VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

OPTIMALIZACE TESTU DIGITÁLNÍHO OBVODU MULTIFUNKČNÍMI PRVKY

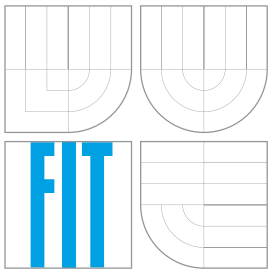
DISERTAČNÍ PRÁCE
PHD THESIS

AUTOR PRÁCE
AUTHOR

Ing. LUKÁŠ STAREČEK

BRNO 2012

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

OPTIMALIZACE TESTU DIGITÁLNÍHO OBVODU MULTIFUNKČNÍMI PRVKY

DIGITAL CIRCUITS TEST OPTIMIZATION BY MULTIFUNCTIONAL COMPONENTS

DISERTAČNÍ PRÁCE
PHD THESIS

AUTOR PRÁCE — Ing. LUKÁŠ STAREČEK
AUTHOR

VEDOUCÍ PRÁCE — Doc. Ing. ZDENĚK KOTÁSEK, CSc.
SUPERVISOR

BRNO 2012

Abstrakt

Tato práce se zabývá možností optimalizace testu číslicových obvodů pomocí multifunkčních logických hradel. Nejdůležitější částí práce je vysvětlení samotného principu optimalizace, který je popsán také formálními matematickými prostředky. Na základě tohoto popisu je v práci prezentováno několik možností využití. Ukázána je optimalizace testovatelnosti obdobná metodě vkládání testovacích bodů a jednoduchá metodika založena na základě SCOAP. Těžištěm práce je však metodika, která byla vytvořena pro optimalizaci testu obvodu. Ta byla implementována v podobě softwarových nástrojů. V práci jsou následně prezentovány výsledky použití těchto nástrojů na úloze snížení počtu testovacích vektorů se zachováním pokrytí poruch pro různé obvody včetně testovací sady ISCAS 85.

Část práce je věnována také různým principům a technologiím tvorby multifunkčních logických hradel. Některá vybraná hradla z těchto technologií jsou podrobena simulacím elektronických vlastností ve SPICE. Na základě principů prezentované metodiky a výsledků simulací multifunkčních hradel je také provedena analýza a rozbor různých problémů jako je platnost testu modifikovaného obvodu a vhodnost jednotlivých technologií multifunkčních hradel pro danou metodiku.

Výsledky analýz a provedených experimentů je potvrzeno, že pomocí multifunkčních hradel lze optimalizovat diagnostické vlastnosti obvodu takovým způsobem, aby došlo k požadovaným úpravám parametrů výsledných testů obvodů při minimálních dopadech na kvalitu a věrohodnost těchto testů.

Klíčová slova

číslicový obvod, testovací vektory, optimalizace testu, multifunkční hradla, polymorfní hradla

Abstract

This thesis deals with the possibilities of digital circuit test optimization using multifunctional logic gates. The most important part of this thesis is the explanation of the optimization principle, which is also described by a formal mathematical apparatus. Based on this apparatus, the work presents several options. The optimization of testability analogous to inserting test points and simple methodology based on SCOAP is shown. The focus of work is a methodology created to optimize circuit tests. It was implemented in the form of software tools. Presented in this work are the results of using these tools to reduce the test vectors volume while maintaining fault coverage on various circuits, including circuits from the ISCAS 85 test set.

Part of the work is devoted to the various principles and technology of creating multifunctional logic gates. Some selected gates of these technologies are subject to simulations of electronic properties in SPICE. Based on the principles of presented methodology and results of multifunctional gates simulations, analysis of various problems such as validity of the modified circuit test and the suitability of each multifunctional gate technology for the methodology was also made.

The results of analysis and experiments confirm it is possible for the multifunctional logic gate to optimize circuit diagnostic properties in such a way that has achieved the required circuit test parameter modification with minimum impact on the quality and credibility of these tests.

Keywords

digital circuit, test vectors, test optimization, multifunctional gates, polymorphic gates

Citace

Lukáš Stareček: Optimalizace testu digitálního obvodu multifunkčními prvky, disertační práce, Brno, FIT VUT v Brně, 2012

Optimalizace testu digitálního obvodu multifunkčními prvky

Prohlášení

Prohlašuji, že jsem tuto doktorskou práci vypracoval samostatně pod vedením pana Doc. Ing. Zdeňka Kotáska CSc. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Lukáš Stareček
22. března 2012

Poděkování

Na tomto místě bych rád poděkoval svému školiteli Doc. Ing. Zdeňku Kotáskovi CSc. za příkladné vedení mého studia a zejména za jeho trpělivost, kterou se mnou musel mít. Dále bych chtěl poděkovat Prof. Ing. Lukáši Sekaninovi Ph.D. za cené rady, nápady a konzultace. Poděkovat bych chtěl také své ženě Ing. Terezii Starečkové Ph.D. za toleranci mé dlouhé časové indispozice. Mé ženě, stejně jako Ing. Jiřímu Vrbovi Ph.D. patří také díky za jejich neúnavnou nápravu mých češtinářských a věcných přehmatů v této práci. Poděkovat bych chtěl také svým dvěma rodičům a poslednímu prarodiči, kteří ve vážných nemocech tolerují moji sníženou starostlivost a omezenou péči.

Práce byla podporována z projektu GAČR, GA102/06/0599 s názvem "Metody návrhu polymorfních číslicových obvodů".

Obsah

Kapitola 1

Úvod

1.1 Úvod

Jednou z charakteristik dvacátého století je velmi rychlý rozvoj v různých technologických oborech. Jednoho z největších rozmachů dosáhla elektronika, za jejíž vznik se často považuje rok přibližně 1906. V této době Lee De Forest vyvinul triodu, kterou roku 1907 nechal patentovat [13]. Od vzniku elektroniky do současnosti bylo vyvinuto velké množství elektronických komponent, jejichž složitost neustále stoupá. Nejsložitější integrované obvody dneška obsahují miliardy tranzistorů (viz např. NVIDIA Fermi s cca třemi miliardami tranzistorů [41]). Charakteristiku exponenciálně rostoucí složitosti elektronických komponent poprvé formuloval spoluzakladatel firmy Intel Gordon E. Moore v roce 1965, kdy uvedl, že složitost součástek se přibližně každý rok zdvojnásobí při zachování stejné ceny [34]. Tento výrok byl později přeformulován do podoby nazývané „Mooreův zákon“ [33] se zněním „počet tranzistorů, které mohou být vloženy do integrovaného obvodu, se přibližně každé dva roky zdvojnásobí“. Od doby formulace Mooreova zákona do současnosti bylo již mnohokrát předpokládáno, že jeho platnost brzy skončí, většinou z důvodů technologických limitů. Zatím však pokaždé byly technologické komplikace vyřešeny, limity prolomeny a v současné době se předpokládá, že platnost tohoto zákona zůstane zachována minimálně do roku 2015 [23]. Mnozí vědci věří, že platnost zůstane zachována minimálně po dobu dalších dvou desetiletí.

Aby bylo možné udržet růst míry integrace (a tedy i platnost Mooreova zákona), je třeba neustále zdokonalovat a vylepšovat tvorbu elektronických komponent a to ve všech aspektech od návrhu až po použití. Při návrhu se využívají různé úrovně abstrakcí a modelů s použitím vyspělých návrhových systémů, ve výrobě se hledají nové technologie a postupy.

Rozsáhlou oblastí je problém diagnostiky. Vzhledem ke složitosti současných obvodů není technologicky možné, aby byly všechny vyrobené kusy funkční. Z hlediska ekonomických nákladů je detekce vadného kusu velmi důležitá. Čím později se defekt odhalí, tím bývá náprava nákladnější. Současně, i když je obvod shledán plně funkčním a je nasazen do reálného použití, může dojít ať už skrytou chybou, vnějšími jevy nebo stárnutím, k jeho poškození takovým způsobem, kdy už není schopen plnit svoji funkci. V závislosti na důležitosti funkce může být i zde důležitá včasná detekce poškození. Rostoucí složitost obvodů však testování komplikuje a z otestování složitého obvodu se stává složitý problém. Pro test obvodu je důležité, aby při obhajitelných nákladech byl test dostatečně účinný. Jsou tedy vyvíjeny techniky a postupy tvorby testů obvodů, jejichž kvality jsou poměřovány metrikami, jako je počet pokrytí poruch, cena testu, doba aplikace testu, příkon potřebný pro provedení testu atp. Důležitost každé metriky se může lišit dle určení testovaného ob-

vodu. Ukazuje se, že u složitějších obvodů vyráběných ve větších sériích začíná být velmi důležitým parametrem doba aplikace testu. Jan Dohnal z ON Semiconductor uvedl, že doba testování obvodů zabírá přibližně třetinu času z celého procesu výroby. Zkrácení doby testu tak může zrychlit výrobu a ušetřit část prostředků.

Hlavním cílem této práce je vytvoření nové metodiky pro zkrácení doby aplikace testu obvodu. Důraz je kladen na jednoduchost a obecnost metodiky a její použitelnost na komplexní obvody. Aplikace nové metodiky by také neměla mít větší negativní vliv na ostatní metriky testu. Předpokladem je vznik nové obecně použitelné metodiky, která umožní zrychlit výrobu a snížit tak náklady.

1.2 Struktura práce

V kapitole 1 je úvod do problematiky a struktura práce. Kapitola 2 obsahuje základní znalosti v oblastech týkajících se této práce a je rozdělena na tři části. První se zabývá problematikou elektronických obvodů se vztahem zejména k číslicové elektronice. Za nejdůležitější lze považovat části týkající se tranzistorů, logických hradel a simulace elektronických obvodů. Ve druhé části jsou uvedny principy testování a diagnostiky číslicových obvodů. Jsou zde vysvětleny principy a postupy testování, modelování poruch, metriky testů a principy ohledně analýzy testovatelnosti. V poslední třetí části jsou představeny principy optimalizace a konkrétní použité algoritmy.

Cílem kapitoly 3 je představit aktuální stav v oblastech multifunkční elektroniky. Věnuje se multifunkčním logickým hradlům a technologickým principům jejich tvorby. Jsou zde diskutovány technologie tvorby hradel konvenčním způsobem, které se opírají o standardní návrh dle obecných principů CMOS technologie. Dále polymorfní hradla, která jsou zkoumána od roku 2000 v NASA Propulsion Laboratory využívající vlastnosti MOSFET tranzistorů, které se běžně pro změnu funkce nevyužívají. Jako poslední je diskutována technologie založená na grafenu, která je zkoumána přibližně od roku 2004 a má potenciál stát se náhradou nebo alespoň doplňkem dnešní křemíkové technologie.

V kapitole 4 jsou vysloveny předpoklady a na jejich základě stanoveny cíle práce. Kapitola 5 je souhrnem mé práce v oblasti multifunkčních hradel. Jsou zde uvedena některá nová mnou vytvořená multifunkční hradla a jsou zde také uvedeny a diskutovány elektrické vlastnosti a výsledky simulací prezentovaných hradel. První část kapitoly se týká hradel konvenčních, druhá rozsáhlejší část se zabývá hradly polymorfními.

Jednou z nejdůležitějších částí práce je kapitola 6. Ta prezentuje principy nové navrhované metodiky a je rozdělena na tři podkapitoly. První podkapitola se zabývá samotným principem optimalizace testu multifunkčními hradly a formálně tento princip definuje. Ve druhé podkapitole jsou pak ukázány možnosti optimalizace testovatelnosti obvodu. V poslední třetí podkapitole je stěžejní část, která se zabývá obecnou optimalizací testu obvodu. Popisuje princip metody a implemnetaci nástrojů pro její ověření.

Kapitola 7 navazuje na kapitolu předchozí a obsahuje výsledky dosažené navrhovanou metodou optimalizace testu. Jsou zde prezentovány výsledky mnou implementovaných optimalizačních algoritmů kompletního a rekurzivního prohledávání, jako i evoluční metody implementované Ing. Šimáčkem pod vedením Prof. Ing. Lukáše Sekaniny Ph.D. Metody byly vyzkoušeny na různých obvodech včetně testovací sady ISCAS 85. Na závěr kapitoly jsou zhodnocena nalezená řešení a je zde diskutována i výkonostní stránka použitých nástrojů.

Poslední kapitolou před samotným závěrem je kapitola 8. Ta je rozdělena na dvě části. První část se zabývá důležitou problematikou ohledně vlastností navrhované metodiky a

věrohodnosti testu modifikovaného obvodu. Ve druhé části je pak diskutována vhodnost jednotlivých technologií tvorby multifunkčních hradel pro navrhovanou metodiku.

Poslední kapitola 9 je závěrečnou kapitolou, která zhodnocuje celou metodu, dosažené výsledky a uvádí některé možné směry případných navazujících prací.

Kapitola 2

Prerekvizity

2.1 Elektronické obvody

Za vznik samostatného oboru zvaného elektronika, se považuje rok přibližně 1906. V této době Lee De Forest vyvinul triodu, kterou roku 1907 nechal patentovat [13]. Od vzniku elektroniky do současnosti bylo vyvinuto velké množství elektronických komponent, jejichž složitost neustále stoupá. Nejsložitější integrované obvody dneška obsahují miliardy tranzistorů (viz např. NVIDIA Fermi s cca třemi miliardami tranzistorů [41]). Charakteristiku rychle rostoucí složitosti elektronických komponent poprvé formuloval spoluzakladatel firmy Intel Gordon E. Moore v roce 1965 [34]. Jeho výrok byl později přeformulován do podoby v dnešní době známé jako „Mooreův zákon“ [33], který zní „počet tranzistorů, které mohou být vloženy do integrovaného obvodu, se přibližně každé dva roky zdvojnásobí“. V současné době se předpokládá, že platnost Mooreova zákona zůstane zachována minimálně do roku 2015 [23] a mnozí vědci věří, že platnost zůstane zachována minimálně po dobu dalších dvou desetiletí.

Elektronické obvody lze rozdělit do dvou základních skupin a to na analogové a digitální (číslicové). Zatímco analogové obvody pracují se spojitým rozsahem napětí, číslicové pracují s diskrétními hodnotami napětí. Tato práce se bude zabývat číslicovými obvody, zejména pak obvody integrovanými.

2.1.1 Tranzistory

První popis zařízení s funkcí obdobnou FET tranzistoru patentoval Julius Edgar Lilienfeld v roce 1925 [29]. Podobné zařízení poté patentoval také Oscar Heil v roce 1934 [21]. První funkční tranzistor prezentoval John Bardeen, Walter Brattain a William Shockley 23. prosince 1947 v AT&T Bellových laboratořích v New Jersey, za který v roce 1956 obdrželi nobelovu cenu [45]. První křemíkový tranzistor vyrobil Texas Instruments v roce 1954 [51]. Vynález tranzistoru se často považuje za jeden z největších vynálezů dvacátého století [45] a je klíčovou komponentou prakticky ve všech moderních elektronických zařízeních.

Každý tranzistor má (nejméně) tři elektrody a jejich základem jsou dva polovodičové PN přechody. V závislosti na tom, jaké typy nosičů elektrického náboje se podílí na funkci, rozlišujeme dva typy tranzistorů:

Bipolární - Využívá obou druhů nosičů elektrického náboje. Jeho elektrody se nazývají kolektor, báze a emitor a velikostí proudu báze se ovládá proud tekoucí mezi kolektorem a emitorem.

Unipolární - Nazývaný též jako „tranzistor řízený elektrickým polem“ (FET). Využívá pouze jednoho druhu nosičů elektrického náboje. Jeho elektrody se nazývají kolektor (angl. drain), hradlo (angl. gate), emitor (angl. source) a substrát (angl. bulk). Jelikož hradlo jakožto řídicí elektroda má velký vstupní odpor, tak jí neprochází (na rozdíl od bipolárního tranzistoru) téměř žádný proud. Tranzistor je tedy řízen velikostí napětí hradlo-emitor, které ovládá proud tekoucí mezi kolektorem a emitorem.

Dále lze tranzistory kategorizovat dle:

Polovodičového materiálu - Křemík (Si), germanium (Ge), galiumarsenid (GaAs), karborundum (karbid křemíku - SiC) atd.

Struktury - Bipolární BJT a IGBT. Unipolárni JFET, MESFET, MOSFET a MISFET.

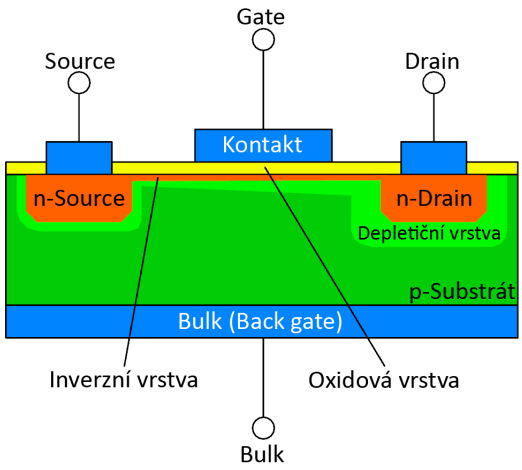
Polarity - Bipolární NPN a PNP, unipolární s n-kanálem a p-kanálem.

A další - Dělení dle výkonu, provozní frekvence, zaměření (spínače, zesilovače ...), zapouzdření, zesílení atp.

Nadále budou v práci uvažovány pouze unipolární tranzistory typu MOSFET.

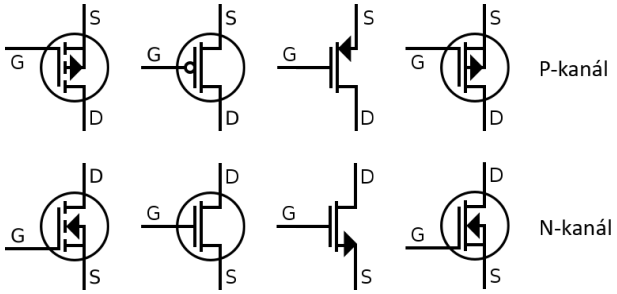
Unipolární tranzistory MOSFET

Unipolární tranzistory MOSFET jsou nejběžnějšími tranzistory v digitální i analogové technice [30]. Mohou být použity jako zesilovače nebo spínače elektrického signálu. Základním principem jejich funkce je vytvoření vodivého kanálu mezi emitorem a kolektorem po přivedení napětí na hradlo. Vytvořený kanál může být polovodičového typu P (P-kanál) resp. N (N-kanál). Na základě typu kanálu jsou tranzistory nazývány PMOSFET resp. NMOSFET (zkráceně pMOS resp. nMOS). Typická implementace na polovodičovém substrátu je zobrazena na obrázku 2.1 a symboly používané při strukturálním popisu obvodu jsou na obrázku 2.2.



Obrázek 2.1: Implementace nMOS tranzistoru na substrátu [71]

Na obrázku 2.3 je vidět charakteristika závislosti I_D (proudu kolektoru) na V_{DS} (napětí kolektor-emitor) pro několik hodnot V_{GS} (napětí hradlo-emitor). Důležitým parametrem



Obrázek 2.2: Symboly MOSFET tranzistorů
[64]

tranzistoru je prahové napětí otevření V_T, které udává minimální napětí V_{GS} pro vytvoření vodivého kanálu tranzistoru a tedy minimální napětí, od kterého může téct proud mezi kolektorem a emitorem. Nejjednodušeji lze funkci tranzistoru popsat z hlediska tří režimů, ve kterých může pracovat [30, 26, 71, 50, 16] (tabulka 2.1):

Uzavření (také podprahový stav nebo slabá inverze) - V charakteristice tranzistoru (obrázek 2.3) je tento stav reprezentován dolní vodorovnou hranu grafu ($I_D = 0$). V tomto režimu tranzistorem teoreticky neprochází žádný proud (ve skutečnosti však mezi kolektorem a emitorem protéká zbytkový proud zvaný "leakage").

Ohmický (také lineární nebo triodový) - V charakteristice tranzistoru (obrázek 2.3) je tento stav reprezentován lineární oblastí. V tomto stavu je v tranzistoru vytvořen vodivý kanál a protéká proud mezi kolektorem a emitorem. Tranzistor se chová jako rezistor řízený V_{DS} a V_{GS} a hodnota těchto dvou napětí přímo ovlivňuje I_D.

Saturace (nebo také aktivní režim) - V charakteristice tranzistoru (obrázek 2.3) je tento stav reprezentován oblastí saturace. V tomto režimu již V_{DS} nemá téměř vliv na I_D, které je závislé pouze na V_{GS}.

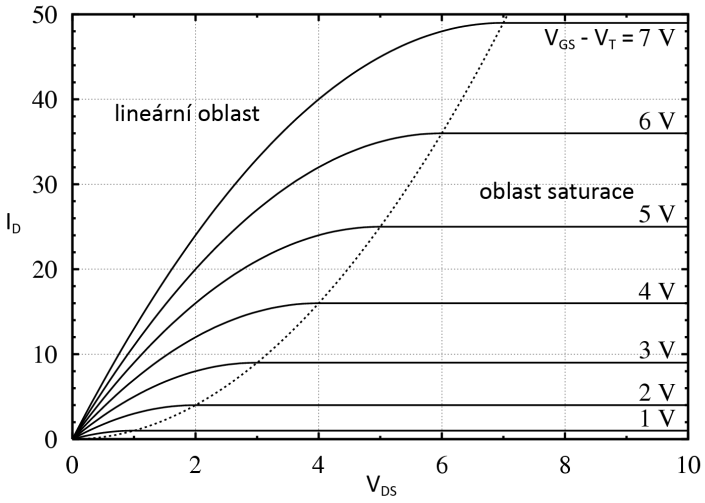
Režim	podmínky nMOS	podmínky pMOS
Uzavření	$V_{GS} < V_T, V_{DS} \geq 0$	$V_{GS} > V_T, V_{DS} \leq 0$
Ohmický	$V_{GS} > V_T, 0 < V_{DS} < V_{GS} - V_T$	$V_{GS} < V_T, 0 > V_{DS} > V_{GS} - V_T$
Saturace	$V_{GS} > V_T, V_{DS} > V_{GS} - V_T$	$V_{GS} < V_T, V_{DS} < V_{GS} - V_T$

Tabulka 2.1: Režimy práce tranzistoru
[50, 16]

Zatímco V_{DS} a V_{GS} nastavuje přímo návrhář při návrhu obvodu, prahové napětí V_T je dáno použitou výrobní technologií. Jeho hodnota navíc není stálá a je proměnná dle teploty [50]. Vliv na V_T má také V_{BS}. Substrát se při návrhu může považovat za druhé hradlo, někdy označovaná jako back-gate. Její vliv na V_T se někdy označuje také jako back-gate effekt. Tohoto efektu se může využít v nízkonapěťových nebo nízkopříkonových obvodech, kdy je V_{BS} využito pro snížení V_T [24].

MOSFET jako spínače

Pomocí MOSFET tranzistorů lze vytvořit různé základní stavební prvky elektronických obvodů [16]. Může se jednat například o:



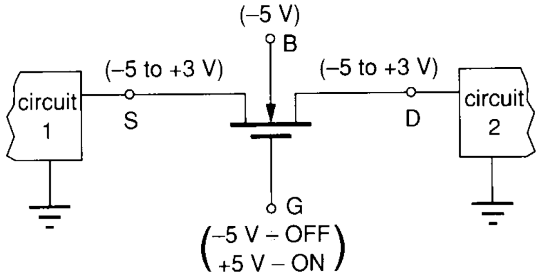
Obrázek 2.3: Charakteristika závislosti I_D na V_{DS} MOSFET tranzistorů [64]

- spínače,
- aktivní rezistory,
- proudové zdroje,
- proudová zrcadla a zesilovače a
- napěťové a proudové reference.

Nadále se bude práce podrobněji zabývat pouze spínači, které jsou základními stavebními prvky digitálních obvodů.

Iedální spínač má nulový odpor v sepnutém stavu (R_{ON}), nekonečný odpor ve stavu rozepnutém (R_{OFF}) a nekonečně krátkou dobu přechodu mezi stavy. Hodnoty dosahované MOS tranzistory se liší v závislosti na geometrii tranzistoru (délka a šířka kanálu) a použité výrobní technologii. Obecně však MOS tranzistory používané v digitální technice dosahují R_{ON} v rozsahu stovek Ω až jednotek $k\Omega$, R_{OFF} kolem $10^{12}\Omega$ [16] a maximální rychlosti v řádu stovek GHz [70].

MOSFET tranzistor jako spínač se pohybuje v režimu uzavření pro vypnutý spínač a v ohmickém režimu pro spínač otevřený. Důležitou vlastností je rozsah spínaných napětí v porovnání k napětí spínacímu. Vezmeme-li v potaz nMOS tranzistor, pak pro režim otevření musí být splněna podmínka $V_{GS} > V_T$. Z této podmínky plyne, že maximální úroveň spínaného napětí je o V_T nižší než napětí spínací. Prakticky pokud máme obvod napájený ze zdroje napětím V_{DD}, pak maximální spínané napětí může být nižší než $V_{DD} - V_T$. Jakmile se spínané napětí začne k této hranici blížit, tranzistor se začne uzavírat. Problém je demonstrován na obrázku 2.4. Pro tranzistor pMOS platí stejné omezení ale s podmínkou $V_{GS} < V_T$.



Obrázek 2.4: nMOS tranzistor jako spínač [16]

2.1.2 Logická hradla

Logická hradla jsou elektronické prvky implementující logické funkce. Obsahují jeden nebo více vstupů a jediný výstup. Hodnota na výstupu je dána funkcí vstupních hodnot (2.1) [65].

$$y = f(x_1, x_2, ..., x_n) \tag{2.1}$$

První úvahy implementovat logické funkce pomocí elektroniky byly nalezeny v dopise Charlese S. Peirce Allanu Marquandovi z roku 1886 [44]. V roce 1898 Nikola Tesla podal patenty na zařízení využívající elektromechanické logické obvody (viz např. [62]). V roce 1937 ve své diplomové práci [48] Claude E. Shannon ukázal využití Booleovy algebry pro analýzu a návrh logických obvodů. Booleova algebra se v následujících letech stala základem návrhových i popisných metod pro logické obvody a v dnešní době se o logických hradlech mluví jako o členech implementujících Booleovy funkce [65]. Více o Booleově algebře např. v [3, 19, 20].

Pro počet všech možných Booleových funkcí (a tedy i logických hradel) platí vztah 2.2, kde y je počet všech možných funkcí a n je počet vstupů funkce (logického hradla). Existují tedy čtyři funkce pro jednu vstupní proměnnou. Trvalá 1, trvalá 0, negace a opakování. Funkcí s dvěma vstupními proměnnými je již 16 a jsou popsány v tabulce 2.2.

$$y = 2^{2^n} \tag{2.2}$$

Z množiny všech Booleových funkcí lze vybrat takové podmnožiny, pomocí nichž je možné implementovat všechny ostatní funkce [69]. O takových podmnožinách se pak říká, že jsou logicky kompletní a je možné s jejich pomocí implementovat jakýkoliv logický systém. V některých případech obsahuje tato podmnožina pouze jedinou funkci, která je logicky kompletní sama o sobě. Logicky kompletní funkce dvou proměnných jsou NAND a NOR [49]. Další logicky kompletní množiny je možné nalézt např. v [69]. Logické kompletnosti se pak využívá při návrhu složitějších systémů, které mohou být složeny jen z jedné logicky kompletní množiny prvků.

Značení

Značení logických členů v obvodě je dáno normou ANSI/IEEE Std 91-1984 a jejím dodatkem ANSI/IEEE Std 91A-1991. Tato norma povoluje dva způsoby značení. První používá

Vstup	A	0	0	1	1
	B	0	1	0	1
Výstup	trvalá 0	0	0	0	0
	A AND B	0	0	0	1
	$\not\rightarrow$	0	0	1	0
	A	0	0	1	1
	$\not\leftarrow$	0	1	0	0
	B	0	1	0	1
	A XOR B	0	1	1	0
	A OR B	0	1	1	1
	A NOR B	1	0	0	0
	A XNOR B	1	0	0	1
	NOT B	1	0	1	0
	$\longleftarrow$	1	0	1	1
	NOT A	1	1	0	0
	$\longrightarrow$	1	1	0	1
	A NAND B	1	1	1	0
	trvalá 1	1	1	1	1

Tabulka 2.2: Booleovy funkce dvou vstupních proměnných

tvarově odlišné symboly logických členů složených z křivek a je odvozeno od normy MIL-STD-806. Je rozšířeno zejména v profesionálních návrhových systémech. Toto značení je na první pohled názornější, ale je schopné vyjádřit pouze ty logické členy, ke kterým je symbol definován.

Druhým způsobem značení založeným na normě IEC 60617-12 je využití obdélníkových tvarů. Každý logický člen je znázorněn obdelníkem, ve kterém je v horní části určení funkce. Funkce se určují posloupností znaků nebo symbolů, z nichž některé základní jsou definovány v normě, u ostatních se předpokládá dodatečný popis chování. Toto značení je schopné vyjádřit všechny logické členy bez omezení a je tedy obecnější. Seznam základních logických členů i s jejich symboly je uveden v příloze A.

Parametry

U logických obvodů jsou důležité jejich provozní parametry, které udávají provozní vlastnosti a určují podmínky, v jakých je zajištěna správná funkce. Základní parametry pro dvouhodnotovou logiku jsou:

Napájecí napětí - rozsah napájecího napětí, se kterým jsou obvody schopné pracovat.

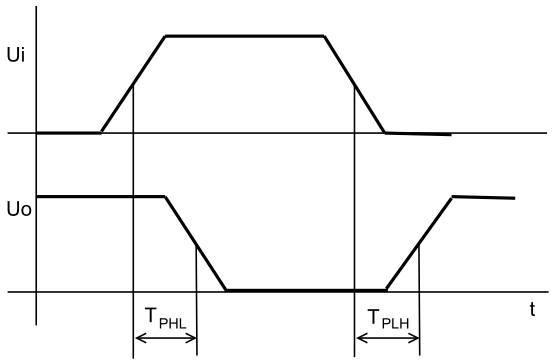
Vstupní a výstupní napětí - minimální a maximální úrovně vstupního a výstupního napětí. Jedná se o čtyři hodnoty:

- $U_{OL_{max}}$ maximální výstupní napětí nízké úrovně,
- $U_{IL_{max}}$ maximální vstupní napětí nízké úrovně,
- $U_{OH_{min}}$ minimální výstupní napětí vysoké úrovně a
- $U_{IH_{min}}$ minimální vstupní napětí vysoké úrovně.

Výstupní proudy - maximální velikosti výstupních proudů, kterých je logický člen schopen. Tento údaj se může udávat zvlášť pro nízkou i vysokou úroveň.

Šumová imunita - udává, jak nepřesné mohou být vstupní signály, aby byla na výstupu správná hodnota. Výrobcem garantovaná šumová imunita je definována jako rozdíl nejhorších mezních hodnot vstupů a výstupů. Pro nízkou úroveň je to tedy $U_{IL_{max}} - U_{OL_{max}}$ a pro vysokou $U_{OH_{min}} - U_{IH_{min}}$. V praxi se však více používá tzv. typická šumová imunita, která udává mezní napěťové úrovně tzn. maximální napětí pro nízkou a minimální napětí pro vysokou úroveň.

Zpoždění hradla - definuje, za jak dlouho nejpozději se na výstupu projeví změna na vstupu. Údaj se někdy udává zvlášť pro přechod do vysoké a pro přechod do nízké úrovně. Ilustrace je na obrázku 2.5.



Obrázek 2.5: Zpoždění logického obvodu

Maximální kmitočet - maximální rychlost změn vstupních signálů za vteřinu, aby byl zajištěn korektní signál na výstupu.

Logický zisk - udává, kolik vstupů dalších obvodů stejné technologie je možné zapojit na výstup prvku.

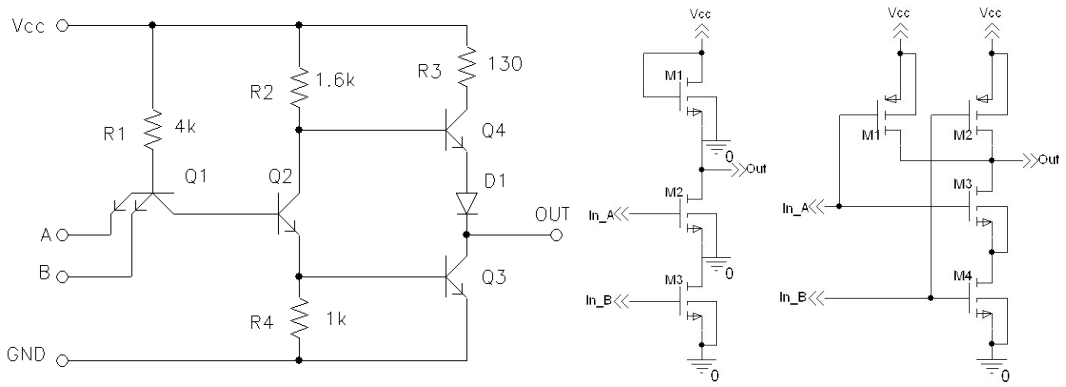
Příkon - příkon, který obvod potřebuje. Často se také udává pouze maximální proud tekoucí hradlem.

Hodnoty některých parametrů jsou přímo závislé na jiných. Vstupní a výstupní napětí jsou přímo ovlivněny velikostí napájecího napětí. Logický zisk je zase dán výstupním proudem a proudovým odběrem vstupů na daný výstup připojených.

Implementace

Způsob implementace logických hradel se v elektronice s postupem času měnil. V prvopočátcích se využívala elektromechanická relé nebo elektronky. Po rozmachu polovodičů se objevují různé polovodičové implementace jako například diodová logika (DL), rezistor-transiztorová logika (RTL) nebo diodo-tranzistorová logika (DTL). Významného rozmachu dosáhla tranzistor-tranzistorová logika (TTL) využívaná v některých obvodech dodnes. Ukázka implementace hradla NAND v TTL logice je na obrázku 2.6a. Všechny tyto implementace s tranzistory používají tranzistory bipolární.

Při implementaci elektronických obvodů dochází postupem času k nahrazování bipolárních tranzistorů tranzistory unipolárními a tak se začínají prosazovat implementace postavené na MOSFET tranzistorech. Obecným problémem využití MOSFET tranzistorů jako spínačů je omezení rozsahu spínatelného napětí popsané v kapitole 2.1.1. Pro správnou funkci spínače tak mají pMOS tranzistory emitor připojen vždy na vysokou úroveň nebo kolektor jiného pMOS tranzistoru a nMOS tranzistory mají emitor připojen na nízkou úroveň nebo kolektor jiného nMOS tranzistoru. Cesta proudu z napájení na výstup tak vede přes pMOS tranzistory a cesta proudu z výstupu na zem přes nMOS tranzistory. Tímto se problém omezené úrovně spínatelného napětí eliminuje.



