(0,3); CS(1,2); CS(4,5);
  CS(7,8); CS(3,6); CS(4,7);
  CS(5,8); CS(1,4); CS(2,5);
  CS(4,7); CS(4,2); CS(6,4);
  CS(4,2)

  return din[4];
}
```

(b)

Fig. 1 Two instances of a median network for 9 inputs. The compare-swap operation is implemented using macro CS(a, b) which assigns the lower value to din[a] and higher value to din[b]. One of the possible implementations of CS is the following one:

$$\textbf{if } (\mathrm{z} \quad \mathrm{din[b]} - \mathrm{din[a]}) < 0 \textbf{ then } \mathrm{din[a]} \leftarrow \mathrm{din[a]} + \mathrm{z};$$
$$\mathrm{din[b]} \leftarrow \mathrm{din[b]} - \mathrm{z}; \textbf{ end}$$

a Median constructed using Batcher's odd-even merge sorting. **b** Optimal implementation of 9-input median [4]

network consists, however, of 19 operations (see Fig. 1b). The corresponding codes in C language are shown in Fig. 1. Note that various implementations can be utilized. If there is a requirement to preserve the input data samples (i.e. to avoid the usage of in-place computations), two temporary variables have to be associated with each output of the CS operation.

Alternatively, each compare-swap operation can be replaced by two basic operations—minimum and maximum. This allows us to furthermore decrease the total number of required instructions because not every output value calculated by the compare-swap element is subsequently utilized. For example, the last operation shown in Fig. 1b calculates din[2] and din[4]. It is evident that it makes no sense to determine the value of din[2] because only din[4] is returned at the end. The representation based on the minimum/maximum operations enables us to reduce the code size of the 9-median shown in Fig. 1b by 21 % provided that the minimum and maximum macros share the code required to determine the relation between both input values.

3.3 Power-aware improvement of median networks

It is clear that the performance as well as power consumption of a particular median network implementation directly depends on the number of operations a given median network consists of. The higher number of operations results in a higher power consumption as well as a longer execution time. This relation can easily be revealed, for example, by inspecting the implementations shown in Fig. 1. The median networks shown consist of a fixed number of operations. Each operation is executed in the same number of clock cycles on average if it is measured at the level of machine code instructions.

Decreasing the number of operations represents the only way to improve the performance and power consumption. Unfortunately, a reduction of the number of operations is not possible without accepting some errors in the outputs produced by a median function. In other words, we have to search for a sequence of compare-swap operations that are capable of approximating the median. Let us call such a sequence a comparator network.

Two possible approaches can be applied to achieve our goal. One way to obtain an improved median network with a reduced number of operations is to construct a comparator network completely from scratch. It means to employ a variant of GP (e.g. linear GP, cartesian GP, etc.) and evolve programs satisfying the required quality as well as target size constraints (i.e. consisting of the required number of operations). The other possibility is to apply evolutionary techniques to reduce the number of operations of already existing median networks provided that the quality is maximized. In this scenario, a fully working median network used as a starting point is gradually modified according to the genetic improvement methodology. At the end of this process, a comparator network of the highest possible quality consisting of the required number of operations is expected. The question is which of these approaches performs better.

It seems to be natural to employ the first approach, however, due to the limited scalability of the evolutionary design, this approach seems to be extremely

inefficient. As shown in [34], randomly seeded GP discovered fully functional solutions for the 9-median, however, no correct solution was discovered for the 25-median. While the evolutionary design of a 9-median is a relatively simple problem, a 25-median consisting of more than 200 operations seems to be outside of the range of possibilities of the evolutionary design approach. If a median consisting of more than 100 operations is required, then direct evolution is unable to accomplish the goal. The authors claimed that solving the larger instances from scratch seems to be impossible for any evolutionary algorithm based on direct encoding.

3.4 Problem formulation

Given an existing median network N, i.e. a sequence of compare-swap operations of length n, and the target number of compare-swap operations m, find an alternative sequence of compare-swap operations M of length m s that this sequence maximizes the functional objective (quality) and minimizes the non-functional objectives.

In general, the problem can be understood as a single-objective as well as a multiple-objective optimization problem. The number of operations and time of execution and power consumption represent the typical software-related non-functional objectives. In addition, the number of stages required to determine the output value can be considered. This parameter is, however, important only when the comparator networks are intended for hardware implementation.

4 The quality of the improved median networks

In this section, we give an overview of approaches that enable us to assess the quality of partially working software and hardware. Then we discuss how to determine the quality of the improved median networks. We introduce and prove the permutation principle which gives a clue on how to determine the quality of median functions efficiently. In order to measure the distance between an original and improved version of the median, a problem-specific quality metric is proposed.

4.1 Common quality metrics

Various approaches to evaluate the quality of partially working software and hardware have been proposed in the literature.

The *error probability* (error rate) and *Hamming distance* represent metrics typically used to measure the quality of digital circuits. The error rate (Hamming distance) is defined as the percentage of input vectors (bits of output) for which the approximate output differs from the original one. In general, 2^{wn} input combinations exist for an n-input median network operating with elements encoded using w-bit integers. Clearly, it is intractable to evaluate all possible input combinations, however, the number of input combinations can substantially be reduced by applying the zero–one principle. The zero–one principle states that if a sorting network with n inputs sorts all 2^n input sequences of 0's and 1's into a

nondecreasing order, it will sort any arbitrary sequence of n elements into a nondecreasing order [14]. As a consequence, 2^n input combinations are sufficient to determine the error rate. Unfortunately, it seems to be difficult to apply this metric in practice. For example, there can exist a candidate implementation slightly modifying one half of the output values, but still providing good performance if used, for example, in image filtering.

The *average error magnitude* is another metric which is used for determining the quality of arithmetic circuits, not only in the field of evolutionary design, but also in the approximate computing. The average error magnitude is defined as the sum of absolute differences in magnitude between the original and approximate circuits, averaged over all inputs. Two complex issues are, however, connected with this parameter when used to evaluate the quality of a median network. Firstly, it is not possible to apply the zero–one principle in this case. As a consequence, we are unable to determine the exact value of this metric. In practice, we have to use a subset of all possible input combinations which helps us to estimate the value of the average error magnitude. The selection of the input vectors must be done carefully because it influences the precision of the estimate. Secondly, the averaging may hide situations in which completely wrong results are returned.

The common problem of the previously discussed generic metrics is that they do not reflect the quality of selecting the median value. In order to investigate the impact of the approximations on the quality of obtained results, regardless of the values of the input items, we introduce a new problem-specific metric. Let us recall that the median of a finite list of numbers can be found by arranging all the numbers from the lowest value to the highest value and picking the middle one. In other words, the median of a finite list of numbers consisting of $2k + 1$ items is equal to the $(k + 1)$th lowest value. The most important property of the median functions implemented in accordance with Sect. 3.2 is that the output always equals one of the input values. Let the output value equal to the jth lowest value. To describe the quality of an approximate median function, we can introduce *distance error* defined as the distance of the item chosen as the output value (i.e. jth lowest value) from the median (i.e. $(k + 1)$th lowest value) calculated as $|j - k + 1|$. Two additional metrics can be inferred from the distance error: *average distance error* defined as the sum of error distances averaged over all input combinations producing an invalid output value and *worst case distance error* defined as the maximal distance error calculated over all input combinations.

Note that it is not necessary to investigate all possible input combinations in practice. The *permutation principle* introduced in Sect. 4.2 permits one to substantially reduce the total number of input combinations that has to be investigated for a given comparator network in order to precisely determine the properties of the network. According to the permutation principle, the aforementioned distance errors can be determined using the permutations of a set S consisting of $2k + 1$ different values. To determine the quality, we propose to use a set $S = \{-k, -k + 1, \ldots, 0, \ldots, k - 1, k\}$. The set S consists of $2k + 1$ successive integers starting at the value $-k$. This particular arrangements enable to calculate the average distance error in the same way as the average error magnitude. This is possible because the median of S is equal to zero and the distance between jth lowest item (i.e. the value $j - (k + 1)$) and $(k + 1)$th lowest item (i.e. median of S)

is equal to $j-(k+1)$. Compared to the process of determining the average error magnitude, however, a substantially lower number of input combinations is required to be processed by a candidate median implementation.

4.2 The permutation principle

Definition 1 Let Σ be an ordered alphabet. A *comparator network* is a directed acyclic graph with n inputs and n outputs ($n \geq 2$), where each node has two inputs (x_1, x_2) and two outputs (y_1, y_2). The function of a node is defined as $y_1 = min(x_1, x_2) \wedge y_2 = max(x_1, x_2)$, where $x_1, x_2 \in \Sigma$.

Definition 2 A *sorting network* is a comparator network that monotically sorts every input sequence.

Definition 3 Let $A = (a_1, \ldots, a_n)$ be a sequence of n different elements, $A \in \Sigma^*$. Let δ_A: $\Sigma^* \rightarrow \mathbb{N}$ be a mapping which assigns each element $a_i \in A$ the position of this element in the sorted variant of A. Let δ_A be defined as follows:

$$\begin{aligned}
&\delta_A(x) = 0 \Leftrightarrow \quad \forall a \in A{:}x < a \\
&\delta_A(x) = |A| - 1 \Leftrightarrow \quad \forall a \in A{:}x > a \\
&\forall\ 1 \leq i, j \leq n{:}a_i < a_j \Leftrightarrow \delta_A(a) < \delta_A(b)
\end{aligned}$$

For simplicity, let $\delta(\mathrm{A})$ denote the sequence $(\delta_A(a_1), \delta_A(a_2), \ldots, \delta_A(a_n))$.

Lemma 1 ([14]) *Let N be a sorting network with n inputs that transforms a sequence $A = (a_1, a_2, a_3, \ldots, a_n)$ to a sequence $B = (b_1, b_2, b_3, \ldots, b_n)$. If a monotonic mapping f is applied to the sequence A, the network N transforms a sequence $A' = (f(a_1), f(a_2), f(a_3), \ldots, f(a_n))$ to $B' = (f(b_1), f(b_2), f(b_3), \ldots, f(b_n))$.*

Theorem 1 *Let N be a comparator network with n inputs. Let S be a set consisting of n distinct values. If every permutation of a set S is sorted by N, then every arbitrary sequence is sorted by N.*

Proof Suppose $A = (a_1, \ldots, a_n)$ is an arbitrary sequence which is not sorted by N. This means $N(A) = B = (b_1, \ldots, b_n)$ is unsorted, i.e. there is a position k such that $b_{k+1} < b_k$. Clearly, mapping δ_A is monotonic. By applying Lemma 1 and δ_A, the following holds $\delta_A(b_{k+1}) < \delta_A(b_k)$, i.e. $\delta(B) = \delta(N(A))$ is unsorted. This means that $N(\delta(A))$ is unsorted or, in other words, that the sequence $\delta(A)$ is not sorted by the comparator network N.