(a) TTL (b) NMOS (c) CMOS

Obrázek 2.6: Implementace hradla NAND v různých techologiích

Technologie NMOS

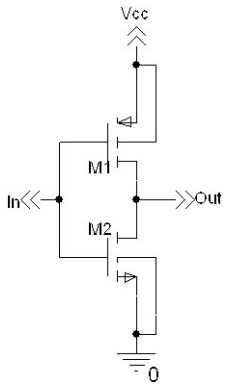
První implementace logických obvodů s MOSFET tranzistory byly založeny na využití tranzistorů s jedním typem kanálu. Většího rozšíření dosáhla logika NMOS založena pouze na tranzistorech s n-kanálem. Jelikož nMOS tranzistory mohou dobře sloužit pouze pro přenos nízkých úrovní, pro přenos úrovní vysokých se vyžívá tzv. "PullUp"rezistorů. V praxi se pak jako rezistor využívá nMOS tranzistor ve speciálním zapojení. Ukázka implementace hradla NAND v NMOS logice je na obrázku 2.6b.

NMOS logika má ale oproti jiným technologiím horší některé důležité parametry. Přestože je obvod v neměnném stabilním (tzv. statickém) stavu, tak v případě nízké úrovně obvodem protéká nezanedbatelný proud. Jeho velikost je dána převážně velikostí PullUp rezistorů. NMOS obvody také mají velké zpoždění při přepnutí z nízké na vysokou úroveň a mají horší šumovou imunitu. Z těchto důvodů byla tato techologie postupně vytlačena techologií CMOS.

Technologie CMOS

Technologie CMOS (Complementary Metal–Oxide–Semiconductor) byla patentována Frankem Wanlassem 1967 [67] a je dnes hlavní technologií pro tvorbu všech větších digitálních obvodů. Jejím principem je využití pMOS a nMOS tranzistorů v symetrických párech, kdy každý nMOS resp. pMOS tranzistor má v obvodě komplementární pMOS resp. nMOS tranzistor. Tranzistor pMOS se stará o přenos v případě vysoké úrovně a komplementární nMOS v případě úrovně nízké.

Nejjednodušším případem logického hradla pomocí CMOS technologie je invertor na obrázku 2.7. Je-li na vstupu nízká úroveň, tranzistor M1 je otevřený a M2 uzavřený. Proud tedy může procházet z napájení na výstup. A opačně, je-li na vstupu úroveň vysoká, tranzistor M1 je uzavřený, M2 otevřený a proud může procházet z výstupu na zem. Tranzistory M1 a M2 se tak navzájem doplňují.



Obrázek 2.7: CMOS invertor

V CMOS logice tak obecně platí, že množina všech cest od napájení na výstup je komplementem množiny všech cest od výstupu na zem. Této vlastnosti lze jednoduše dosáhnout definováním množiny cest a poté její negací. Z De Morganových pravidel pak plyne, že k pMOS tranzistorům zapojeným paralelně jsou komplementární nMOS tranzistory zapojeny sériově a naopak [25]. Tento princip je zřejmý z obrázku CMOS NAND hradla 2.6c, kde k paralelnímu zapojení pMOS tranzistorů M1 a M2 jsou sériově zapojeny tranzistory M3 a M4.

K přednostem CMOS technologie patří velká šumová imunita a zejména malý statický odběr. Významný odběr má pouze při přepínání stavů, kdy tranzistory, které se mají otevřít jsou již pootevřeny a tranzistory, které se mají uzavřít nejsou ještě plně uzavřeny. Obvodem pak může procházet proud přímo z napájení na zem. To má za následek příkonové špičky při změně stavu obvodu.

2.1.3 Úrovně integrace

Zvyšující se míra integrace integrovaných obvodů byla klasifikována dle počtu vnitřních prvků (tabulka 2.3). Jelikož se v integrovaných obvodech vyskytují v podstatě pouze tranzistory, je počet vnitřních prvků vztažen právě na ně. V tabulce je také uveden přibližný rok dosažení dané míry integrace. Pojem ULSI se neujal a pro velké obvody se více používá označení VLSI.

Název	Počet tranzistorů	Rok
SSI	1	1960
MSI	100	1965
LSI	10 000	1974
VLSI	100 000	1980
ULSI	1 000 000	1986

Tabulka 2.3: Úrovně integrace integrovaných obvodů

Postupem času se také začaly objevovat speciální pojmenované případy integrovaných obvodů. WSI (Wafer scale integration) označuje obvody využívající při výrobě celý výrobní plát (wafer). 3D-IC (Three dimensional integrated circuit) značí obvody implementovány ve více vrstvách, jejichž prvky jsou orientovány jak vertikálně, tak horizontálně. Objevují se také obvody, které obsahují celý konkrétní systém. Označují se SoC (System on a chip) [40].

2.1.4 Model číslicových obvodů

Jak se postupně zvyšovala složitost integrovaných číslicových obvodů, přestalo být prakticky možné navrhovat obvody popisem pomocí elementárních prvků. Z hlediska efektivity a chybovosti je důležité, aby návrhář řešil „co se bude dělat“ a neřešil „jak se to bude dělat“. Začalo se proto využívat abstrakcí, kdy návrhář pracuje s prvky se složitější funkcí, jejichž vnitřní implementace je skryta. Po navržení obvodu z abstraktních prvků se tyto rozpracují až na potřebnou úroveň. Z počátku však nebyly určeny konvence a v odborných kruzích se vedly diskuze jak při návrhu postupovat. Názory nebyly jednotné. Za zlomový okamžik se považuje rok 1983, kdy Daniel D. Gajski ve svém článku [15] poprvé publikoval tzv. Y-diagram (obrázek 2.8). Tento diagram identifikoval tři základní pohledy na popis obvodu a nastartoval diskuzi ohledně specifikace používaných úrovní abstrakce. Třemi základními popisy obvodu jsou:

Popis strukturální - Obvod je popsán pomocí známých strukturních stavebních prvků a jejich propojení. Stavební prvky mohou být tranzistory, logická hradla, registry, výpočetní jednotky (sčítačky, odčítačky, ALU ...), sběrnice, subsystémy atp.

Popis chování - Obvod je popsán podle svého chování. Může se jednat o přenosové rovnice, Booleovu algebru, logické funkce, algoritmy atp.

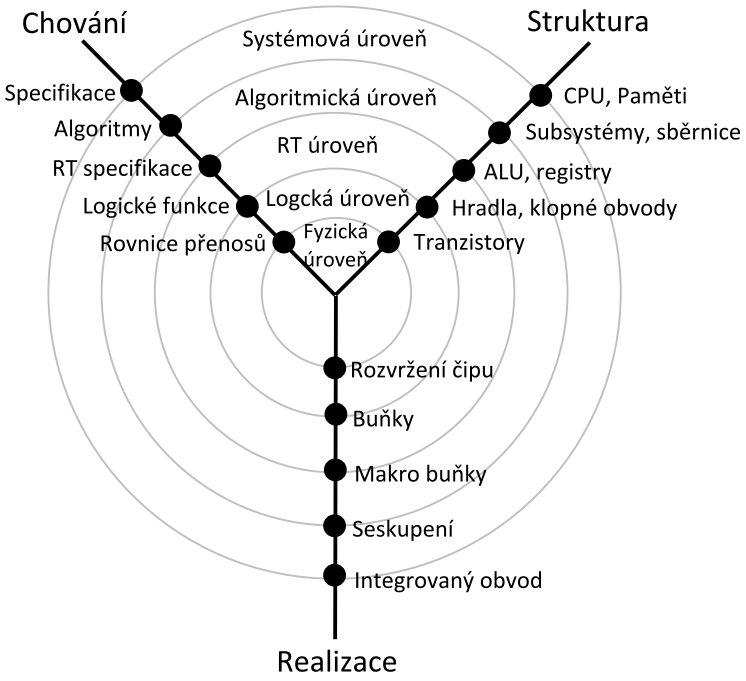
Popis realizace čipu - Obvod se popisuje tak, jak bude realizován na výsledném čipu. Prostředky jsou P a N polovodičové plošky a propojky na výsledném substrátu, buňky, makro buňky atp.

Těmito třemi způsoby popisu může být dobře definována většina nejdůležitějších vlastností výsledného obvodu.

Y-diagram je dále tvořen soustřednými kružnicemi, které charakterizují hierarchickou úroveň abstrakce. Úroveň abstrakce vzrůstá směrem od středu ven. Průsečík kružnice s polopřímkou způsobu popisu obvodu reprezentuje pojmenovanou úroveň abstrakce pro daný způsob popisu. Ohledně počtu a pojmenování jednotlivých úrovní abstrakce se nedošlo k úplnému konsensu a v různých literaturách jsou uváděny odlišně. Ve většině však lze identifikovat pět následujících:

Systémová úroveň - Na této úrovni jsou stanoveny základní koncepty. Z hlediska chování se jedná o definici funkce celého systému a jeho vlastností. Na strukturní úrovni se může jednat např. o procesor, paměť atp. Z hlediska realizace se jedná o popis celého integrovaného obvodu nebo jeho fyzických částí.

Algoritmická úroveň - Udává, jak bude řešení provedeno. Tato úroveň abstrakce je důležitá pro algoritmický popis chování. Obecně může být pro tento účel použit jakýkoliv prostředek (přirozený jazyk, diagram, programovací jazyk, formální model



Obrázek 2.8: Y-diagram

atp.). V praxi se však využívá HDL jazyků (VHDL, Verilog ...). Z hlediska strukturního popisu tato úroveň udává, jaké jsou použity subsystémy a jak jsou mezi sebou propojeny.

RT úroveň - Nazývaná také jako úroveň meziregistrových přenosů. Tato úroveň je charakteristická popisem toku dat v obvodě a jeho řízením. Typicky je zde patrné oddělení datové části od řídicí. Datová část bývá tvořena funkčními bloky navzájem oddělenými dalšími prvky (registry, paměti atp.), řídicí část bývá řešena pomocí stavového automatu. Pro strukturální popis se využívá prvků, jako jsou sčítačky, násobičky, ALU, registry atp. Pro popis chování se obvykle používá model datového toku. Pro popis realizace bývají použity tzv. makro buňky typicky reprezentující strukturální prvky (ALU, sčítačka ...).

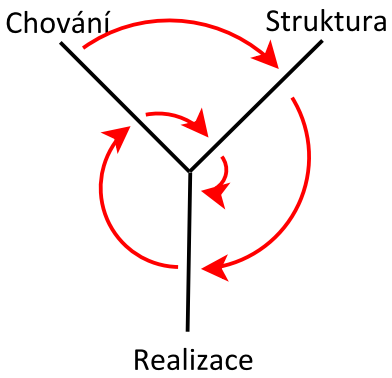
Logická úroveň - Je reprezentována nejmenšími číslicovými prvky. Ze strukturálního pohledu se zpravidla jedná o logická hradla a klopné obvody. Při popisu chování se obvykle využívá Booleovy algebry. Při popisu realizace se používají buňky typicky reprezentující strukturální prvky (hradla a klopné obvody).

Fyzická úroveň - Je nejnižší úrovní v uvažované hierarchii abstrakce. Z hlediska struktury se zde uvažují základní elektronické prvky jako tranzistor, rezistor, kondenzátor, cívka atp. Pro popis chování se obvykle používají diferenciální přenosové rovnice. Vyjadřovacím prostředkem realizačního popisu jsou typicky N a P polovodičové plošky a propojky jednotlivých prvků (tzv. layout) na výsledném substrátu (obvykle křemíkovém waferu), které následně slouží jako popis masky pro fyzickou výrobu čipu.

2.1.5 Postup návrhu obvodů

Jedním z nutných výstupů návrhu obvodu pro jeho výrobu je popis výrobní masky. Ať už je tedy obvod navržen jakkoliv, je nutné jej před samotnou výrobou reprezentovat pomocí popisu realizace obvodu na nejnižší úrovni abstrakce.

Uznávaný postup návrhu číslicových obvodů (tzv. Design flow) lze obecně chápat jako pohyb v Y-diagramu (obrázek 2.9). Typicky se jedná o spirálu ve směru hodinových ručiček se začátkem ve vnějších částech u popisu chování a s koncem u středu popisu realizace. Jedná se tedy o cestu v Y-diagramu směřující z vnějšku dovnitř. Jak přesně cesta vypadá závisí na návrhářích a použitých nástrojích.



Obrázek 2.9: Postup v Y-diagramu při návrhu číslicových obvodů

Z postupu je zřejmé, že při návrhu dochází ke změnám způsobu reprezentace obvodu a to jak ve způsobu popisu obvodu, tak i v úrovni abstrakce. V závislosti na tom, zdali pro danou transformační změnu existuje nástroj (příp. i knihovna použité technologie), je možné tuto transformaci provést automatizovaně (tzv. syntéza), nebo je nutné ji provést manuálně. Pro každý krok musí platit, že nová reprezentace je z hlediska vlastností obvodu ekvivalentní s reprezentací předchozí a tedy, že se jedná o tentýž obvod zapsaný jiným způsobem. Je důležité uvést, že převod od abstraktnějšího návrhu ke konkrétnímu není jednoznačný a tedy, že abstraktní model může být reprezentován mnoha různými implementacemi na nižší úrovni abstrakce.

Příkladem postupu může být návrh obvodu pomocí sepsání algoritmu v HDL jazyce (popis chování), který se syntézou převede do strukturálního RTL popisu, ten se další syntézou převede do strukturálního logického popisu (tzn. na logická hradla), jenž se pomocí poslední syntézy převede na masku výsledného obvodu pro výrobu.

Důležitou částí návrhu je také ověření jeho správnosti. V této souvislosti se objevují dva pojmy:

Verifikace - Jejím úkolem je ověřit, jestli je obvod funkční a neobsahuje chyby. Při transformacích se ověřuje, zdali je obvod shodný s jeho předchozí reprezentací, tzn. jestli se jedná o tentýž obvod reprezentovaný jinak. Verifikace může být prováděna formálními prostředky, simulacemi atp.

Validace - Jejím úkolem je ověřit, zdali obvod vyhovuje požadavkům.

2.1.6 Simulace elektronických obvodů

Simulace je proces návrhu modelu zkoumaného systému a provedení experimentů nad tímto modelem [52]. Při simulování se pak zjišťují charakteristiky, chování nebo strategie pro provoz systému. Simulace se využívá pokud není možné nebo praktické zjistit požadované informace přímo ze systému.

Důležitou roli v simulaci hraje způsob a přesnost popisu systému. Čím je systém složitější, tím je méně pravděpodobný jeho přesný popis a přistupuje se k jeho aproximaci. Ta musí odpovídat systému do takové míry, aby v simulaci byly získány výsledky s požadovanou přesností. Pro jeden zkoumaný systém tak může existovat více modelů, které se mohou lišit způsobem popisu, mírou abstrakce a přesností. Jednotlivé modely také mohou popisovat různé vlastnosti. U modelu kovu např. můžeme sledovat jeho tepelnou vodivost nebo elektrickou vodivost. Konkrétní model systému se proto volí dle záměru simulace.

Simulace elektronických obvodů se dnes obvykle provádějí na počítačích ve specializovaných nástrojích. Mohou probíhat analogově, digitálně nebo smíšeně. Nejznámější digitální simulátory bývají založeny na Verilog HDL (Hardware Description Language) nebo VHDL (VHSIC Hardware Description Language). Jako modely slouží nejčastěji algoritmické popisy na RT úrovni nebo Booleova algebra. Nejznámější analogové simulátory bývají založeny na open-source simulátoru SPICE [63].

SPICE

První implementace SPICE (Simulation Program with Integrated Circuit Emphasis) byla vytvořena Laurencem Nagelem v Electronics Research Laboratory Kalifornské Univerzity Berkeley kolem roku 1960 jako open-source derivát simulátoru CANCER (Computer Analysis of Nonlinear Circuits, Excluding Radiation). Prezentována byla v roce 1973 [37]. Většího rozšíření dosáhla až druhá verze představená v roce 1975 [36]. Aktuální třetí verze byla uvedena v roce 1983 [46] a je chápána jako standard pro analogové a smíšené simulace na fyzické úrovni jak na akademické půdě, tak v průmyslu.

Vstupem simulátoru je textový soubor s popisem obvodu tzv. netlist. Tento soubor obsahuje identifikaci jednotlivých prvků obvodu a jejich propojení. Pro každý použitý prvek musí existovat matematický model. Na základě matematických modelů jednotlivých prvků a informaci o jejich propojení je vygenerován matematický model celého obvodu, kerý je použit při simulaci. Podrobnější popis jakým způsobem výpočet funguje je možné nalézt v [68].

Open-source přístup a velká popularita vedli k simulátorům, které ze SPICE vychází a bývají do jisté míry kompatibilní. Vznikají tak další open-source nebo komerční simulátory jako ISPICE, HSPICE, XSPICE atd. Jednotlivé implementace se mohou lišit v různých vlastnostech, jako například rychlosti nebo přesnosti simulace. Pro potřeby této práce bude využit převážně PSPICE ze sady nástrojů OrCad a implementace SPICE ve Spectre ze sady nástrojů Cadence Virtuoso.

Typy analýz

Velkou výhodou SPICE simulátorů je podpora různých typů analýz obvodu. Ze základních uvedu následující:

DC analýza - slouží pro zjištění operačního bodu obvodu se zkratovanými indukcemi a odpojenými kapacitami. Ukazuje, jaké jsou v obvodě hodnoty napětí a proudu bez nelineárních prvků.

AC analýza - ukazuje vlastnosti obvodu v závislosti na frekvenci.

Přechodová analýza - ukazuje vlastnosti obvodu v přechodových dějích. Hodnoty jsou funkcí času.

U jednotlivých typů analýz bývá možné definovat další požadavky na měření. Často se využívá tzv. "sweep"parametrů, kdy se měření provede vícekrát pro různé definované hodnoty konkrétního parametru. Parametrem mohou být vstupní proměnné obvodu (např. napětí), vlastnosti prostředí (např. teplota) nebo vlastnosti obvodu (např. hodnota odporu).

Pro praxi důležitou analýzou je tzv. Corner analýza. I při návrhu a simulaci obvodu je potřeba počítat s tím, že v reálném prostředí nejsou podmínky přesné, ale pohybují se v určitých mezích a tolerancích. Corner analýza simuluje obvod v "rozích"tedy nejhorších podmínkách, které ještě odpovídají stanoveným mezím a tolerancím. Příkladem může být simulace vlivu minimální a maximální teploty prostředí, rozsahu napájecího napětí nebo nepřesností výrobního procesu.

Modely MOSFET tranzistorů

Pro simulaci elektrických vlastností obvodu dnes existuje mnoho modelů MOSFET tranzistoru. Jedním z prvních nejjednodušších modelů je model SPICE level 1 popsaný rovnicemi v tabulce 2.4 a popisem jednotlivých parametrů v tabulce 2.5 [50].

Režim	Rovnice I_{DS}
Uzavření	$I_{DS} = 0$
Ohmický	$I_{DS} = UO\frac{\varepsilon_0\varepsilon_r}{TOX}\frac{W}{L}(V_{GS} - V_T)^2$
Saturace	$I_{DS} = UO\frac{\varepsilon_0\varepsilon_r}{TOX}\frac{W}{L}(V_{GS} - V_T)^2$

Tabulka 2.4: Model tranzistoru SPICE level 1 [50]

Parametr	Definice
V_T	$V_T = VTO + GAMMA(\sqrt{PHI - V_{BS}} - \sqrt{PHI})$
ε_0	Absolutní permitivita. $\varepsilon_0 = 8.85 \times 10^{-12}$
ε_r	Relativní permitivita
VTO	Prahové napětí
UO	Mobilita nosičů
TOX	Tloušťka izolace brány
PHI	Potenciál povrchu při silné inverzi
GAMMA	Práh bulk
W	Šířka MOS kanálu
L	Délka MOS kanálu

Tabulka 2.5: Parametry modelu tranzistoru SPICE level 1 [50]

Podobné jednoduché modely jsou ale jen přibližné a pro potřeby přesných simulací se prakticky nedají použít. Současně vývoj ve výrobních procesech vede k novým technologiím, které mohou funkci tranzistoru ovlivňovat. Postupně tak vznikají nové modely tranzsitorů, které jsou složitější a přesnější než jejich předchůdci. Vznikají tak novější SPICE modely,

uznávaná řada modelů BSIM [6] a další [14]. Nebude-li uvedeno jinak, bude v práci použit model SPICE level 7 s hodnotami parametrů technologie AMI 0.7 μm uvedených v příloze B.

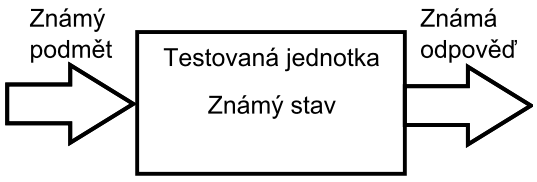
2.2 Diagnostika číslicových obvodů

S rostoucí složitostí číslicových obvodů roste také pravděpodobnost odchylek jejich funkce od funkce požadované. Přesáhne-li odchylka stanovenou mez nebo je-li přímo v rozporu s požadovanou funkcí, stává se obvod nefunkční. Pro číslicové obvody podobně jako pro jiné produkty platí, čím později je problém odhalen, tím větší bývají náklady na jeho odstranění. Včasné nalezení případných problémů se tak stává jednou z klíčových fází vývoje a provozu číslicových systémů.

Odchylky a chyby mohou vzniknout z různých důvodů v různých fázích životního cyklu obvodu. Zatímco na chyby vzniklé při návrhu obvodu se zaměřuje samostatná disciplína verifikace, diagnostika se zabývá chybami vzniklými *defekty* ve fyzické implementaci obvodu (nejčastěji na křemíkovém substrátu). Příkladem defektu může být zkrat oxidové vrstvy oddělující hradlo tranzistoru, zkrat mezi propojkami, nespojená propojka atp. Defekt se pak může projevit jako *porucha* v obvodě. Na tranzistorové úrovni je poruchou například propojení hradla a kolektoru tranzistoru, na úrovni logických hradel se jedná například o trvalou 0 na výstupu hradla. Porucha může způsobit *chybu* obvodu, která je definována jako neshoda mezi správnou a skutečnou hodnotou proměnné zjištěné na výstupu. Platí, že defekt se nemusí projevit jako porucha a porucha se nemusí projevit jako chyba. Pokud se porucha jako chyba neprojeví, hovoří se o tzv. *latentní poruše*. Hlavními úlohami diagnostiky jsou detekce a následná lokalizace poruchy.

2.2.1 Testování

U elektronických obvodů se testování provádí souborem měření za účelem zjištění, jestli testovaný obvod odpovídá požadovaným parametrům. Testování lze popsat jako aplikaci známých vstupních podnětů na testovanou jednotku ve známem stavu a porovnání odezvy s odezvou očekávanou [11]. Tento princip je zobrazen na obrázku 2.10.



Obrázek 2.10: Princip testování elektronických obvodů
[11]

Není-li v době testu přímý přístup ke vstupům a výstupům testované jednotky, je třeba řešit, jakým způsobem k této jednotce přes dostupné vstupy dopravit podněty a jakým způsobem dopravit její odezvy na dostupné výstupy. Cesty pro dopravu podnětů a odezev testované jednotky přes ostatní prvky systému se nazývají *transparetní cesty*. Ze znalosti transparetních cest se podněty a odezvy testované jednotky transformují na podněty a odezvy dostupných vstupů a výstupů. Aplikací podnětů na dostupný vstup se po přenosu transparentní cestou dosáhne aplikace požadovaného podnětu na testovanou jednotku.

Současně se přenosem odezvy testované jednotky po transparentní cestě získá odezva na dostupném výstupu. Možnost nastavit požadovaný podnět na vstupech testované jednotky se nazývá *řiditelnost* a možnost odečíst hodnotu odezvy na výstupu testované jednotky se nazývá *pozorovatelnost*.

Pro aplikaci testu také musí být testovaná jednotka v požadovaném stavu. Nastavení do požadovaného stavu se nejčastěji provádí buď posloupností vstupních podnětů a nebo použitím nějaké formy hardwarových příkazů pro nastavení nebo nulování.

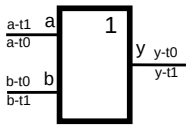
K testování číslicových obvodů se jako vstupní podnět využívají vysoké a nízké úrovně na vstupech testované jednotky. Obvykle se zapisují jako jeden řetězec složený ze znaků "1"a "0"nazývaný testovací vektor. Pro očekávanou odezvu testované jednotky se využívá stejná forma. V očekávané odezvě však mohou být nejen vysoké a nízké úrovně, ale také neznámý stav, pro který se obvykle využívá znak "X". Při testu např. sběrnic se používá i stav vysoké impedance obvykle označený jako "Z".

2.2.2 Modelování poruch

Poruchy obvodu jsou způsobeny defekty ve fyzické implementaci. Model poruchy je matematický model fyzických defektů, který může být zpracováván algoritmicky a je využíván simulačními nástroji [11].

Nejznámějším a nejvíce používaným modelem poruchy je trvalá 1 a trvalá 0, který se obvykle využívá při testování na úrovni hradel. Tento model předpokládá, že se jakýkoliv defekt projeví jako trvalá 0 (*t0*) nebo trvalá 1 (*t1*) na některém ze vstupů nebo výstupů alespoň jednoho hradla obvodu. Pro zjednodušení také předpokládá, že v obvodě existuje pouze jedna porucha.

Ne všechny poruchy *t0*/*t1* jsou ale vzájemně rozlišitelné pomocí primárních výstupů testované jednotky. Některé poruchy se na výstupu prvku projeví stejným způsobem jako poruchy jiné. Různé poruchy se shodným projevem na výstupu prvku se nazývají *ekvivalentní poruchy*. Pro test obvodu je možné množinu všech poruch redukovat na základě ekvivalence poruch. Vezmeme-li v potaz například dvouvstupové hradlo OR na obrázku 2.11, pak existuje šest možných poruch popsaných množinou $P = \{a\text{–}t0, a\text{–}t1, b\text{–}t0, b\text{–}t1, y\text{–}t0, y\text{–}t1\}$. Pokud je na výstupu hradla porucha *t1*, nelze pomocí nastavení hodnot vstupů a odečítáním výstupu zjistit, zdali se jedná o poruchu $y\text{–}t1$, $a\text{–}t1$ nebo $b\text{–}t1$. Množinu všech poruch P tak lze převést na redukovanou množinu poruch $PR = \{a\text{–}t0, b\text{–}t0, y\text{–}t0, y\text{–}t1\}$ o čtyřech prvcích. Jelikož se poruchy $a\text{–}t1$ a $b\text{–}t1$ vyřazené z množiny PR projeví jako porucha $y\text{–}t1$, která je v množině PR zahrnuta, pak tato úprava nemá pro detekci existence poruchy v obvodě vliv.



Obrázek 2.11: Model poruch t0/t1 hradla OR

2.2.3 Typy testů

Testy obvodu mohou být postaveny odlišně na základě rozdílných přístupů k samotnému způsobu testu. Testy se mohou zaměřit na funkci obvodu, strukturu obvodu, nebo vyzkoušet

všechny možné vstupní kombinace (tzv. úplný test).

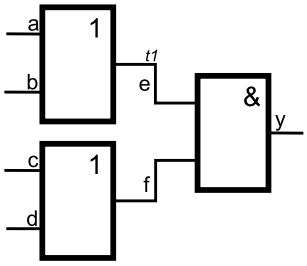
Test funkce

Funkční testy slouží k ověření, že se obvod chová požadovaným způsobem. Podněty na obvod při testování simulují reálné požadavky. Testuje-li se například dvouvstupová sčítačka, pak test na vstup obvodu přivede dva reálné operandy a na výstupu zkontroluje, jestli je výsledek jejich součtem. Při testu se pak otestují ty vstupní kombinace, které jsou pro obvod kritické. Jsou-li kritické všechny platné vstupy, musí test obvodu všechny tyto kombinace otestovat.

Test struktury

Strukturní testy slouží k ověření spojů a funkcí prvků v obvodu. Nejčastěji se používá model poruchy *t0/t1*. Principem testu pak je otestovat, zdali nějaké místo v obvodě obsahuje poruchu *t0* nebo *t1*. To se provádí způsobem, kdy se testované místo přes vstupy obvodu nastaví na opačnou hodnotu, než je předpokládaná porucha a hodnota z tohoto místa se poté přivede na výstupy obvodu. Pokud místo poruchu obsahuje, na výstupu bude jiná než předpokládaná hodnota.

Jako příklad může sloužit obvod na obrázku 2.12. Nyní předpokládejme poruchu typu *t1* na spoji *e*. Tuto poruchu můžeme detekovat ve chvíli, kdy by na spoji *e* měla správně být logická 0. Ta na spoji *e* nastane, pokud je na vstupech *a* a *b* logická 0. Nyní pokud existuje porucha *t1* na spoji *e*, pak je tento spoj ve stavu logické 1, i když by měl být ve stavu logické 0. Porucha je nasimulována. Abychom ale mohli poruchu detekovat, musíme ji přenést na výstup celého obvodu. Logickou hodnotu ze spoje *e* přeneseme na výstup *y* tehdy, pokud na spoji *f* bude logická 1. Tu dosáhneme nastavením vstupu *c* nebo *d* na logickou 1. Pro otestování této poruchy tak může posloužit například testovací vektor $0_a0_b1_c1_d$. Pokud při aplikaci tohoto testovacího vektoru na vstup obvodu je na výstupu logická hodnota 1, vykazuje obvod chybu, která může být způsobena poruchou *t1* na spoji *e*.



Obrázek 2.12: Strukturní testování obvodu s modelem poruch t0/t1

Úplný test

Úplné testování je založeno na vyzkoušení všech možností, které v obvodě mohou nastat. U kombinačních obvodů se jedná o vyzkoušení všech možných vstupních kombinací. Jejich počet je roven 2^n, kde n je počet vstupů obvodu. U obvodů, které nejsou čistě kombinační, ale obsahují vnitřní stav, se musí otestovat všechny možné vstupní kombinace pro všechny možné vnitřní stavy obvodu. Pro otestování stavového automatu s m stavy je potřeba 2^m testů. Výsledná složitost testu obvodu s n vstupy a m stavy tak je 2^{n+m} [11].

Složitost typů testu

Z exponenciální složitosti úplného testu plyne, že se vzrůstajícím počtem vstupů příp. vnitřních stavů významně roste složitost tohoto typu testu. Vezmeme-li například čistě kombinační obvod 64 bitové sčítačky s přenosem se 129ti vstupy, tak počet všech možných vstupních kombinací je roven 2^{129}. Pokud by jsme tento obvod testovali na rychlosti 1GHz (tzn. 10^9 výpočtů za sekundu), test by trval přibližně 10^{29} let. Složitost úplného testu bývá možné snížit například pomocí metody peudotriviálních testů, někdy zvanou též metodou diagnostických kuželů [32]. Výsledná složitost však často zůstává vysoká.

Pokud bychom pro stejný případ využili funkční test s požadavkem na úplnou funkčnost sčítačky, stal by se z funkčního testu test úplný se stejnou složitostí. Pro strukturní test podobné sčítačky však počet vstupních kombinací nepřekročí řád několika tisíc vstupních kombinací [43]. Strukturní testy jsou tak hlavním typem testu využívaným při testování čílicových elektronických obvodů.

2.2.4 Metriky testu

Aby bylo možné určit kvalitu testu, je třeba metriky, která test kvalitativně popíše. Základní široce používanou metrikou je *pokrytí poruch*. Pokrytí poruch popsané rovnicí 2.3 udává, kolik je test schopen odhalit poruch z množiny všech možných poruch obvodu. Dle [11] je v průmyslu u strukturního testu standardní pokrytí poruch mezi 95 a 99.9%.

$$Pokrytí\ poruch = \frac{Počet\ detekovatelných\ poruch}{Počet\ všech\ poruch} \cdot 100[\%] \tag{2.3}$$

Při porovnávání hodnot pokrytí poruch je však nutné dávat pozor. Různé nástroje mohou pro jeho výpočet vycházet u stejných obvodů z odlišných hodnot počtů poruch. Některé nástroje vychází z redukované množiny poruch, jiné mohou vycházet z neredukované množiny. Dále také některé nástroje do počtu poruch nezapočítávají latentní poruchy. Někdy se také liší samotná podstata předpokladu poruch v obvodu. U chyb typu *t0/t1* mohou některé nástroje předpokládat poruchy pouze na vstupech a výstupech hradla, jiné i na spojích mezi hradly.

2.2.5 Automatické generování testu (ATPG)

Z prvopočátku se testy vytvářely ručně. Tento přístup však byl zdlouhavý a náročný. V [11] se uvádí, že vytváření testu obvodu trvalo i 18 měsíců. Na straně jedné rostoucí složitost obvodů a požadavky na kvalitu obvodu vytvářely tlak na kvalitní testy, na straně druhé ekonomické zájmy způsobovaly tlak na rychlejší vývoj, TTM[1], a cenu. Bylo tak potřeba navrhovat efektivnější testy za kratší čas.

Velké zlepšení situace přineslo automatické generování testu (anglicky Automatic Test Pattern Generation - ATPG). Jedná se o přístup, kdy jsou testy vytvářeny automatizovanými nástroji. Jsou-li dodržena jistá pravidla návrhu obvodů, dovedou ATPG nástroje vytvořit testy i pro velmi složité obvody v řádu jednotek týdnů [11]. Současně testy vytvořené pomocí ATPG nástrojů dosahují lepšího pokrytí poruch při potřebě menšího počtu testovacích vektorů, než dosahovaly testy vytvářené ručně. Průběh testu tak bývá rychlejší a testery potřebují méně paměti pro uložení testovacích vzorků. To přináší další snížení nákladů na testování.

[1] Z anglického Time To Market - Doba uvedení na trh

Nevýhodou využití ATPG nástrojů je nutnost jejich vývoje a údržby. Současně je také potřeba pro ATPG nástroje udržovat knihovny prvků používaných v obvodech. Výhody použití však obecně převažují nad nevýhodami a ATPG nástroje dnes bývají ústředním prvkem při vytváření testu obvodu.

Samotný pojem ATPG je někdy používán pro generátory různých typů testu. Obecně se však používá hlavně ve spojitosti s generátory strukturních typů.

Kombinační ATPG pro model poruchy *t0/t1*

Generování testovacích vektorů ATPG nástrojem pro model poruchy *t0/t1* lze popsat několika dílčími kroky. Základním předpokladem je vytvoření množiny poruch, pro které bude ATPG nástroj hledat testy. Většinou se využívá redukovaná množina všech poruch obvodu.

Prvním krokem algoritmu je výběr poruchy, pro kterou se bude vytvářet test. Jelikož vytvoření detekční cesty je náročnější, než vytvoření cesty aktivační, volí se obvykle poruchy co nejblíže detekčním místům [11].

Jakmile je porucha zvolena, je potřeba ji aktivovat. Aktivace poruchy znamená nastavit opačnou hodnotu logické úrovně v místě poruchy, než je předpokládaná porucha. To vyžaduje prozkoumání obvodu zpětně od místa poruchy až ke vstupům a nalezení takových vstupních kombinací, aby byla porucha aktivována.

Jakmile je porucha aktivována, je potřeba ji detekovat. Detekce poruchy vyžaduje sestavení detekční cesty, tedy cesty obvodem od místa poruchy do místa detekce. To vyžaduje prozkoumání obvodu od místa poruchy až k detekčnímu místu a nastavení všech prvků po cestě do tzv. transparentního režimu. Mezi vytvářenými cestami může docházet ke konfliktům, keré musí nástroj řešit.

Po sestavení aktivační a detekční cesty bývá obvykle obvod prozkoumán a ze seznamu poruch pro otestování jsou odstraněny všechny ty, které jsou aktuálním testem také pokryty. Dochází tak ke značné redukci počtu potřebných testovacích vektorů.