We have shown that, if there is an arbitrary sequence A that is not sorted by N, then there is a sequence $\delta(A)$, i.e. a sequence of $(0, \ldots, n-1)$ values, that is not sorted by N. Equivalently, if there is no $(0, \ldots, n-1)$-sequence that is not sorted by N, then there can be no sequence A whatsoever that is not sorted by N. Equivalently again, if all $(0, \ldots, n-1)$-sequences are sorted by N, then all arbitrary sequences are sorted by N.

Clearly, there exists a bijection between all permutations of S and all $(0, \ldots, n-1)$-sequences as follows from the definition of S. In particular, δ_S

ensures the bijective mapping. This means that if all permutations of S are sorted by N, then all arbitrary sequences are sorted by N. □

4.3 The permutation principle and distance error

The permutation principle introduced in the previous section can be employed to determine the distance between an arbitrary comparator network (e.g. partially working sorting network) and a sorting network as follows.

Theorem 2 *Let C be a comparator network, and N be a sorting network, both with n inputs. Let A be an arbitrary sequence $A = (a_1, \ldots, a_n)$. Let $D = C(\delta(A)) - N(\delta(A)) = (d_1, \ldots, d_n)$ be a mapping which assigns each element a_i a number d_i. Then, d_i is error expressed as the number of positions that are required to shift a_i to the right in sequence $C(\delta(A))$ to obtain sorted variant of sequence A.*

Proof Let $B = N(A)$ and $B' = C(A)$. For each element $b'_k \in B'$ holds that difference between a correct position and position of b'_k in a sorted variant of sequence A is equal to $d_k = \delta_A(b'_k) - k$. As N is a sorting network, it holds that $\delta_A(b_k) = k$ which implies that d_k can be expressed as $d_k = \delta_A(b'_k) - k = \delta_A(b'_k) - \delta_A(b_k)$. By applying Lemma 1, it holds that $D = C(\delta(A)) - N(\delta(A))$. □

Definition 4 Let A and B be two sequences of elements. Let $\sim_\delta$ denote an equivalence relation on the set of all sequences defined as follows:

$$A \sim_\delta B \Leftrightarrow |A| = |B| \wedge \delta(A) = \delta(B)$$

To conclude this part, let us give a simple example which illustrates the principle of determining the position error for a comparator network with 4 inputs and 4 outputs and two chosen sequences A and B.

Example 1 Let A and B be two sequences consisting of 4 items defined as $A = (25, 14, 36, 8)$, $B = (16, 12, 20, 2)$. Let N denotes the sorting network and C be a comparator network both with 4 inputs and 4 outputs, where C is defined as follows: $C(a_1, a_2, a_3, a_4) = (min(a_1, a_2), max(a_1, a_2), a_3, a_4)$.

According to the Definition 3, $\delta(A) = (2, 1, 3, 0)$ and $N(\delta_A(A)) = (0, 1, 2, 3)$ which follows from $N(A) = (8, 14, 25, 36)$ where $N(A)$ denotes the sorted sequence A. As the output of C is equal to $C(A) = (14, 25, 36, 8)$, the $C(\delta(A)) = \delta(C(A)) = (1, 2, 3, 0)$. To calculate the positional differences $D_C(A)$, we apply Theorem 2 which yields the following result $D_C(A) = C(\delta(A)) - N(\delta(A)) = (1, 2, 3, 0) - (0, 1, 2, 3) = (1, 1, 1, -3)$. The result can be interpreted in such a way that each of the first three elements of the partially sorted sequence $C(A)$ should be shifted one position to the right and the last element should be shifted three positions to the left. If all the shifts are applied, we obtain a sorted sequence.

The same sequence of steps applied to B yields $D_{C(B)} = C(\delta(B)) - N(\delta(B)) = (1, 2, 3, 0) - (0, 1, 2, 3) = (1, 1, 1, -3)$. It reveals that $D_{C(B)} = D_{C(A)}$, i.e. the same

sequence as for A was obtained. It means that we have applied the sequence within the same equivalence class, i.e. $A \sim_{\delta} B$. This fact can be easily checked by comparing the output of $\delta(A)$ and $\delta(B)$. It holds that $\delta(B) = \delta(A)$.

4.4 Final remarks

In general, there exist 2^{wn} input combinations that can be processed by an n-input comparator network operating at w-bits. We have shown that it is sufficient to reduce the number of the possible input combinations to $n!$ to prove the validity of a sorting network due to the existence of permutation principle (see Theorem 1). According to the zero–one principle, the validity of a sorting network can, however, be checked using 2^n binary vectors. As it can easily be checked, the 2^n is for $n \geq 4$ lower than $n!$, hence it seems that the proposed permutation principle does not offer an advantage. However, the problem of zero–one principle is that the binary vectors cannot probably be used to evaluate the quality of a comparator network. The reason is that we are not able to distinguish which value comes from what input (there are only two values—0's and 1's). To address this problem, Theorem 2 helps to determine the so called position error (distance error) which can be used as a basis of an error metric.

The impact of the introduced permutation principle can be seen from theoretical as well as practical point of view. From the theoretical point of view, it was proven that we can use this principle to evaluate the quality of candidate solutions without loss of generality (i.e. it is not necessary to evaluate responses for all w-bit input combinations). The permutation principle significantly reduces the number input vectors that have to be applied to obtain the fitness. In particular, 362, 880 vectors instead of 256^9 vectors are sufficient to precisely determine quality of a 9-input comparator network operating at 8-bits. From the practical point of view, the permutation principle (if properly applied) extremely simplifies the evaluation of candidate solutions because the response of a comparison network (i.e. output value) is equal to the distance from the median value. It means that we can avoid precomputing and storing of a training dataset.

Example 1 illustrates that no additional information about an investigated network is obtained when we try to check some property of a comparator network using sequences belonging to the same equivalence classes. This may happen when randomly generated test cases are used to determine this property. In fact, the randomly generated input sequenced may introduce a bias when used to evaluate quality of a comparison network. The probability of occurrence such cases is relatively high, because only the relation among the values within a generated sequence is important (i.e. not values themselves). Let us give an example. We created 10^6 test vectors consisting of nine randomly generated 8-bit values. Then, we calculated the number of covered equivalence classes according to Definition 4. It revealed that only 337,751 out of 362,880 (i.e. 93 %) of all possible equivalence classes were covered despite the fact that we generated approximately three times more test vectors than the number of equivalence classes. It means that there is many test vectors belonging to the same equivalence class.

The permutation principle and the obtained conclusions can directly be applied not only to comparator networks discussed in this section but also to comparator networks with a single output. In other words, the permutation principle can be used to assess the quality of partially working median as well as sorting networks.

5 The proposed method

In order to search for solutions with some improved level of a non-functional property, we must be able to quantify that property. In our case, the execution time and power consumption are considered. To estimate these non-functional properties, we can use the number of operations of which the median function consists of. This is a fairly reliable high-level estimate not only of execution time but also of power consumption. A more detailed simulation employing an accurate simulator would be necessary in general, however, our programs are designed as a sequence of min and max operations. It is supposed that each operation is transformed by compiler to a sequence of instructions that requires exactly the same number of clock cycles to perform this operation. In addition to that it also reflects the nature of modern embedded systems (e.g. ARM) whose instruction set predominantly consists of instructions that can be executed within one clock cycle.

In this paper, the task is formulated as a single objective optimization problem where the number of operations $\widetilde{n}$ represents a constraint specified by designer. Because both considered non-fuctional objectives linearly depend on this constraint, it is not necessary to include these objectives in the fitness function. This represents the main advantage of the constraint-oriented approach. In addition to this, the constrain-oriented approach is relevant to practice where the designers usually wants to achieve a particular power reduction in order to improve the performance of the whole embedded system.

To achieve our goal, we propose to use cartesian GP (CGP) in its linear form [18]. The linear form seems to be preferred approach compared to the traditional form of CGP representing the solved problems using two-dimensional array of nodes.

5.1 Representation of comparator networks

Each comparator network with n inputs can be represented using a directed acyclic graph consisting of k nodes. In order to encode such a graph, we can map the nodes to a 1D array of N nodes ($N \geq k$) that can be encoded using cartesian GP representation as follows. The number of rows, which is one of CGP parameters, is set to $n_r = 1$. The number of columns n_c is equal to the number of nodes, i.e. $n_c = N$.

The 1D array of nodes can be encoded using a string of integers, the so-called chromosome. The inputs are labelled $(0, 1, \ldots, n-1)$ and the nodes are labelled $(n, n+1, \ldots, n+n_c-1)$. Each node has two inputs and is encoded in the chromosome using three integers—two labels specifying where the node inputs are connected to and a single label specifying the function of the node. Finally, the

chromosome contains a single integer specifying the label of a node where the output is connected. The chromosome consists of $3n_c + 1$ integers (i.e. genes).

The main feature of CGP is that n_c as well as n_r (i.e. the total number of nodes N) are constant during evolution. It means that the size of the chromosome is constant because it depends only on n_c. On the other hand, the size of graph represented by this chromosome is variable as some nodes may become inactive. The nodes which do not contribute to the output value are called the inactive nodes. Example is given in Fig. 2 where only 4 out of 5 nodes are active.

Each node can act as minimum or maximum function. The inputs of a node can be connected either to the output of a node placed in previous l columns or to one of the input variables. This restriction ensures that no feedbacks are allowed. The output can be connected to output of any node. The l-back parameter will be unrestricted, i.e. $l = n_c$.

5.2 Quality of candidate solutions

The computational effort of EA directly depends on the number of test cases that are used for fitness evaluation of the GP individuals. Even if we apply the permutation principle which substantially reduces the number of all possible test cases that have to be investigated, we cannot run all of these due to time constraints. Thus, the number of test cases is fixed and specified at the beginning of the evolution. Based on the preliminary experiments and in accordance with observations related to the minimum number of test vectors required to evaluate candidate solutions [34], we determine the number of test cases as $T = 10^3 \times n^2$, where n is the number of input variables. The number of test cases is chosen in such a way that we are able to relatively precisely estimate the quality of the individuals while the time of fitness evaluation remains reasonable. Surprisingly, this simplification does not have any significant effect in practice provided that a reasonable number of unique permutations is used. The example given in Fig. 3 demonstrates how the number of test cases influences the shape of distribution of distance error for an approximate version of 25-input median. If more than 10^4 test vectors are used, the distributions are almost identical.

The first generated test case is the sequence $S = (-k, -k+1, \ldots, 0, \ldots, k-1, k)$ where $n = 2k + 1$. The next test case is obtained by swapping two randomly chosen items. This step ensures that a permutation of S is obtained. Note that the process of

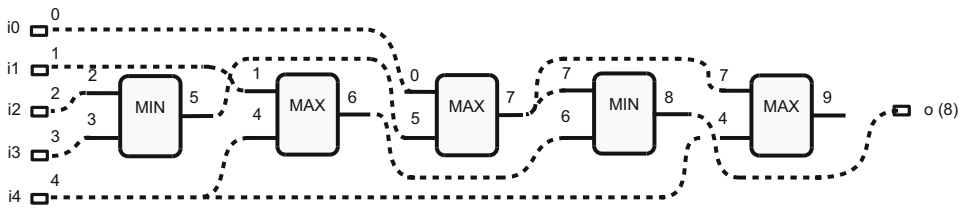

Fig. 2 Example of a 5-input comparator network encoded using cartesian GP with parameters: $k = 5$, $n_c = 5$, $l = 4$. Chromosome: 2, 3, min; 1, 4, max; 0, 5, max; 7, 6, min; 7, 4, max; 8. Node 9 is not used. The behaviour of the encoded comparator network is defined as: $o = \min(\max(i_0, \min(i_2, i_3)), \max(i_1, i_4))$

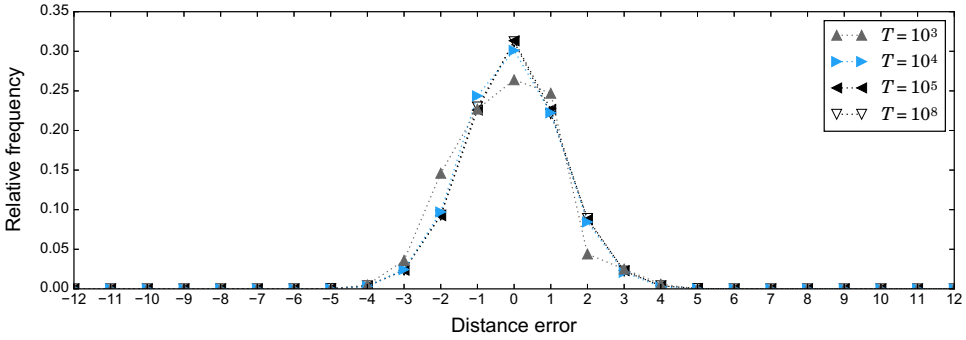

Fig. 3 Distribution of distance error for an approximate version of 25-input median as a function of the number of test cases T

generating the test cases have to be deterministic because we have to guarantee that exactly the same fitness score is obtained for individuals that represent the same behaviour. In order to satisfy this requirement, a separate random generator is used to perform the random exchanges. This generator is reinitialized with the same seed whenever we begin to generate the permutations.

The quality of the GP individuals is determined as follows. For each test case, the chromosome is interpreted. This step requires to successively determine the value at the output of each node. Finally, the response of a comparator network encoded by the individual is calculated. Because the permutations of a set proposed in Sect. 4.1 are applied to the inputs, the obtained response equals to the distance error determined for a given test case. In order to prefer the implementations with the lowest worst case distance error, we propose to calculate a histogram of distance errors and summarize the obtained results as follows:

$$q(C) = h(C,0) - \sum_{i=-k}^{k} h(C,i)i^2, \tag{1}$$

where $q(C)$ denotes the quality of a comparator network C and $h(C, i)$ represents the number of occurrences of a case for which the distance error equals to i, formally:

$$h(C,i) = \sum_{t \in T} \begin{cases} 1, & \text{ifC(t)} = \text{i.} \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

where $C(t)$ denotes the response of the comparator network C obtained for a test case $t \in T$ and T is a set of considered permutations of S. There are two reasons for including i^2. Firstly, only positive numbers are summed. Secondly, a natural weight is provided in order to emphasize the most important part of the histogram.

5.3 Search method

The search method follows the standard CGP approach [18], i.e. the evolutionary strategy $1 + \lambda$ is applied. The initial population is seeded by an existing median

network. In order to generate a new population, λ offspring individuals are created by a point mutation operator modifying h genes of the parent individual. The best individual of the current population (i.e. the parent individual together with λ offspring) serves as the parent of new population. The process is repeated until a given number of generations is not exhausted.

One mutation can alter either the function of a node, node input connection, or output connection. If a mutation hits a non-active node, this is detected and the candidate solution is not evaluated in terms of functionality because it has the same fitness as its parent. Mutations that do not affect the fitness score are called neutral and seem to be important in CGP because a series of neutral mutations can accumulate useful structures in the part of the chromosome which is not currently active (see detailed analysis in [8, 19]). In order to support this kind of neutrality, neutral mutants always replace their parent in CGP. One adaptive mutation can then connect these structures with active nodes which could lead to discovering new useful implementations.

In order to obtain an approximate version of median function M, we propose to apply a two-stage procedure. At the beginning, the designer specifies the target reduction that should be achieved, e.g. 15 %. The specified value is internally understood as the number of operations L of the approximate median function. In our example, L is equal to 85 % of the number of operations in the original median function.

The first stage starts with a fully functional solution. As has been discussed in Sect. 3.2, the initial solution can always be obtained in practice. In this stage, the goal is to gradually modify the initial sequence consisting of minimum and maximum operations and produce a reduced sequence of length L providing that a 5 % difference is tolerated with respect to L (tolerating a small deviance in the number of operations is acceptable; otherwise the search could easily stuck in a local extreme). The fitness function fit_1 used in the first stage is thus solely based on the number of operations

$$fit_1(C) = |C|, \tag{3}$$

where C denotes a candidate solution implemented using $|C|$ operations.

In the second stage, which begins after obtaining an implementation consisting of the target number of operations, the fitness function reflects not only the size, but also the quality:

$$fit_2(C) = \begin{cases} q(C), & \text{if } 0.95\mathrm{L} \leq |\mathrm{C}| \leq 1.05\mathrm{L}. \\ -\infty, & \text{otherwise.} \end{cases} \tag{4}$$

It is requested that the number of operations remains within 5 % tolerance with respect to L. Candidate circuits violating this hard constraint are discarded.

The proposed two-stage method eliminates the problem with seeding of initial population which may be considered as a limitation of the resource-oriented method [20]. The advantage of our method is that we do not need to implement a heuristics for generating the initial solution consisting of L operations. Instead, a fully working median function obtained by pruning a sorting-network is used as the start point. In

addition to this, it was demonstrated that the randomly seeded CGP was unable to produce reasonable solutions when the complexity of the problem to be solved increases. The role of seeding was investigated for example in [34]. The benefits of the two-stage method are not only in improving the quality of evolved circuits, but also in reducing the time of evolution.

6 Experimental setup

In order to evaluate the performance of the proposed approach, i.e. the ability to improve the considered non-functional parameters of the existing median functions, namely time of execution and power consumption, we have chosen four instances of the median filter that are common in practice. The results of optimization for 9-median, 11-median, 13-median, and 25-median will be reported. While the 9-input and 25-input medians are typically employed in image processing, the 9-input, 11-input and 13-input medians represent instances used to filter data coming from sensors. We did not consider the lower number of inputs because there is nearly no potential for improvement due to the small code complexity.

As previously mentioned, the designer has to specify the target reduction that ought to be achieved by reducing the number of instructions. Eight to eleven design points (i.e. different values of L) were considered for each problem. We carried out 100 repetitions of CGP at each design point to evaluate the variation in the output caused by the random seed. In total, 4000 experimental runs were performed. To be able to evaluate all runs in a reasonable time, the number of generations was limited to $g_{max} = 1 \times 10^4$. The number of generations is based on the initial experiments and represents a compromise between the ability to demonstrate the advantage of the genetic improvement in the solving of the chosen problem and the amount of required computational resources. If the objective is to find the best possible implementation for a certain design point, we recommend to increase the number of generations. In order to improve the efficiency of the fitness function, approach proposed in [35] was employed.

The following settings was used for the search strategy: Twenty offspring individuals are generated from the parent (i.e. $\lambda = 20$) using the mutation operator that modifies up to 5 % of the chromosome genes. The number of columns n_c is fixed for each design point and is initialized according to the number of operations of the original median function. In the case of 9-input and 25-input median, the optimal implementations consisting of the minimal number of compare-swap elements were used from [4]. Each compare-swap element was replaced by minimum and maximum operation and the worthless operations were removed as mentioned in Sect. 3.2. The obtained sequence of minimum and maximum operations was used as a starting point for seeding the evolutionary algorithm. In remaining cases, the initial fully working median networks were derived from a 25-input median network by reducing the number of inputs and removing redundant operations. We have verified that this approach produces more compact median networks compared to the results obtained using the approach employing a sorting algorithm. The parameters of the initial networks are summarized in Table 1.

Table 1 Parameters of the fully working median functions used to seed the evolution and the range in which the design points are sampled

Parameter	9-Median	11-Median	13-Median	25-Median
Number of compare-swaps elements	19	33	43	99
Number of min/max operations	30	56	74	174
Number of min operations	15 (50 %)	28 (50 %)	37 (50 %)	87 (50 %)
Number of max operations	15 (50 %)	28 (50 %)	37 (50 %)	87 (50 %)
Minimum value of L	8	8	10	50
Maximum value of L	30	56	74	174
Number of design points	11	8	8	13

7 Results

The results of the evolution are summarized in Fig. 4. For each problem and each design point, the normalized fitness score is given. This score is calculated according to Eq. 4, however, the results are normalized by the total number of test

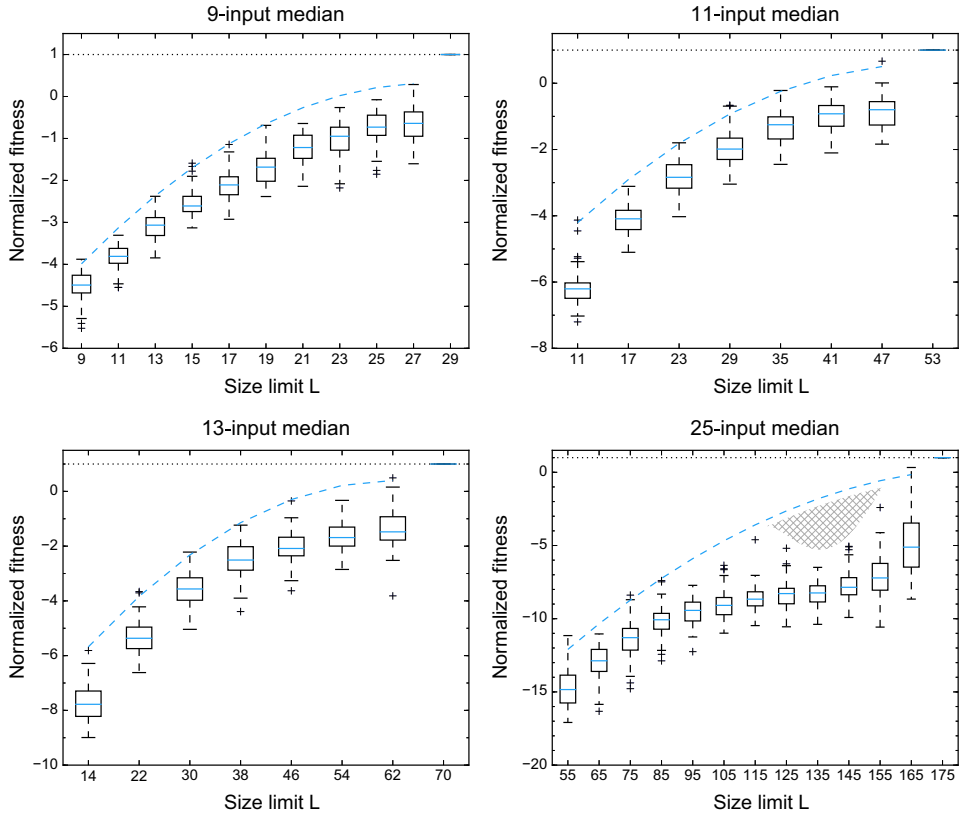

Fig. 4 The fitness score of the evolved comparator networks (approximate median function) based on 100 experimental runs performed for each design point. The *dash line* represents target Pareto frontier

cases. The interpretation of the y-axis is as follows. While the fully working median functions represented by the fittest solutions have their fitness score equal to one, the solutions of lower quality have assigned the fitness score lower than one.