V současnosti existuje pro generování testů s modelem poruchy *t0/t1* mnoho kvalitních algoritmů. Mezi základní se řadí například D-algoritmus nebo PODEM (Path-Oriented Decision Making). Jejich popis je však již nad rámec této práce

Sekvenční ATPG pro model poruchy *t0/t1*

Oproti generování testu pro kombinační obvody je generování testu pro sekvenční obvody daleko náročnější úlohou. Generátor musí brát v potaz nejen vstupní a výstupní kombinace, ale také vnitřní stav a způsob, jakým vnitřní sekvenční obvody fungují a jak se ovládají. Vytváření testu pro velké sekvenční obvody bez použití technik návrhu obvodu pro snadnou testovatelnost (anglicky Design for Testability - DfT) je stále velmi náročným úkolem a není uspokojivě vyřešen.

2.2.6 Testovací sady obvodů

Aby bylo možné vzájemně porovnávat jednotlivé nástroje pro tvorbu testu, bylo potřeba vytvořit jednotné testovací obvody, na kterých by se nástroje zkoušely. Postupně tak vznikají různé sady testovacích obvodů (anglicky benchmark circuits) určené pro různé účely. Z hlediska diagnostiky patří k nejznámějším a současně k jedněm z nejstarších obvody ze sady ISCAS 85 (tabulka 2.6) představené na konferenci International Symposium on Circuits and Systems (ISCAS) roku 1985 [5]. Jedná se o sadu deseti kombinačních obvodů v rozsahu od

160 do 3513 hradel. Se sadou je také spjat speciální obvod c17 určený pro analýzy. I když se jedná o poměrně zastaralou testovací sadu, pro potřeby této práce je dostačující.

Jméno	Funkce	Hradel	Vstupů	Výstupů
c17		6	5	2
c432	27ch interrupt cont.	160	36	7
c499	32b SEC	202	41	32
c880a	8b ALU	383	60	26
c1355	32b SEC	506	41	32
c1908	16b SEC/DED	880	33	25
c2670	12b ALU, control.	1269	233	140
c3540	8b ALU	1669	50	22
c5315	9b ALU	2307	178	123
c6288	16x16 multiplier	2416	32	32
c7552	32b adder/compar.	3513	207	108

Tabulka 2.6: Obvody sady ISCAS 85

2.2.7 Testovatelnost a analýza testovatelnosti

Ačkoliv je pojem "testovatelnost"vzhledem k elektronickým digitálním obvodům hojně využíván, není standardně definován. Obecně se chápe jako míra jednoduchosti tvorby a aplikace testu pro určitý obvod. Zde však jednotnost končí a při konkrétnějším popisu testovatelnosti určitého obvodu již dochází k odlišnostem. Existuje tak několik různých popisů testovatelnosti obvodu, pro které jsou vytvořeny různé metodiky jejich výpočtu obvykle vytvořených pro různé úrovně abstrakce popisu obvodu.

Analýzy testovatelnosti, tedy metody výpočtu údajů o testovatelnosti, mají obvykle dvě vlastnosti. Při výpočtu zohledňují pouze topologii obvodu bez použití jakýchkoliv testovacích vektorů a mají lineární složitost výpočtu [8]. Pokud by metody analýzy testovatelnosti nedosahovaly lineární složitosti, pak by byly zbytečné a charakteristiku testovatelnosti obvodu by bylo možné získávat čistě z ATPG nebo simulací poruch.

Mezi nejvíce využívané popisy testovatelnosti patří popis na principu vyjádření řiditelnosti a pozorovatelnosti [11, 8]. Pro tento typ popisu existují různé principy jak řiditelnost a pozorovatelnost vyjádřit a počítat. Mezi nejznámější se řadí metoda SCOAP vyvinutá Goldsteinem [18]. Ta pracuje na principu ohodnocení každého spoje obvodu celými čísly, které vyjadřují míru řiditelnosti resp. pozorovatelnosti. U řiditelnosti se udávají vždy dvě hodnoty, kdy jedna udává řiditelnost logické 0 a druhá řiditelnost logické 1. Současně metoda rozlišuje řiditelnost resp. pozorovatelnost kombinační a sekvenční. Ohodnocení každého spoje obvodu l tak sestává z následujících šesti údajů:

- Kombinační 0-řiditelnost značená $CC0(l)$.
- Kombinační 1-řiditelnost značená $CC1(l)$.
- Kombinační pozorovatelnost $CO(l)$.
- Sekvenční 0-řiditelnost značená $SC0(l)$.
- Sekvenční 1-řiditelnost značená $SC1(l)$.

- Sekvenční pozorovatelnost $SO(l)$.

Hodnoty řiditelnosti se vždy nacházejí v intervalu $\langle 1, \infty \rangle$ a hodnoty pozorovatelnosti v intervalu $\langle 0, \infty \rangle$. Čím jsou hodnoty nižší, tím je řiditelnost resp. pozorovatelnost lepší.

Při výpočtu konkrétních hodnot pro každý spoj obvodu se v prvním kroku algoritmu přiřadí každému vstupu obvodu hodnoty $CC0 = 1$ a $CC1 = 1$. Každému dalšímu spoji se stanoví hodnoty CC na základě výpočtu ze vstupních hodnot CC u prvku, jehož výstup do aktuálně vyhodnocovaného spoje vede. V algoritmu se obvykle postupuje vzestupně po hloubce logiky, která je určená počtem hradel mezi primárními vstupy obvodu a daným hradlem. Ve druhém kroku algoritmu tak jsou vypočítány řiditelnosti na výstupech hradel, jejichž vstupy jsou napojeny na primární vstupy. Ve třetím kroku pak řiditelnosti na výstupech hradel, která jsou spojena s primárními vstupy nebo s hradly ohodnocenými v předchozích krocích atd.

Výpočet hodnoty CC na výstupu hradla se provádí následovně. Pokud je požadovaná výstupní hodnota dosažena pomocí nastavení jakéhokoliv jednoho vstupu na určitou úroveň, potom je použit vzorec 2.4. Pokud je požadovaná výstupní hodnota dosažena pomocí nastavení všech vstupů na určitou úroveň, potom je použit vzorec 2.5. Pokud může být požadovaná výstupní úroveň dosažena různými kombinacemi vstupů (např. u hradla XOR), potom je použit vzorec 2.6. Je-li spoj v obvodě větven, pak hodnota řiditelnosti jednotlivých větví je rovna řiditelnosti kmene, tedy hodnoty na výstupu hradla, které do daného spoje vede. Konkrétní rovnice pro výpočet nejběžněji využívaných logických hradel jsou uvedeny v tabulce 2.7.

$$\text{výstupní řiditelnost} = min(\text{vstupní řiditelnosti}) + 1 \tag{2.4}$$

$$\text{výstupní řiditelnost} = \sum(\text{vstupní řiditelnosti}) + 1 \tag{2.5}$$

$$\text{výstupní řiditelnost} = min(\text{řiditelnosti vstupních množin}) + 1 \tag{2.6}$$

Hradlo	0-řiditelnost	1-řiditelnost
AND	$min(CC0(a), CCO(b)) + 1$	$CC1(a) + CC1(b) + 1$
OR	$CC0(a) + CC0(b) + 1$	$min(CC1(a), CC1(b)) + 1$
XOR	$min(CC0(a) + CC0(b), CC1(a) + CC1(b)) + 1$	$min(CC1(a) + CC0(b), CC0(a) + CC1(b)) + 1$
NAND	$CC1(a) + CC1(b) + 1$	$min(CC0(a), CCO(b)) + 1$
NOR	$min(CC1(a), CC1(b)) + 1$	$CC0(a) + CC0(b) + 1$
XNOR	$min(CC1(a) + CC0(b), CC0(a) + CC1(b)) + 1$	$min(CC0(a) + CC0(b), CC1(a) + CC1(b)) + 1$
BUF	$CC0(a) + 1$	$CC1(a) + 1$
INV	$CC1(a) + 1$	$CC0(a) + 1$
Větev	$CC0(kmen)$	$CC1(kmen)$

Tabulka 2.7: Rovnice řiditelnosti nejběžnějších hradel se vstupy a a b

Po stanovení řiditelnosti pro všechny spoje obvodu je vypočtena pozorovatelnost zpětně od výstupů na vstupy. Každému výstupu l se přiřadí pozorovatelnost $CO(l) = 0$. Poté se postupuje zpětně obvodem dle hloubky logiky. Pro každé hradlo obvodu jsou vypočteny vstupní CO jako součet CO na výstupu daného hradla $+1$ a součet vstupních řiditelností pro logickou úroveň, která musí být na vstupu hradla, aby bylo hradlo v transparentním režimu. Tzn. aby změna logické úrovně na právě ohodnocovaném spoji změnila hodnotu na výstupu. Konkrétní rovnice pro výpočet nejběžněji využívaných logických hradel jsou uvedeny v tabulce 2.8.

Popis ohodnocení obvodu kombinačními ukazateli je pro tuto práci dostačující a sekvenční ohodnocení nebude dále diskutováno. Jeho popis je možné nalézt například v [8].

Hradlo	Pozorovatelnost	
	a	b
AND	$CO(y) + CC1(b) + 1$	$CO(y) + CC1(a) + 1$
OR	$CO(y) + CC0(b) + 1$	$CO(y) + CC0(a) + 1$
XOR	$CO(y) + min(CC0(b), CC1(b)) + 1$	$CO(y) + min(CC0(a), CC1(a)) + 1$
NAND	$CO(y) + CC1(b) + 1$	$CO(y) + CC1(a) + 1$
NOR	$CO(y) + CC0(b) + 1$	$CO(y) + CC0(a) + 1$
XNOR	$CO(y) + min(CC0(b), CC1(b)) + 1$	$CO(y) + min(CC0(a), CC1(a)) + 1$
BUF	$CO(y) + 1$	
INV	$CO(y) + 1$	
Větev	$min(CO(a), CO(b), \ldots, CO(n))$	

Tabulka 2.8: Rovnice pozorovatelnosti nejběžnějších hradel se vstupy a a b a výstupem y

2.2.8 Návrh pro snadnou testovatelnost

Pojem *návrh pro snadnou testovatelnost* též označovaný zkratkou DfT z anglického *Design for Testability* je označení situace, kdy je návrh obvodu uzpůsoben tak, aby se obvod snadněji testoval. Dochází k úpravě struktury obvodu a obvykle k vkládání přídavné logiky. Zlepšení testovatelnosti tak bývá vykoupeno zvětšením plochy čipu, změnou dynamických vlastností obvodu, příkonem, počtem pinů pouzdra atp. Správným návrhem úprav pro snadnou testovatelnost však lze negativní dopady značně eliminovat [11].

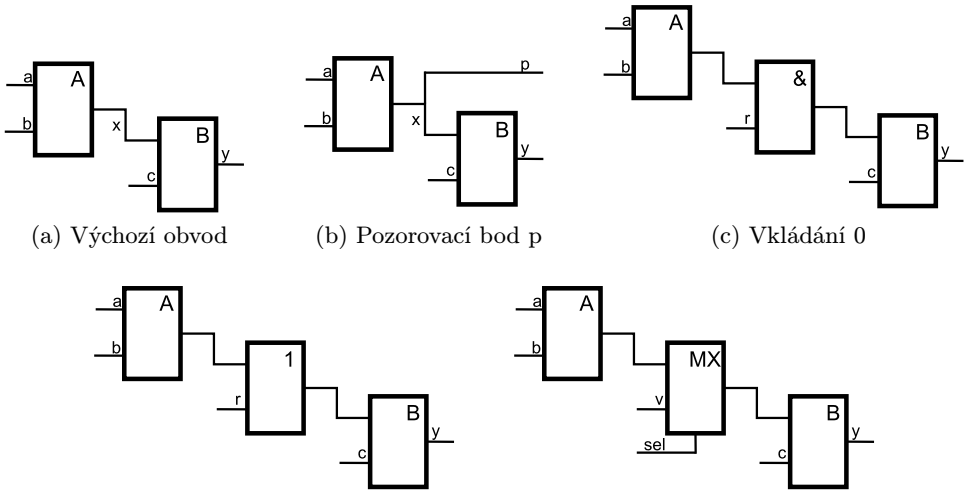
Techniky pro snadnou testovatelnost lze rozdělit do dvou základních skupin a to na ad-hoc techniky a techniky strukturovaného návrhu. Ad-hoc techniky jsou použitelné v konkrétních speciálních případech. Techniky strukturovaného návrhu jsou sofistikovanými obecněji použitelnými metodami využitelnými na různých úrovních abstrakce obvodu.

Ad-hoc techniky

Mezi nejznámější ad-hoc techniky patří technika vkládání testovacích bodů. Jejím cílem je zlepšení řiditelnosti nebo pozorovatelnosti určitého místa obvodu. Prakticky se jedná o možnost nastavit logickou hodnotu na určité místo obvodu nebo ji z tohoto místa odečíst. Existují tak dva základní typy testovacích bodů. Jeden slouží pro nastavení a druhý pro odečtení logické hodnoty. Nejjednodušším je testovací bod pro odečtení. Ten lze implementovat pomocí spoje z určeného místa vyvedeného přímo na primární výstup obvodu (obrázek 2.13b).

Testovací body pro nastavení logické hodnoty v určeném místě mohou být trojího typu. Mohou umožňovat nastavení logické 0, logické 1 nebo obě hodnoty. Všechny typy rozšiřují obvod o další logické prvky. Body pro možnost nastavení logické 0 je možné implementovat pomocí hradla logického součinu (obrázek 2.13c), body pro možnost nastavení logické 1 pomocí hradla logického součtu (obrázek 2.13d) a body pro možnost nastavení obou logických hodnot (tzv. úplné řízení) například pomocí multiplexoru (obrázek 2.13e).

Nevýhodou této techniky je nárůst počtu prvků obvodu a tedy plochy na čipu. Dále pak změna dynamických vlastností, neboť na cestě signálu je navíc další logický prvek, který signál opozdí. Hlavní nevýhodou je však lineární nárůst počtu potřebných vstupů a výstupů pouzdra obvodu vzhledem k počtu testovacích bodů. Pro každý testovací bod je potřeba jeden vývod pouzdra obvodu; v případě úplného řízení dva. Tento nedostatek je možné zmírnit pomocí přidání další logiky ve formě multiplexorů a demultiplexorů. n pozorovacích bodů je možné připojit na alespoň n-vstupový multiplexor s jedním výstupem a $m = \lceil \log_2 n \rceil$ řídicích vstupů (obrázek 2.14b). Při tomto řešení je ale možné v jednom okamžiku sledovat hodnotu pouze jednoho pozorovacího bodu.

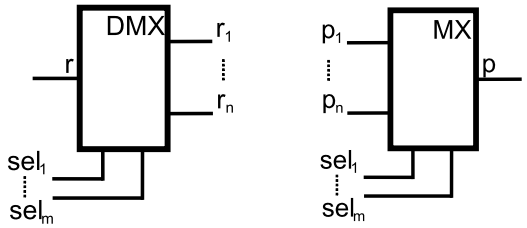


(a) Výchozí obvod (b) Pozorovací bod p (c) Vkládání 0

(d) Vkládání 1 (e) Vkládání 0 nebo 1

Obrázek 2.13: Metoda vkládání testovacích bodů

Pro snížení počtu n vstupů pro řídící body je možné použít n-výstupý demultiplexor o jednom datovém vstupu a $m = \lceil \log_2 n \rceil$ řídicích vstupů (obrázek 2.14a). Při tomto řešení je možné nastavovat současně pouze jediný řídicí bod. Při použití registrů zapojených na výstup demultiplexoru je možné nastavovat hodnoty i pro několik řídicích bodů. U obou přístupů je také možné další snížení počtů vývodů obvodu pomocí čítače připojeného na řídící vstupy multiplexoru resp. demultiplexoru. Další možnosti snížení počtu vývodů obvodu je možné nalézt například v [58, 1]



(a) Snížení počtu vstupů (b) Snížení počtu výstupů

Obrázek 2.14: Snížení počtu vývodů obvodu pro testovací body

K dalším ad-hoc technikám patří například možnost inicializace vnitřního stavu obvodu do určitého stavu, segmentace obvodu a rozbití globálních zpětných vazeb. Další je možné nalézt v [1, 58].

Techniky strukturovaného návrhu

Základním principem technik strukturovaného návrhu je rozdělení obvodu na kombinační a sekvenční část. Sekvenční část je pak upravena tak, aby bylo možné přímo nastavovat nebo pozorovat hodnoty u prvků z této části. Je-li toto možné u všech klopných obvodů, jedná se o tzv. úplný scan. Při úplném scan se sekvenční část při testu chová jako kombinační a je na ni možné použít nástroje pro kombinační obvody. Z důvodu úspory přídavné logiky příp.

jiných se také využívá tzv. částečného scan, kdy není možné nastavovat/pozorovat hodnoty u všech klopných obvodů. V tomto případě se stále musí používat testovací metodiky pro sekvenční obvody. Částečný scan ale způsobí snížení sekvenční hloubky a tedy náročnosti generování a aplikace testu. Další informace ohledně strukturovaného návrhu je možné nalézt například v [11, 8, 1].

2.3 Matematická optimalizace

Optimalizace je obecně hledání nebo vytváření řešení, které nabývá lepších (ideálně nejlepších možných) parametrů, než řešení výchozí nebo referenční. Z hlediska matematiky je optimalizace hledání takových hodnot proměnných, pro které *účelová funkce* nad těmito hodnotami dosahuje minima resp. maxima. Máme-li účelovou funkci definovanou jako $f : A \to \mathbb{R}$, pak hledáme takové $x_0 \in A$, že $f(x_0) \leq f(x)$ resp. $f(x_0) \geq f(x)$ pro všechna $x \in A$ [42]. Množina A často bývá podmnožinou Euclidovského prostoru $\mathbb{R}^n$ definovanou množinou omezujících podmínek. Omezující podmínky jsou pak definovány jako funkce $f_i : \mathbb{R}^n \to \mathbb{R}$ a pro každé $x \in A$ musí platit $f_i(x) \leq b_i, i = 1, ..., m$ kde $b_1, ..., b_m$ jsou konstanty omezující hranice [4]. Definiční obor množiny A se nazývá *prohledávaný prostor* nebo *přípustná množina* a prvky množiny se nazývají *kandidátními* nebo *možnými řešeními*.

Optimalizační problémy lze rozdělit na dvě kategorie. Jedna kategorie se zabývá optimalizací spojitých proměnných, druhá proměnných diskrétních, která se nazývá kombinatorická [42]. Ve spojité optimalizaci se obvykle hledá množina reálných čísel nebo funkce, v optimalizaci kombinatorické se hledá řešení z konečné nebo spočitatelně nekonečné množiny kandidátních řešení. Dále budou uvažovány pouze kombinatorické optimalizace, které jsou využity dále v práci.

Příkladem problémů řešených kombinatorickou optimalizací může být například problém obchodního cestujícího, úloha osmi dam atp. Pro řešení kombinatorických problémů byla vyvinuta řada postupů a algoritmů. Jedná se například o backtracking, simulované žíhání, horolezecký algoritmus, evoluční algoritmy atp.

2.3.1 Kompletní prohledání

Kompletní prohledání (anglicky exhaustive search) je jednoduchou prohledávací metodou. Jejím principem je postupně vyzkoušet všechna kandidátní řešení. Výhodou tohoto přístupu je, že vždy nalezne nejlepší řešení. Může tak taky sloužit jako důkaz, že lepší řešení neexistuje. Další výhodou je jednoduchost implementace. Velkou nevýhodou a limitujícím faktorem použití této metody je složitost výpočtu úměrně rostoucí s počtem kandidátních řešení. Roste-li množina kandidátních řešení s velikostí problému exponenciálně, pak i náročnost výpočtu má exponenciální charakter. Vezmeme-li v potaz například úlohu osmi dam, počet všech kandidátních řešení bez aplikace dalších omezujících pravidel a heuristik je 64^8 tj přibližně 10^{14} možných řešení. Prakticky se tak dá tato metoda použít pouze na malé problémy a nebo existují-li jiné kvalitní techniky (například heuristiky), které množinu kandidátních řešení zmenší na únosnou velikost. Příklad implementace je vidět v algoritmu 2.1.

2.3.2 Náhodné hledání

Náhodné hledání (anglicky random search) pracuje na principu náhodného výběru prvků z množiny kandidátních řešení. Náhodný výběr ale bývá omezen na prvky, které se nachází do určité vzdálenosti od aktuálního řešení (jsou v tzv. hyperkouli). Výhodou algoritmu je, že

```
procedure kompletniProhledani(problem P) {
  reseni = ziskejPrvniReseni(P);
  nejlepsireseni = reseni;
  while reseni != NULL do {
    if kvalita(reseni) > kvalita(nejlepsireseni) {
      nejlepsireseni = reseni;
    }
    reseni = ziskejDalsiReseni(P, reseni);
  }
  return nejlepsireseni;
}
```

Algoritmus 2.1: Optimalizace kompletním prohledáním

nemusí pracovat s čistě konvexními resp. konkávnimi problémy a dovede se vymanit z lokálních minim resp. maxim. Pokud se však v množině kandidátních řešení nachází málo lepších řešení, princip náhodného výběru má malou pravděpodobnost nalézt lepší řešení. Současně nachází-li se jednotlivá lepší řešení ve větší vzdálenosti, než které dovoluje překročit hyperkoule, algoritmus uvázne v lokálním minimu resp. maximu. Příklad implementace je vidět v algoritmu 2.2.

```
procedure nahodneHledani(problem P, maxvzdalenost l) {
  reseni = ziskejNahodneReseni(P);
  nejlepsireseni = reseni;
  while !ukoncitHledani(P, nejlepsireseni) do {
    reseni = ziskejNahodneReseni(P, nejlepsireseni, l);
    if kvalita(reseni) > kvalita(nejlepsireseni) {
      nejlepsireseni = reseni;
    }
  }
  return nejlepsireseni;
}
```

Algoritmus 2.2: Optimalizace náhodným hledáním

2.3.3 Horolezecký algoritmus

Horolezecký algoritmus (anglicky Hill-climbing) je metodou, která pracuje na principu posunu po gradientu funkce. Jejím principem je postupný posun od aktuálního řešení k sousednímu lepšímu řešení dokud takové řešení existuje. Při startu algoritmu se náhodně nebo na základě nějaké logiky vybere jedno řešení a to je ustaveno jako aktuálně nejlepší možné. Poté algoritmus začne prohledávat nejbližší okolí a případné lepší řešení se stane řešením aktuálním. Algoritmus se opakuje, dokud je možné kolem aktuálního řešení nalézt řešení lepší.

Hodně užívanou variantou je případ, kdy algoritmus nepřevezme jako aktuální řešení první lepší v okolí nalezené řešení, ale k aktuálnímu řešení ohodnotí všechna okolní a vybere z nich to nejlepší. Princip je popsán v algoritmu 2.3.

Výhodou algoritmu je jeho jednoduchost a rychlost konvergence k nejlepšímu řešení. Nevýhodou je, že algoritmus dobře pracuje pouze s konvexními nebo konkávními problémy. Jakmile jsou ale funkce složitější a obsahují lokální minima resp. maxima, algoritmus v nich uvázne.

```
procedure horolezekeHledani(problem P) {
  nejlepsireseni = ziskejStartovaciReseni(P);
  do {
    noveReseniNalezeno = false;
    okolniReseni = ziskejOkolniReseni(P, nejlepsireseni);
    reseni = ziskejPrvniReseni(okolniReseni);
    while reseni != NULL do {
      if kvalita(reseni) > kvalita(nejlepsireseni) {
        nejlepsireseni = reseni;
        noveReseniNalezeno = true;
      }
      reseni = ziskejDalsiReseni(okolniReseni);
    }
  } while (noveReseniNalezeno);
  return nejlepsireseni;
}
```

Algoritmus 2.3: Optimalizace horolezeckým algoritmem

2.3.4 Zpětné vyhledávání

Metoda zpětného vyhledávání (anglicky Backtracking) je vylepšením kompletního prohledávání o možnost vyloučení velké části kandidátních řešení bez přímého vyzkoušení. Pracuje na principu postupné tvorby kandidátních řešení a vyloučí z řešení všechna částečná řešení c (backtracks) jakmile zjistí, že c nemůže být dopracováno na platné řešení [66]. Vezmeme-li například úlohu osmi dam, pak jakmile algoritmus dospěje k částečnému řešení, kde se vzájemně ohrožují dvě dámy, pak může algoritmus ukončit jakékoliv další dopracovávání tohoto řešení, neboť již nemůže být splněno zadání. Příklad implementace je vidět v algoritmu 2.4.

Výhodou zpětného vyhledávání je, že stejně jako kompletní vyhledávání dokáže nalézt řešení, pokud existuje. Oproti kompletnímu vyhledávání však může jeho složitost s rostoucí složitostí problému růst pomaleji v závislosti na tom, jak moc je možné pomocí částečných řešení redukovat kandidátní řešení. Nevýhodou této metody je, že může pracovat pouze s problémy, u kterých je možné pracovat s částečnými řešeními. Současně efektivní je tato metoda pouze tehdy, pokud je u částečného řešení možné rychle určit, zdali jde dopracovat na platné řešení. Pro efektivitu je také důležité, aby dokázala tato metoda efektivně redukovat množinu všech kandidátních řešení o řešení neplatná.

```
procedure zpetneVyhledavani(problem P, castecnereseni c) {
  nejlepsireseni = NULL;
  if kompletni(P, c) {
    nejlepsireseni = c;
  } else if !zamitnute(P, c) {
    creseni = prvniReseni(P, c);
    while creseni != NULL do {
      reseni = zpetneVyhledavani(P, creseni);
      if kvalita(reseni) > kvalita(nejlepsireseni) {
        nejlepsireseni = reseni;
      }
      creseni = dalsiReseni(P, creseni);
    }
  }
  return nejlepsireseni;
}
```

Algoritmus 2.4: Optimalizace zpětným vyhledáním

Kapitola 3

Multifunkční elektronika

Pod pojmem multifunkční elektronika je v této práci chápána taková, která umí měnit svoji funkci a to předvídatelným, požadovaným a kontrolovatelným způsobem. Jelikož se práce zabývá obvody popsanými na úrovni hradel, budou diskutována multifunkční logická hradla.

Multifunkční logická hradla jsou taková, která umí změnit svoji logickou funkci v závislosti na řídicí podmínce. Řídící podmínkou může být obecně cokoliv, co dovede funkci hradla změnit. Označení funkce multifunkčního hradla s n funkcemi je $X_1/X_2/\ldots/X_n$ kde každé X_i je standardní logickou funkcí (např. pro funkce dvou vstupních proměnných se jedná o funkci z tabulky 2.2) a říká se jí i-tá logická funkce. Někdy se také o i-té funkci mluví jako o funkci v i-tém režimu multifunkčního logického hradla. Hodnoty řídicí proměnné se mohou zapsat v podobném formátu jako $Y_1/Y_2/\ldots/Y_n$ kde každé Y_i je hodnotou řídicí proměnné (nebo řídicích proměnných). Platí, že pro správnou funkci musí být v jednom okamžiku platná právě jedna Y_i a v době její platnosti má hradlo funkci X_i. Příkladem může být hradlo AND/OR řízené napájecím napětím 1.5/3.3 V. Jedná se o multifunkční logické hradlo se dvěmi funkcemi, kde v případě napájecího napětí 1.5V plní hradlo logickou funkci AND a v případě napájecího napětí 3.3V plní logickou funkci OR.

Samotná implementace multifunkčních hradel může být různá. Hradla mohou být implementována pomocí konvenčních postupů k tvorbě logických hradel nebo mohou být implementována pomocí nových metodik jako je například polymorfní elektronika nebo elektronika založená na grafénu.

3.1 Konvenční hradla

Implementace konvenčních multifunkčních logických hradel využívá standardní technologie a standardní postup návrhu logických hradel. Řízení funkce hradla je prováděno řídícími vstupy, které mají stejné vlastnosti jako vstupy funkční. Tedy ovládají se standardními hodnotami logických úrovní, způsobují obdobnou zátěž předchozího budícího stupně atp. Z hlediska funkčních a řídicích vstupů se tak hradlo chová jako klasické hradlo popsané například pravdivostní tabulkou, kdy jeden nebo více vstupů se chápou jako vstupy řídicí a ostatní jako vstupy funkční. Změnou logické úrovně na řídicích vstupech se pak mění logická funkce hradla mezi funkčními vstupy a výstupem. Jelikož mezi funkčními a řídicími vstupy není fyzikálního rozdílu, dá se obecně říct, že jakékoliv klasické minimálně dvouvstupové hradlo může být chápáno jako multifunkční, kdy některé vstupy jsou stanoveny jako vstupy řídicí a ostatní jako vstupy funkční. Označení konvenčně implementovaného hradla

za multifunkční je tak dáno způsobem pohledu na určité klasické hradlo a ne speciálními vlastnostmi daného hradla.

3.1.1 Princip

Nejjednoduššími multifunkčními hradly mohou být dvouvstupová hradla. Jako příklad nyní uvažujme hradlo XOR, jehož pravdivostní tabulku je možné nalézt v tabulce 2.2. Budeme-li toto hradlo chápat jako multifunkční, kdy vstup B je vstupem řídicím, pak v případě $B = 0$ plní hradlo mezi vstupem A a výstupem funkci BUF, tedy přenosu stejného vstupu na výstup, a v případě $B = 1$ funkci INV, tedy inverze vstupu A. Z multifunkčního pohledu tak hradlo XOR může fungovat jako řízený invertor, kdy je možné ovlivnit, zda vstupní hodnotu invertuje nebo nikoliv. Hradlo XOR je tak možné označit jako multifunkční hradlo BUF/INV.

S rostoucím počtem vstupů hradla roste také počet možností, jakým způsobem dané hradlo jako multifunkční využít. Můžeme volit kolik a které vstupy jsou řídící, které funkční a kolik a jaké funkce hradla využijeme. Obecně však vždy musí platit $rv \geq \lceil \log_2 f \rceil$, kde rv je počet řídicích vstupů a f je počet funkcí multifunkčního hradla. Pokud vyžadujeme, aby byla logická funkce v každém režimu hradla unikátní (tedy aby hradlo neplnilo v různých dvou režimech stejnou funkci), bude také platit $2^{2^{fv}} \geq f$, kde fv je počet funkčních vstupů hradla. Počet možností, jak zvolit řízení hradla, je roven $\binom{v}{1} + \binom{v}{2} + \cdots + \binom{v}{rv}$, kde v je počet všech vstupů hradla.

Mějme například konvenční čtyřvstupové hradlo. Budeme-li z něj vytvářet hradlo multifunkční, můžeme jako řídicí vstupy zvolit jakýkoliv jeden samostatný vstup, libovolnou kombinaci dvojic vstupů bez opakování, případně i libovolnou kombinaci trojic bez opakování. Počet všech možností řízení (a tedy potenciálních možností funkcí) tak je $\binom{4}{1} + \binom{4}{2} + \binom{4}{3} = 4 + 6 + 4 = 14$. Můžeme tedy získat čtyři různá multifunkční hradla funkce tří proměnných (tři funkční a jeden řídicí vstup), šest různých hradel funkce dvou proměnných (dva funkční a dva řídicí vstupy) případně čtyři různá hradla funkce jedné proměnné (jeden funkční a tři řídicí vstupy). Je však třeba mít na paměti, že různé řízení nezaručuje různou funkci multifunkčního hradla. Různé způsoby řízení mohou vést k funkčně stejným multifunkčním hradlům a celkový počet možných funkcí multifunkčního hradla tak může být nižší.

Jako konkrétní případ nyní uvažujme hradlo popsané tabulkou 3.1. Pokud zvolíme vstupy A a B jako řídicí a zbylé vstupy jako funkční, potom získáme multifunkční hradlo AND/OR/NAND/NOR. Funkce AND plní hradlo v případě $AB = 00$, funkci OR v případě $AB = 01$, funkci NAND v případě $AB = 10$ a funkci NOR v případě $AB = 11$. Další možností je zvolit jako řídicí vstup pouze vstup A a zbylé vstupy jako funkční. Potom hradlo v případě $A = 0$ plní funkci majority a v případě $A = 1$ plní funkci negované majority. Podobně lze popsat funkce i pro další možnosti volby řídicích vstupů.

3.1.2 Existující hradla

Jak bylo řečeno v předchozí kapitole, konvenční multifunkční logická hradla jsou klasickými hradly, u kterých jsou některé vstupy zvoleny jako řídicí. Každé existující konvenční hradlo s alespoň dvěma vstupy tak může být chápáno jako multifunkční a tudíž existuje velká spousta potenciálně multifunkčních hradel. V některých případech jsou však hradla navrhována od počátku jako multifunkční. Spíše se však ale jedná o situace, kdy je takové hradlo uloženo samostatně v balení s vývody pro umístění na desku plošných spojů. Pro ilustraci a jako příklad níže uvedu dva typy podobných hradel. Rozhodně se však nejedná o vyčerpávající popis a existují další podobná řešení.

A	B	C	D	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Tabulka 3.1: Pravdivostní tabulka multifunkčního hradla

Toshiba TC7SPXX

Toshiba TC7SPXX jsou třívstupá CMOS hradla plnící různé logické funkce. Každá z dostupných funkcí je volitelná pomocí vstupních kombinací. Hradlo je umístěno v pouzdře se šesti vývody. Existují čtyři druhy a to TC7SP57, TC7SP58, TC7SP97 a TC7SP98. Popis jejich funkce je uveden v pravdivostní tabulce 3.2. Multifunkční hradla dosažitelná těmito obvody při jednom řídicím a dvou funkčních vstupech jsou uvedena v tabulce 3.3. Podrobnější popis hradel je možné nalézt na stránce výrobce [35] a v produktových listech [60, 61].

Vstupy			Výstupy			
			TC7SP57	TC7SP58	TC7SP97	TC7SP98
A	B	C	Y	Y	Y	Y
L	L	L	H	L	L	H
L	L	H	L	H	L	H
L	H	L	H	L	H	L
L	H	H	H	L	L	H
H	L	L	L	H	L	H
H	L	H	L	H	H	L
H	H	L	L	H	H	L
H	H	H	H	L	H	L

Tabulka 3.2: Pravdivostní tabulka Toshiba TC7SPXX

Texas Instruments CD4048B

Texas Instruments CD4048B je multifunkční logické hradlo s osmi funkčními vstupy označenými A-H, čtyřmi řídicími vstupy označenými K_a, K_b, K_c a K_d, jedním výstupem a jedním vstupem "EXPAND". Hradlo je umístěno v pouzdře se šestnácti vývody. Řídicí vstupy K_a,

Řídící vstup	TC7SP57	TC7SP58	TC7SP97	TC7SP98
A	$\longleftarrow$/AND	$\not\longleftarrow$/NAND	$\not\longrightarrow$/OR	$\longrightarrow$/NOR
B	NOR/$\longrightarrow$	OR/$\not\longrightarrow$	AND/$\longleftarrow$	NAND/$\not\longleftarrow$
C	NOT A/B	A/NOT B	B/A	NOT B/NOT A

Tabulka 3.3: Dosažitelná multifunkční hradla pro obvody Toshiba TC7SPXX

K_b a K_c slouží k přepínání mezi osmi různými funkcemi hradla uvedenými v tabulce 3.4. Poslední čtvrtý řídicí vstup slouží k možnosti přepnutí výstupu hradla do stavu vysoké impedance pro možnost připojení obvodu do sběrnice. Pro $K_d = 1$ je hradlo ve standardním režimu a pro $K_d = 0$ je výstup ve stavu vysoké impedance. Vstup "EXPAND"slouží pro možnost rozšíření počtu vstupů hradla připojením výstupu dalšího stejného hradla na tento vstup. Například dvěmi těmito hradly lze získat šestnáctivstupové multifunkční hradlo. Podrobnější popis hradla je možné nalézt v produktovém listu [10].

K_a	K_b	K_c	Funkce	Rovnice
0	0	0	NOR	$Y = \overline{A + B + C + D + E + F + G + H}$
0	0	1	OR	$Y = A + B + C + D + E + F + G + H$
0	1	0	OR + AND	$Y = (A + B + C + D) \cdot (E + F + G + H)$
0	1	1	OR + NAND	$Y = \overline{(A + B + C + D) \cdot (E + F + G + H)}$
1	0	0	AND	$Y = A \cdot B \cdot C \cdot D \cdot E \cdot F \cdot G \cdot H$
1	0	1	NAND	$Y = \overline{A \cdot B \cdot C \cdot D \cdot E \cdot F \cdot G \cdot H}$
1	1	0	AND + NOR	$Y = \overline{A \cdot B \cdot C \cdot D + E \cdot F \cdot G \cdot H}$
1	1	1	AND + OR	$Y = A \cdot B \cdot C \cdot D + E \cdot F \cdot G \cdot H$

Tabulka 3.4: Tabulka funkcí hradla Texas Instruments CD4048B

3.2 Polymorfní hradla

Polymorfní elektronika (polytronika) je druh elektroniky, která je schopna měnit svoji funkci dle okolních podmínek (např. teplo, světlo, napájecí napětí, řídicí vstup, radiace atp.) [56]. Tento princip byl představen v [53], patentován v roce 2000 a zkoumán v NASA Propulsion Laboratory v Pasadeně. Základní myšlenkou polytroniky je vytvořit elektronické obvody, které by byly schopny reagovat na vnější podněty změnou své funkce a to pouze na základě fyzikálních vlastností technologie použité pro jejich implementaci bez speciálních senzorů.

U zatím známých polymorfních hradel implementovaných CMOS technologií se využívá zejména vlastností parametru V_T tedy prahového napětí otevření tranzistoru diskutovaného v kapitole 2.1.1. U něj se využívá jak vliv na otevření/uzavření tranzistoru vzhledem k velikosti napětí, tak i jeho závislosti na teplotě. Díky těmto vlastnostem je možné implementovat polymorfní logická hradla pomocí technologie CMOS závislá na napájecím napětí, řídícím napěťovém vstupu nebo na teplotě prostředí. Polymorfní hradla závislá na jiných okolních podmínkách (např. světlo, radiace ...) se mohou objevit jako výsledek dalších výzkumů.

Většina publikovaných polymorfních hradel je uvedena v tabulce 3.5. Implementace některých z nich je pak na obrázku 3.1. Polymorfní hradla představená Adrianem Stoicou a spol. prezentovaná v [55, 56, 57, 54] byla získána hledáním řešení pomocí evolučních technik. Hradlo prezentované v [47] bylo vyvinuto přímým návrhem Romanem Prokopem z UMEL

FEKT VUT Brno. Bližší prozkoumání uvedených hradel bylo provedeno v rámci mé práce a je uvedeno v kapitole 5.2.

Funkce	Řídicí úrovně	Řídicí proměnná	Tran.	Obr.	Zdroj
AND/OR	27/125°C	teplota	6		[56]
AND/OR/XOR	3.3/0.0/1.5V	napětí řídicího vstupu	10		[56]
AND/OR	3.3/0.0V	napětí řídicího vstupu	6	3.1b	[56]
NAND/NOR/XOR/AND	0.0/0.9/1.1/1.8V	napětí řídicího vstupu	11		[54]
AND/OR	1.2/3.3V	napájecí napětí	8	3.1a	[56]
NAND/NOR	3.3/1.8V	napájecí napětí	6	3.1c	[55]
NAND/NOR	5/3.3V	napájecí napětí	8	3.1d	[47]

Tabulka 3.5: Publikovaná polymorfní hradla

3.3 Grafenová hradla

Grafen je pojmenování materiálu, jehož strukturu tvoří uhlíkové atomy uspořádané do jednovrstvé šestiúhelníkové mřížky (viz obrázek 3.2). Jedná se tak o 2D strukturu uspořádanou shodně jako medové plástve [17]. Do roku 2004 se předpokládalo, že materiál podobné struktury nemůže samostatně existovat a využívalo se jej převážně na akademické půdě jako modelu pro popis dalších uhlíkových struktur. V roce 2004 se však Andre Geimovi a Konstantinu Novoselovovi podařilo izolovat samostatný grafenový plát a v roce 2010 za experimenty týkající se dvourozměrného grafenu dostali nobelovu cenu za fyziku [38].

Po zjištění, že podobná struktura může samostatně stabilně existovat a popsání jednoho z principů, jak podobnou strukturu získat, se rozeběhlo velké množství výzkumů v různých technologických oborech. Výzkumy se provádí například v oblasti získání grafenu, využití v konstrukčních kompozitních materiálech, bateriích, polovodičové elektronice aj. [17].

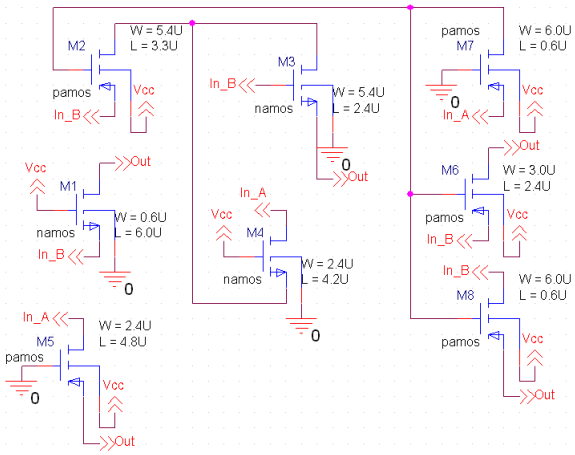
3.3.1 Grafen v elektronice

Z hlediska elektroniky a digitálních číslicových obvodů se do grafenu vkládají velké naděje. Bylo ukázáno, že grafen může fungovat jako FET tranzistor a tudíž může být použit jako podkladová technologie pro elektronické obvody stejně jako dnešní technologie založená převážně na křemítku [39, 28, 7]. V roce 2006 byl firmou IBM vytvořen kompletní integrovaný logický obvod sestavený z tranzistorů implementovaných na uhlíkových nanotrubičkách [9]. Ve vědeckých kruzích panuje předpoklad, že grafenové technologie časem postupně nahradí dnešní křemíkové [9, 17, 7]. Dle odhadu by k tomu ale nemělo dojít dříve než za 20 let [17].

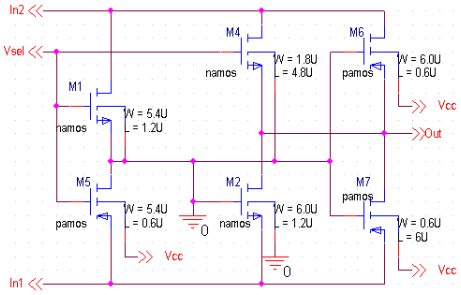
Jednou z velmi zajímavých elektrických vlastností grafenu je, že i při pokojových teplotách (kolem 300°K) se elektrony pohybují velmi rychle a ztrácí velmi málo energie. Prakticky se tak chovají jako nehmotné částice [17, 27]. To současně s minimální velikostí grafenu vede k předpokladu možnosti tvorby tranzistorů atomární velikosti o veliké rychlosti spínání. Obecně se předopkládá dosažení terraherzových rychlostí [17].

3.3.2 Grafenové multifunkční hradlo

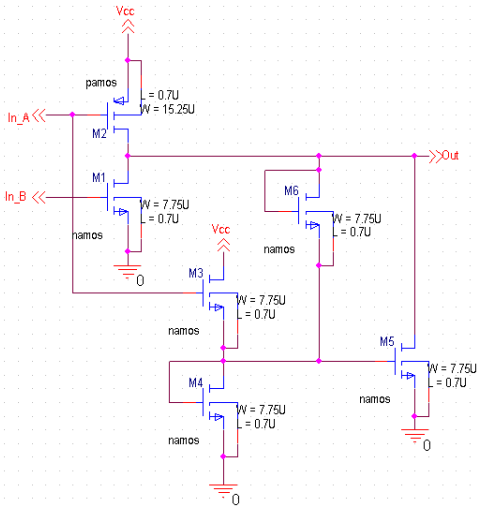
Jedna z prvních implementací grafenového multifunkčního hradla byla představená v [59] a je zobrazena na obrázku 3.3. Hradlo je vytvořeno na polovodičovém substrátu, ve kterém jsou vytvořeny tři oblasti $\overline{U}$, A a U v trojúhelníkovém tvaru pod úhlem 45°. Na hladké horní části substrátu je nanesena grafenová vrstva, na které jsou umístěny elektrody B, F a



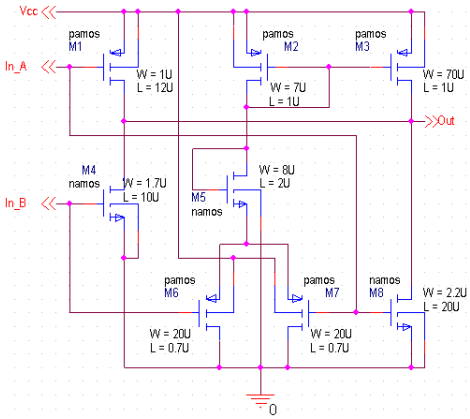
(a) AND/OR řízený napětím



(b) AND/OR řízený vstupem Vsel



(c) NAND/NOR řízený napětím (A. Stoica)

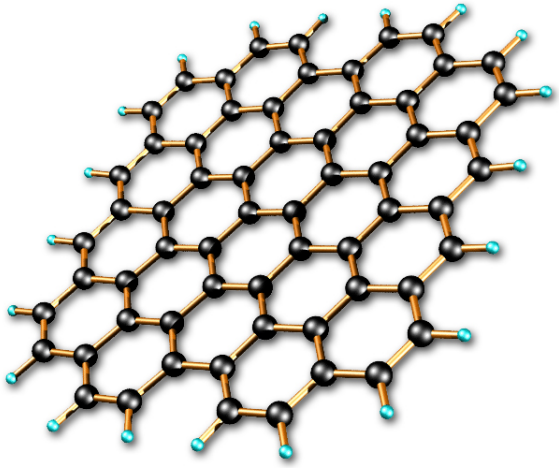


(d) NAND/NOR řízený napětím (R. Prokop)

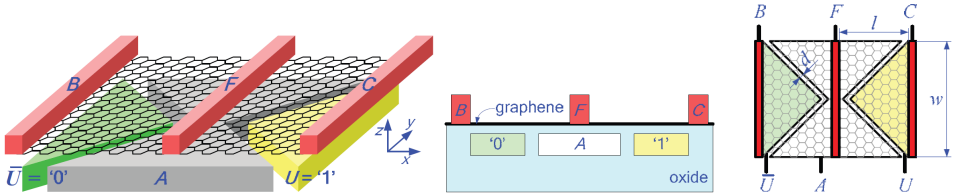
Obrázek 3.1: Implementace některých polymorfních hradel v CMOS technologii

C. Kontakty připojené k A, B a C tvoří vstupy hradla, kontakt připojený k F tvoří výstup hradla a kontakty U a $\overline{U}$ slouží k napájení. Při klasickém zapojení je $\overline{U}$ připojeno na nízký potenciál (zem) a U na vysoký potenciál (napájení). V tomto režimu označeném $U = 1$ plní hradlo logickou funkci popsanou rovnicí $Y = AC + \overline{A}B$. Hradlo však také umožňuje opačné zapojení, kdy na $\overline{U}$ je připojen vysoký potenciál (napájení) a na U potenciál nízký (zem). V takovém případě je režim označen $U = 0$ a hradlo plní funkci popsanou rovnicí $Y = \overline{A}C + AB$. Funkce pro oba režimy je popsána pravdivostní tabulkou 3.6a. Budeme-li jeden vstup chápat jako řídicí, pak je možné implementovat multifunkční hradla uvedená tabulce 3.6b.

Jednou z výhod uvedeného multifunkčního hradla je, že základem pro implementaci je shodný substrát jako pro tvorbu CMOS obvodů. Hlavním technologickým rozdílem je pouze nanesená grafenová vrstva mezi substrátem a elektrodami hradla. Dle [59] by tímto způsobem měla být dosažitelná integrace klasické CMOS technologie s technologií grafenovou a mělo by tak být možné vytvářet nové hybridní CMOS-grafenové obvody.



Obrázek 3.2: Struktura grafenu
[27]



(a) 3D pohled (b) Řez (c) Horní pohled

Obrázek 3.3: Struktura multifunkčního hradla s grafenem
[59]

Vstupy			Výstupy $U=1$	Výstupy $U=0$
A	B	C	Y	Y
0	0	0	0	0
0	0	1	0	1
0	1	0	1	0
0	1	1	1	1
1	0	0	0	0
1	0	1	1	0
1	1	0	0	1
1	1	1	1	1

(a) Pravdivostní tabulka

Řídicí vstup	Funkce $U=1$	Funkce $U=0$
A	B/C	C/B
B	AND/$\longrightarrow$	$\not\leftarrow$/OR
C	$\not\leftarrow$/OR	AND/$\longrightarrow$

(b) Dosažitelná multifunkční hradla

Tabulka 3.6: Tabulky funkce grafenového hradla

Kapitola 4

Cíle disertační práce

4.1 Motivace

Při současné složitosti číslicových obvodů je technologicky nemožné, aby byly všechny vyráběné kusy funkční. Zejména u nových technologií a velkých obvodů pak může být počet těch nefunkčních značný. Z ekonomického hlediska je však důležité, aby byly vadné kusy identifikovány co nejdříve. Čím později jsou odhaleny, tím větší vzniká škoda. Stále rostoucí složitost číslicových obvodů však testování komplikuje. Je důležité, aby bylo při rozumných nákladech dostatečně účinné. Jsou tak vyvíjený nové techniky a postupy jak testy vytvářet a jak modifikovat obvody, aby se snadněji testovaly. Kromě samotné tvorby obvodu a jeho testu začíná být velmi důležitým parametrem také doba aplikace testu na výsledný obvod. Jan Dohnal z ON Semiconductor uvedl, že doba testování obvodů zabírá přibližně třetinu času z celého procesu výroby. Zkrácení doby aplikace testu tak může ušetřit část prostředků a zrychlit celkový proces výroby.

V posledních několika letech se také začínají diskutovat technologie a principy pro tvorbu multifunkčních logických hradel. V roce 2000 byly patentovány principy polymorfních hradel a v roce 2010 bylo prezentováno hradlo založené na grafenu (více v kapitole 3). Podobné technologie umožňují mimo jiné také změnu funkce vnitřních prvků obvodu bez nutnosti modifikace jeho struktury na úrovni hradel. Změna funkce hradla uvnitř obvodu způsobí změnu transparentních cest, ekvivalentních poruch a v neposlední řadě také poruch, které jsou otestovány aktuálně sestavenou aktivační a detekční cestou. Tyto vlastnosti vedou k úvahám o možnostech využití multifunkčních hradel pro různé úpravy diagnostických vlastností obvodu, které by vedly k optimalizaci parametrů výsledných testů. Optimalizovanými parametry mohou být například počet testovacích vektorů nebo pokrytí poruch výsledného testu.

4.2 Cíle práce

Hlavním cílem této práce je ověření předpokladu, že multifunkčními prvky lze ovlivnit diagnostické vlastnosti obvodu takovým způsobem, že je možné dosáhnout požadovaných změn parametrů výsledného testu obvodu. Současně si práce klade za cíl navrhnout a implementovat metodiku pro optimalizaci parametrů testu založenou na tomto předpokladu a pomocí navržené metodiky předpoklad ověřit na úloze minimalizace počtu testovacích vektorů potřebných pro otestování obvodu při zachování ostatních kvalitativních parametrů testu. V rámci řešení se předpokládá splnění následujících dílčích cílů:

1. Navrhnout a popsat principy využití multifunkčních hradel v obvodech pro účely optimalizace testu.
2. Popsat tyto principy formálními prostředky a definovat nad těmito principy metodiku optimalizace testu.
3. Navrženou metodiku implementovat.
4. Pomocí implementace ověřit metodiku na úloze minimalizace počtu testovacích vektorů pro různé obvody včetně uznávané testovací sady ISCAS 85.
5. Analyzovat známé technologie tvorby multifunkčních hradel. Zjistit jejich vlastnosti a zhodnotit jejich vhodnost pro navrhovanou metodiku.

Důraz při návrhu a implementaci musí být kladen na jednoduchost a obecnost metodiky. Dále, aby bylo možné metodiku použít i pro komplexní obvody. Současně nesmí mít navržená metodika větší negativní vliv na ostatní kvalitativní parametry testu. Vytvořená metodika a její ověřovací implementace také musí být jednoduše začlenitelná do dnes standardního procesu tvorby obvodů a jejich testů, a musí být jednoduše použitelná se standardními návrhovými systémy.

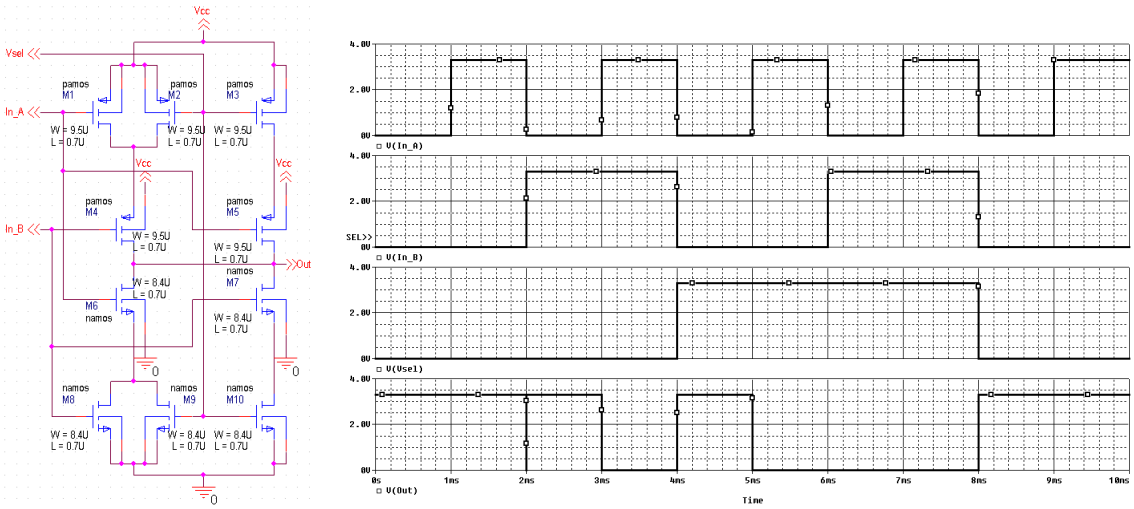
Kapitola 5

Multifunkční logická hradla

V kapitole 3 byla uvedena informace o současném stavu v oblasti multifunkčních logických hradel. V rámci této práce byly provedeny výzkumy a měření v oblasti zejména hradel polymorfních. Část pozornosti byla věnována také hradlům konvenčním.

5.1 Konvenční hradla

Jak bylo uvedeno v kapitole 3.1, konvenční multifunkční logická hradla jsou prakticky klasickými logickými hradly popsanými standardní pravdivostní tabulkou a implementovanými standardními technologiemi. V rámci práce jsem navrhnul multifunkční hradlo na bázi CMOS technologie zobrazené na obrázku 5.1a, jehož funkce je popsaná pravdivostní tabulkou 5.1. Toto hradlo má tři vstupy A, B a $Vsel$ a jeden výstup Out. Prakticky se jedná o třívstupové hradlo plnící funkci negované majority ze tří. Ať už je u tohoto hradla zvolen jako jeden řídicí vstup jakýkoliv, hradlo je vždy multifunkčním hradlem NAND/NOR. Tuto funkci považuji za jednu z nejdůležitějších, neboť funkce NAND i NOR jsou logicky kompletní a je možné s jejich pomocí vytvořit jakoukoliv logickou funkci či složitější číslicový obvod (viz kapitola 2.1.2).



(a) schéma (b) simulace OrCad pSpice

Obrázek 5.1: CMOS Implementace konvenčního multifunkčního NAND/NOR hradla

Vsel	B	A	Out
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Tabulka 5.1: Pravdivostní tabulka multifunkčního NAND/NOR hradla

Simulace průběhů hodnot napětí v OrCad pSpice je možné vidět na obrázku 5.1b. Jelikož se prakticky jedná o klasické CMOS logické hradlo, jsou i jeho elektrické vlastnosti obdobné s hradly ostatními. Hradlo tak na výstupu dosahuje kvalitních úrovní napětí pro obě logické úrovně a větší odběr proudu (a tedy příkon) má pouze při přepínání stavů. Ani dalšími parametry jako je šumová imunita, zpoždění hradla či logický zisk se při simulacích nijak nelišilo od jiných běžných CMOS hradel.

5.2 Polymorfní hradla

V kapitole 3.2 tabulce 3.5 je uvedena většina publikovaných polymorfních hradel. Pro každé hradlo vyjma hradla AND/OR ovládaného teplotou, jsem v OrCad PSPICE vytvořil model a odsimuloval jejich logické i elektrické vlastnosti. Je důležité poznamenamenat, že Adrian Stoica a spol. tvořili a simulovali hradla s využitím technologie HP 0.35 μm. Já pro simulace využil tranzistory z technologie AMI 0.7 μm, jejichž model je uveden v příloze B.

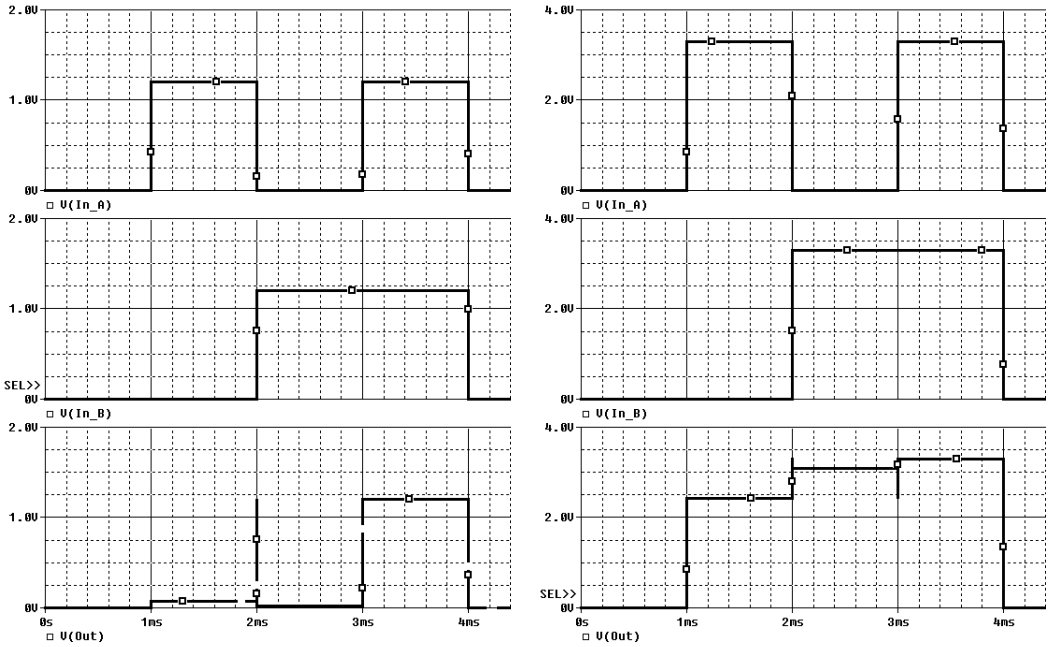
Při mých simulacích se ukázal zásadní problém se všemi hradly ovládanými napětím na řídicím vstupu. Žádné neplnilo uvedené logické funkce a nepodařilo se je zprovoznit. Předpokládám, že hlavní příčinou je pravděpodobně rozdílná hodnota parametru V_T jmenovaných výrobních technologií, případně odchylky v dalších parametrech. Tento první poznatek ukazuje jednu zásadní vlastnost polymorfních hradel a to, že jsou velmi závislá na výrobní technologii. Rozdíly parametrů výrobních technologií, které u konvenčních hradel neznamenají větší problém, mohou u polymorfních hradel působit zásadní potíže jako i úplnou nefunkčnost. Jelikož se tato hradla nepodařilo úspěšně odsimulovat, nebudou dále podrobněji analyzována. Níže jsou tak diskutována pouze hradla, jejichž simulace v OrCad PSPICE byla úspěšná.

5.2.1 AND/OR ovládané napájecím napětím

Hradlo AND/OR ovládané napájecím napětím zobrazené na obrázku 3.1a bylo úspěšně odsimulováno jak v nástroji OrCad PSPICE, tak v nástroji Spectre. Průběhy hodnot napětí jsou uvedeny na obrázku 5.2. Ze simulací a základních sledovaných parametrů uvedených v tabulce 5.3 je možné vyvodit následující problematické vlastnosti:

- Výstupní úrovně napětí nejsou příliš kvalitní. Největší rozdíl od ideální úrovně činí až 0.9V a to konkrétně v režimu OR při vstupech logická 1 na vstupu A a logická 0 na vstupu B viz obrázek 5.2b. Tento značný rozdíl způsobuje poměrně velký odběr proudu z napájení při zapojení dalších prvků na toto hradlo.

- Proudové zatížení předchozího prvku v ustáleném stavu může činit až 113 μA. To klade značné nároky na výkon předchozího stupně a způsobuje značný odběr i v ustáleném stavu.
- Maximální kmitočet je zejména v režimu AND silně závislý na zátěži hradla na výstupu.



(a) AND 1.2V Vcc

(b) OR 3.3V Vcc

Obrázek 5.2: Průběhy napětí na prázdno polymorfního hradla AND/OR

Jelikož bylo hradlo funkční i v simulacích v nástroji Spectre, bylo podrobeno dalšímu zkoumání, z něhož vyplynuly další dvě problematické vlastnosti. Pokud je na vstupu A logická 0 a na vstupu B logická 1, pak při přepínání režimu by mělo docházet k přepínání logické úrovně na výstupu. To je zřejmé z funkce hradla, neboť v režimu OR je správná výstupní hodnota logická 1 a v režimu AND logická 0. Toto však hradlo nedovede a při přepnutí režimu z OR na AND zůstane na výstupu hodnota logická 1. Po přepnutí funkce tak může hradlo na výstupu poskytovat neplatnou hodnotu.

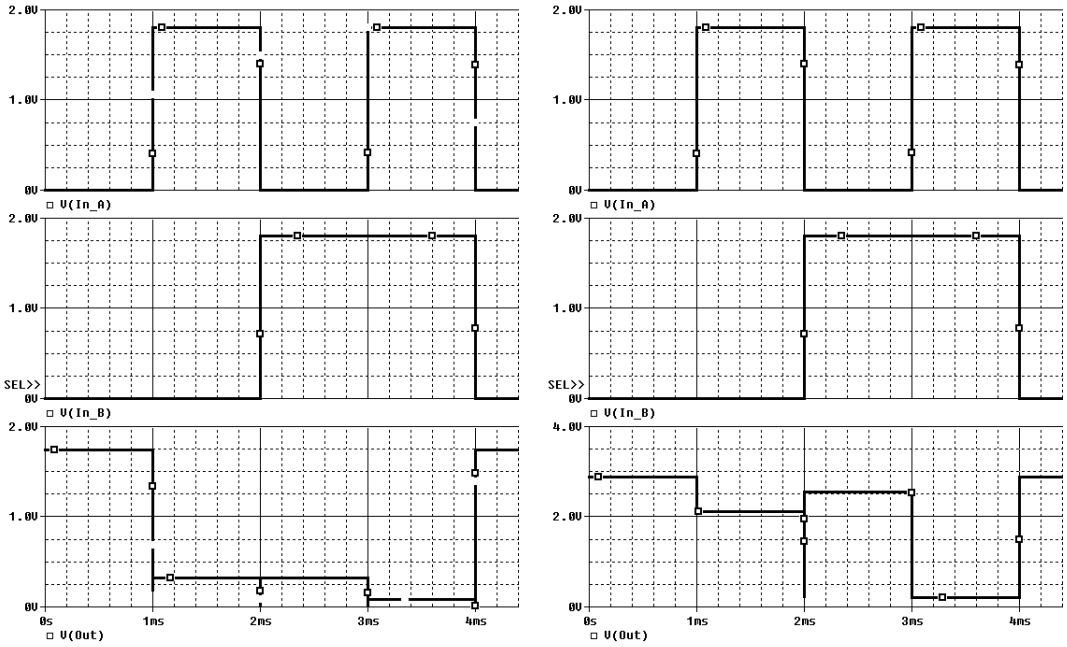
Druhou problematickou vlastností je situace, kdy na jednom vstupu je neměnná logická úroveň a na druhém vstupu se logické úrovně mění. V této situaci u hradla dochází k postupné degradaci výstupní úrovně a po každé změně na vstupu dochází postupně k přibližování výstupní úrovně k opačné logické úrovni, než je ta správná. Po větším počtu přepnutí je logická úroveň na výstupu opačná, než by měla být pro danou kombinaci vstupů. Pro správnou funkci hradla tak je potřeba, aby nedocházelo dlouhodobě ke stavu, kdy se mění logická úroveň pouze na jednom vstupu.

5.2.2 NAND/NOR ovládané napájecím napětím (A. Stoica)

Polymorfní hradlo NAND/NOR prezentované Adrianem Stoicou a spol. zobrazené na obrázku 3.1c je jedním z nejznámějších polymorfních hradel. Jeho návrh byl ověřen fyzickou

implementací [55]. Bohužel k tomuto hradlu nebyly uvedeny parametry šířky a délky kanálu a pro účely mé práce jsem je získal experimentálně. V mých simulacích bylo hradlo funkční pouze v OrCad PSPICE. V nástroji Spectre byly simulace neúspěšné.

Zvláštností tohoto hradla je, že úrovně napětí na vstupech jsou stále 1.8V nezávisle na tom, zdali plní hradlo funkci NOR nebo NAND, tedy zdali je napájecí napětí 1.8V nebo 3.3V. Výstupní úrovně jsou však již shodné s napájecím napětím. Prvky před hradlem tak musí pracovat stále s napětím 1.8V a prvky za hradlem s napětím shodným s napájecím napětím hradla. To vede ke značné komplikaci využití tohoto hradla v obvodech a není možné zapojit více těchto hradel za sebe.



(a) NOR 1.8V Vcc

(b) NAND 3.3V Vcc

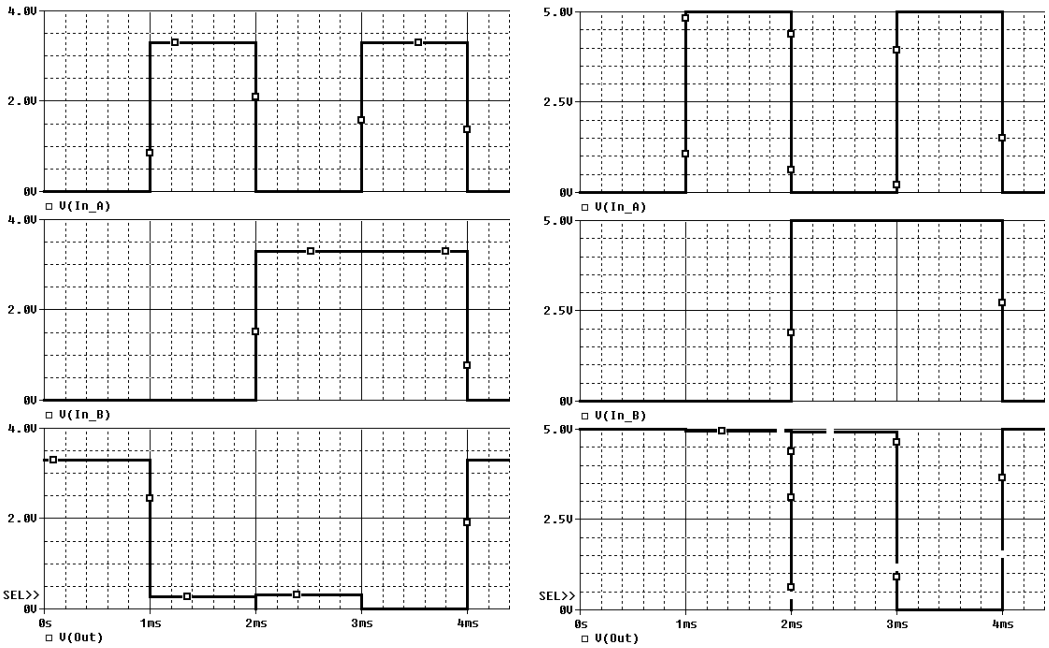
Obrázek 5.3: Průběhy napětí na prázdno polymorfního hradla NAND/NOR (A. Stoica)

Ze simulací a základních sledovaných parametrů uvedených v tabulce 5.3 je možné vyvodit následující problematické vlastnosti:

- Výstupní úrovně napětí nejsou příliš kvalitní. Největší rozdíl od ideální úrovně činí až 1.2V a to konkrétně v režimu NAND při vstupech logická 1 na vstupu A a logická 0 na vstupu B viz obrázek 5.3b. Tento značný rozdíl způsobuje poměrně velký odběr proudu z napájení při zapojení dalších prvků na toto hradlo.
- Hradlo má velmi velký odběr proudu z napájení, který může činit až stovky μA. To způsobuje značný odběr i v ustáleném stavu. Problém je zapříčiněn mj. i vstupními tranzistory M1 a M2, které jsou zapojeny stejně jako invertor, ale jejich brány jsou vyvedeny každá na jiný vstup. Velký odběr pak nastává v situaci, kdy na vstupu A je logická 0 a na vstupu B logická 1, tedy kdy jsou oba dva tranzistory otevřeny a proud může procházet přes oba tranzistory z napájení na zem.

5.2.3 NAND/NOR ovládané napájecím napětím (R. Prokop)

Polymorfní hradlo NAND/NOR bylo vytvořeno Romanem Prokopem na Fakultě UMEL FEKT VUT Brno ve spolupráci na projektu GAČR, GA102/06/0599 s názvem Metody návrhu polymorfních číslicových obvodů. Toto hradlo je jediné testované, které bylo navrhováno přímo pro použitou technologi AMI 0.7 μm a které bylo vytvořeno návrhem od návrháře, nikoli evolučními algoritmy. Simulace tohoto hradla byly úspěšné jak v OrCad PSPICE, tak ve Spectre.