For each problem, we sampled the design space equidistantly to be able to construct the Pareto frontier which helps us to discuss the performance of the method. As mentioned earlier, the maximum value of L is bounded by the size of the initial solution. Conversely, it makes no sense to explore situations where L is lower than the number of input variables because it means that some inputs will not be involved in computation. In the case of 25-input median, we restricted the lower bound even more because it would be computationally expensive to perform evolution for all cases. According to the measurements, 8.8 ms are required in average to calculate fit_2 for 9-input median. This time, however, increases up to 368.3 ms in the case of 25-input median. The experiments were conducted on a 64-bit Linux machine running on Intel Xeon X5670 CPU (2.93 GHz, 12 MB cache) equipped with 32 GB RAM.

Interestingly, compared to the resource-oriented method [34], our method is extremely efficient if the time required to obtain an implementation consisting of L operations is considered. According to the experiment, the average duration of the first stage is less than 10 ms in the case of 9-median and less than 373 ms in the case of 25-median.

The obtained results given in Fig. 4 are presented using boxplots which illustrate distribution of the normalized fitness calculated independently for each considered design point. As it can be seen, the variance of the fitness score is quite low for each design point. Taking into account that the number of generations was relatively low, these results demonstrate the robustness and stability of our method. The only exception is the 25-input median where we can see higher variance primarily at both extremes of L. In order to analyse this situation more thoroughly, we created a target pareto frontier (see dashed lines in Fig. 4) representing the goal of evolution. This Pareto frontier was obtained by interpolation of the fittest implementations obtained for 9-input, 11-input and 13-input median. The obtained regression models were generalized and projected backward to the plots. In most cases, we were able to find solutions that are very close to this imaginary pareto frontier. Unfortunately, in the case of 25-median (see Fig. 4, bottom right), we can see that there are cases in which the fitness score of the obtained results is far from the expected one. This is evident especially for cases where L is between 125 and 155. To investigate the reason of this gap, we tried to prolong the time of evolution for few of these cases and we discovered that this problem is caused by the insufficient number of generations. In order to obtain better results, it would be necessary to increase n_g adequately (at least by two orders of magnitude).

Whilst the initial implementations of 9-input and 25-input median networks remained unchanged, which was in fact expected as it is believed that the corresponding sequences of compare-swap elements are optimal, the evolution discovered improved versions of 11-input fully functional median function consisting of 50 operations and improved version of 13-input fully functional median counting 66 operations which yields 11 % reduction in both cases.

A more detailed analysis of the quality of discovered solutions is shown in Fig. 5 where we present histograms of the error distribution for each problem. The histograms are created using the best solutions obtained from all experimental runs. It means that for each design point, the fittest solution was identified and chosen. The quality is expressed in terms of the distance error. The histogram of distance errors are calculated for each discovered solution using 1000 times more permutations compared to the number of permutations utilized to determine fit_2; this enables to obtain precise results exhibiting the error in the order of 10^{-3}.

Let us discuss, for example, the results for 11-input median (see Fig. 5, top right). If we reduce the number of operations by 12 % (i.e. to 44 operations), the output value is determined correctly in more than 93 % all possible cases. In the rest of the cases (i.e. 6 %), the output value is determined incorrectly as the 4th lowest item of a sorted list of numbers. In less than 0.9 % of cases, the 6th lowest item is returned. Because the median value corresponds with 5th lowest item, the distance between median and output value is equal to 1 in both cases. If the number of operations is reduced to 20 (60 %), the worst case error increases to 2. According to the distribution of errors, this error, which is caused by outputting 3th or 7th lowest item, occurs in 3.6 % of all input cases only. The remaining 47.2 % erroneous outputs are caused by selecting 4th or 6th lowest item.

An interesting feature of the discovered solutions is the asymmetric distribution of the errors. This is more evident if we look at the histogram for 25-input

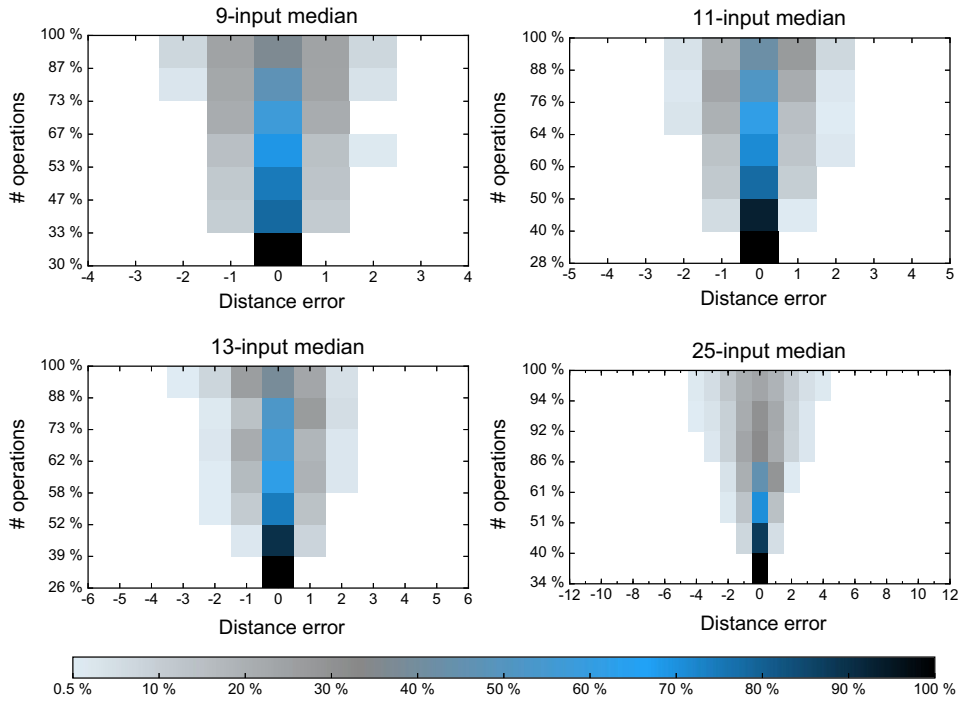

Fig. 5 The quality of the best discovered solutions consisting of a different number of operations expressed in terms of the distance error. The zero error means that the 5th, 6th, 7th, and 13th lowest item of input sequence was returned for 9-input, 11-input, 13-input, and 25-input median respectively

Table 2 Parameters of the improved implementations of 9-input median

No. of operations	Achieved improvement (%)	Error probability(%)	Distance error		
			Mean	Left/right	Worst-case
9	70.00	68.11	0.871 ± 0.702	−2	2
10	66.67	63.36	0.776 ± 0.677	−2	2
14	53.33	53.12	0.591 ± 0.601	−2	2
16	46.67	42.85	0.428 ± 0.495	−1	1
20	33.33	30.93	0.321 ± 0.491	−1	2
22	26.67	25.26	0.253 ± 0.434	−1	1
26	13.33	21.53	0.215 ± 0.411	−1	1

comparator network consisting of 150, i.e. 86 %, operations (see Fig. 5, bottom right). While the 4th lowest item is returned in 20 % of cases, the 6th lowest item is returned in more than 29 % of cases. We have not investigated the exact reason because it does not represent a real problem, however, it is worth noting that we have obtained many solutions with the same fitness score and it may happen that there is a solution with the same or slightly lower fitness score having a symmetric distribution of errors.

We can conclude that the obtained reduced median networks are of high quality. Even in the extreme case, where approximately 50 % of operations are removed, the error is not worse than one position for 9-input median, 2 positions for 11-input median, and 3 positions for 13-input and 25-input median respectively. The 25-input median consisting of more than 170 operations offers the largest possibilities for improvement. We can remove more than 25 % operations without a significant decrease in the quality. For more than 75 % of all possible input combinations, the median value or the values next to the median are returned.

To have a notion of properties of the discussed error metrics, Table 2 reports the error probability, mean distance error, and left and right worst case distance error for 9-input median. It can be seen that as the number of operations decreases, the error probability as well as the distance error are increasing. The mean value increases, however, it is not easy to a priori specify the required target value. The same is valid even for the error probability which gives the number of invalid output values, i.e. the amount of cases in which a value different from median was returned. Looking at the results shown in Fig. 5, it can easily be revealed that the issue with the mean value is that the distribution of errors is not the Gaussian distribution, especially for cases with a small reduction of the number of operations.

8 Improved medians in real embedded systems

Because the medians are typically employed to solve some real problem, we take the best discovered approximated median filters whose quality was discussed in the previous section and evaluated their performance in two different real-world

problems—processing of data acquired by sensor devices, and removing of noise in image data.

For each case study, the problem is briefly introduced first. Then, non-functional parameters of evolved as well as commonly used implementations are analysed and discussed. Finally, the impact of the approximate medians on quality and performance is evaluated providing that the approximate medians are employed as the main component which process data.

Four microcontrollers were chosen to evaluate the non-functional parameters of the evolved median functions. The microcontrollers were programmed using the complied C codes of discovered implementations discussed in the previous sections. Two non-functional parameters were measured: (a) the time that each microcontroller spends in a routine which computes the median value, and (b) energy consumed by the microcontroller to execute this routine.

A specific program was implemented, compiled and executed by the microcontrollers to perform the measurements. The program is designed as follows. Firstly, an input vector consisting of n integers is randomly initialized and fed to the routines calculating the median. Note that n is equal to the number of inputs of median. Then, an infinite loop is executed, which contains calling of the routine calculating the median value followed by a code modifying a randomly chosen value of the input vector to another value. Passing one iteration of the loop is indicated by inverting the logic value on a given pin. The execution time is then obtained using an oscilloscope by monitoring the period of the signal on the pin. The average execution time is reported.