(a) NOR 3.3V Vcc

(b) NAND 5.0V Vcc

Obrázek 5.4: Průběhy napětí na prázdno polymorfního hradla NAND/NOR (R. Prokop)

Ze simulací a základních sledovaných parametrů uvedených v tabulce 5.3 je možné identifikovat jednu problematickou vlastnost a to velký odběr proudu z napájení i v ustáleném stavu, který může činit až 123 μA. Tento odběr je způsobený více faktory. Prvním problematickým místem jsou vstupní tranzistory M1 a M4, které jsou zapojeny jako invertor, ale jejich brány jsou zapojeny každá na jiný vstup. V případě, kdy na vstupu A je logická 0 a na vstupu B logická 1 jsou oba otevřeny a proud může procházet přímo z napájení na zem. Druhým problematickým místem jsou tranzitory M3 a M8, kde v případě funkce NAND (napájecí napětí 5V) se otevírá i tranzistor M3 a proud může přes oba tyto tranzistory procházet z napájení přímo na zem. Výstupní úrovně je tak dosaženo tím, že tranzistor M3 má větší kanál a propustí více proudu a tedy výstupní úroveň je rovna logické 1 [47].

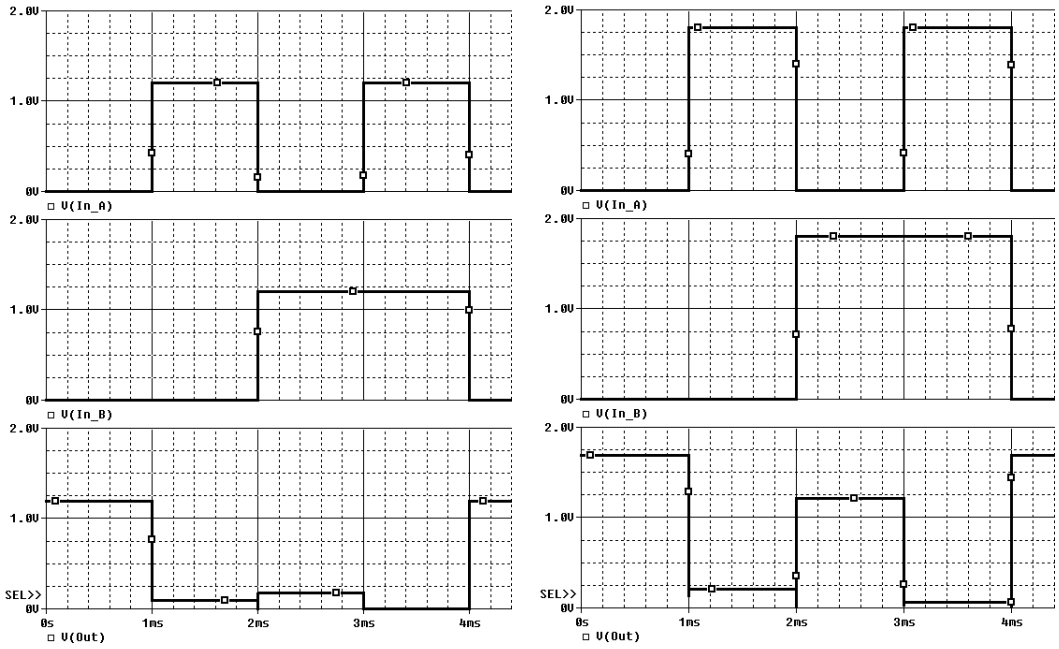
5.2.4 NOR/NOT A ovládané napájecím napětím

Při pokusech s hradlem NAND/NOR prezentovaném Adrianem Stoicou a spol. uvedeném v kapitole 5.2.2 jsem objevil takové kombinace parametrů tranzistorů, že se hradlo chovalo pouze jako invertor vstupu A. Po dalších experimentech byly stanoveny takové parametry šířek a délek kanálů tranzistorů, že se hradlo při napájecím napětí 1.2V chovalo jako NOR

a při napájecím napětí 1.8V jako invertor vstupu A. Hradlo má stejnou strukturu jako původní hradlo na obrázku 3.1c. Liší se pouze hodnotami šířek a délek kanálů tranzistorů, které jsou uvedeny v tabulce 5.2. Oproti výchozímu hradlu jsou vstupní úrovně pro logickou 1 shodné s napájecím napětím ve všech režimech a hradlo je tak možné lépe používat jako komponentu pro tvorbu větších obvodů.

Tranzistor	Šířka	Délka
M1	7μm	2μm
M2	10μm	0.7μm
M3	3μm	1μm
M4	20μm	0.7μm
M5	20μm	0.7μm
M6	7.75μm	0.7μm

Tabulka 5.2: Šířky a délky kanálů tranzistorů polymorfního hradla NOR/NOT A



(a) NOR 1.2V Vcc

(b) NOT A 1.8V Vcc

Obrázek 5.5: Průběhy napětí na prázdno polymorfního hradla NOR/NOT A

Ze simulací a základních sledovaných parametrů uvedených v tabulce 5.3 je možné vyvodit následující problematické vlastnosti:

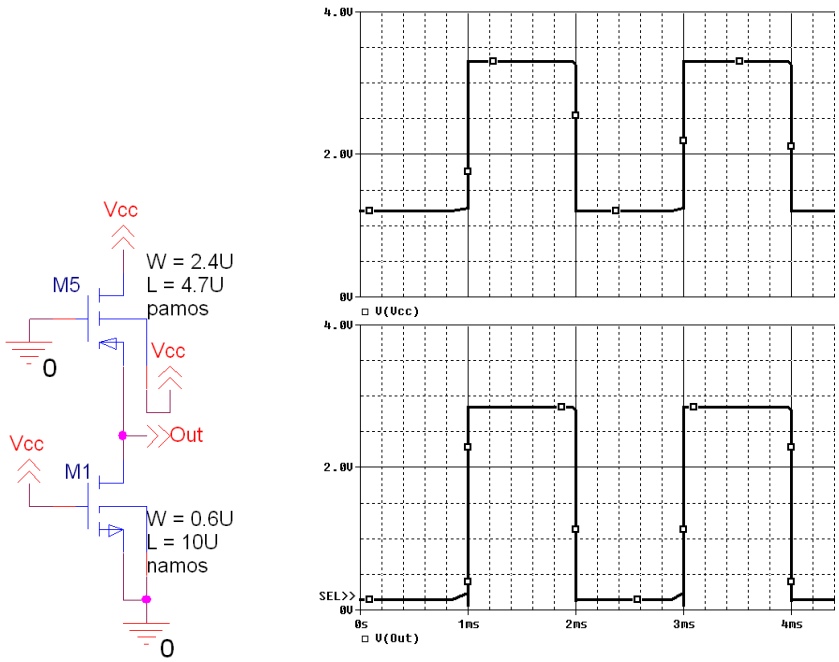
- Výstupní úrovně napětí nejsou příliš kvalitní. Největší rozdíl od ideální úrovně činí téměř 600mV a to konkrétně v režimu NOT A při vstupech logická 0 na vstupu A a logická 1 na vstupu B viz obrázek 5.5. Tento rozdíl je již v oblasti prahového napětí otevření tranzistoru V_T a může způsobovat velký odběr proudu z napájení při zapojení dalších prvků na toto hradlo.
- Hradlo má velmi velký odběr proudu z napájení, který může činit až 126μA v ustále-

ném stavu. Problém je zapříčiněn stejnými vlastnostmi, jaké má výchozí strukturně shodné NAND/NOR hradlo.

Hradlo jsem podrobil také simulacím v nástroji Spectre, kde se jevilo jako funkční. Problematická se však ukázala corner analýza. Ta byla použita na simulaci vlivu odchylek parametrů tranzistorů od ideálních nastavených hodnot, které mohou vzniknout při fyzické výrobě hradla. Simulovány byly maximální povolené odchylky použité technologie a hradlo bylo v režimu naprázdno. Výsledky simulace jsou zobrazeny na obrázku 5.7. V levé půlce jsou průběhy pro režim NOT A, v pravé půlce pro režim NOR. Horní polovina zobrazuje průběh hodnoty napětí na výstupu, spodní polovina průběh hodnoty odběru proudu z napájení. Z obrázku je patrné, že v několika případech hradlo nedovede plnit svoji funkci a výroba tohoto hradla použitou technologií by tak byla velmi problematická a vedla by pravděpodobně k velké chybovosti.

5.2.5 Detektor napájecího napětí

Detektor napájecího napětí je polymorfní hradlo, které nemá žádný vstup a má jediný výstup. V případě, kdy je hradlo připojeno na napájecí napětí 1.2V, má na výstupu logickou 0 a v případě napájecího napětí 3.3V má na výstupu logickou 1. Funguje tedy jako detektor napájecího napětí. Toto hradlo jsem objevil při podrobných analýzách a dekompozici hradla AND/OR prezentovaného v kapitole 5.2.1. V mých simulacích bylo hradlo funkční pouze v OrCad PSPICE. V nástroji Spectre byly simulace neúspěšné.



(a) Schéma (b) Průběh napětí naprázdno

Obrázek 5.6: Polymorfního hradlo detektor napájecího napětí

Ze simulací a základních sledovaných parametrů uvedených v tabulce 5.3 je možné vyvodit následující problematické vlastnosti:

- Výstupní úrovně napětí nejsou příliš kvalitní. Největší rozdíl od ideální úrovně činí téměř 500mV, viz obrázek 5.5. Tento rozdíl je v blízkosti prahového napětí otevření tranzistoru V_T a může způsobovat velký odběr proudu z napájení při zapojení dalších prvků na toto hradlo.
- Hradlo má velmi velký odběr proudu z napájení, který může činit až 11μA v ustáleném stavu. Problém se objevuje zejména v případě hodnoty napájení 3.3V, kdy jsou otevřeny oba tranzistory a proud může procházet z napájení přes oba tranzistory přímo na zem.
- Maximální kmitočet je silně závislý na výstupní zátěži hradla.

5.2.6 Shrnutí

V předchozích kapitolách byla uvedena všechna mnou v OrCad PSPICE úspěšně simulovaná polymorfní hradla. Základní naměřené informace pro tato hradla je možné nalézt v souhrnné tabulce 5.3. Jako zátěž bylo použito deset invertorů (CMOS invertor viz obrázek 2.7). Z této tabulky a předchozích kapitol je zřejmé, že popisovaná polymorfní hradla trpí problematickými vlastnostmi, které značně snižují možnost jejich rozáhlejšiho využití ve velkých obvodech. Obecně se dá říci, že hradla mají velké odběry proudu z napájení, některá nepřiměřeně zatěžují předchozí stupně, většina má nepřesné napěťové úrovně na výstupu a v neposlední řadě u některých hradel při zatížení výrazně klesá maximální pracovní kmitočet. Nepřesné napěťové úrovně na výstupu hradel následně vedou ke zvýšenému odběru proudu z napájení v případě zapojení dalších, i konvenčních, hradel na jejich výstup.

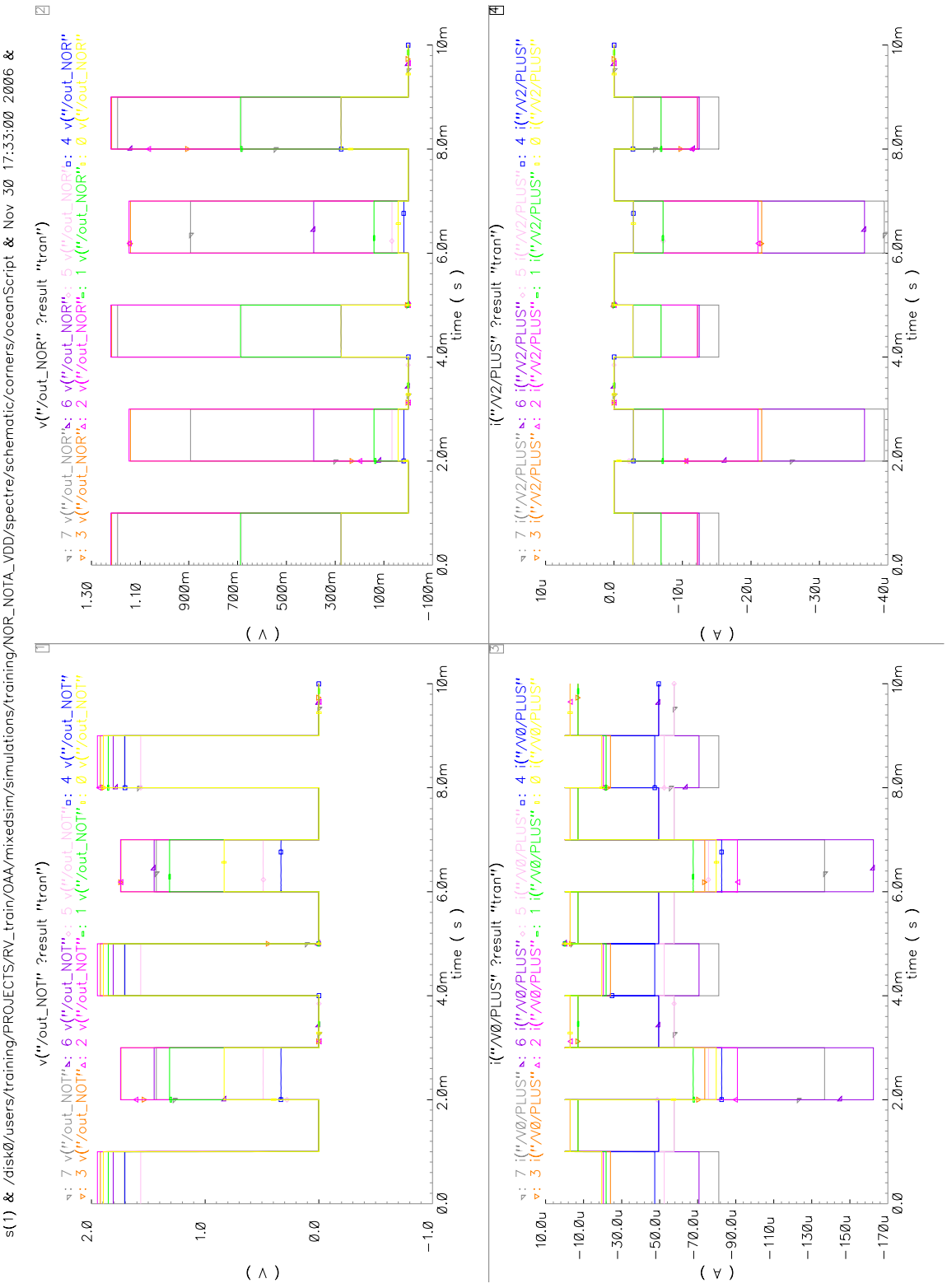
Polymorfní hradla také většinou mají malou šumovou imunitu a jsou velmi citlivá na přesné napěťové úrovně na vstupu. To současně s nepřesností jejich výstupních napěťových úrovní často způsobí, že nelze zapojit více podobných hradel přímo za sebe a je třeba, aby se mezi dvěma polymorfními hradly vyskytovalo alespoň jedno hradlo konvenční. Konvenční hradlo dovede lépe zpracovat nepřesné napěťové úrovně a na výstupu opět poskytne napěťové úrovně s minimálními odchylkami od úrovní ideálních, se kterými již polymorfní hradla dovedou korektně fungovat.

Jako hlavní příčiny uvedených problémů lze identifikovat nedodržení několika základních pravidel navrhování logických elektronických hradel na základě MOSFET. Prvním problémem je využití nMOS tranzistorů i pro přenos vysokých úrovní a pMOS tranzistorů pro přenos nízkých úrovní napětí. Vzhledem k vlastnostem MOSFET jako spínačů uvedených v kapitole 2.1.1 toto vede k tomu, že tranzistory nejsou schopny plně spínat požadovaný rozsah napětí a k problémům se zvýšeným odběrem proudu nebo nepřesnou napěťovou úrovní na výstupu hradla.

Druhým problémem je konstrukce hradel, kdy tranzistory nefungují v symetrických párech jako v CMOS technologii (viz kapitola 2.1.2). V hradle často bývá i ve stabilním neměnném stavu otevřená cesta pro proud z napájení přímo na zem a tranzistory na této cestě pak fungují jako odporový dělič. Tranzistory na cestě k výstupu, které dovedou přenést větší proud určují výstupní logickou úroveň. Pokud je přivedeno více proudu z napájení na výstup, než odvedeno z výstupu na zem, pak je na výstupu hodnota logická 1 a naopak. Tím je sice zajištěna požadovaná napěťová výstupní úroveň, ale vede to současně k trvalému zvýšenému odběru proudu z napájení i ve stabilním stavu a nepřesným výstupním napěťovým úrovním. U hradla AND/OR ovládaného napájecím napětím existují podobné cesty pro proud i mezi vstupy tohoto hradla a výstupem předchozího prvku a hradlo tak silně

proudově ztěžuje předchozí prvek, který musí být na podobný proudový odběr dostatečně dimenzován.

Z prezentovaných hradel se jako nejlepší z hlediska sledovaných parametrů jeví hradlo NAND/NOR ovládané napájecím napětím R. Prokopa. Toto hradlo oproti ostatním trpí pouze vysokým odběrem proudu z napájení a o něco nižším maximálním kmitočtem spínání oproti konvenčním hradlům, který je u této technologie v řádu stovek MHz. Předpokládám, že toto hradlo dopadlo nejlépe ze dvou hlavních důvodů. Bylo navrženo přímo pro technologii AMI 0.7 μm, ve které byly simulace prováděny, a současně bylo hradlo vytvořeno návrhem od návrháře s dobrou znalostí vlastností MOSFET tranzistorů a CMOS technologie a nikoliv evolučními postupy jako hradla ostatní.



Obrázek 5.7: Corner analýza na prázdno polymorfního hradla NOR/NOT A

	AND/OR napáj. napětí		NAND/NOR Stoica		NAND/NOR Prokop		NOR/NOT_A		Detektor napětí
	AND	OR	NAND	NOR	NAND	NOR	NOR	NOT_A	-
Rozdíl napětí výstupu	70 mV	0.9 V	1.2 V	326 mV	74 mV	303 mV	205 mV	593 mV	460 mV
Proud napájení	12 pA	32 pA	667 μA	235 μA	123 μA	10 μA	19 μA	126 μA	11 μA
Proud vstupu A	346 nA	113 μA	15 pA	4 pA	2 pA	6 pA	1 pA	3 pA	-
Proud vstupu B	346 nA	113 μA	5 pA	20 fA	1 pA	2 pA	148 fA	3 pA	-
Kmitočet na prázdno	10 MHz	10 MHz	1 GHz	100 MHz	10 MHz	10 MHz	10 MHz	10 MHz	10 MHz
Kmitočet při zátěži	100 kHz	1 MHz	100 MHz	10 MHz	10 MHz	1 MHz	1 MHz	1 MHz	10 kHz

Tabulka 5.3: Základní údaje polymorfních hradel zjištěné simulací v OrCad PSPICE

Kapitola 6

Optimalizace testu číslicových obvodů

Při tvorbě testovací posloupnosti logického obvodu se postupně vytvářejí testovací vektory tak, aby pokrývaly poruchy v analyzovaném obvodě. Výsledná posloupnost testovacích vektorů pak nabývá určitých kvalitativních parametrů popsaných různými vlastnostmi jako je například počet testovacích vektorů, pokrytí poruch aj. Použití multifunkčních hradel nabízí možnost upravit funkci vnitřních prvků obvodu a tím ovlivnit jeho chování i vzhledem k tvorbě testu. Navrhovaný způsob vychází ze základního předpokladu, že změnou funkce vnitřních prvků dojde ke změnám diagnostických vlastností celého obvodu. Dá se tedy hovořit o konkrétní hypotéze, jejíž ověření je předmětem této práce. Pro tyto účely byl vytvořen formální model, který definuje pojmy využité následně při tvorbě metodiky, který byl označen pojmem „optimalizace testu číslicových obvodů“. Na tomto formálním modelu pak operují procedury a algoritmy, které jsou v této kapitole rovněž popsány.

6.1 Princip metody

Metoda je vybudována na možnosti změnit funkci některých vnitřních hradel takovým způsobem, aby došlo ke zlepšení požadovaných parametrů testu. Jelikož je pro provoz obvodu samozřejmě nutné zachovat jeho původní funkci, je potřeba, aby upravená hradla umožňovala plnit i původní logickou funkci. Upravená hradla tak musí ve *funkčním* režimu plnit funkci tak, jak je potřeba pro funkci obvodu a v *testovacím* režimu plnit funkci vhodnější pro test. Této vlastnosti lze docílit pomocí multifunkčních hradel diskutovaných v kapitole 3 a 5.

Hlavní výhodou tohoto přístupu je, že přidáváním testovací logiky nedochází ke změně vnitřní struktury ve funkční logice obvodu. Pomineme-li logiku na řízení funkce multifunkčních hradel, počty vstupů, výstupů nebo spojení vnitřních prvků jsou nezměněné. Jediná změna je tak náhrada logického hradla za hradlo multifunkční s funkcí řízenou pro účely testu. Mají-li použitá multifunkční hradla stejné vlastnosti jako hradla původní, nedojde ani ke změně dynamických parametrů obvodu. To je výhodou oproti jiným technikám DfT, které mohou dynamické parametry měnit (například metoda vkládání testovacích bodů viz kapitola 2.2.8).

Další vlastností této metody je, že na základě řešeného problému, požadavků na test a omezujících podmínek umožňuje volbu, zdali bude možné přepínat hradla i v průběhu aplikace testu. Nejjednodušším způsobem je změnu funkce hradel v průběhu testu neumož-

nit. Pak stačí spojit řízení funkce všech multifunkčních hradel a vyvést je jako jeden vstup obvodu. Před samotným testem se funkce hradel v obvodě přepne do testovacího režimu, spustí se test a po jeho dokončení se obvod přepne zpět do režimu funkčního.

Druhou možností je umožnit změnu funkce hradel i v průběhu testu. Zde ovšem vznikají další možnosti ohledně zapojení řízení jednotlivých multifunkčních hradel. Obecně lze říci, že multifunkční hradla je možné rozdělit do skupin, v rámci každé skupiny spojit řízení hradel a pro každou skupinu samostatně vyvést řídicí vstup. V průběhu testu je tak možné řídit každou skupinu hradel samostatně. V extrémních případech se může stát, že existuje pouze jedna skupina hradel a tedy všechna multifunkční hradla obvodu se přepínají stejně, a nebo naopak kdy každá skupina obsahuje pouze jediné hradlo. Pak je pro každé hradlo vyvedený ovládací vstup samostatně a princip funkce je podobný metodě vkládání testovacích řídících bodů. To již může klást značné požadavky na počet vstupů obvodu. Počet potřebných vstupů obvodu pro řízení multifunkčních hradel lze však snížit stejnými technikami jako u metody vkládání řídících bodů (viz kapitola 2.2.8).

Celá metoda tak sestává ze tří základních dílčích úkolů. Identifikace hradel v obvodě, jejichž funkci je vhodné změnit, volba jejich testovací funkce a volba způsobu jejich řízení. Jelikož je struktura optimalizovaného obvodu daná, je také pevně daná množina všech jeho hradel a je konečná. Jelikož počet funkcí, které může jedno hradlo nabývat, je konečný (viz rovnice 2.2 z kapitoly 2.1.2) a množina všech hradel obvodu je také konečná, pak i množina všech funkcí, která mohou hradla v obvodě nabývat, je také konečná. Počet možností řízení konečné množiny hradel je také konečný. Celý problém tak nabývá konečného počtu možných řešení.

6.1.1 Formální definice prerekvizit

Abychom mohli pro řešení úlohy využít matematického aparátu, potřebujeme formálně definovat všechny základní prvky, nad kterými bude možné tento matematický aparát vybudovat. Budeme postupovat od elementárních prvků (např. množinu logických úrovní) přes složitější komponenty (např. logická hralda) až po celé logické obvody.

Jako první definujeme množinu logických úrovní, nad kterou budou všechny ostatní prvky pracovat.

Definice 6.1. *Nechť 0 a 1 jsou logické úrovně vyskytující se v analyzovaném obvodě, pak množinu* $L = \{0, 1\}$ *budeme nazývat množinou logických úrovní.*

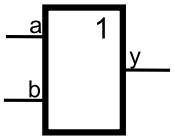
Jakmile máme definovánu množinu logických úrovní, můžeme definovat logické hradlo, základní stavební prvek logických obvodů. Logické hradlo je obecně tvořeno množinou vstupů a relací mapujících vstupní hodnoty na výstupní hodnotu.

Definice 6.2. *Nechť* I *je uspořádaná množina vstupů, nechť existuje relace* $gf : L^{|I|} \to L$, *pak* $G = (I, gf)$ *představuje kombinační logické hradlo.*

Příklad 6.1. $I = \{a, b\}$
$gf = (0,0) \to 0, (0,1) \to 1, (1,0) \to 1, (1,1) \to 1$

Příklad 6.1 je formálním zápisem dvouvstupového hradla OR z obrázku 6.1.

Na základě různorodosti relace gf mohou existovat logická hradla s různými logickými funkcemi. Tato hradla je třeba nějakým způsobem rozlišovat. Proto provedeme následující úmluvu jak hradla s různými funkcemi označovat.



Obrázek 6.1: Hradlo OR

Úmluva 6.1. *V textu budou logická hradla zapisována označením jejich funkce velkými písmeny, za kterými může následovat číslo udávající počet jejich vstupů. Označení funkcí je standardní a některé je možné nalézt např v tabulce 2.2 nebo v příloze A.*

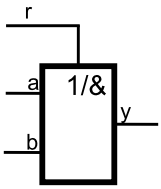
Příklad 6.2. *INV je jednovstupové hradlo s funkcí inverze vstupu.*
AND2 je dvouvstupové hradlo s funkcí AND.
NOR3 je třívstupové hradlo s funkcí NOR.

Nyní již máme definovánu množinu logických úrovní a logická hradla plnící specifické funkce nad touto množinou. Na základě těchto definic můžeme přistoupit k definici hradla multifunkčního, které je stěžejním prvkem navrhované metodiky.

Definice 6.3. *Nechť IF je množina vstupů pro řízení funkce hradla,*
nechť I je množina funkčních vstupů, pro kterou platí $I \cap IF = \emptyset$,
nechť existuje relace $gfm : L^{|I|} \to L$,
nechť $GFM = \{gfm_1, gfm_2, \ldots, gfm_n\}$ je množinou navzájem různých relací gfm,
nechť existuje relace $gfs : L^{|IF|} \to GFM$,
pak $GM = (I, IF, GFM, gfs)$ představuje multifunkční kombinační logické hradlo.

Příklad 6.3. $I = \{a, b\}$
$IF = \{r\}$
$gfm_1 = (0,0) \to 0, (0,1) \to 1, (1,0) \to 1, (1,1) \to 1$
$gfm_2 = (0,0) \to 0, (0,1) \to 0, (1,0) \to 0, (1,1) \to 1$
$GFM = \{gfm_1, gfm_2\}$
$gfs = (0) \to gfm_1, (1) \to gfm_2$

Příklad 6.3 je formálním zápisem multifunkčního hradla z obrázku 6.2, které v jednom režimu plní logickou funkci OR a ve druhém logickou funkci AND. Funkci OR hradlo plní v případě, kdy je na vstupu r hodnota logické 0 a funkci AND v případě, kdy je na vstupu r hodnota logické 1.



Obrázek 6.2: Multifunkční hradlo OR/AND

Stejně jako u logického hradla z definice 6.2 potřebujeme nějakým způsobem odlišovat multifunkční hradla s různými logickými funkcemi. Provedeme proto následující úmluvu.

Úmluva 6.2. *V textu budou multifunkční logická hradla zapisována jako funkce v jednotlivých režimech oddělené lomítkem. Za označením jednotlivých funkcí nebo na konci může být uvedeno číslo udávající počet jejich vstupů.*

Příklad 6.4. *INV/BUFF je jednovstupové hradlo s funkcí inverze v prvním a opakováním vstupu v druhém režimu.*

AND2/OR2 je dvouvstupové hradlo s funkcí AND v prvním a funkcí OR ve druhém režimu.

NAND/NOR/XOR2 je dvouvstupové hradlo s funkcí NAND v prvním, funkcí NOR ve druhém a funkcí XOR ve třetím režimu.

Nyní přistoupíme k definici celého logického obvodu. Ten bude definován jako orientovaný graf rozšířený o relaci přiřazující jeho vrcholům logická hradla. Prakticky jsou tak vrcholům přiřazovány logické funkce, které dané vrcholy plní. Vstupy jsou reprezentovány jako logická hradla bez vstupu s jedním výstupem a výstupy jsou reprezentovány logickými hradly s jedním vstupem a žádným výstupem. Definice předpokládá znalosti z teorie grafů, které je možné nalézt například zde [12, 2].

Definice 6.4. *Nechť Γ je množina všech dostupných logických hradel,*

nechť I je množina vstupů obvodu, O množina výstupů obvodu a G množina vnitřních prvků obvodu, které jsou po dvou disjunktní,

nechť $V = I \cup O \cup G$ je množina vrcholů obvodu,

nechť E je množina orientovaných hran, tedy uspořádaných párů z množiny V,

nechť existuje relace $\varepsilon : V \to \Gamma$,

pak $C = (I, O, G, E, \varepsilon)$ představuje hradly tvořený obvod.

Aby C bylo platnou reprezentací obvodu, musí být splněny následující podmínky:

- $\forall i \in I : deg^+(i) = 0$.
- $\forall o \in O : deg^-(o) = 0; deg^+(o) = 1$.
- $\forall g \in G : deg^+(g) = |I|$, kde I je množina vstupů hradla $\varepsilon(g)$.

$deg^+(v) = |\{u : (u, v) \in E\}|$ určuje počet vrcholů, ze kterých do v vede hrana a $deg^-(v) = |\{u : (v, u) \in E\}|$ určuje počet vrcholů, do kterých z v vede hrana. Nadále budeme uvažovat pouze platná C.

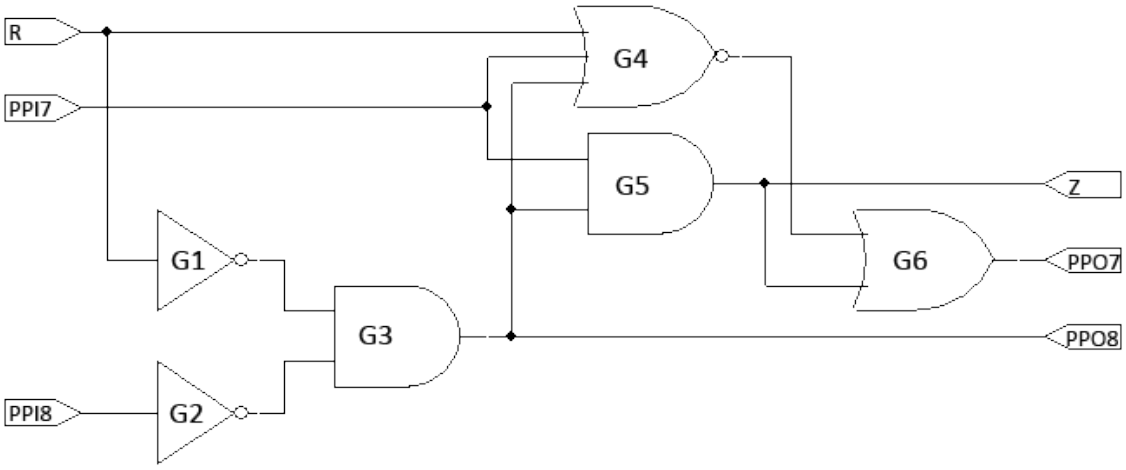
Příklad 6.5. $\Gamma = \{In, Out, INV, AND2, OR2, NOR3\}$
$I = \{R, PPI7, PPI8\}$
$O = \{Z, PPO7, PPO8\}$
$G = \{G1, G2, G3, G4, G5, G6\}$
$E = \{(R, G1), (R, G4), (PPI7, G4), (PPI7, G5), (PPI8, G2),$
$(G1, G3), (G2, G3), (G3, PPO8), (G3, G4), (G3, G5), (G4, G6), (G5, Z), (G5, G6), (G6, PPO7)\}$
$\varepsilon = R \to In, PPI7 \to In, PPI8 \to In, Z \to Out, PPO8 \to Out, PPO7 \to Out, G1 \to INV, G2 \to INV, G3 \to AND2, G4 \to NOR3, G5 \to AND2, G6 \to OR2$

Příklad 6.5 je formálním zápisem obvodu na obrázku 6.3 převzatého z [8].

Výsledná metodika bude také potřebovat pracovat s množinou logických obvodů. Ta bude definována následovně.

Definice 6.5. *Nechť $\Omega = \{C_1, C_2, \ldots, C_i\}$ je množina obvodů C, kde $\forall C \in \Omega : C$ je obvod dle definice 6.4,*

potom Ω je množinou obvodů tvořených logickými hradly.



Obrázek 6.3: Kombinační obvod
[8]

6.1.2 Formální definice principu metody

Nyní, když již máme formálně definovány všechny potřebné prerekvizity, můžeme přistoupit k formální definici principu samotné metody. Jako první je třeba vyřešit vzájemnou nahraditelnost jednotlivých logických hradel. To vyřešíme tak, že jednotlivá logická hradla rozdělíme do množin, kde v každé množině jsou pouze ta hradla, která jsou vzájemně nahraditelná. Z hlediska metody jsou vzájemně nahraditelná ta hradla, která mají stejný počet vstupů.

Definice 6.6. *Nechť existuje relace* $\eta : \Gamma \to \mathbb{N}$*, pro kterou platí* $\forall g \in \Gamma : \eta(g) = |I| \wedge$ I *je množina vstupů hradla* g*,*
nechť Δ *je množina hradel, kde platí* $\forall j, k \in \Delta : \eta(j) = \eta(k)$*,*
potom buď množina Δ *nazývána množinou vzájemně nahraditelných hradel.*

Příklad 6.6. $\Delta_1 = \{AND3, OR3\}$
$\Delta_2 = \{AND2, NOR2, NAND2\}$
$\Delta_3 = \{NAND2, OR2\}$

Příklad 6.6 ukazuje množiny vzájemně nahraditelných hradel. Z příkladu je zřejmé, že jedno logické hradlo může patřit do různých skupin nahraditelných hradel. Jelikož toto není pro navrhovanou metodu vhodné, provedeme následující omezující definici.

Definice 6.7. *Nechť* $\Lambda = \{\Delta_1, \Delta_2, \ldots, \Delta_i\}$ *je množina množin* Δ*, kde* $i \in \Theta$ *je indexová množina a kde platí* $\forall j, k \in \Theta \wedge j \neq k : \Delta_j \cap \Delta_k = \emptyset$ *a současně* $\bigcup_{i \in \Theta} \Delta_i = \Gamma$*,*
potom množina Λ *je množinou disujnktních množin vzájemně nahraditelných hradel.*

Množiny z příkladu 6.6 nemohou náležet do jedné množiny Λ, neboť $\Delta_2 \cap \Delta_3 = NAND2$. V příkladu 6.7 jsou však již množiny nahraditelných hradel navzájemn disjunktní a mohou do množiny Λ náležet.

Příklad 6.7. $\Gamma = \{AND3, OR3, AND2, NOR2, NAND2, OR2\}$
$\Delta_1 = \{AND3, OR3\}$

$\Delta_2 = \{AND2, NOR2\}$
$\Delta_3 = \{NAND2, OR2\}$
$\Lambda = \{\Delta_1, \Delta_2, \Delta_3\}$

Na příkladu 6.7 je také vidět, že může existovat více množin nahraditelných hradel v Λ, ve kterých jsou hradla se stejným počtem vstupů a jsou tedy potenciálně vzájemně nahraditelná. U tohoto konkrétního případu se jedná o množiny Δ_2 a Δ_3. Pokud lze obecně říci, že všechna hradla se stejným počtem vstupů jsou navzájem nahraditelná, je možné pro množinu Λ stanovit omezení, kdy pro každé $\Delta_j, \Delta_k \in \Lambda$, kde $j \neq k$ a současně pro jakékoliv $g \in \Delta_j$ a $h \in \Delta_k$ platí $\eta(g) \neq \eta(h)$. Tato vlastnost však není pro navrhovanou metodu stěžejní a je ponechána jako volitelná při nasazení u konkrétního problému.

Jako další provedeme definici relace pro získání množiny vzájemně nahraditelných hradel k jakémukoliv hradlu. To je klíčové pro následující definice a rovnice.

Definice 6.8. *Nechť existuje relace $\sigma : \Gamma \to \Lambda$ pro kterou platí $\forall g \in \Gamma : \sigma(g) = \Delta \wedge g \in \Delta$, potom je σ funkcí pro získání množiny nahraditelných hradel pro každé hradlo $g \in \Gamma$.*

Příklad 6.8. $\sigma(AND3) = \Delta_1$
$\sigma(OR2) = \Delta_3$

V příkladu 6.8 jsou uvažovány množiny Δ_i uvedené v příkladu 6.7.

Na základě relace σ jsme nyní schopni jednoduše definovat jednu z nejdůležitější vlastnosti metodiky. Ta říká, že vnitřní struktura logického obvodu je při optimalizaci neměnná a mění se pouze funkce logických hradel. Tuto vlastnost definujeme pomocí relace φ, která vrcholům grafu reprezentujícího změněný obvod přiřazuje pouze taková logická hradla, která jsou ze stejné množiny nahraditelných hradel jako hradla na stejném vrcholu v obvodu výchozím. Vezmeme-li tak existující obvod reprezentovaný grafem dle výše uvedených definic a nahradíme-li jeho relaci ε za relaci splňující definici relace φ, tak je jisté, že mohlo dojít pouze ke změnám funkce logických hradel a nikoliv ke změně struktury obvodu.

Definice 6.9. *Nechť existuje relace $\varphi : V \to \Gamma$ pro kterou platí $\forall v \in V : \sigma(\varphi(v)) = \sigma(\varepsilon(v))$, pak $C' = (I, O, G, E, \varphi)$ představuje strukturně shodný obvod s obvodem $C = (I, O, G, E, \varepsilon)$, který se může lišit pouze funkcí vzájemně nahraditelných hradel.*

Příklad 6.9. *Mějme obvod $C = (I, O, G, E, \varepsilon)$ definovaný dle příkladu 6.5 a $\{OR2, AND2\} \in \Lambda$. Pak obvod $C' = (I, O, G, E, \varphi)$, kde*
$\varphi = R \to In, PPI7 \to In, PPI8 \to In, Z \to Out, PPO8 \to Out, PPO7 \to Out, G1 \to INV, G2 \to INV, G3 \to AND2, G4 \to NOR3, G5 \to AND2, G6 \to AND2$
splňuje definici 6.9 a liší se pouze funkcí hradla G6.

Počet všech možných obvodů C' strukturně shodných s obvodem C dle definice 6.9 je dán logickými hradly použitými v obvodě C a množinami vzájemně nahraditelných hradel. Celkový počet je možné vyjádřit vztahem 6.1.

$$\prod_{v \in V} |\sigma(\varepsilon(v))| \tag{6.1}$$

Definice 6.9 je dostatečná pro přístupy, které nepředpokládají přepínání funkce hradel v průběhu testu. Úkolem optimalizační metody je nalézt φ takové, aby obvod C' dosahoval lepších sledovaných vlastností, než původní obvod C. Vrcholy grafu obvodu, jimž relace φ

přiřazuje jiná logická hradla (logické funkce) než relace ε, jsou ve výsledném obvodě implementovány jako multifunkční hradla se spojeným řízením funkce vyvedeným jako jediný nový vstup obvodu. Pokud však potřebujeme přepínat hradla i v průběnu testu, potřebujeme formálně popsat změny v obvodě vzhledem ke vzniku nových vstupů a hran v grafu obvodu. To řeší následující definice.

Definice 6.10. *Nechť Γ^m je množina všech dostupných multifunkčních hradel,*
nechť Γ je množina všech dostupných hradel, pro kterou platí $\Gamma \cap \Gamma^m = \emptyset$,
nechť $\Gamma' = \Gamma \cup \Gamma^m$ je množina všech dostupných hradel,
nechť I^m je množina vstupů obvodu pro řízení multifunkčních hradel,
nechť I je množina vstupů obvodu, pro kterou platí $I \cap I^m = \emptyset$,
nechť $I' = I \cup I^m$ je množina vstupů obvodu,
nechť $V' = I' \cup O \cup G$ je množina vrcholů obvodu,
nechť existuje relace $\vartheta : V' \to \Gamma'$, kde

$$\forall v \in V \begin{cases} \vartheta(v) = \varepsilon(v) & \text{pro } \vartheta(v) \in \Gamma \\ \exists x \forall w \in N^+(v) \wedge w \in I^m \forall u \in N^-(w) : gfs(x) = gf & \text{pro } \vartheta(v) \in \Gamma^m \end{cases}, \text{ kde } gfs$$

je relací dle definice 6.3 pro hradla $\vartheta(u)$ a gf je relací dle definice 6.2 pro hradlo $\varepsilon(v)$,
nechť E^m je množina orientovaných hran, tedy uspořádaných párů z množiny V', kde $\forall (u,w) \in E^m : u \in I^m \wedge \vartheta(w) \in \Gamma^m$,
nechť E je množina orientovaných hran, tedy uspořádaných párů z množiny V', kde $\forall (u,w) \in E : u \notin I^m$,
nechť $E' = E \cup E^m$ je množina orientovaných hran, tedy uspořádaných párů z množiny V',
potom obvod $C' = (I', O, G, E', \vartheta)$ je obvodem $C = (I, O, G, E, \varepsilon)$, ve kterém mohla být některá hradla nahrazena za multifunkční, jejichž řízení bylo vyvedeno na nové vstupy I^m a současně existuje kombinace na vstupech I^m taková, že všechna multifunkční hradla obvodu C' plní stejnou funkci jako hradla na stejných vrcholech v obvodě C.

$N^+(v) = \{u : (u,v) \in E'\}$ je okolí v, ze kterého do vrcholu v vedou hrany a $N^-(v) = \{u : (v,u) \in E'\}$ je okolí v do kterého z v vedou hrany.

Příklad 6.10. $\Gamma^m = \{OR2/AND2\}$
$I^m = \{TST\}$
$\vartheta = R \to In, PPI7 \to In, PPI8 \to In, Z \to Out, PPO8 \to Out, PPO7 \to Out, G1 \to INV, G2 \to INV, G3 \to AND2, G4 \to NOR3, G5 \to AND2, G6 \to OR2/AND2,$
$E^m = \{(TST, OR2/AND2)\}$
$C' = (I', O, G, E', \vartheta)$
Ostatní množiny a relace jsou shodné dle příkladu 6.5 a definice 6.10.

V příkladu 6.10 je obvod C' lišící se od obvodu $C = (I, O, G, E, \varepsilon)$ z příkladu 6.5 pouze změnou hradla $G6$ na multifunkční s vyvedením jeho vstupu na nový vstup obvodu TST. Budeme-li uvažovat multifunkční logické hradlo OR2/AND2 dle příkladu 6.3, pak pokud je na vstupu obvodu TST hodnota logická 0, pak má hradlo $G6$ funkci OR2 a obvod C' plní přes vstupy I a výstupy O stejnou logickou funkci jako obvod C.

Pro následující kapitoly bude ještě potřeba definovat okolí obvodu C.

Definice 6.11. *Nechť $C = (I, O, G, E, \varepsilon)$ je obvod dle definice 6.4,*
nechť $C' = (I, O, G, E, \varphi)$ je obvod dle definice 6.9,
nechť existuje množina $\Upsilon_i = \{C' : |\varphi \cap \varepsilon| = |\varphi| - i\}$,
potom je množina Υ_i množinou i-tého okolí obvodu C.

Příklad 6.11. *Mějme obvod C definovaný dle příkladu 6.5 a obvod C' definovaný dle příkladu 6.9,*
potom vzhledem k obvodu C platí $C' \in \Upsilon_1$.

Definice 6.11 stanovuje množiny Υ_i jako i-tá okolí obvodu C. Do množiny Υ_i patří ty obvody, které se od C liší právě i hradly. V příkladě 6.11 je pak znázorněno, že obvody C a C' se od sebe liší právě jedním hradlem, proto C' náleží do množiny Υ_1.

Definice 6.12. *Nechť existuje množina $\Upsilon_{ci} = \{C' : |\varphi \cap \varepsilon| = |\varphi| - i\}$ obsahující všechna možná C' i-tého okolí,*
potom je množina Υ_{ci} kompletní množinou i-tého okolí obvodu C.

Definice 6.12 stanovuje podobně jako definice 6.11 okolí obvodu C, avšak Υ_{ci} musí obsahovat všechny obvody C', které se v i-tém okolí mohou nacházet. Vezmeme-li například 1-okolí, pak platí $\Upsilon_{c1} = \{C' : |\varphi \cap \varepsilon| = |\varphi| - 1 \wedge (\forall v \in V : (\forall g \in \sigma(\varepsilon(v)) \wedge g \neq \varepsilon(v) : \varphi = \varepsilon \backslash \{(v, \varepsilon(v))\} \cup \{(v, g)\}))\}$. Současně lze také stanovit velikost množiny $|\Upsilon_{c1}| = \sum_{v \in V} (|\sigma(\varepsilon(v))| - 1)$. Podobné formulace lze vyjádřit i pro další množiny Υ_{ci}. Pro tuto práci je však 1-okolí dostačující.

6.2 Optimalizace testovatelnosti

Jak bylo řečeno v kapitole 2.2.7, existuje více pohledů na analýzu testovatelnosti obvodu. Nadále budu uvažovat testovatelnost založenou na analýze řiditelnosti a pozorovatelnosti bodů obvodu diskutovanou v odkazované kapitole.

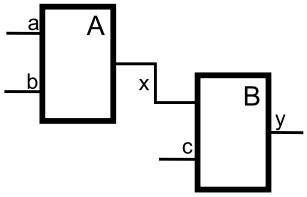
Optimalizace testovatelnosti pomocí multifunkčních prvků je založena na principu vkládání testovacích bodů (viz kapitola 2.2.8) pro zlepšení řiditelnosti nebo pozorovatelnosti vybraného místa. Základní odlišností však je, že navrhovaná metodika nepřidává nová hradla do obvodu, ale pouze modifikuje hradla existující na hradla multifunkční. Výběr hradla, jeho funkce v testovacím režimu a způsob jeho řízení je nutné realizovat na základě požadavků na zlepšení řiditelnosti.

6.2.1 Přímé řízení

Klasická implementace přímého řízení určitého bodu obvodu metodou vkládání testovacích bodů spočívá v přidání logického hradla před tento určený bod. Funkce vloženého hradla závisí na tom, zdali je potřeba řídit jen jednu konkrétní logickou úroveň a jakou nebo je potřeba mít nad místem úplnou kontrolu (viz kapitola 2.2.8). Vložení nového hradla má za následek změnu dynamických vlastností v okolí nového hradla, případně změnu dynamických vlastností i celého obvodu.

Navrhovaná metodika však nové hradlo do obvodu nepřidává. Požadavek na přímé řízení je možné vyřešit náhradou hradla (jehož výstup vede do určeného bodu) za hradlo multifunkční. Nyní předpokládejme situaci na obrázku 6.4 a požadavek na přímé řízení logické úrovně bodu x. Pak náhradou hradla A za multifunkční hradlo můžeme logickou úroveň v bodě x řídit.

Volba testovací funkce multifunkčního hradla závisí na požadavku, jakým způsobem je potřeba kogickou úroveň v bodě x řídit. Obecně však vždy musí platit, že ať už je na vstupu hradla jakákoliv vstupní kombinace, na výstupu musíme být schopni nastavit požadovanou logickou úroveň. V případě, kdy potřebujeme řídit pouze jednu logickou úroveň $k \in L$ a



Obrázek 6.4: Obvod s požadavkem na přímé řízení logické úrovně bodu x

nahrazujeme hradlo $G = (I, gf)$ multifunkčním hradlem $G' = \{I', IF, GFM, gfs\}$, musí být hradlo G' definováno následovně:

- $I' = I$, tzn. nové hradlo má stejné vstupy jako hradlo původní.
- $IF = \{TST\}$, kde TST je vstup, kterým bude funkce hradla ovládána.
- $GFM = \{gfm_1, gfm_2\}$, tzn. nové hradlo dovede plnit dvě funkce.
- $gfm_1 = gf$, tzn. první funkce nového hradla je shodná s funkcí hradla původního.
- $gfm_2 \supseteq \{(l_1, l_2, \ldots, l_n, k) | n = |I| \wedge (l_1, l_2, \ldots, l_n, j) \in gf \wedge j \neq k\}$, tzn. druhá funkce nového hradla má na výstupu hodnotu k pro všechny vstupní kombinace, pro které původní funkce hradla (a tedy i funkce gfm_1) není rovna k.
- $gfs = \{(u, gfm_1), (v, gfm_2)\}$, kde $u, v \in L \wedge u \neq v$.

Nejjednodušším případem je situace, kdy funkce v režimu test multifunkčního hradla je trvalá k. Při požadavku nastavení logické k na bodu x, pak je možné pouze přepnout hradlo do testovacího režimu a požadovaná hodnota je zajištěna.

Z hlediska výše uvedených požadavků však funkce trvalá k není nutná. Důležité je pouze to, aby požadovaná úroveň byla nastavitelná při jakékoliv vstupní kombinaci. Pro jakoukoliv vstupní kombinaci tedy stačí, aby byla na výstupu úroveň k ve funkčním nebo testovacím režimu multifunkčního hradla. V případě požadavku na hodnotu k v bodě x se pak musí multifunkční hradlo přepnout do toho režimu, ve kterém je na výstupu hradla hodnota k pro aktuální vstupní kombinaci.

Příklad 6.12. *Mějme obvod z obrázku 6.4, ve kterém požadujeme řídit logickou 0 v bodě x a hradlo A plní funkci AND. Potom hradlo A nahradíme za multifunkční, kde v testovacím režimu bude plnit jednu z funkcí x_i z tabulky 6.1.*

a	b	AND	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
0	0	0	0	1	0	1	0	1	0	1
0	1	0	0	0	1	1	0	0	1	1
1	0	0	0	0	0	0	1	1	1	1
1	1	1	0	0	0	0	0	0	0	0

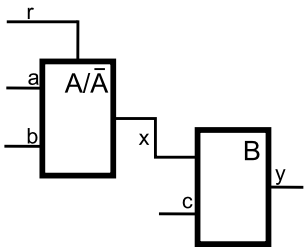
Tabulka 6.1: Možné funkce testovacího režimu hradla G' příkladu 6.12

V případě požadavku na úplné řízení bodu x, je situace obdobná s předchozí. Budeme-li uvažovat předchozí situaci, pak jediným rozdílem je definice relace gfm_2. Ta musí být definována následovně:

- $\forall(l_1, l_2, \ldots, l_n, k) \in gf : (l_1, l_2, \ldots, l_n, \neg k) \in gfm_2$.

Kde relace $\neg : L \to L$ je definována jako $\neg = 0 \to 1, 1 \to 0$. Funkce v testovacím režimu tedy musí být komplementem funkce v režimu funkčním. Pak je možné při zachování funkce ve funkčním režimu nastavit v bodě x požadovanou logickou úroveň k při jakékoliv vstupní kombinaci. Samotné nastavení požadované logické úrovně pak spočívá ve změně funkce hradla, pokud je pro aktuální vstupní kombinaci na výstupu jiná logická úroveň než požadovaná. Bod x je tak úplně řiditelný.

Příklad 6.13. *Mějme obvod z obrázku 6.4, ve kterém požadujeme úplně řídit bod x a hradlo A plní funkci AND. Potom hradlo A nahradíme za multifunkční, kde v testovacím režimu musí plnit funkci x_8 z tabulky 6.1, tedy funkci NAND. Výsledný obvod je zobrazen na obrázku 6.5. Tabulka 6.2 ukazuje do jakého režimu musí být přepnuto multifunkční hradlo (tedy jaká musí být na vstupu r logická úroveň) v případě $gfs = \{(0, gfm_1), (1, gfm_2)\}$, pokud požadujeme na výstupu logickou 0 (sloupec r_0 a x_0) nebo logickou 1 (sloupec r_1 a x_1)*



Obrázek 6.5: Úplné řízení multifunkčním logickým hradlem

a	b	AND	NAND	r_0	x_0	r_1	x_1
0	0	0	1	0	0	1	1
0	1	0	1	0	0	1	1
1	0	0	1	0	0	1	1
1	1	1	0	1	0	0	1

Tabulka 6.2: Úplné řízení bodu x dle příkladu 6.13

6.2.2 Zlepšení testovatelnosti dle SCOAP

Dalším řešeným problémem může být požadavek na obecné zlepšení testovatelnosti v obvodě. Přidržíme-li se metody SCOAP, je navrhované řešení založeno na snížení hodnoty řiditelnosti vybraných míst obvodu. Díky matematickému popisu řiditelnosti na výstupu hradel lze jednoduše identifikovat hradla, která jsou a která nejsou pro řiditelnost vhodná.

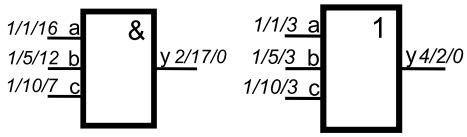
Nejlépe se řídí výstup hradel, jejichž rovnice řiditelnosti je rovna minimu vstupních řiditelností (rovnice 2.4). U těchto hradel je nárust hodnoty řiditelnosti nejnižší. Naopak nejhůře se řídí výstup hradel, jejichž rovnice řiditelnosti je rovna součtu vstupních řiditelností (rovnice 2.5). U těchto hradel je nárůst hodnoty nejvyšší.

Z rovnic pro řiditelnost výstupu hradla uvedených v kapitole 2.2.7 dále plyne, že není běžně užívané dvou a více vstupové hradlo, které by mělo dobré vlastnosti řiditelnosti pro

obě logické úrovně. Vždy je na výstupu hradla dobře řiditelná logická 0 nebo logická 1. Nikdy však ne obě hodnoty. Je proto třeba uvažovat pro jakou hodnotu logické úrovně se optimalizace provádí a jaké důsledky změna hradel přinese pro opačnou logickou úroveň.

Úmluva 6.3. *Řiditelnost bodu obvodu budeme zapisovat jako x/y, kde x je řiditelnost logické 0 a y logické 1. Je-li uvedena trojice x/y/z, pak x a y jsou řiditelnosti tak jak byly popsány a z je pozorovatelnost.*

Jako příklad mějme třívstupové hradlo AND na obrázku 6.6a. Obrázek předpokládá, že je hradlo součástí většího obvodu a tak nemá všechny vstupní řiditelnosti rovné 1. Pokud bychom chtěli optimalizovat řiditelnost logické 1 na výstupu, která je 17, můžeme hradlo nahradit hradlem OR, jehož řiditelnost na výstupu je rovna minimální hodnotě řiditelnosti na vstupu +1. Výsledek je zobrazen na obrázku 6.6b. Z obrázku je patrné, že hodnota řiditelnosti logické 1 na výstupu se snížila z hodnoty 17 na hodnotu 2. Současně však došlo k nárůstu hodnoty řiditelnosti logické 0 z hodnoty 2 na 4. Oproti razantnímu snížení hodnoty řiditelnosti pro logickou 1 je však tento nárůst minoritní. Současně s razantním snížením hodnoty řiditelnosti logické 1 došlo také ke snížení hodnoty pozorovatelnosti na vstupech hradla.



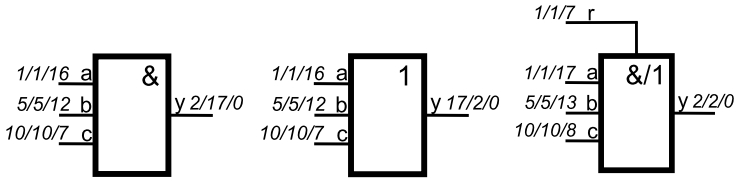
(a) AND (b) OR

Obrázek 6.6: Třívstupové hradlo AND a OR ohodnocené metodou SCOAP

Příklad z obrázku 6.6 je případem, který je pro prezentovanou metodu nejvhodnější. Tedy kdy hodnota řiditelností na vstupech hradla je pro jednu logickou úroveň velmi malá, pro druhou velká, a hradlo má na výstupu hodnotu řiditelnosti rovnu součtu řiditelnosti logických úrovní s velkou hodnotou řiditelnosti na vstupu. Optimalizace pak spočívá ve změně funkce hradla tak, aby z vysokých hodnot řiditelnosti na vstupu byla řiditelnost na výstupu rovna minimu, a z nízkých hodnot řiditelnosti na vstupu byla řiditelnost na výstupu rovna součtu. Tato situace však vždy nenastává.

Nyní mějme opět třívstupové hradlo AND, avšak ohodnocené dle obrázku 6.7a. Vstupní řiditelnost pro logickou 0 je nyní shodná jako pro logickou 1. Výstupní řiditelnost zůstala zachována. Změnou hradla za OR (obrázek 6.7b) dojde opět ke snížení hodnoty řiditelnosti logické 1 na výstupu ze 17 na 2, ale současně dojde k nárůstu hodnoty řiditelnosti pro logickou 0 z 2 na 17. Situace špatné hodnoty řiditelnosti se tedy prohodila a optimalizací řiditelnosti logické 1 jsme výrazně zhoršili řiditelnost logické 0. Pokud však umožníme změnu funkce hradla i v průběhu testu, pak je možné funkce hradel měnit tak, aby měly dobré vlastnosti řiditelnosti pro právě řízenou logickou úroveň. Tento princip je zobrazen na obrázku 6.7c. Výsledná hodnota řiditelnosti na výstupu hradla je pak pro logickou 0 i logickou 1 shodně 2.

Předochzí statě ukázaly, jak optimalizovat testovatelnost v případě, kdy máme vybrané místo obvodu pro optimalizaci. Pokud optimalizujeme celý obvod, můžeme také řešit problém, jak tato místa vybírat. Z hlediska optimalizace dle navrhované metody je důležité určit ta hradla a jejich testovací funkce, aby došlo k co největšímu zlepšení celkové řiditelnosti a pozorovatelnosti obvodu. K největšímu zlepšení dojde tehdy, pokud dojde k maximálnímu možnému snížení hodnot řiditelnosti a pozorovatelnosti.



(a) AND (b) OR (c) AND/OR

Obrázek 6.7: Třívstupové hradlo AND, OR a AND/OR ohodnocené metodou SCOAP

Navrhovaná metoda určení hradla pro náhradu pracuje v několika krocích s postupným ohodnocováním jednotlivých spojů obvodu. Každý spoj je mimo standardních SCOAP hodnot ohodnocen ještě následující čtveřicí:

- Nejlepší dosažitelná kombinační řiditelnost 0 označená $CCB0(l)$[1].
- Nejlepší dosažitelná kombinační řiditelnost 1 označená $CCB1(l)$.
- Rozdíl kombinační řiditelnosti 0 označená $CCD0(l)$[2].
- Rozdíl kombinační řiditelnosti 1 označená $CCD1(l)$.

Ukazatel nejlepší dosažitelné hodnoty řiditelnosti *CCB* je dán nejnižší hodnotou, která je v aktuální situaci dosažitelná nejvhodnějším hradlem. Tento ukazatel je možné zjistit například metodou hrubé síly, kdy se v dané situaci vyzkouší každé dostupné hradlo a použije se nejnižší dosažená hodnota. Nejsou vyloučeny sofistikovanější metody, ty ale nebudou v práci diskutovány.

Jakmile je získán ukazatel *CCB*, je možné zjistit ukazatel *CCD* jednoduchým výpočtem dle rovnice 6.2 a 6.3.

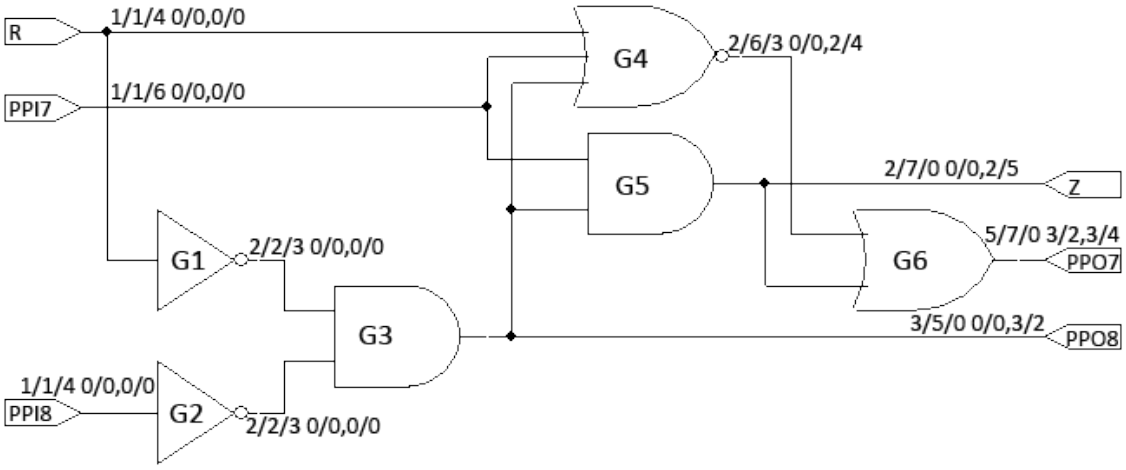
$$CCD0 = CC0 - CCB0 \tag{6.2}$$

$$CCD1 = CC1 - CCB1 \tag{6.3}$$

Po ohodnocení všech spojů obvodu ukazateli *CCB* a *CCD* se vybere ten spoj, který má nejvyšší hodnotu *CCD* nezávisle na tom, zdali se jedná o *CCD0* nebo *CCD1*. Tento spoj je vhodné optimalizovat a hradlo, které má výstup na tento spoj, se nahradí za multifunkční. Jeho funkce v testovacím režimu je pak taková, pomocí které bylo *CCB* dosaženo.

Pokud se po náhradě hradla za multifunkční pokračuje v hledání dalších míst pro optimalizaci, je potřeba provést celý algoritmus od začátku včetně SCOAP ohodnocení obvodu. Současně pokud nejsou povolena multifunkční hradla s více jak dvěmi funkcemi, musí být v následujících iteracích algoritmu vyloučena již nahrazená hradla z výběru maximální hodnoty *CCD*.

Jako příklad mějme obvod na obrázku 6.8 převzatý z [8]. Každý spoj v obvodě je ohodnocen pomocí SCOAP metody a také hodnotami *CCB* a *CCD*. Z ohodnocení je zřejmé, že nejvyšší hodnotou *CCD* je $CCD1 = 5$ na výstupu *Z*. Aktuální hodnota $CC1 = 7$, přičemž nejnižší možná $CCB1 = 2$. Výstup Z je tvořen výstupem hradla *G5*. Změnou funkce hradla *G5* z AND na funkci například OR dojde k dosažení $CC1 = CCB1 = 2$. Výsledná úprava obvodu je zobrazena na obrázku 6.9.



Obrázek 6.8: Kombinační obvod s ohodnocením spojů ve tvaru CC0/CC1/CO CCB0/CCD0,CCB1/CCD1

Pokud provedeme další iteraci metody nad upraveným obvodem, pak pro náhradu bude vytipováno hradlo G4, které na výstupu má nejvyšší hodnotu $CCD1 = 4$. Hodnotu $CC1 = CB1 = 2$ získáme změnou funkce hradla z NOR na funkci například NAND. Výsledná úprava obvodu a jeho ohodnocení je na obrázku 6.10.

Porovnáme-li výchozí a konečný obvod (po dvou iteracích metody), použitím dvou multifunkčních hradel došlo ke snížení hodnot řiditelnosti na třech spojích celkem o hodnotu 13, dále došlo ke snížení hodnoty pozorovatelnosti na jednom spoji o 1 a vznikl jeden nový vstup "Test"s hodnotou testovatelnosti *1/1/3*.

6.3 Optimalizace testu

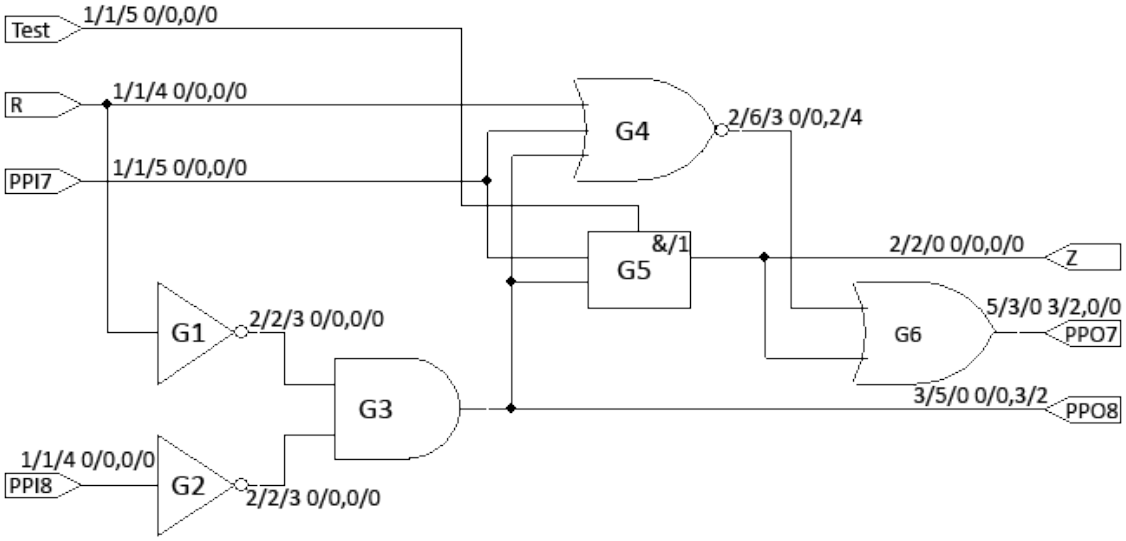
Princip optimalizace testu multifunkčními prvky staví na předpokladech uvedených v kapitole 4. Pokud platí, pak lze vhodnými změnami vnitřních prvků obvodu dosáhnout požadované optimalizace výsledného testu. Požadovanou optimalizací mohou být různé aspekty testu, jako například počet testovacích vektorů, potřebný příkon pro test, pokrytí poruch atp. Ačkoliv je navrhovaná metodika obecně použitelná na více problémů, bude použita zejména na snížení počtu testovacích vektorů.

6.3.1 Princip metody

Jak bylo uvedeno v kapitole 6.1, je třeba řešit tři hlavní problémy. Výběr hradel, výběr jejich testovací funkce a způsob jejich řízení. Navržená metoda volí takový způsob řízení, kdy funkci hradel není možné v průběhu testu měnit. Úloha metody tak sestává pouze z identifikace hradel, která by bylo vhodné změnit, a volby jejich funkce.

Vstupem metody je obvod C popsaný na úrovni hradel, u něhož byl vznesen požadavek na optimalizaci jeho testu. Výstupem metody je strukturně stejný obvod C', u něhož ale došlo ke změně funkce některých jeho hradel, a který dosahuje lepších sledovaných vlastností. Jedná se tak o hledání obvodu C' dle definice 6.9. Změněná hradla jsou ve výsledném

[1]CCB - Combinational Controllability Best
[2]CCD - Combinational Controllability Difference



Obrázek 6.9: Kombinační obvod s ohodnocením spojů po první iteraci optimalizace

obvodě reprezentována jako multifunkční, kdy ve funkčním režimu mají funkci jako hradla původního obvodu a v testovacím režimu mají funkci jako hradla z obvodu optimalizovaného. Pro optimalizaci obvodu C a získání obvodu C' s lepšími sledovanými vlastnostmi využívá metoda kombinatorické optimalizace φ.

6.3.2 Účelová funkce a omezující podmínky

Účelová funkce a omezující podmínky musí co nejlépe vystihovat požadovanou optimalizaci. Je tedy třeba určit jaké vlastnosti obvodu se optimalizují, který z nich má jakou důležitost a určit omezující podmínky kandidátních řešení.

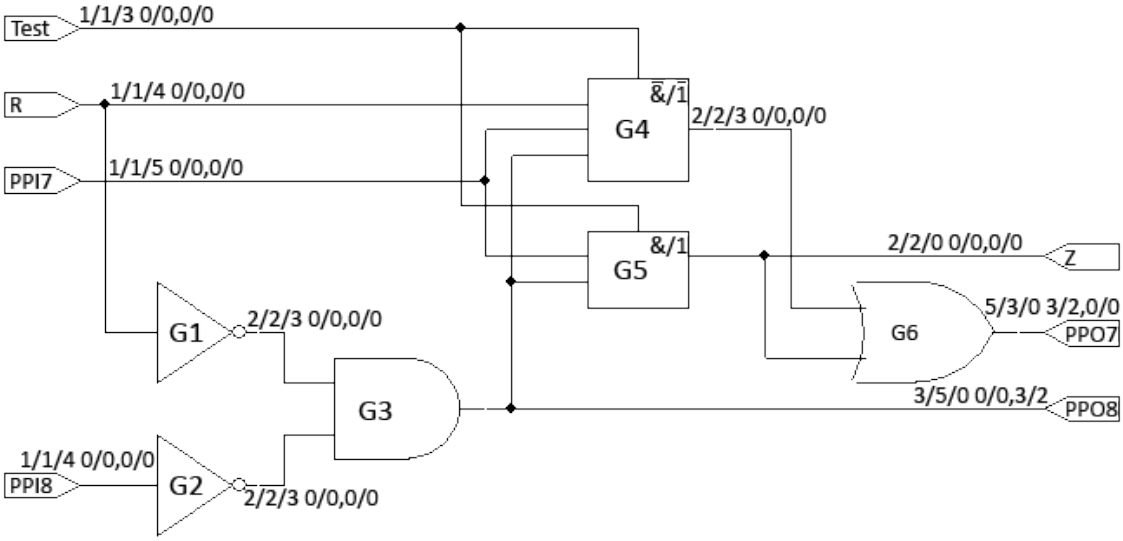
Při řešení optimalizace počtu testovacích vektorů jsem využil parametry:

- Počet testovacích vektorů. Definujme jej jako relaci $vc : \Omega \to \mathbb{N}$[3].
- Pokrytí redukovaných poruch. Definujme jej jako relaci $fc : \Omega \to \mathbb{R}$[4].
- Počet multifunkčních hradel. Definujme jej jako relaci $mc : \Omega \to \mathbb{N}$[5].
- Počet CMOS tranzistorů při implementaci obvodu. Definujme jej jako relaci $tc : \Omega \to \mathbb{N}$[6].