In order to precisely determine an average energy needed to calculate the median value, all unused peripheral devices are switched off. Only those external components remain used which are necessary for program execution. Energy consumption was measured using Agilent N6705B DC Power Analyzer displaying the error lower than 0.025 % for voltage as well as current measurements.

8.1 Microcontrollers used for testing

In order to evaluate the non-functional parameters, we have chosen the following common-off-the-shelf microcontrollers available in our lab: 8-bit microcontroller of Microchip PIC family with code name PIC16F628A, 16-bit PIC24F08KA102, low-power 16-bit microcontroller MSP430F2617 from Texas Instruments and 32-bit ARM-based microcontroller STM32F100RB produced by STMicroelectronics. The goal is to present results for various architectures because there typically exist variations in the performance caused by different instruction sets on the one side and different internal architecture on the other side. To be able to interpret the obtained results, the main features of the microcontrollers are briefly discussed in this section.

The 8-bit PIC equipped with 3.5 kB of FLASH and 224 B of RAM is optimized for low-cost applications. Hence, a simple accumulator architecture without a stack is used. The instruction set consists of 35 instructions encoded using a 14-bit wide instruction word. The two-stage instruction pipeline allows all instructions to be executed in a single cycle, except for program branches. The chosen chip has an internal oscillator running at 4 MHz and consuming about 10 nA in the sleep mode

and about 565 μA in the active mode. Note that these values were measured when all the peripherals were deactivated.

The 16-bit PIC represents a class of microcontrollers with a register architecture consisting of 16 general-purpose 16-bit registers and 7 special registers. The instructions are encoded using a 24-bit instruction word with a variable length of the opcode field. The chosen chip contains 8 kB of FLASH memory, 1.5 kB of RAM memory and employs an internal oscillator running at 8 MHz. The instructions require from 1 to 3 clock cycles and are executed at 4 MHz. Our chip consumes about 4 mA in the active mode and 25 nA in the sleep mode.

The MSP430F2 is a 16-bit ultralow-power RISC microcontroller with register architecture optimized for processing data from sensor devices. The chosen CPU consists of 16 registers, is equipped with 92 kB of FLASH memory and 4kB of RAM, and can operate at 16 MHz. The calibrated digitally controlled internal oscillator can be configured to generate up to 8 MHz signal for system clock. The instruction set consists of 51 instructions with three formats and seven address modes. In contrast with PIC, there are instructions that enable to access two memory operands. The instructions require from 1 to 6 cycles to be executed. The instructions working with registers require a single clock cycle, the instructions addressing memory require 3 or 6 (when two memory accesses are required) cycles. The chip consumes 365 μA in the active mode at 1 MHz and 500 nA in the standby mode. In order to exploit the low-power capabilities, we configure the internal oscillator to operate at 1 MHz.

The STM32F100RB incorporates a high-performance RISC ARM Cortex M3 core offering twelve 32-bit general-purpose registers. This core builds on the ARMv7-M architecture and shows higher computational power compared to the aforementioned chips. For example, a single-cycle multiplication and a hardware division are supported. STM32 is equipped with 128 kB of FLASH memory, 8 kB of RAM and operates at 24 MHz. The maximum current consumption in the sleep mode is approx. 3.8 mA. When the peripherals are enabled, the current increases to 9.6 mA. The current in active mode ranges from 10 to 150 mA depending on the state of peripherals.

8.2 Evolved code on different microcontrollers

The process of obtaining C code from a chromosome is straightforward. Every active node, starting from one with the lowest index, corresponds with a single line of code containing a call of *min* or *max* function whose operands are taken from the input sequence or the outputs of preceding operations.

The *min* and *max* functions are defined as two macros outputting the minimal and maximal value for two operands. The compiler is then able to unroll the code and optimize it in terms of register assignment and overall performance.

8.3 Processing data from sensor devices with approximated median filters

When we look at signals coming from various devices such as A/D converters, temperature sensors, or accelerometers, the data are noisy even in a perfect

environment. In a real situation, where the accelerometers are, for example, used to stabilize various flight vehicles, the situation is even worse because of various vibrations caused by motors or propellers that are for example out of balance. When such a sensor acts as a central element controlling a process, it is necessary to remove the noise so as to prevent unwanted behaviour.

There are many filters that can be applied to smooth the measured data, for example, a variant of *low-pass filter*. The filter tries to keep the low frequency data while removing the high frequency noise (i.e. spikes). A low-pass filter usually is implemented in a situation where a limited number of computational resources are available because its implementation is simple. It can be implemented as an exponentially weighted moving average $x_{t+1} = \alpha y_t + (1 - \alpha)x_t$ where y_t represents data measured at time t and x_t the output value obtained at time t. Alternatively, a more robust Kalman filter may be used [13]. In contrast to the low-pass filter which has a fixed parameter α, Kalman filter is an adaptive estimator which minimizes the mean square error of the estimated parameters according to the previous state and actual measured value. Given only the mean and standard deviation of noise, the Kalman filter is the best linear estimator.

Unfortunately, there are two issues connected with the usage of linear filters. The first problem is that the data is being delayed by the filter which is a feature of linear filters when they are set to have a strong filtering effect. The second issue is that the filtered signal does not seem to follow the original measured data very well. To avoid the delay and provide results of high quality, we can employ an instance of median filter to smooth the measured data.

A relatively small number of samples are sufficient to be able to filter the measured data and remove the outliers. To demonstrate the benefits of the discovered approximations, we will apply the evolved 11-input and 13-input medians to filter the outliers presented in a signal captured by an accelerometer sensor. The obtained non-functional parameters are summarized in Tables 3 and 4. As the non-functional parameters are manually evaluated on real systems, only some of the Pareto dominant discovered solutions are investigated. It should be noted that only the number of operations and the quality defined by Eq. 1 was considered during construction of the Pareto set. We have implemented and measured not only the evolved solutions, but also three common approaches to determine median value—the *quicksort* algorithm, *quickselect* algorithm and the so called *running median*. While *quicksort* represents a sorting algorithm, the *quickselect* is a selection algorithm which is able to find the kth smallest element in an unordered list [4]. The quickselect uses the same overall approach as quicksort, however, it only recurses into one side of the input sequence which reduces the average complexity. The *running median* attempts to minimize processing time by maintaining a data list that is sorted from the smallest value to the largest value [25]. When a new sample is submitted, it replaces the oldest sample. The new sample is then shifted in the sorted list to bring it to the correct location.

Firstly, let us discuss size of the machine code of the complied C codes. If we compare the amount of bytes occupied by median networks and common approaches such as quicksort, quickselect and running median, we can easily

Table 3 Non-functional parameters of accurate (emphasized) and approximated implementations of 11-input median function measured on different MCUs

Impl.	Machine code size [B]				Execution time [µs]				Consumed energy [nWs]			
	STM32	PIC24	PIC16	TI430	STM32	PIC24	PIC16	TI430	STM32	PIC24	PIC16	TI430
14-ops	118	324	328	156	4.0	88	341	310	120	604	682	249
20-ops	158	441	452	224	4.6	108	450	356	140	745	900	286
25-ops	206	549	567	276	5.5	127	552	375	169	878	1105	301
30-ops	232	648	684	318	5.9	144	659	410	179	995	1318	329
32-ops	254	696	845	342	6.2	153	791	420	188	1057	1582	337
38-ops	294	819	1065	400	6.8	175	982	450	207	1208	1964	361
44-ops	328	900	1200	434	7.5	187	1105	465	230	1290	2210	373
50-ops	378	1032	1320	472	8.6	210	1220	480	261	1449	2440	385
qsort	128	333	–	196	40.5	958	–	1515	1235	6610	–	1217
qselect	212	849	607	276	17.5	488	2910	705	535	3367	5820	566
running	236	729	412	344	14.2	274	785	690	435	1887	1570	554

An implementation labelled as *n*-ops denotes evolved comparator network consisting of *n* operations

Table 4 Non-functional parameters of accurate (emphasized) and approximated implementations of 13-input median function measured on different MCUs

Impl.	Machine code size [B]				Execution time [µs]				Consumed energy [nWs]			
	STM32	PIC24	PIC16	TI430	STM32	PIC24	PIC16	TI430	STM32	PIC24	PIC16	TI430
17-ops	138	378	387	192	4.4	96	375	335	135	666	750	269
26-ops	214	588	594	288	5.6	136	587	390	172	935	1174	313
34-ops	288	747	922	402	7.0	163	880	470	215	1125	1760	377
38-ops	330	825	1054	434	8.0	176	980	480	244	1218	1960	385
41-ops	332	894	1144	516	8.0	190	1058	530	245	1309	2115	426
48-ops	398	1020	1306	574	8.8	210	1205	565	270	1452	2410	454
58-ops	478	1209	1590	642	9.5	242	1690	585	290	1672	3380	470
66-ops	496	1353	1854	666	10.2	266	1690	560	311	1835	3380	450
qsort	128	333	–	196	51.6	1178	–	1800	1574	8128	–	1445
qselect	212	849	607	276	21.4	610	4060	800	652	4212	8120	642
running	236	732	394	344	13.1	552	1100	750	400	3809	2200	602

determine that these implementations are more compact compared to the accurate median filter implemented using the median network consisting of 50 min/max operations for the 11-input median and 66 operations for the 13-input median. The size of the quicksort routine is equal to the size of the 11-input approximate median consisting of 14 operations. To sum up, quicksort is the most compact algorithm. Nevertheless, it is interesting to note that PIC16 does not allow one to execute the quicksort algorithm because its implementation relies on the recursion which cannot fit the in-memory emulated stack. The implementation of the running median occupies approximately a 1.8 times higher number of bytes on average compared to the quicksort. Quickselect consumes a bit more except for STM32 and MSP430 where the algorithm requires a lower number of bytes to be implemented.

The number of operations of the approximate median functions correlates with the machine code size. There is only one exception. The 13-input comparator network consisting of 38 operations implemented on STM32 exhibits nearly no reduction compared to the code consisting of 41 operations. It seems that some optimization tricks were discovered by the ARM compiler.

If we compare the size of machine code across all considered microcontrollers, the ARM architecture has an extremely efficient mechanism of instruction encoding. In addition, it revealed that the ARM compiler contains a very effective optimization engine. Similarly, the code generated by the MSP430 GCC compiler is very compact compared to the code for PIC microcontrollers. It is worth noting that it is extremely important to enable GCC optimizations. Otherwise, not only the size of the machine code, but also computation time increases by 60 % on average without changing a line of C code. When we take into account the fact that MSP430 is equipped with an exceptionally large FLASH memory (see Sect. 8.1), it seems to be a very powerful low-cost microcontroller.

The average execution time and average energy consumption measured for various implementations of 11-input and 13-input accurate as well as approximate median functions are given in the second and third part of Tables 3 and 4. Let us first discuss the parameters of the accurate implementations. During the measurements, it turned out that the energy consumption pattern remains almost invariant because all approximations use identical sequences of instructions. Consumed energy thus mainly depends on the execution time which is shorter when more aggressive approximations are applied. The average power consumption, when an accurate median is calculated, is 0.8 mW for MSP430, 2 mW for PIC16, 6.9 mW for PIC24 and 30.5 mW for ARM. In the case of the 11-input median function implemented on PIC24, the median network is 4.5 times faster than the quicksort algorithm and the consumed energy was reduced from 6610 to 1449 nWs (i.e. by 78 %). The median network is 4.7 times faster than quicksort on the STM32 and the energy was also reduced by 78 %. A little bit worse situation is at MSP430. The median network is 3.1 times faster than quicksort, but its energy consumption decreases by 68 %. Similar results were obtained for the 13-input median. The median network implemented on PIC24 is 4.4 times faster than the quicksort algorithm and the consumed energy was reduced by 77 %. At STM32, the quicksort algorithm exhibits 5 times worse execution time and an 80 % higher energy consumption compared to the median calculated using 66 min/max operations. At

MSP430, the median network is executed 3.2 times faster than quicksort. While there is a relative large performance gain of median networks compared to the quicksort algorithm, the execution time of running median is comparable with the median networks. The best improvement is achieved at STM32 where the implementations of median networks are executed 1.7 times faster than running median. For the rest, the gain varies around 1.4 on average.

Let us now move on to the execution time and energy consumption of the evolved approximate median functions. At first glance, it is evident that the execution time decreases with the decreasing number of operations. The situation is, however, a little bit complicated here. Let us compare the execution time of, for example, an 11-input accurate median network consisting of 50 operations and an 11-input reduced network consisting of 25 operations. While the number of operations is reduced by 50 %, the execution time is adequately decreased only at PIC16, where a 54 % improvement was achieved. STM32 and PIC microcontrollers exhibit improvement which is less than 39 %. In the case of MSP430, only a 22 % reduction was achieved. In order to better understand this phenomenon, we have to firstly investigate the dependence between the number of min/max operations and the number of generated instructions for MSP430. The implementation of an accurate median network consists of 219 instructions and the reduced median network consists of 122 instructions which leads to a 44 % improvement. Unfortunately, the difference between the improvement at the level of instructions (44 %) and improvement at the level of operations (50 %) is relatively small. In order to determine the root source of such a discrepancy, it is necessary to perform an analysis at the level of a machine code. It has been revealed that two different mechanisms were used to implement the min/max operations. Some operations are implemented using indirect addressing, other operations are optimized and consists of instructions only manipulated with registers. This makes a huge difference in the number of clock cycles required to execute a single min/max operation. Some operations are evaluated within 5 cycles, others require up to 11 clock cycles. Despite this finding, there is linear dependence between the energy consumption and time of execution and longer times imply a higher energy demand.

Despite the fact that STM32 has the largest current consumption in active mode, it provides the best results from the perspective of energy consumption. Even if the MSP430 is declared as an ultralow-power microcontroller, it requires about a 1.7 times higher amount of energy to execute the same code. It is necessary to note, however, that higher energy consumption is in close relation with the time of execution which is more than 60 times higher compared to STM32. Compared to PIC16 and PIC24, MSP430 consumes from 3 to 6 times lower energy to accomplish the same task. On the other hand, PIC24 is able to produce about five times more results within the same period of time at the cost of 30 % higher energy consumption compared to MSP430. In order to avoid misinterpretation, it is worth noting that MSP430 operates at 1 MHz while PIC24 operates at 4 MHz. When we increase the frequency to 4 MHz, the time of execution decreases four times with no additional cost (the energy consumption remains at the same level).

Figure 6 gives an example of real data filtered by various implementations of 11-input and 13-input median filters. The data were obtained from an accelerometer

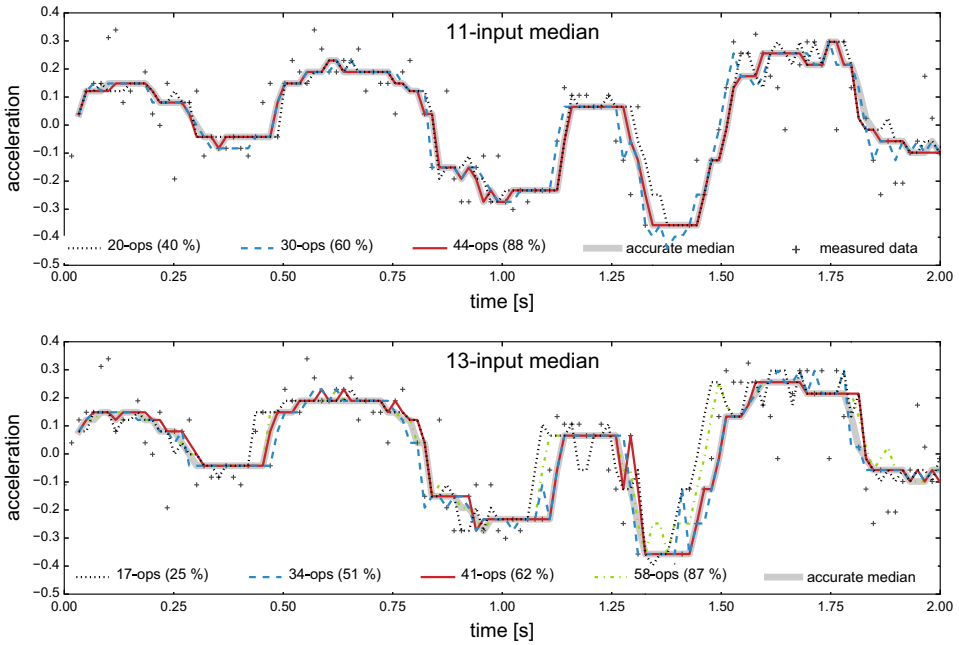

Fig. 6 Example of data filtered using accurate as well as approximate versions of 11-input and 13-input median filter. Note that only some of the *measured points* are shown because of readability

whose output signal was sampled at 8 kHz. Taking into account the sample rate, the considered accurate median filters introduce a delay not worse than 1.7 ms which represents a reasonable value. When six operations (12 %) are removed from the 11-input median network, the resulting approximate median produces an output that is nearly similar to the output of accurate implementation. There are only neglible differences that do not prevent us from applying this inaccurate implementation in an embedded application to filter the outliers and save energy. In the case of implementation at STM32, for example, we can reduce the consumed energy by 11.8 % by introducing the approximated median network consisting of 44 operations.

Interestingly, the median network which consists of 20 operations (60 %) produces a signal which is very similar to the output of an accurate median, despite the fact that the measured signal is very noisy. It seems that the output is of a better visual quality compared to the output of a network having 30 operations. In contrast to the output of an accurate median, there are some small oscillations around 1.7 seconds caused by the oscillations in a signal coming from the accelerometer. Nevertheless, the trend in data is reliably followed. In case these small differences do not represent a real problem for a target application, it is worth implementing the improved median network which is able to offer a 40 % reduction of power consumption on the one hand, its approximately 1.8 times faster execution time on the other hand.

The approximate versions of the 13-input median also performs very well. Only small differences are observable when a median network with 38 % removed

operations is used. Compared to a commonly used running median, we obtain a solution which has 38 % lower power consumption when implemented on a STM32 microcontroller. Interestingly, the approximate median networks which consists of 17 and 58 operations exhibit lower delay compared to the fully working 13-input median. It can be seen that the filtered data appears to be shifted to the left when these filters are used.

It can be concluded that the observations on real examples are consistent with conclusions given in Sect. 7. As the quality of an approximate median defined by Eq. 1 decreases, the amount of inaccuracies in the output signal increases. The processing of the sensor data seems to be an application with great potential for genetic improvement. As was previously shown, we are able to significantly improve energy consumed by the filters for a cost of small differences in the output data. In fact, any of the presented approximations can be used to filter the input signal because no golden solution is available for the validation of the obtained outputs. The filtration is typically used to avoid high variances in output signal (i.e. to reduce sensitivity of the output signal to the outliers). In this sense, we can employ approximate medians consisting of 50 % (or even less) operations to accomplish this task because they are able to sufficiently remove the outliers.

8.4 Median in image processing

The processes of acquiring, transmitting and storing images in computer systems are not always ideal and hence some pixels or groups of pixels can be corrupted. Hence, noise elimination is a typical low level image processing task. In many applications, the noise elimination has to be implemented by non-linear functions because the noise contained in the images is inherently non-linear [6]. A typical representative of non-linear noise is a shot noise which manifests itself by setting some individual pixels to a random value. Median-based non-linear filters play a prominent role among the filters utilized to suppress the shot noise [2]. Traditionally, a simple median filter applied to every pixel of the input image is employed. In advanced image filters (e.g. switching filters [32]) the filtering function is only applied if a noise detector, implemented typically using a median, detects some noise.

The image filters operate with pixel values in the neighbourhood of the centre pixel. The process of filtration is based on a sliding window, a square window of an odd size $(2k + 1)$, that moves along the image. More formally, let I be an image consisting of $m \times n$ pixels $x(i,j) \in I$, where $1 \leq i \leq m, 1 \leq j \leq n$. Then, each pixel of the filtered image I' is calculated as $y(m,n) = \text{median}(W_I(m,n))$, where $W_I(m,n) = \{x(m+i, n+j) \in I \mid -k < i,j < k\}$ is a sliding window function. It is evident that the median value is calculated using $(2k+1)^2$ pixels. The typical sliding windows employed in image processing consists of 3×3 and 5×5 pixels which corresponds with 9-input and 25-input filtering functions.

The measured non-functional parameters of various implementations of 9-input accurate as well as approximate median filters are summarized in Table 5. Apart from the evolved implementations, two common approaches to determine a median value are evaluated—the quicksort algorithm and the quickselect algorithm. The

Table 5 Non-functional parameters of accurate (emphasized) and approximated implementations of 9-input median function measured on different MCUs

Impl.	Machine code size [B]				Execution time [µs]				Consumed energy [nWs]			
	STM32	PIC24	PIC16	TI430	STM32	PIC24	PIC16	TI430	STM32	PIC24	PIC16	TI430
9-ops	78	204	207	96	3.2	65	228	274	97	450	457	220
10-ops	84	234	238	108	3.3	71	256	280	102	492	512	225
14-ops	112	315	324	156	3.9	86	338	310	118	590	675	249
16-ops	126	372	376	176	4.1	96	386	324	126	666	771	260
20-ops	158	441	454	208	4.6	108	452	340	141	745	905	273
22-ops	180	495	502	234	5.0	118	506	360	151	818	1012	289
26-ops	208	573	586	280	5.4	132	576	388	165	909	1153	312
30-ops	240	645	676	330	6.4	144	650	412	196	994	1299	331
qsort	128	333	–	196	26.8	830	–	1325	816	5727	–	1064
qselect	212	849	607	272	15.3	466	2255	690	467	3219	4510	554

running median discussed in the previous section is not applicable in this case because more than one value has to be removed/added between two subsequent processing windows. The discussion that has been given for the implementation of the 11-input median and its variants is also valid for the 9-input alternative whose parameters are included in Table 5. There is nearly a linear dependency between the number of operations used to approximate the median value and the execution time as well as power consumption.