Jelikož je řešení postaveno na hledání C' pro obvod C dle definice 6.9, není nutné zabývat se problémem strukturní konzistence obvodu. Každé řešení nalezené dle této definice je platným obvodem.

Dle definice z kapitoly 2.3 ohodnocuje účelová funkce kandidátní řešení hodnotou z $\mathbb{R}$. Při implementaci jsem však přistoupil k přímé variantě porovnání dvou obvodů a rozhodnutí, jestli je novější obvod lepší nebo nikoliv.

[3] vc - Vectors Count
[4] fc - Fault Coverage
[5] mc - Multifunctional logic gates Count
[6] tc - Transistors Count



Obrázek 6.10: Kombinační obvod s ohodnocením spojů po druhé iteraci optimalizace

Definice 6.13. *Nechť existuje relace* $fb : (\Omega \times \Omega) \rightarrow \{false, true\}$, *potom buď* fb *nazývána porovnávací funkcí.*

Samotná porovnávací funkce popsaná algoritmem 6.1 vrací *true*, pokud má obvod C' lepší parametry, než obvod C. Obvod má lepší parametry, pokud se sníží počet testovacích vektorů a nedojde ke snížení pokrytí poruch. Je-li počet testovacích vektorů shodný, pak je lepší tehdy, dojde-li ke zvýšení pokrytí poruch. Jestliže je i tento parametr shodný, pak je lepší, obsahuje-li méně multifunkčních hradel. Jsou-li i počty multifunkčních hradel shodné, je lepší, pokud potřebuje méně CMOS tranzistorů pro implementaci.

```
procedure provnejObvody(obvod C, obvod Cn) {
  if vc(Cn) > vc(C) OR fc(Cn) < fc(C) {
    return false;
  }

  if vc(Cn) < vc(C) OR fc(Cn) > fc(C) OR mc(Cn) < fc(C) {
    return true;
  }

  if tc(Cn) < tc(C) AND mc(Cn) <= fc(C) {
    return true;
  }

  return false;
}
```

Algoritmus 6.1: Porovnávací funkce

Klasická implementace účelové funkce

V případě potřeby účelové funkce implementované dle kapitoly 2.3 jako relace $f : \Omega \to \mathbb{R}$, lze implementaci provést například dle rovnice 6.4, kde vcw, fcw, mcw a tcw jsou váhy jednotlivých hodnot. Čím nižší hodnoty funkce nabývá, tím je řešení kvalitnější. Optimalizace tedy spočívá v minimalizaci této funkce.

$$f(C) = vc(C) \cdot vcw + (100 - fc(C)) \cdot fcw + mc(C) \cdot mcw + tc(C) \cdot tcw \tag{6.4}$$

Případný požadavek na nepřekročení meze některé z vlastností je možné implementovat jako omezující podmínku optimalizace. V prezentovaném případě by takovou omezující podmínkou byl požadavek, aby nedošlo ke snížení pokrytí poruch. Každé kandidátní řešení by tak muselo splnit podmínku $fc(C') \geq fc(C)$, kde C' je kandidátní řešení a C je výchozí optimalizovaný obvod.

Jelikož podobné funkce nebylo využito, nebude nadále tato problematika diskutována.

6.3.3 Optimalizační algoritmy

V předchozích kapitolách byl definován model obvodu a porovnávací funkce jako náhrada funkce účelové. Nad těmito prvky lze spouštět optimalizační algoritmy pro hledání nejlepších možných řešení. V rámci práce byly vyzkoušeny tři metody, z nichž dvě úspěšně nacházely kvalitní řešení. Relativně neúspěšnou metodou bylo náhodné hledání, jejíž algoritmus je popsaný v kapitole 2.3.2 a nebude dále diskutována.

Kompletní prohledání

Prvním úspěšně použitým algoritmem bylo kompletní prohledání stavového prostoru řešení. Počet kandidátních řešení problému je vyjádřen rovnicí 6.1. Zjednodušeně lze počet aproximovat exponencíální rovnicí c^g, kde c je průměrná velikost množiny vzájemně nahraditelných hradel a g je počet hradel obvodu. Jelikož složitost kompletního prohledávání roste úměrně složitosti problému, je složitost nalezení řešení tímto algoritmem exponenciální. Z toho lze usuzovat, že tento algoritmus bude použitelný pouze na relativně malé obvody. Samotná implementace pak vychází z popisu v kapitole 2.3.1.

Rekurzivní algoritmus

Jako druhý byl použit optimalizační algoritmus popsaný algoritmem 6.2, který je založený na prohledávání do hloubky a byl inspirovaný horolezeckým algoritmem a zpětným prohledáváním. Vychází z obvodu pro optimalizaci a hledá v jeho celém 1-okolí lepší řešení. Pomocí metody "ziskejDalsiReseni", která pro obvod C postupně vrací všechny obvody C' z Υ_{c1}, postupně zkoumá obvody v okolí a platí-li $provnejObvody(C, C') = true$, pak se nad obvodem C' spustí algoritmus rekurzivně. Po vynoření z rekurze pokračuje dalším obvodem z okolí. Každé spuštění si uchovává nejlepší řešení a vzájemným porovnáním nejlepších řešení jednotlivých spuštění je nalezen nejlepší obvod.

Z algoritmu je zřejmé, že se z výchozího obvodu snaží "vyšplhat"všemi možnými směry po lepších obvodech v okolí. Jelikož hledá v 1-okolí, je algoritmus schopen nalézt pouze ta řešení, ke kterým z výchozího obvodu vede cesta přes řešení, kdy v každém kroku došlo ke změně pouze jednoho hradla a každé následující řešení je lepší než předchozí. Algoritmus tak není schopen nalézt řešení, která jsou mimo tyto cesty a uvázne v lokálních extrémech

```
procedure dhc(obvod C) {
  nejlepsireseni = C;
  Cn = ziskejDalsiReseni(C);
  while Cn != NULL do {
    if provnejObvody(C, Cn) {
      reseni = dhc(Cn);
      if porovnejObvody(nejlepsireseni, reseni) {
        nejlepsireseni = reseni;
      }
    }
    Cn = ziskejDalsiReseni(C, Cn);
  }
  return nejlepsireseni;
}
```

Algoritmus 6.2: Optimalizační algoritmus inspirovaný horolezeckým algoritmem a zpětným prohledáváním

v okolí výchozího řešení. Výhodou algoritmu je, že velmi rychle dovede zjistit, zdali se v 1-okolí nalézá lepší řešení a pokud ano, dovede k němu poměrně rychle konvergovat. Algoritmus je tak svojí jednoduchostí a rychlostí konvergence vhodný pro potvrzení vyslovených předpokladů.

6.3.4 Implementace metodiky

Aby bylo možné implementovat nástroje pro ověření metodiky, bylo třeba zvolit formáty dat, se kterými budou nástroje pracovat. Dále bylo potřeba vyřešit jakým způsobem získat parametry obvodu potřebné pro hodnotící funkci a implementovat samotné optimalizační algoritmy.

Formát popisu obvodu

Jako vstupní a výstupní formát popisu obvodu byl zvolen strukturální Verilog na úrovni hradel. Díky využití standardního formátu je možné spolupracovat s jinými standardními nástroji.

Jako knihovna prvků (hradel) obvodu byla použita DfT/ATPG ADK (Asic Design Kit) verze 1.6 využívaná v nástrojích od Mentor Graphics. Při experimentech však byly používány i obvody obsahující hradla, která v této knihovně chybí. Vytvořil jsem tak novou knihovnu prvků sestávající z prvků standardní ADK knihovny rozšířenou o prvky popsané v příloze C tabulce C.2.

Využité nástroje a parametry obvodu

Některé použité obvody nebyly ve formátu Verilog na úrovni hradel pro použitou knihovnu prvků. Pro jejich převod byl využit nástroj Leonardo Spectrum.

Pro zjištění počtu testovacích vektorů a pokrytí poruch obvodu byl využit ATPG nástroj FlexText od Mentor Graphics. Počet multifunkčních hradel obvodu je roven počtu

rozdílných hradel mezi výchozím a ohodnocovaným obvodem. Počet CMOS tranzistorů pro implementaci byl stanoven dle standardní implementace knihovny AMI 0.7 μm viz příloha C. U změněných hradel obvodu se nepočítal počet tranzistorů pro implementaci multifunkčního hradla, ale pouze pro implementaci hradla s aktuálně použitou funkcí. Toto zjednodušení je založeno na předpokladu, že složitost implementace multifunkčního hradla s danou funkcí bude přímo úměrná složitosti implementace hradla pouze s touto jedinou funkcí. Do jaké míry je tento zjednodušující předpoklad platný, je závislé jak na funkcích hradla, tak i na implementační technologii.

Optimalizační nástroj

Samotný optimalizační nástroj nazvaný "Testgen"byl naprogramován v jazyce Perl. Jeho vstupem je obvod popsaný pomocí strukturálního Verilogu a výstupem je stejný obvod ve stejném formátu, který má ale některá hradla změněna. Změněná hradla by pak při reálné implementaci obvodu byla vytvořena jako multifunkční, jejichž vstup pro řízení funkce by byl spojen a vyveden jako jediný vstup obvodu.

Samotný nástroj sestává ze tří hlavních komponent. První komponeta se stará o práci s obvodem. Dovede načíst i zapsat obvod ve formátu Verilog a poskytovat o načteném obvodě potřebné informace, jako jsou například hradla obvodu, spojení, vstupy, výstupy atp. Druhá komponenta se stará o spolupraci s nástrojem pro zjištění vlastností testu obvodu. Využit byl FlexTest od Mentor Graphics, pomocí kterého byly pro obvod získávány počty testovacích vektorů a pokrytí poruch. Třetí a poslední hlavní komponenta byl samotný optimalizátor, který na základě optimalizačního algoritmu hledal řešení problému.

Kapitola 7

Výsledky

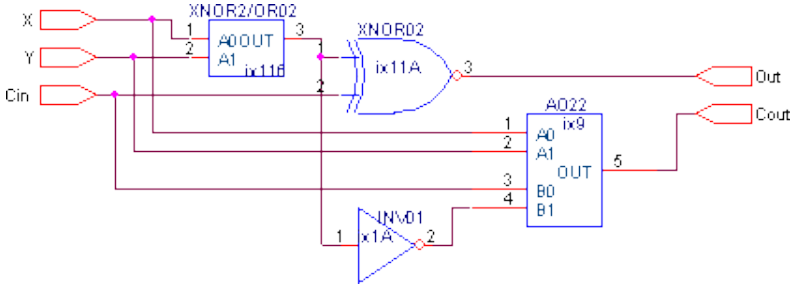
Implementace algoritmů prezentovaných v kapitole 6.3 byla využita pro získání rozsáhlých experimentálních výsledků. První experimenty využívaly jednoduché obvody, které byly optimalizovány pomocí kompletního prohledávání. Následně byla vyvinuta metoda prohledávání do hloubky popsaná v kapitole 6.3.3, pomocí které se optimalizovaly složitější obvody včetně obvodů ze sady ISCAS 85 popsaných v kapitole 2.2.6.

Hradla byla rozdělena do množin vzájemně nahraditelných hradel Δ_i dle tabulky 7.1. Hradla, která se v obvodě objevila a nebyla v této tabulce, nebyla algoritmem nahrazována a tedy chovala se stejně jako by tvořila samostatnou množinu Δ_i obsahující jeden prvek.

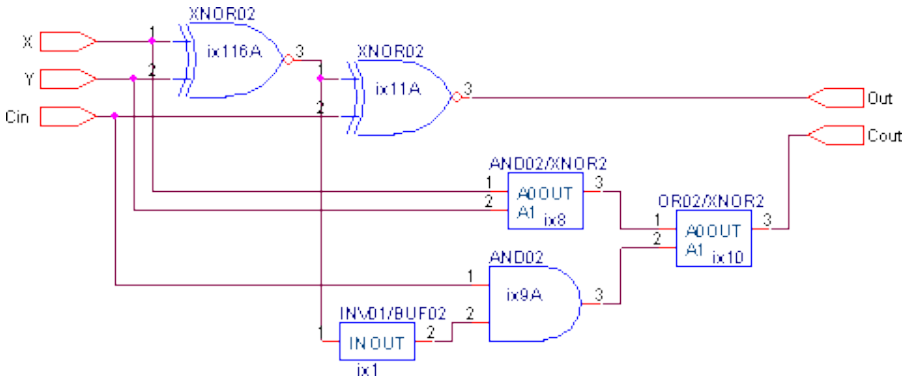
Δ_i	Hradla	Δ_i	Hradla
Δ_1	buf02, inv01	Δ_{14}	and05, nand05, or05
Δ_2	buf04, inv02	Δ_{15}	aoi222, oai222
Δ_3	buf08, inv04	Δ_{16}	aoi321, oai321
Δ_4	buf12, inv08	Δ_{17}	aoi33, oai33
Δ_5	buf16, inv12	Δ_{18}	aoi322, oai322
Δ_6	and02, nand02, nor02, or02, xor2, xnor2	Δ_{19}	aoi43, oai43
Δ_7	and03, nand03, nor03, or03	Δ_{20}	aoi332, oai332
Δ_8	mux21	Δ_{21}	aoi44, oai44
Δ_9	ao21, aoi21, oai21	Δ_{22}	and08, nand08, nor08
Δ_{10}	and04, nand04, nor04, or04	Δ_{23}	aoi333, oai333
Δ_{11}	ao22, aoi22, oai22	Δ_{24}	and09
Δ_{12}	ao221, aoi221, oai221	Δ_{25}	hadd1
Δ_{13}	ao32, aoi32, oai32	Δ_{26}	fadd1

Tabulka 7.1: Množiny vzájemně nahraditelných hradel

V jednotlivých výsledcích se také objevují odhady počtu CMOS transistorů potřebných pro implementaci obvodu pomocí multifunkčních hradel. Tento odhad je dán součtem počtu tranzistorů potřebných pro implementaci jednotlivých funkcí multifunkčního hradla dle přílohy C. Počet CMOS tranzistorů pro implementaci například hradla nand02/nor02 tak vychází $4 + 4 = 8$.



Obrázek 7.1: Obvod fulladd1



Obrázek 7.2: Obvod fulladd2

7.1 Komletní prohledání

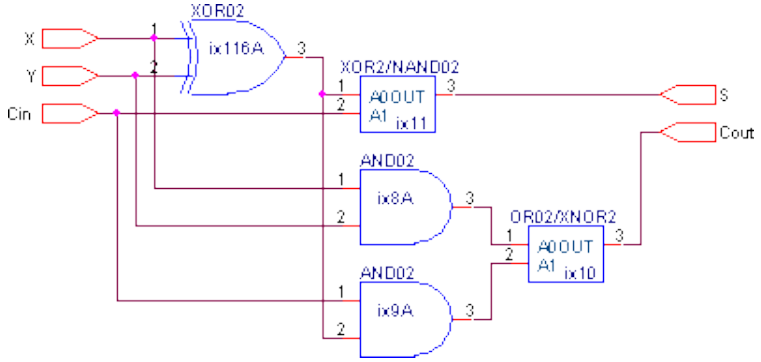
Optimalizační metoda kompletního prohledávání byla využita pro nalezení řešení u šesti malých obvodů popsaných tabulkou 7.2. Souhrnné výsledky jsou zaznamenány v tabulce 7.3, jednotlivé obvody jsou zobrazeny na obrázcích 7.1 až 7.6 a testovací vektory jednotlivých obvodů jsou v tabulkách 7.4.

Název	Popis	Hradel	Tranzistorů	Vstupů	Výstupů
fulladd1	jednobitová úplná sčítačka	4	32	3	2
fulladd2	jednobitová úplná sčítačka	6	40	3	2
fulladd3	jednobitová úplná sčítačka	5	42	3	2
comp3bit	3bit komparátor	11	62	6	1
enc8to3	enkodér 8 na 3	13	74	8	3
dec3to8	dekodér 3 na 8	11	56	3	8

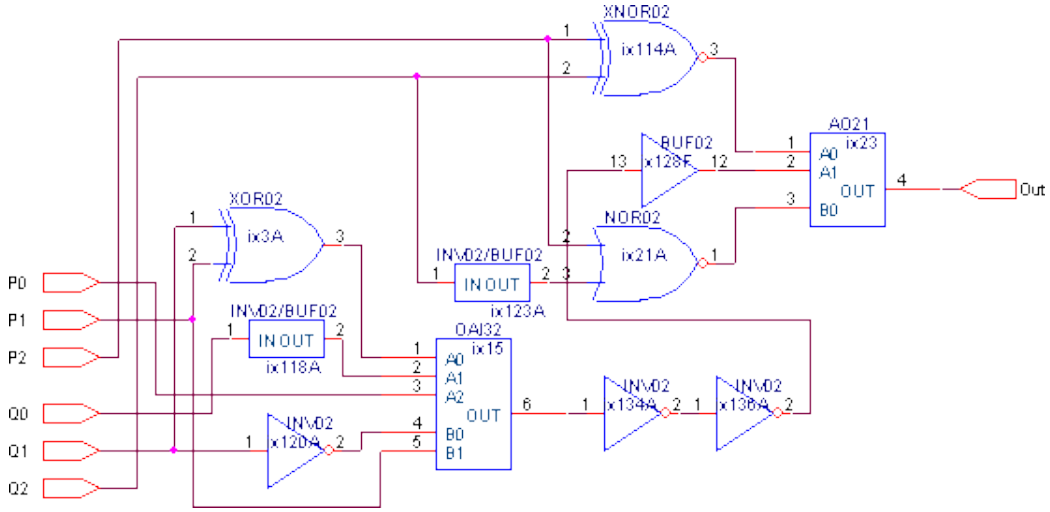
Tabulka 7.2: Jednoduché obvody pro metodu kompletního prohledávání

Jelikož pokrytí poruch výchozích obvodů je 100 % a optimalizace byla omezena tak, aby nemohlo dojít ke snížení pokrytí, jsou výsledky zajímavé pouze z hlediska snížení počtu testovacích vektorů. U všech obvodů došlo ke snížení počtu testovacích vektorů o 9,09 % až 50 %. Z hlediska nárůstu "ceny"obvodu vyjádřené odhadem počtu CMOS tranzistorů potřebných pro implementaci obvodu došlo k nárůstu o 13 % až 60 %.

K největšímu snížení potřebných testovacích vektorů došlo u obvodu dec3to8. Snížení činilo 50 % při odhadovaném nárůstu počtu CMOS tranzistorů na implementaci o 42,85 %. U tohoto obvodu je zajímavé nalezené řešení, kdy po překlopení multifunkčních hradel do



Obrázek 7.3: Obvod fulladd3



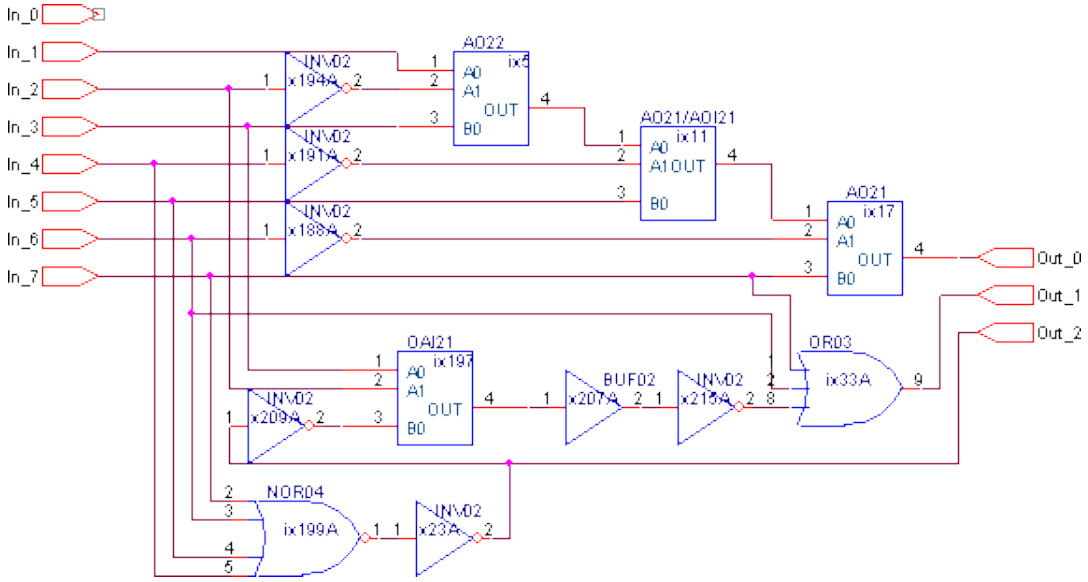
Obrázek 7.4: Obvod comp3bit

testovacího režimu má obvod na všech výstupech vždy stejnou hodnotu a chová se jako hradlo NOR se třemi vstupy. Jelikož testem pro samostatné hradlo NOR se třemi vstupy jsou otestovány všechny spoje i v obvodu dec3to8, je test tohoto obvodu shodný s testem samostatného NOR hradla.

Protože metoda kompletního prohledávání vyzkouší všechna kandidátní řešení, je možné zjistit všechna řešení s lepšími parametry než má výchozí obvod a určit jejich kvalitu. Dle kvality je pak možné tato řešení rozdělit do skupin a stanovit jejich počty. Na základě těchto dat lze vyvozovat další předpoklady ohledně složitějších obvodů. Souhrnná data o počtu kandidátních řešení a řešení lepších, než bylo dosaženo ve výchozím obvodě, je možné nalézt v tabulce 7.5.

Z dat metody byly také pro každý obvody sestaveny kontingenční tabulky s údaji o počtu nalezených řešení vzhledem k počtu testovacích vektorů a počtu změněných hradel. Ty jsou zobrazeny v souhrnné tabulce 7.6. Pro každý obvod byly vytvořeny dvě kontingenční tabulky, kdy v horní jsou uvedena absolutní čísla o počtu řešení a ve spodní počet řešení v procentech vzhledem ke všem kandidátním řešením.

Z těchto kontingenčních tabulek lze vyvodit trend ohledně počtu řešení vzhledem k jejich kvalitě. Z hlediska počtu testovacích vektorů je zřejmé, že čím je větší redukce počtu testovacích vektorů, tím je počet řešení menší. Z hlediska počtu změněných hradel v obvodě



Obrázek 7.5: Obvod enc8to3

data ukazují, že nejvíce řešení se nachází v případě změny přibližně poloviny hradel obvodu. Jelikož jsou tato data získána na malé skupině malých obvodů, nelze rozhodnout, jestli se dá obecně říci, že se nejvíce řešení nachází právě v oblastech při změně poloviny hradel obvodu a nebo při změně menšího či většího počtu hradel.

Obecně však lze říci, že čím jsou řešení kvalitnější, tím je jich menší počet a těch nejlepších je vždy minimum. Lze předpokládat, že tyto trendy budou shodné i pro větší obvody a nalezení nejlepšího řešení bude s rostoucí složitostí obvodu náročnější. Nicméně i řešení, která nejsou nejlepší, mohou být dostatečně kvalitní a není tak nutné nacházet pouze ta nejlepší.

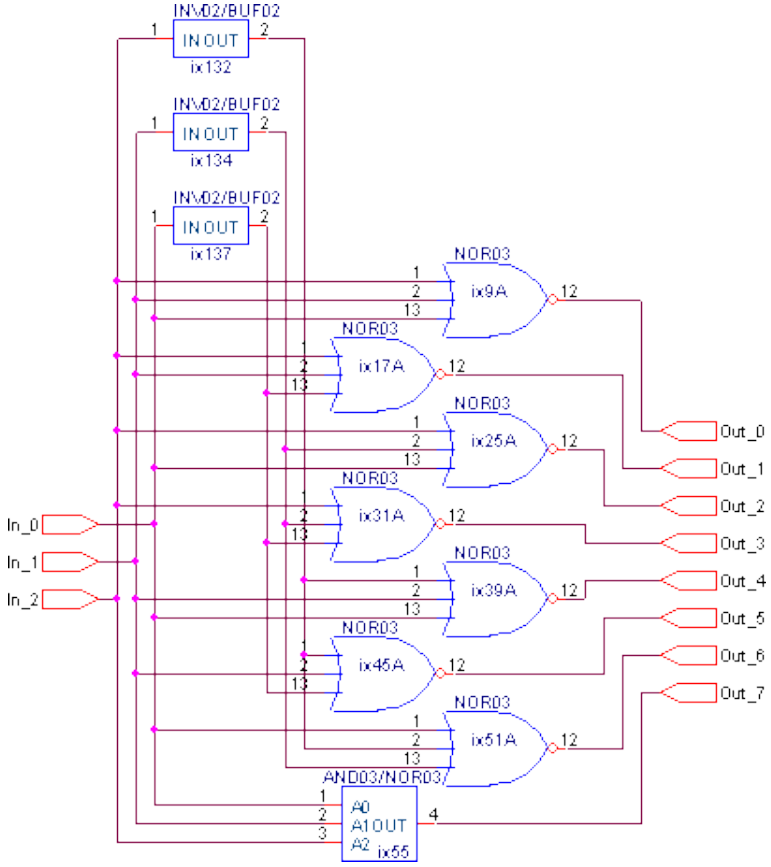
7.2 Rekurzivní algoritmus

Po získání výsledků a jejich analýze z kompletního prohledávání byla implementována optimalizační metoda na principu rekurzivního algoritmu popsaná v kapitole 6.3.3. Ta byla použita pro optimalizaci testu složitějších obvodů ze sady ISCAS 85 popsaných v kapitole 2.2.6 a dalších obvodů popsaných v tabulce 7.7. Výsledky jsou zaznamenány v tabulce 7.8.

Algoritmus doběhl do konce (tedy vyzkoušel všechna jemu dostupná řešení) u všech obvodů z tabulky 7.7 vyjma obvodu mul8 a pouze u obvodu c17 a c1355 ze sady ISCAS 85. U ostatních byl ukončen předčasně, neboť nedoběhl v požadovaném čase. Později bylo zjištěno, že konkrétní implementace nástroje "Testgen"v jazyce Perl nebyla optimální a výkon metody je možné značně optimalizovat. Podrobnosti jsou diskutovány v kapitole 7.4.2.

I když algoritmus u některých obvodů nedoběhl do konce a tedy pravděpodobně nenašel nejlepší řešení, které je schopen nalézt, došlo ke snížení počtu testovacích vektorů u všech obvodů vyjma obvodu c1355. Snížení počtu testovacích vektorů se pohybuje mezi 12,67 % až 58,21 % při odhadovaném nárůstu počtu CMOS tranzistorů pro implementaci o 0,73 % až 75 %.

U sady ISCAS 85 došlo průměrně ke snížení počtu testovacích vektorů o 27,98 % při změně 2,79 % funkcí hradel při zvýšení pokrytí poruch o 0,51 % a nárůstu odhadova-



Obrázek 7.6: Obvod dec3to8

ného počtu CMOS tranzistorů pro implementaci o 5,29 %. U ostatních obvodů pak došlo průměrně ke snížení počtu testovacích vektorů o 26,73 % při změně 7,41 % funkcí hradel při zvýšení pokrytí poruch o 0,22 % a nárůstu odhadovaného počtu CMOS tranzistorů o 7,91 %. Celkově pro všechny obvody došlo průměrně ke snížení počtu testovacích vektorů o 27,83 % při změně 2,98 % hradel při zvýšení pokrytí poruch o 0,39 % a nárůstu odhadovaného počtu tranzistorů o 5,45 %,

Největší snížení počtu testovacích vektorů se podařilo dosáhnout u obvodu c499 a to o 58,21 % z původních 67 na 28 při odhadovaném nárůstu počtu CMOS tranzistorů pro implmenetaci o 13,86 % z 1764 na 2062. Zajímavého výsledku bylo také dosaženo například u obvodu c6288, u kterého došlo ke snížení počtu testovacích vektorů o 21,74 % z 46 na 36 při odhadovaném nárůstu počtu CMOS tranzistorů pro implementaci o pouhých 0,73 % z 10112 na 10186 při změně funkce deseti hradel.

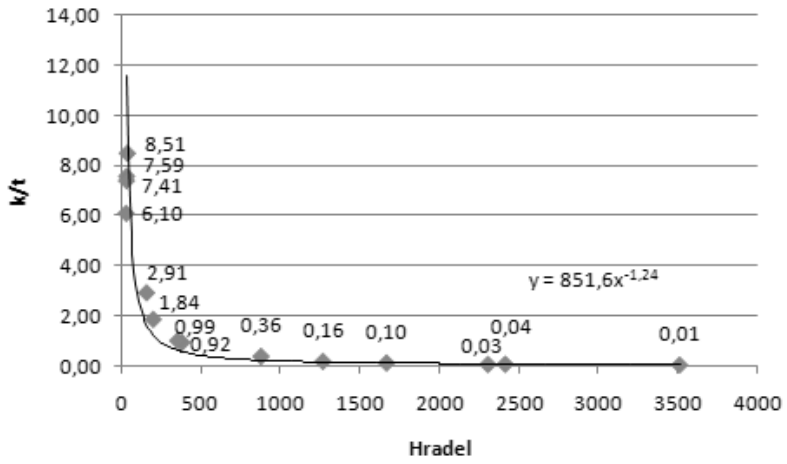
U některých obvodů došlo také ke zlepšení pokrytí poruch, které se pohybuje mezi 0,16 % a 1,76 %. Zajímavým výsledkem je například obvod comp8, u kterého došlo ke zvýšení pokrytí poruch o 1,76 % z 98,27 % na rovných 100 %.

U obvodu c1355 doběhl algoritmus až do konce, ale nenalezl žádné řešení. Pro tento obvod však řešení nalezla evoluční metoda diskutovaná v kapitole 7.3. Je tedy zřejmé, že se tento obvod nachází v lokálním extrému, u kterého neexistuje obvod se změnou jednoho hradla, který by naplňoval stanovená kritéria lépe než obvod výchozí. Metoda tak uvázla v lokálním extrému přímo na výchozím obvodě a nebyla schopná nalézt žádné další řešení.

V tabulce výsledků 7.8 jsou také uvedeny údaje o čase a kroku algoritmu, ve kterém

bylo řešení nalezeno. V každém kroku algoritmu je vytvořen modifikovaný obvod, ten je ohodnocen a porovnán s obvodem předchozím. V každém kroku tak dochází k právě jednomu spuštění nástroje FlexTest. Z informací o čase a počtu kroků je vypočten průměrný počet kroků za sekundu, který je uveden ve sloupci k/t. Závislost velikosti obvodu vyjádřenou počtem hradel a hodnoty k/t je vynesena v grafu na obrázku 7.7. Do dat grafu byly zahrnuty pouze ty obvody, jejichž řešení bylo nalezeno později než v kroku 1 000. Z dat, grafu a rovnice trendu $851,6x^{-1,24}$ je zřejmé, že s velikostí obvodu prudce klesá počet evaluací, které je nástroj schopen provést za sekundu.

Výpočty byly prováděny na počítači s dvěma procesory Dual Core AMD Opteron 2220. Jelikož nástroj Testgen není implementován vícevláknově, algoritmus běžel vždy pouze v jednom procesoru na jednom jádře.



Obrázek 7.7: Graf počtu kroků za sekundu vzhledem k počtu hradel obvodu

7.3 Evoluční algoritmus

Pro prezentovanou metodiku popsanou v kapitole 6.3 byl Ing. Jiřím Šimáčkem pod vedením Prof. Ing. Lukáše Sekaniny Ph.D. vytvořen také optimalizační algoritmus založený na evolučním principu, který byl prezentovaný v [22]. Algoritmus využívá účelovou funkci popsanou rovnicí 7.1, kde jednotlivé relace a symboly znamenají relace a symboly popsané v kapitole 6 a G je množina vnitřních prvků (hradel) obvodu C.

$$f(C, C') = fcw \cdot (1 - fc(C')) + vcw \cdot \frac{vc(C')}{vc(C)} + mcw \cdot \frac{mc(C')}{|G|} \tag{7.1}$$

Pro získání výsledků byly použity váhy $fcw = 1000$, $vcw = 100$ a $mcw = 10$, pravděpodobnost mutace byla $p_{mut} = 0,005$ a využito bylo 1000 populací, kdy v každé populaci bylo 1000 jedinců. Pro získání výsledků si Ing. Šimáček napsal vlastní nástroj v jazyce Python, který umožňoval vícevláknový běh a byl spouštěn ve čtyřech vláknech. Výpočty byly provedeny na stejném počítači jako rekurzivní algoritmus prezentovaný v kapitole 7.2, který obsahoval dva procesory Dual Core AMD Opteron 2220. Výsledky získané z metody jsou prezentovány v tabulce 7.9.

Hlavní odlišnost výsledků oproti předchozím metodám je v tom, že bylo umožněno snížení hodnoty pokrytí poruch. To poskytlo evolučnímu algoritmu větší prostor v nalezených řešeních a snížilo riziko uváznutí v lokálním extrému. Budeme-li porovnávat pouze

výsledky pro sadu ISCAS 85, došlo průměrně ke snížení pokrytí poruch o 1,96 %, zatímco u rekurzivního algoritmu došlo ke zvýšení pokrytí o 0.51 %. Ohledně počtu testovacích vektorů dokázal evoluční algoritmus nalézt řešení s průměrnou úsporou 48,22 % při změně 6,37 % hradel, zatímco algoritmus rekurzivní nalezl řešení s průměrnou úsporou 27,98 % při změně 2,79 % hradel. Evoluční algoritmus tak nachází lepší řešení v počtu testovacích vektorů a horší vzhledem k pokrytí poruch při vyšším nárůstu počtu změněných hradel. Velkou výhodou evolučního algoritmu se zdá být jeho rychlost. To však s aktuálními daty nelze přesně rozhodnout, neboť rychlost samotného nástroje Ing. Šimáčka oproti mému nástroji Testgen je několikanásobně vyšší, nehledě na použitou optimalizační metodu (viz rozbor v kapitole 7.4.2).

7.4 Zhodnocení výsledků

7.4.1 Nalezená řešení

Implementované optimalizační metody nalezly řešení, která potřebují méně testovacích vektorů při zachování nebo mírném zlepšení pokrytí poruch. Algoritmus kompletního prohledávání dokázal na všech použitých malých obvodech nalézt řešení. Metoda prohledávání do hloubky nalezla řešení u všech obvodů vyjma obvodu c1355 ze sady ISCAS 85, pro který ale nalezla řešení evoluční metoda implementovaná Ing. Šimáčkem. Na problému obvodu c1355 se ukázalo, že problém není čistě konvexní a i u této metodiky je potřeba navrhovat optimalizační algoritmy, které počítají s nekonvexním průběhem optimalizovaného problému a dovedou unikat z lokálních extrémů.

Z výsledků je také patrné, že v prohledávaném prostoru existuje mnoho rozličných řešení, která dosahují různých parametrů. Úpravou účelové funkce nebo nastavením omezujících podmínek je možné získávat řešení, která přesně vyhovují stanoveným požadavkům.