The results for the 25-input median and its alternative implementations are given in Table 6. In contrast with the 13-input approximate medians, the difference between the improvement achieved at the level of operations and improvement at the level of instructions does not exceed 5 %. Similarly, the time of execution decreases linearly with a decreasing number of operations with one exception—implementation compiled for MSP430 which suffers from issues observed also for 13-input approximate medians. There is an 18 % difference between the reduction at the level of instructions and the reduction of execution time (see the execution time for 174-ops and 60-ops implementations). Since the response of other architectures to a reduced number of operations is as expected, it may suggest that there may be a problem with the quality of the compiled code. We did not analyse this problem in detail as it is outside the scope of this paper.

The chosen problem nicely demonstrates the overhead of median networks implemented in the software. The accurate median function implemented using 174 operations occupies ten times more bytes than the quicksort algorithm. Even if we remove half of the operations, the machine code is more than six times larger. This is the price that must be sacrificed for greater speed of the algorithm based on a median network. As regards the execution time, the median can be calculated 70 % faster when the median network which consists of 174 operations is used instead of the quicksort algorithm and 31 % faster when compared to the quickselect

Table 6 Non-functional parameters of accurate (emphasized) and approximated implementations of 25-input median function measured on different MCUs

Impl.	Machine code size [B]			Execution time [μs]			Consumed energy [nWs]		
	STM32	PIC24	TI430	STM32	PIC24	TI430	STM32	PIC24	TI430
60-ops	502	1302	742	10.9	262	665	333	1808	534
70-ops	596	1527	912	12.3	303	785	375	2091	630
88-ops	796	1887	1180	16.4	366	955	501	2525	767
107-ops	920	2250	1438	18.4	428	1100	562	2953	883
150-ops	1264	3015	1688	23.9	554	1130	727	3823	907
160-ops	1378	3195	1818	24.6	584	1200	751	4030	964
164-ops	1454	3255	1826	26.0	596	1240	793	4109	996
174-ops	1524	3423	1864	27.6	619	1270	841	4271	1020
qsort	128	333	196	104.0	2430	2610	3172	16,767	2096
qselect	212	849	276	39.1	1040	1685	1194	7176	1353

Note that PIC16 is not included in this table due to small amount of available RAM memory

algorithm. The 25-median implemented using 150 operations enables us to reduce the energy by more than 10 %. According to the distribution of errors shown in Fig. 5, this implementation provides an output of high quality with a low percentage of erroneous outputs that are close to the median value.

In order to evaluate the filtering quality as well as robustness of the evolved approximate medians, the medians were employed as median filters and evaluated using 25 randomly selected test images (384x256 pixels) from [17] that were corrupted by random valued shot noise. Because the removal of random valued shot noise represents a difficult problem, it usually is used to compare the performance of various median filters [5]. Considering the fact that a sliding window is used, more than two million test cases were in fact used for quality assessment. There exists several approaches to measure the quality of filtered images. The structural similarity index (SSIM) represents probably the most advanced approach which attempts to quantify the visibility of errors (differences) between a distorted image and a reference image[37].

Boxplots of the structural similarity index calculated for 9-input and 25-input accurate as well as approximate median networks used as image filters are given in Fig. 7. As is evident from the results, there is a relatively large variance in the similarity index of accurate as well as inaccurate median filters. The index of similarity for images produced by accurate an 9-input median filter is 88.6 % in average. The average similarity index decreases with the decreasing number of operations. Interestingly, it decreases very slightly without any radical change in variance. When we reduce the number of operation to 16 (53 %), the average similarity index decreases to 87.5 %. The results suggest that it is possible to use an approximate median network consisting of half the number of operations instead of an accurate median. The impact on quality of the filtered images is negligible.

Figure 8 illustrates filtering capabilities of various filters on an image corrupted by random valued shot noise where 10 % of the pixels are affected. The output of the median filter and approximate median filter is visually indistinguishable. Nearly all of the noisy pixels were successfully detected and removed, even for a median with 16 operations. When we reduce the number of operations to 14, a few noisy pixels remain in the filtered image. This behaviour corresponds with the distribution

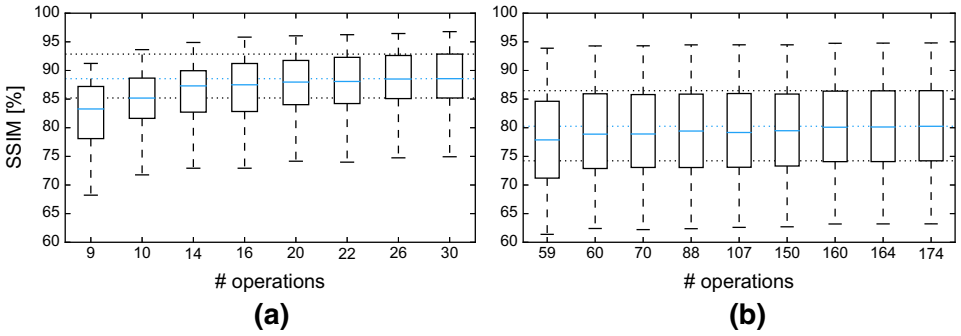

Fig. 7 *Boxplots* illustrating the distribution of structural similarity index for evolved median networks calculated using a set of test images corrupted by 10 % random valued shot noise. **a** 9-input median. **b** 25-Input median

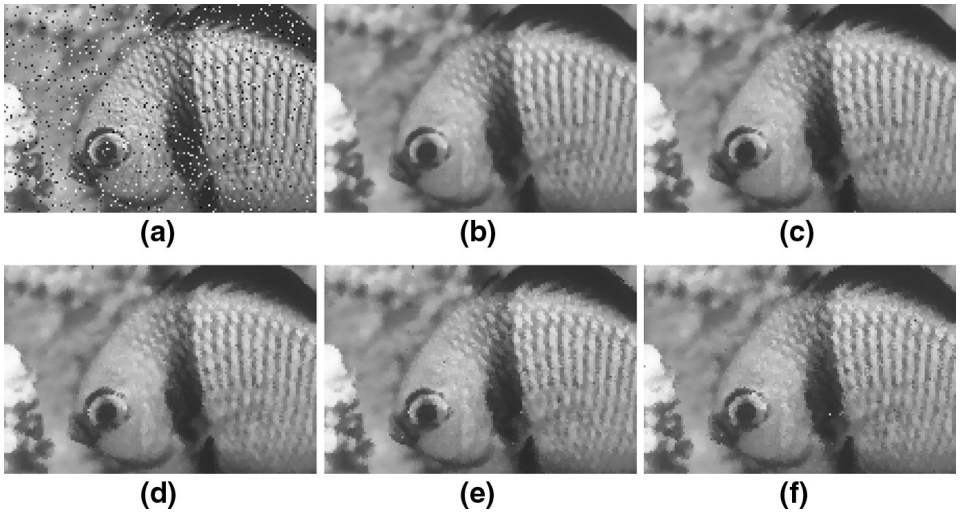

Fig. 8 Detail of an image **a** corrupted by 10 % random valued shot noise filtered by **b** 9-input accurate median filter and approximated median filters consisting of **c** 22 (73 %) operations, **d** 16 (53 %), **e** 14 (46 %) and **f** 10 (33 %) operations

of errors shown in Fig. 5 and a detailed analysis provided in Table 2. The 9-input approximate median with 16 operations exhibits the worst-case distance error equal to one, while the 14-ops implementation has the worst-case distance error equal to two.

If we compare the distribution of the similarity index for a 9-input and 25-input median filter, it is evident that the 25-input median filters provide results of lower quality. The similarity index of the accurate implementation consisting of 174 operations is equal to 80.3 %. The reason is obvious. Increasing the size of the filtering window allows for the common median filter to remove a great deal of noisy pixels, however, because the standard median filters modify almost all pixels, images become smudged and less detailed. Nevertheless, this fact does not prevent the employment of the 25-input median filter as a robust noise detector. Interestingly, there is only a small degradation in quality of the reduced 25-input median filters. When we remove 50 % of operations, the similarity index decreases to 79.4 % on average. This approximation yields a 40 % reduction in power consumption when implemented on STM32 microcontroller.

Similar conclusions may be inferred even if we use the peak signal-to-noise ratio (PSNR) which represents another commonly used quality metric. In contrast to structural similarity, PSNR does not respect a psycho-visual model of the human optical system. While PSNR of the images filtered by the accurate 9-input median filter is equal to 29.4 dB in average, PSNR of the images obtained by the 14-ops (9-ops) filter drops by 1.3 dB (3.5 dB). The PSNR of the images filtered by the accurate 25-input median filter is equal to 25.9 dB. When the number of operations is reduced to 59, the PSNR only decreases by 0.7 dB.

The results demonstrate how robust the evolved implementations are and that there is great space for improvement of the non-functional parameters in practice. In

most cases, it is not even necessary to exactly determine the median value which helps us to reduce the power consumption or increase the performance (i.e. speed) of a given piece of software.

9 Conclusions

In this paper, we presented a new approach to improve non-functional properties of software. In particular, we concentrated on improvements in the execution time and power consumption of various instances of the median function. In general, it is impossible to improve non-functional parameters of the median function without accepting occasional errors in results since optimal implementations of typical instances are available. In order to address this problem, we adopted the approximate computing scenario which allows us to accept some errors in the outputs.

In approximate computing, software and hardware is approximated, i.e. simplified with respect to fully accurate implementations, in order to reduce power consumption or increase performance. As a consequence, errors can emerge during computations which is tolerable in many real applications. When an approximation should be introduced, the common approach is to remove the less significant bits and reduce data widths. This paper shows that the approximation conducted at the level of function (algorithm) that are based on EA is able to deliver significantly better results.

The median is implemented using a sequence of elementary operations that forms a median network. The task is formulated as a single objective optimization problem where the number of operations represents constraints specified by the designer. The constrains-oriented approach is relevant to practice where the designers usually wants to achieve a particular power reduction in order to improve the performance of the whole embedded system. The method is based on cartesian GP and exploits the fact that GP is able to find a good trade-off between the error and number of operations, even if the number of operations is intentionally constrained.

In order to avoid problem with seeding (only fully functional implementations of various instances of median filter exist), we proposed to apply a two-stage procedure. The first stage starts with a fully functional median network and gradually reduces the number of employed operations in order to satisfy constraints given by a designer. As soon as a satisfactory candidate solution is found, the second stage responsible for maximizing the quality of partially working implementations is used.

The accuracy of determining a median value is measured by means of a problem-specific quality metric. The proposed metric is based on the positional error calculated using the permutation principle introduced in this paper. The impact of the permutation principle was discussed from a theoretical as well as a practical point of view. Firstly, the permutation principle helps us to reduce the computational complexity of the fitness evaluation. Secondly, the permutation principle enables one to construct a metric approaching the quality of selecting the median value and, what is important, which can be efficiently calculated. Finally, the

permutation principle helps to understand how to avoid biased solutions that may be produced when we generate test vectors used to determine the fitness score inappropriately (randomly). In order to understand this phenomenon, it is necessary to realize that the median value is determined according to a set of values (i.e. the ordering of input values is completely ignored). It was illustrated and discussed that it is necessary to generate test vectors from different equivalence classes so as to avoid any bias.

The problem of trading between quality and non-functional parameters was demonstrated in four different instances of the median function that are typically employed in practice. The performance of the best discovered approximated median filters was evaluated in two real-world problems—sensor data processing and image processing. The non-functional parameters were measured for four microcontrollers so as to avoid misleading conclusions. The results confirmed that median functions are very good examples of functions for which it makes sense to introduce their approximate versions. When the approximate medians are employed in a particular application, the output quality remains relatively high, even for significant reductions of the number of operations. Hence significant improvements in energy consumption can be obtained.

Even though the permutation principle as well as the proposed error metric are problem specific, this paper demonstrated the ability of GI to provide competitive solutions for a chosen real-world problem from the area of approximate computing. This opens a complete new application area for GI. The ability to deliver partially working solutions seems to be natural for evolutionary techniques. Hence the approximate computing seems to have a great potential for these techniques.

There are several directions for future research. Execution time and power consumption are two possible non-functional criteria that can be optimized. There are additional criteria such as delay that need to be considered, especially if median networks would be implemented in the hardware. Despite the fact that the proposed permutation principle helps to significantly improve the time required to determine the fitness value, the test based approach used to calculate the fitness score represents a bottleneck of the whole framework. Unfortunately, this is a general problem of all generate-and-test-based evolutionary approaches. As a consequence of that, only a subset of all possible permutations was used for quality assessment. This simplification introduces two issues. Firstly, it means that we are not able to guarantee the worst-case error unless all the input permutations are tested. Secondly, it may happen that the quality of a given network is worse than determined. Suprisingly, the experiments revealed that our simplification does not have any significant effect in practice. We are convinced, however, that these issues can be completely eliminated by introducing a formal method based on BDDs to the fitness function.

Acknowledgments This work was supported by the Czech science foundation project 14-04197S—Advanced Methods for Evolutionary Design of Complex Digital Circuits.

References

1. A. Agapitos, S.M. Lucas, Evolving efficient recursive sorting algorithms, in *IEEE Congress on Evolutionary Computation*, pp. 2677–2684 (2006)
2. R.H. Chan, C.W. Ho, M. Nikolova, Salt-and-pepper noise removal by median-type noise detectors and edge-preserving regularization. IEEE Trans. Image Process. **14**, 1479–1485 (2005)
3. B. Cody-Kenny, E.G. Lopez, S. Barrett, locoGP: improving performance by genetic programming java source code, in *Genetic Improvement 2015 Workshop*, ed. by W.B. Langdon, J. Petke, D.R. White (ACM, Madrid, 2015), pp. 811–818
4. N. Devillard, *Fast Median Search: An ANSI C Implementation* (1998). http://ndevilla.free.fr/median/median.pdf
5. Y. Dong, A new directional weighted median filter for removal of random-valued impulse noise. IEEE Signal Process. Lett. **14**(3), 193–196 (2007)
6. E.R. Dougherty, J.T. Astola, (eds.) *Nonlinear Filters for Image Processing. SPIE/IEEE Series on Imaging Science and Engineering*. SPIE/IEEE (1999)
7. H. Esmaeilzadeh, A. Sampson, L. Ceze, D. Burger, Neural acceleration for general-purpose approximate programs. Commun. ACM **58**(1), 105–115 (2014)
8. B.W. Goldman, W.F. Punch, Analysis of cartesian genetic programming's evolutionary mechanisms. IEEE Trans. Evol. Comput. **19**(3), 359–373 (2015)
9. J. Han, M. Orshansky, Approximate computing: An emerging paradigm for energy-efficient design, in *Proceedings of the 18th IEEE European Test Symposium*, pp. 1–6. IEEE (2013)
10. M. Harman, B.J. Jones, Search-based software engineering. Inf. Softw. Technol. **43**, 833–839 (2001)
11. W.D. Hillis, Co-evolving parasites improve simulated evolution as an optimization procedure. Phys. D **42**(1–3), 228–234 (1990)
12. H. Juille, Evolution of non-deterministic incremental algorithms as a new approach for search in state spaces, in *Genetic Algorithms: Proceedings of the 6th International Conference (ICGA95)*, ed. by L. Eshelman (Morgan Kaufmann, Pittsburgh, PA, USA, 1995), pp. 351–358
13. R.E. Kalman, A new approach to linear filtering and prediction problems. Trans. ASME J. Basic Eng. **82**(Series D), 35–45 (1960)
14. D.E. Knuth, *The Art of Computer Programming*, vol. 3, 2nd edn. (Sorting and Searching. Addison Wesley Longman Publishing Co., Inc, Redwood City, 1998)
15. W.B. Langdon, M. Harman, Optimizing existing software with genetic programming. IEEE Trans. Evol. Comput. **19**(1), 118–135 (2015)
16. R. Maronna, D. Martin, V. Yohai, *Robust Statistics: Theory and Methods, Wiley Series in Probability and Statistics* (Wiley, New Jersey, 2006)
17. D. Martin, C. Fowlkes, D. Tal, J. Malik, A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics, in *Proceedings of the 8th International Conference Computer Vision*, vol. 2, pp. 416–423 (2001)
18. J.F. Miller, *Cartesian Genetic Programming* (Springer, Berlin, 2011)
19. J.F. Miller, S.L. Smith, Redundancy and computational efficiency in cartesian genetic programming. IEEE Trans. Evol. Comput. **10**(2), 167–174 (2006)
20. V. Mrazek, Z. Vasicek, L. Sekanina, Evolutionary approximation of software for embedded systems: Median function, in *Genetic Improvement 2015 Workshop*, ed. by W.B. Langdon, J. Petke, D.R. White (ACM, Madrid, 2015), pp. 795–801
21. K. Nepal, Y. Li, R.I. Bahar, S. Reda, Abacus: A technique for automated behavioral synthesis of approximate computing circuits, in *Proceedings of the Conference on Design, Automation and Test in Europe, DATE '14*, pp. 1–6. EDA Consortium (2014)
22. J. Petke, M. Harman, W.B. Langdon, W. Weimer, Using genetic improvement and code transplants to specialise a C++ program to a problem class, in *17th European Conference on Genetic Programming, LNCS*, vol. 8599, ed. by Miguel Nicolau, et al. (Springer, Granada, Spain, 2014), pp. 137–149
23. R. Poli, W.B. Langdon, N.F. McPhee, *A Field Guide to Genetic Programming*.Published via http://lulu.com and http://www.gp-field-guide.org.uk (2008)
24. A. Sampson, W. Dietl, E. Fortuna, Gnanapragasam, D., Ceze, L., Grossman, D.: Enerj: Approximate data types for safe and general low-power computation, in *Proceedings of the 32nd ACM SIGPLAN Conference on Programming Language Design and Implementation*, pp. 164–174. ACM (2011)

25. P. Schmidt, Simple median filter library designed for the arduino platform (2014). https://github.com/daPhoosa/MedianFilter
26. E. Schulte, J. Dorn, S. Harding, S. Forrest, W. Weimer, Post-compiler software optimization for reducing energy, in *Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS'14 (ACM, Salt Lake City, 2014), pp. 639–652
27. L. Sekanina, Evolutionary design space exploration for median circuits, in *Applications of Evolutionary Computing, LNCS 3005*, pp. 240–249. Springer (2004)
28. L. Sekanina, M. Bidlo, Evolutionary design of arbitrarily large sorting networks using development. Genet. Progr. Evolv. Mach. **6**(3), 319–347 (2005)
29. L. Sekanina, Z. Vasicek, Approximate circuits by means of evolvable hardware. in *Proceedings of the 2013 IEEE Symposium Series on Computational Intelligence (SSCI), 2013 IEEE International Conference on Evolvable Systems*, pp. 21–28. IEEE CIS (2013)
30. P. Sitthi-Amorn, N. Modly, W. Weimer, J. Lawrence, Genetic programming for shader simplification. ACM Trans. Gr. **30**(6), 152:1–152:12 (2011)
31. J.L. Smith, Implementing median filters in xc4000e fpgas. XCell **23**(1), 16 (1996)
32. T. Sun, Y. Neuvo, Detail-preserving median based filters in image processing. Pattern Recognit. Lett. **16**, 341–347 (1994)
33. V.K. Valsalam, R. Miikkulainen, Using symmetry and evolutionary search to minimize sorting networks. J. Mach. Learn. Res. **14**(1), 303–331 (2013)
34. Z. Vasicek, L. Sekanina, Evolutionary approach to approximate digital circuits design. IEEE Trans. Evol. Comput. **19**(3), 432–444 (2015)
35. Z. Vasicek, K. Slany, Efficient phenotype evaluation in cartesian genetic programming, in *Proceedings of the 15th European Conference on Genetic Programming, LNCS 7244*, pp. 266–278. Springer Verlag (2012)
36. S. Venkataramani, A. Sabne, V.J. Kozhikkottu, K. Roy, A. Raghunathan, Salsa: systematic logic synthesis of approximate circuits, in *The 49th Annual Design Automation Conference 2012*, DAC '12, pp. 796–801. ACM (2012)
37. Z. Wang, A. Bovik, H. Sheikh, E. Simoncelli, Image quality assessment: from error visibility to structural similarity. IEEE Trans. Image Process. **13**(4), 600–612 (2004)
38. D.R. White, A. Arcuri, A. John, Evolutionary improvement of programs. IEEE Trans. Evol. Comput. **15**(4), 515–538 (2011)
39. A. Yazdanbakhsh, D. Mahajan, B. Thwaites, J. Park, A. Nagendrakumar, S. Sethuraman, K. Ramkrishnan, N. Ravindran, R. Jariwala, A. Rahimi, H. Esmailzadeh, K. Bazargan, Axilog: Language support for approximate hardware design, in *Design, Automation and Test in Europe, DATE'15*, pp. 1–6. EDA Consortium (2015)