7.4.2 Výkonnostní hledisko

Vedle kvality nalezených řešení je důležitým parametrem optimalizačních metod také rychlost, s jakou řešení dovedou nalézt. Budeme-li zjednodušeně předpokládat, že samotné algoritmy optimalizačních metod prezentovaných v této práci mají přibližně stejnou výpočetní náročnost, je důležitý jediný parametr, a to počet evaluací kandidátních řešení, tedy kolik kandidátních řešení musí optimalizační metoda prozkoumat, než rozhodne o nejlepším nalezeném řešení. Čím méně kandidátních řešení musí algoritmus prozkoumat, tím rychleji dovede dodat výsledek. Z tohoto hlediska je nejhorším případem algoritmus kompletního prohledání, který musí prozkoumat všechna kandidátní řešení. Pokud by jiný algoritmus dosahoval ještě horších výsledků, byl by pro danou úlohu zbytečný a mohl by být nahrazen algoritmem kompletního prohledání.

Jakmile je optimalizační metoda implementována do konkrétního nástroje a nasazena na řešení konkrétních problémů, musí se při porovnávání rychlosti a kvality různých optimalizačních metod brát také v potaz další prvky, jako je rychlost počítače, na kterém je nástroj spouštěn, a kvalita samotné implementace nástroje, který optimalizační metodu používá. Nyní vezměme v úvahu nástroj Ing. Šimáčka s evolučním algoritmem z kapitoly 7.3 a nástroj Testgen s výsledky z kapitoly 7.2, jejichž výsledky byly získány na stejném počítači. Výkonostní parametry těchto dvou nástrojů jsou uvedeny v tabulce 7.10. Tato tabulka obsahuje tři skupiny sloupců s údaji pro FlexTest, Testgen a pro nástroj Ing. Šimáčka s evolučním algoritmem. V každé skupině sloupců jsou uvedeny počty kroků, ze kterých

byly údaje získány, doba, jakou tyto kroky trvaly, počet jader procesorů, na kterých výpočty běžely a poměry počtu kroků za sekundu. U nástroje FlexTest byla data získána pro generování testu 1 000 krát po sobě pro výchozí obvod, u nástroje Testgen je počet kroků a čas získán ve chvíli, kdy bylo nalezeno nejlepší řešení, u nástroje Ing. Šimáčka jsou údaje získány z celého běhu nástroje. Sloupec *f/tg* zobrazuje poměr rychlosti, tedy počtu kroků na jedno vlákno, nástroje FlexTest vůči nástroji Testgen, sloupec *e/tg* poměr rychlosti nástroje Ing. Šimáčka k nástroji Testgen a sloupec *f/e* poměr rychlosti FlexTest k nástroji Ing. Šimáčka.

Jako první se můžeme zaměřit na časy nacházení řešení obou nástrojů. Jelikož jsou u nástroje Testgen udávány časy nalezení nejlepšího řešení a u nástroje Ing. Šimáčka časy celkového běhu, nehledě na čas nalezení nejlepšího řešení, není možné tyto algoritmy tímto způsobem přímo srovnávat. U složitějších obvodů ale trvá nalezení nejlepších řešení nástrojem Testgen déle než celý běh nástroje Ing. Šimáčka.

Když se ale oprostíme od porovnávání časů a zaměříme se na počet kandidátních řešení, které jsou schopny nástroje ohodnotit za jednotku času, zjistíme zásadní rozdíl. Nástroj Ing. Šimáčka ohodnotí více řešení v násobcích několika řádů v závislosti na složitosti řešení. Z hlediska výkonu na jedno vlákno dosahuje jeho nástroj 4,87 až 224,86 krát více ohodnocení za sekundu, než nástroj Testgen. Přihlédneme-li k vícevláknovosti a získání výsledků při čtyřech souběžných vláknech, pak nástroj Ing. Šimáčka ohodnocoval 19,48 až 899,44 krát více kandidátních řešení za sekundu, než nástroj Testgen. Z popisu nástroje však není jasné, zdali byl pro každé kandidátní řešení připravované pro novou populaci spouštěn nástroj FlexTest nebo nikoliv. Z obecného hlediska to nutné není, neboť se v nové populaci objevují jedinci z populace minulé, kteří už jsou ohodnoceni a tak není nutné opět spouštět rutiny pro jejich ohodnocení.

Zaměříme-li se ale na poměry *f/tg* a *f/e*, tedy poměr mezi počtem běhů za sekundu nástroje FlexTest a zkoumaných optimalizačních nástrojů, zjistíme, že zatímco poměr u nástroje Ing. Šimáčka je přibližně konstantní a pohybuje se kolem hodnoty 0.35, poměr u nástroje Testgen velmi rychle narůstá. Z toho je možné usuzovat, že kvalita implementace nástroje Testgen v jazyce Perl je po stránce výkonové výrazně horší, nežli implementace Ing. Šimáčka v jazyce Python. Zatímco druhý jmenovaný se složitostí problému škáluje s FlexTest rovnoměrně, nástroj Testgen škáluje velmi špatně a jeho rychlost se složitostí problému výrazně klesá. Kvalitu optimalizačních metod tak nelze přímo porovnávat na základě dosažených časů jednotlivých nástrojů. Zaměřme se nyní například na obvod c3540, kde jeden běh nástroje Testgen trvá 77,71 krát déle než nástroje FlexTest. Pokud by nástroj Testgen dokázal udržet škálování vůči FlexTest na úrovni cca 2 jako u nejjednodušších obvodů, trvalo by nalezení řešení u obvodu c5315 35 krát méně času, než nyní, tedy cca 17 hodin a 48 minut. Pokud by byl výpočet spuštěn ve čtyřech vláknech podobně jako nástroj Ing. Šimáčka, což algoritmus použité optimalizační metody umožňuje, trval by výpočet přibližně 4 hodiny a 27 minut, tedy polovinu času běhu nástroje Ing. Šimáčka.

Důležitým závěrem z tohoto srovnání není samotné porovnání výsledků nebo kvality daných nástrojů, ale poznatek, že kvalitní implementací samotného optimalizačního nástroje je možné dosáhnout o několik řádů vyšších rychlostí než u prezentovaného nástroje Testgen. S kvalitně implementovanými nástroji s optimalizací pro výkon při použití mnohovláknových výpočtů by bylo možné s prezentovanými optimalizačními metodami nacházet kvalitní řešení v akceptovatelném čase pro řádově složitější obvody, než jsou ty ze sady ISCAS 85.

Obvod	Hradel			Vektorů			Pokrytí Poruch			CMOS Tranzistorů				Obr.	tv
	C	C'	%	$vc(C)$	$vc(C')$	%	$fc(C)$	$fc(C')$	%	$tc(C)$	$tc(C')$	P	%		
fulladd1	4	1	25,00	6	5	83,33	100,00	100,00	100,00	32	28	38	118,75	7.1	7.4a
fulladd2	6	3	50,00	6	4	66,67	100,00	100,00	100,00	40	50	64	160,00	7.2	7.4b
fulladd3	5	2	40,00	6	4	66,67	100,00	100,00	100,00	42	38	56	133,33	7.3	7.4c
comp3bit	11	2	18,18	11	9	81,81	100,00	100,00	100,00	62	70	74	119,35	7.4	7.4d
enc8to3	13	1	7,69	11	10	90,91	100,00	100,00	100,00	74	72	84	113,51	7.5	7.4e
dec3to8	11	4	36,36	8	4	50,00	100,00	100,00	100,00	56	66	80	142,85	7.6	7.4f

Hradel C	Počet hradel obvodu
Hradel C'	Počet modifikovaných hradel obvodu
$vc(C)$	Počet testovacích vektorů původního obvodu
$vc(C')$	Počet testovacích vektorů modifikovaného obvodu
Vektorů %	$vc(C')$ v procentech oproti $vc(C)$
$fc(C)$	Pokrytí poruch původního obvodu
$fc(C')$	Pokrytí poruch modifikovaného obvodu
Pokr. poruch %	$fc(C')$ v procentech oproti $fc(C')$
$tc(C)$	Počet CMOS tranzistorů původního obvodu
$tc(C')$	Počet CMOS tranzistorů modifikovaného obvodu
P	Odhadovaný počet CMOS tranzistorů obvodu s multifunkčními hradly
CMOS tran. %	P v procentech oproti $tc(C)$
Obr.	Odkaz na obrázek, na kterém je obvod zobrazen
tv	Odkaz na tabulku s testovacími vektory

Tabulka 7.3: Výsledky metody kompletního prohledávání

	C		C'	
	I	O	I	O
1	100	10	100	01
2	000	00	000	10
3	111	11	111	11
4	110	01	110	10
5	001	10	101	10
6	010	10		

(a) fulladd1

	C		C'	
	I	O	I	O
1	100	10	100	11
2	000	00	000	00
3	111	11	111	11
4	110	01	110	01
5	001	10		
6	010	10		

(b) fulladd2

	C		C'	
	I	O	I	O
1	100	10	100	11
2	000	00	111	10
3	111	11	110	00
4	110	01	001	11
5	001	10		
6	010	10		

(c) fulladd3

	C		C'	
	I	O	I	O
1	100011	0	100011	0
2	000110	1	000110	0
3	001100	1	011000	1
4	011000	0	110110	1
5	000000	0	101100	0
6	011011	0	100111	1
7	110110	0	100100	1
8	000010	1	110111	0
9	100101	1	110100	0
10	010001	0		
11	010011	1		

(d) comp3bit

	C		C'	
	I	O	I	O
1	10001100	111	10001100	111
2	00011000	100	00011000	101
3	00110000	101	001100000	100
4	01100000	110	01100000	110
5	11000000	111	00000000	001
6	00000000	000	00100000	100
7	00100000	101	01000000	110
8	01000000	110	00001001	010
9	00001001	011	00000111	011
10	00000111	010	00000010	000
11	00000010	001		

(e) enc8to3

	C		C'	
	I	O	I	O
1	100	00010000	100	00000000
2	000	00000001	000	11111111
3	111	10000000	010	00000000
4	110	01000000	001	00000000
5	011	00001000		
6	101	00100000		
7	010	00000100		
8	001	00000010		

(f) dec3to8

C Původní obvod
C' Modifikovaný obvod
I Testovací vektory na vstupu
O Odezvy na výstupu

Tabulka 7.4: Testovací vektory obvodů

Název	Kandidátních řešení	Lepších řešení	%
fulladd1	216	18	8,33
fulladd2	15552	2634	16,94
fulladd3	7776	1322	17,00
comp3bit	124416	1736	1,40
enc8to3	165888	32	0,02
dec3to8	524288	10752	2,05

Tabulka 7.5: Počty řešení

v\h	1	2	3	4	
5	1	3	7	7	18
	1	3	7	7	18

v\h	1	2	3	4	[%]
5	0,46	1,39	3,24	3,24	8,33
	0,46	1,39	3,24	3,24	8,33

(a) fulladd1

v\h	1	2	3	4	5	6	7	
10	1	2	5	8	7	6	3	32
	1	2	5	8	7	6	3	32

v\h	1	2	3	4	5	6	7	[%]
10	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,02
	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,02

(b) enc8to3

v\h	1	2	3	4	5	6	
4	0	0	8	60	118	90	276
5	2	33	210	618	845	650	2358
	2	33	218	678	963	740	2634

v\h	1	2	3	4	5	6	[%]
4	0,00	0,00	0,05	0,39	0,76	0,58	1,77
5	0,01	0,21	1,35	3,97	5,43	4,18	15,16
	0,01	0,21	1,40	4,36	6,19	4,76	16,94

(c) fulladd2

v\h	1	2	3	4	5	
4	0	4	34	50	36	124
5	4	36	178	531	449	1198
	4	40	212	581	485	1322

v\h	1	2	3	4	5	[%]
4	0,00	0,05	0,44	0,64	0,46	1,59
5	0,05	0,46	2,29	6,83	5,77	15,41
	0,05	0,51	2,73	7,47	6,24	17,00

(d) fulladd3

v\h	1	2	3	4	5	6	7	8	9	10	
9	0	1	4	22	70	119	115	67	23	3	424
10	1	6	24	97	200	289	323	230	120	22	1312
	1	7	28	119	270	408	438	297	143	25	1736

v\h	1	2	3	4	5	6	7	8	9	10	[%]
9	0,00	0,00	0,00	0,02	0,06	0,10	0,09	0,05	0,02	0,00	0,34
10	0,00	0,00	0,02	0,08	0,16	0,23	0,26	0,18	0,10	0,02	1,05
	0,00	0,01	0,02	0,10	0,22	0,33	0,35	0,24	0,11	0,02	1,40

(e) comp3bit

v\h	3	4	5	6	7	8	9	10	11	
4	0	2	14	42	70	70	42	14	2	256
5	0	0	0	0	0	0	0	128	128	256
6	6	42	126	274	466	510	554	326	0	2304
7	20	280	988	1712	2124	1816	836	160	0	7936
	26	324	1128	2028	2660	2396	1432	628	130	10752

v\h	3	4	5	6	7	8	9	10	11	[%]
4	0,00	0,00	0,00	0,01	0,01	0,01	0,01	0,00	0,00	0,05
5	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,02	0,02	0,05
6	0,00	0,01	0,02	0,05	0,09	0,10	0,11	0,06	0,00	0,44
7	0,00	0,05	0,19	0,33	0,41	0,35	0,16	0,03	0,00	1,51
	0,00	0,06	0,22	0,39	0,51	0,46	0,27	0,12	0,02	2,05

(f) dec3to8

v počet testovacích vektorů
h počet modifikovaných hradel

Tabulka 7.6: Kontingenční tabulky počtu řešení

Název	Popis	Hradel	Tranzistorů	Vstupů	Výstupů
add8	8bit sčítačka	32	250	16	8
addsub	5bit sčitačka/odčítačka	43	274	11	6
comp8	8bit komparátor	39	244	16	8
mul8	8bit násobička	356	2362	16	16
mux16	16bit multiplexor	34	292	33	16
mux8	8bit multiplexor	17	146	17	8
shifter	8bit shifter	24	96	11	8
sub8	8bit odečítačka	35	256	16	8

Tabulka 7.7: Obvody pro rekurzivní algoritmus

Obvod	Hradel			Vektorů			Pokrytí Poruch			CMOS Tranzistorů				čas	krok	k/t
	C	C'	%	$vc(C)$	$vc(C')$	%	$fc(C)$	$fc(C')$	%	$tc(C)$	$tc(C')$	P	%			
c17	6	3	50,00	9	5	55,56	100,00	100,00	100,00	24	30	42	175,00	0d 00:03:23	693	3,41
c432	160	34	21,25	102	54	52,94	99,24	99,83	100,59	824	1028	1168	141,75	23d 21:31:39	6 013 417	2,91
c499	202	35	17,33	67	28	41,79	98,94	100,00	101,07	1764	1818	2062	116,89	13d 14:13:02	2 165 389	1,84
c880a	383	34	8,88	104	63	60,58	100,00	100,00	100,00	1802	1902	2092	116,09	20d 14:08:59	1 632 908	0,92
c1355	506	0	0,00	108	108	100,00	99,49	99,49	100,00	2244	2244	2244	100,00	0d 01:53:47	0	0
c1908	880	34	3,86	163	112	68,71	99,52	99,79	100,27	3446	3568	3762	109,17	38d 09:04:04	1 191 912	0,36
c2670	1269	59	4,65	189	109	57,67	95,74	96,73	101,03	5668	5872	6240	110,09	3d 03:31:16	42 353	0,16
c3540	1669	64	3,83	252	190	75,40	96,00	97,29	101,34	7504	7702	8034	107,06	13d 07:55:33	110 394	0,1
c5315	2307	62	2,69	190	124	65,26	98,88	99,04	100,16	11262	11404	11790	104,69	25d 22:56:46	76 956	0,03
c6288	2416	10	0,41	46	36	78,26	99,56	99,56	100,00	10112	10126	10186	100,73	1d 07:36:02	4 963	0,04
c7552	3513	36	1,02	371	324	87,33	98,26	99,41	101,17	15400	15392	15608	101,35	9d 12:12:37	10 515	0,01
add8	32	5	15,63	17	10	58,82	100,00	100,00	100,00	250	248	290	116,00	0d 13:54:47	305 370	6,1
addsub	43	3	6,98	20	17	85,00	100,00	100,00	100,00	274	272	296	108,03	0d 00:01:07	415	6,19
comp8	39	6	15,38	26	19	73,08	98,27	100,00	101,76	244	250	280	114,75	0d 02:41:07	82 260	8,51
mul8	356	14	3,93	47	31	65,96	100,00	100,00	100,00	2362	2360	2458	104,06	42d 04:45:07	3 604 279	0,99
mux16	34	9	26,47	21	11	52,38	100,00	100,00	100,00	292	320	370	126,71	0d 02:16:39	60 750	7,41
mux8	17	2	11,76	15	13	86,67	100,00	100,00	100,00	146	150	162	110,96	0d 00:00:12	94	7,83
shifter	24	2	8,33	58	49	84,48	88,33	88,33	100,00	96	104	108	112,50	0d 00:00:06	126	21
sub8	35	2	5,71	13	9	69,23	100,00	100,00	100,00	256	246	266	103,91	0d 00:41:41	18 976	7,59

Hradel C	Počet hradel obvodu
Hradel C'	Počet modifikovaných hradel obvodu
$vc(C)$	Počet testovacích vektorů původního obvodu
$vc(C')$	Počet testovacích vektorů modifikovaného obvodu
Vektorů %	$vc(C')$ v procentech oproti $vc(C)$
$fc(C)$	Pokrytí poruch původního obvodu [%]
$fc(C')$	Pokrytí poruch modifikovaného obvodu [%]
Pokr. poruch %	$fc(C')$ v procentech oproti $fc(C')$
$tc(C)$	Počet CMOS tranzistorů původního obvodu
$tc(C')$	Počet CMOS tranzistorů modifikovaného obvodu
P	Odhadovaný počet CMOS tranzistorů obvodu s multifunkčními hradly
CMOS tran. %	P v procentech oproti $tc(C)$
čas	Doba od spuštění, ve které bylo řešení nalezeno
krok	Krok, ve kterém bylo řešení nalezeno
k/t	Průměrný počet kroků za sekundu vypočtený jako krok/čas

Tabulka 7.8: Výsledky rekurzivního algoritmu

Obvod	Hradel			Vektorů			Pokrytí Poruch			čas	ohodnocení	k/t
	C	C'	%	$vc(C)$	$vc(C')$	%	$fc(C)$	$fc(C')$	%			
c17	6	4	66,67	9	5	55,56	100,00	100,00	100,00	3:30:00	1 000 000	79,37
c432	160	22	13,75	102	50	49,02	99,24	99,07	99,83	4:54:00	1 000 000	56,69
c499	202	31	15,35	67	30	44,78	98,94	100,00	101,07	4:54:00	1 000 000	56,69
c880a	383	63	16,45	104	49	47,12	100,00	99,50	99,50	5:00:00	1 000 000	55,56
c1355	506	67	13,24	108	31	28,70	99,49	98,52	99,03	7:42:00	1 000 000	36,08
c1908	880	90	10,23	163	48	29,45	99,52	95,54	96,00	7:00:00	1 000 000	39,68
c2670	1269	99	7,80	189	92	48,68	95,74	95,33	99,57	8:42:00	1 000 000	31,93
c3540	1669	103	6,17	252	171	67,86	96,00	94,32	98,25	9:48:00	1 000 000	28,34
c5315	2307	123	5,33	190	111	58,42	98,88	92,32	93,37	9:00:00	1 000 000	30,86
c6288	2416	94	3,89	46	35	76,09	99,56	98,31	98,74	9:30:00	1 000 000	29,24
c7552	3513	152	4,33	371	207	55,80	98,26	91,49	93,11	1d 10:42:00	1 000 000	8,01

Hradel C	Počet hradel obvodu
Hradel C'	Počet modifikovaných hradel obvodu
$vc(C)$	Počet testovacích vektorů původního obvodu
$vc(C')$	Počet testovacích vektorů modifikovaného obvodu
Vektorů %	$vc(C')$ v procentech oproti $vc(C)$
$fc(C)$	Pokrytí poruch původního obvodu [%]
$fc(C')$	Pokrytí poruch modifikovaného obvodu [%]
Pokr. poruch %	$fc(C')$ v procentech oproti $fc(C')$
čas	Doba běhu algoritmu
ohodnocení	Počet ohodnocení vykonaných algoritmem (1000 populací × 1000 obvodů v každé populaci)
k/t	Průměrný počet ohodnocení za sekundu vypočtený jako ohodnocení/čas

Tabulka 7.9: Výsledky evolučního algoritmu

Obvod	FlexTest					Testgen						Nástroj s evolučním algoritmem						
	kroků	čas	k/t	v	k/tv	kroků	čas	k/t	v	k/tv	f/tg	kroků	čas	k/t	v	k/tv	e/tg	f/e
c17	1 000	2:51	5,85	1	5,85	693	03:23	3,41	1	3,41	1,71	1 000 000	3:30:00	79,37	4	19,84	5,81	0,29
c432	1 000	3:17	5,08	1	5,08	6 013 417	23d 21:31:39	2,91	1	2,91	1,74	1 000 000	4:54:00	56,69	4	14,17	4,87	0,36
c499	1 000	3:20	5,00	1	5,00	2 165 389	13d 14:13:02	1,84	1	1,84	2,71	1 000 000	4:54:00	56,69	4	14,17	7,69	0,35
c880a	1 000	3:18	5,05	1	5,05	1 632 908	20d 14:08:59	0,92	1	0,92	5,50	1 000 000	5:00:00	55,56	4	13,89	15,13	0,36
c1355	1 000	4:22	3,80	1	3,80							1 000 000	7:42:00	36,08	4	9,02		0,42
c1908	1 000	5:00	3,33	1	3,33	1 191 912	38d 09:04:04	0,36	1	0,36	9,27	1 000 000	7:00:00	39,68	4	9,92	27,60	0,34
c2670	1 000	6:00	2,78	1	2,78	42 353	3d 03:31:16	0,16	1	0,16	17,83	1 000 000	8:42:00	31,93	4	7,98	51,24	0,35
c3540	1 000	6:56	2,40	1	2,40	110 394	13d 07:55:33	0,10	1	0,10	25,08	1 000 000	9:48:00	28,34	4	7,09	73,93	0,34
c5315	1 000	6:15	2,67	1	2,67	76 956	25d 22:56:46	0,03	1	0,03	77,71	1 000 000	9:00:00	30,86	4	7,72	224,86	0,35
c6288	1 000	5:53	2,83	1	2,83	4 963	1d 07:36:02	0,04	1	0,04	64,93	1 000 000	9:30:00	29,24	4	7,31	167,56	0,39
c7552	1 000	21:59	0,76	1	0,76	10 515	9d 12:12:37	0,01	1	0,01	59,24	1 000 000	1d 10:42:00	8,01	4	2,00	156,36	0,38

kroků	počet kroků (ohodnocení) algoritmu
čas	doba potřebná pro provedení kroků
k/t	průměrný počet ohodnocení za sekundu vypočtený jako kroků/čas
v	počet vláken, ve kterých algoritmus běžel
k/tv	průměrný počet ohodnocení za sekundu na jedno vlákono vypočtený jako (kroků/čas)/v
f/tg	poměr rychlosti FlexTest a nástroje Testgen na jedno vlákno
e/tg	poměr rychlosti nástroje Testgen a nástroje s evolučním algoritmem na jedno vlákno
f/e	poměr rychlosti FlexTest a nástroje s evolučním algoritmem na jedno vlákno

Tabulka 7.10: Porovnání výkonu nástroje Testgen s prohledáváním do hloubky s nástrojem s evolučním algoritmem

Kapitola 8

Diskuze

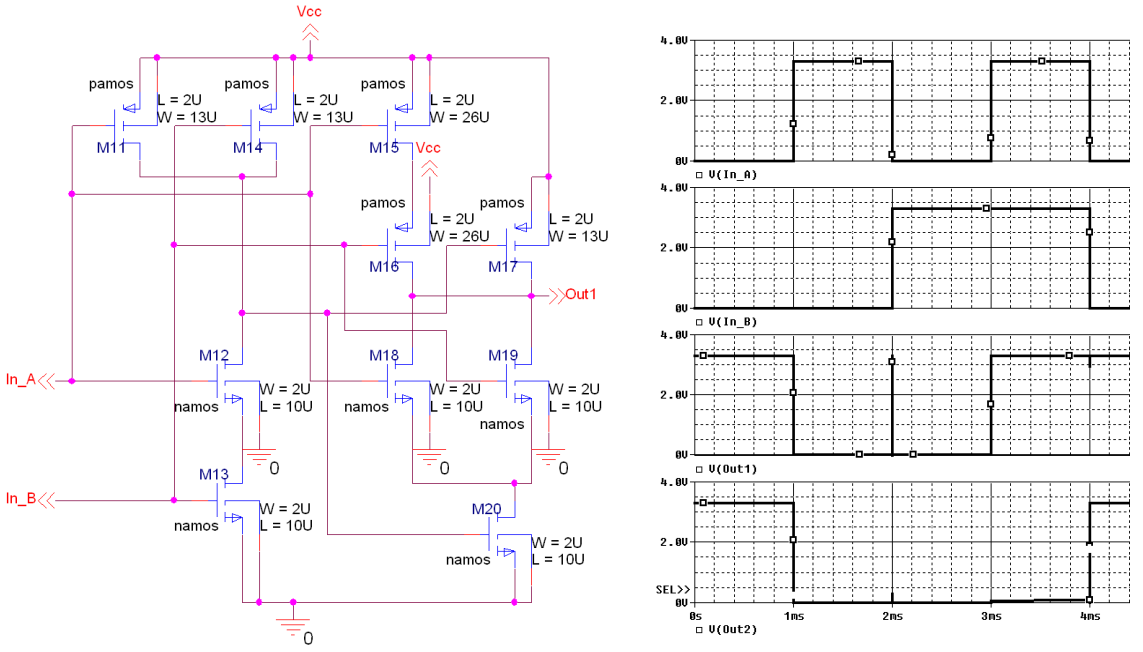
8.1 Platnost testu

Pro celou metodiku je kritická odpověď na otázku, jakou měrou vypovídá test modifikovaného obvodu o obvodu původním. Prezentovaná metodika říká, že obvod obsahuje multifunkční hradla, která jsou v době testu přepnuta a plní jinou funkci, než když obvod plní funkci, pro kterou byl navrhován, a která je potřeba otestovat. Je tedy test modifikovaného obvodu dostatečně vypovídající o výchozím obvodu? Pokud obvod v testovacím režimu splní bezchybně test určité kvality, je zaručena tato kvalita i po přepnutí obvodu do režimu funkčního?

Odpověď na tuto otázku není jednoduchá a je třeba brát v úvahu několik faktorů. Jako první je třeba vzít v úvahu, že je metodika postavena pro strukturní testování. Tedy situace, kdy není testována funkce obvodu jako taková, ale testuje se korektnost struktury obvodu. Pro test tak není třeba, aby obvod plnil svoji funkci. Dále je metodika navrhována pro dnes hojně využívané testy na úrovni hradel s modelem poruchy *t0*/*t1*. Při tomto typu testu se obecně předpokládá, že se jakákoliv porucha uvnitř hradla projeví jako porucha typu *t0* nebo *t1* na některém z vývodů tohoto hradla a testují se tak pouze poruchy na vývodech hradel příp. na spojích. Samotná implementace hradel je skryta a netestuje se. Jelikož je v prezentované metodice struktura obvodu neměnná, je zřejmé, že poruchy na vnitřních spojích stejně jako na vývodech hradel jsou stejné. Otestování poruch obvodu v jednom režimu tak pokrývá tyto poruchy i v režimu druhém. Z tohoto hlediska je tedy test obvodu v testovacím režimu plně vypovídající i pro obvod v režimu funkčním.

Nejproblematičtějším je však předpoklad, že se jakákoliv porucha uvnitř hradla projeví jako *t0* nebo *t1* na alespoň jednom vývodu hradla. I když se jedná o běžně uznávaný zjednodušující předpoklad pro tento typ testování, ve skutečnosti není vždy platný. Jako příklad může posloužit dvouvstupové hradlo XNOR na obrázku 8.1a, jehož průběh výstupního napětí *Out1* (třetí řádek) je na obrázku 8.1b. Nástroj FlexTest pro toto samotné hradlo vygeneruje tři testovací vektory se vstupy AB = 00/01/10 a výstupy 1/0/0. Z tohoto testu je zřejmé, že se zkouší logické 0 i logické 1 na všech vývodech hradla (vstupy i výstupy) a pokud by se zde vyskytovala porucha *t0* nebo *t1*, test by ji dokázal identifikovat. Pokud však v prezentovaném hradle dojde například k defektu připojení hradla tranzistoru M17 na napájecí napětí a ne na příslušný vnitřní spoj (vytvoříme tedy poruchu trvalá 1 na hradle tohoto tranzistoru), dojde ke změně výstupního průběhu napětí dle obrázku 8.1b výstup *Out2* (čtvrtý řádek). Rozdíl mezi výstupy bezporuchového (*Out1*) a poruchového (*Out2*) hradla je pouze v případě vstupní kombinace AB = 11. Tato vnitřní porucha se tedy neprojeví jako *t0* nebo *t1* na některém z vývodů hradla a současně se projeví pouze

při kombinaci vstupů, která není testována. Testu by tak tato porucha unikla.



(a) Schéma

(b) Průběhy napětí na prázdno. *Out2* je výstup hradla s defektem

Obrázek 8.1: Problém testu hradla XNOR

Když už je zřejmé, že ne každá porucha uvnitř hradla se musí projevit jako porucha *t0* nebo *t1* na jeho vývodech, je možné položit si otázku, do jaké míry ovlivní tento fakt navrhovanou metodiku. Zjednodušeně lze říci, že některé poruchy uvnitř hradel nejsou detekovány ani u klasických obvodů (viz prezentovaný problém s hradlem XNOR), proto výskyt těchto poruch u multifunkčních hradel použitých pro optimalizaci testu obvodu vlastnosti testu nijak významně neovlivní. Podíváme-li se ale na problém do hloubky, důležitá bude míra kolik a jak pravděpodobných vnitřních poruch použitých hradel může být testy nedetekováno. Lze předpokládat, že se složitostí vnitřní struktury hradla poroste i množství možných defektů, které se projeví poruchami hradla. Ačkoliv jsem na podobné téma výzkum neprováděl, domnívám se, že s rostoucím počtem možných poruch poroste i počet těch poruch, které mohou testům uniknout. Pro prezentovanou metodiku tak bude také důležité, aby použitá multifunkční hradla obsahovala co nejméně potenciálních poruch, které mohou uniknout testům, a tedy přeneseně, aby měla co nejjednodušší vnitřní strukturu. O co bude použité multifunkční hradlo složitější, o to se může zvýšit množství testem potenciálně nedetekovaných poruch.

Jelikož ne každá porucha uvnitř hradla se musí projevit jako porucha *t0* nebo *t1* na jeho vývodech, je také důležité zamyslet se nad problémem poruch hradel, které se projeví jen v jednom režimu. Tedy kdy v testovacím režimu hradlo funguje správně (resp. úspěšně projde testy), kdežto ve funkčním je poruchové. Pro tento problém však platí stejné závěry jako v předchozím odstavci a výsledné vlastnosti budou dány návrhem použitých multifunkčních hradel.

Posledním důležitým problémem týkajícím se platnosti testu je možnost poruchovosti samotného přepínání funkce multifunkčních hradel mezi testovacím a funkčním režimem. Z hlediska ověření funkce obvodu je nejhorším případem situace, kdy je hradlo v testova-

cím režimu, ale nelze je přepnout do režimu funkčního. Tento problém lze demonstrovat například na konvenčním NAND/NOR hradle prezentovaným v kapitole 5.1 na obrázku 5.1 a poruchou *t0* nebo *t1* na vstupu *Vsel*. Tedy vstupu, který řídí funkci hradla. V případě nemožnosti přepnout hradlo z režimu testovacího do režimu funkčního by test nenašel poruchu, ale obvod by pravděpodobně funkční nebyl.

Možností, jak jmenované problémy řešit, může být více. První možností může být prostý předpoklad, že podobné situace nenastanou. Tedy stejný předpoklad jako u klasické metody, že vše funguje a nebo se poruchy projeví jako *t0* nebo *t1* na některém z vývodů hradla. Pro naše potřeby doplníme tento předpoklad o tvrzení, že se případné poruchy projeví na funkčních vývodech. Další možností je řešení v podobě jednoduchého doplňkového testu ve funkčním režimu. Tento test by měl za cíl co nejjednodušeji otestovat potenciální poruchy hradel vyskytující se pouze v tomto režimu a zdali jsou hradla korektně přepnuta.

Jako poslední zmíním možnost řešení v podobě přepínání hradel i v průběhu testu, kdy podmínka řídící funkci hradel by byla chápána jako další vstup obvodu podléhající testu. Test by tak otestoval jak samotné přepínání multifunkčních hradel, tak oba jejich režimy. Došlo by tak k výrazné eliminaci výše zmiňovaných problémů. Podobné přístupy nebyly v rámci práce podrobně zkoumány a jsou ponechány jako možné náměty na případné další práce. Považuji však za důležité na ně upozornit.

8.2 Využití multifunkčních hradel

Důležitou částí prezentované optimalizace testu je také výběr multifunkčních hradel, která budou použita. V kapitole 3 bylo uvedeno několik možných technologií, z nichž některé byly podrobněji prozkoumány v kapitole 5. Dále budou zhodnoceny vlastnosti jednotlivých technologií vzhledem k problematice optimalizace testu.

8.2.1 Konvenční hradla

Základní možností je pro optimalizaci testu využít konveční multifunkční hradla. Ta jsou navržena stejným způsobem jako zbytek obvodu a použití konvenčních hradel tak nebude mít větší vliv na funkční parametry obvodu. U tohoto postupu však může být problematická složitost výsledného multifunkčního hradla. Vezmeme-li například obvod, ve kterém je pro účely testu potřeba nahradit hradlo NAND (obrázek 2.6c) za multifunkční hradlo NAND/NOR (obrázek 5.1), vzroste počet tranzistorů na implementaci tohoto hradla ze čtyř na deset. To má za následek nejen zvětšení plochy hradla při implementaci, ale vzhledem ke skutečnostem uvedeným v kapitole 8.1 to také může vést ke zvýšenému riziku poruchy hradla nedetekované testem. Použitelnost tak bude záležet na požadavcích na testy a existenci multifunkčních hradel s jednoduchou vnitřní strukturou.

8.2.2 Polymorfní hradla

Další z možností je pro optimalizaci testu využít polymorfní hradla. Zajímavou alternativou by byla například polymorfní hradla, jejichž funkce je závislá na napájecím napětí. Při určitém napájecím napětí by mohlo být hradlo v režimu testu a při jiném ve funkčním režimu. Odpadla by tak nutnost speciálního vývodu pouzdra pro přepínání režimu testu a v obvodě by nemusel existovat spoj pro rozvedení tohoto signálu. Ze simulací v kapitole 5.2 je však zřejmé, že polymorfní hradla trpí různými problémy a na rozdíl od klasický hradel CMOS je nelze chápat jako obecné stavební prvky složitějších číslicových obvodů.

Jedním z hlavních problému prezentovaných polymorfních hradel je například jejich spotřeba. Pro jednoduchost stanovme, že průměrný odběr proudu z napájení prezentovaných polymorfních hradel je 100 μA. Při aplikaci metody optimalizace testu v kapitole 6.3 byla změněna funkce přibližně u 3 % hradel. Pokud budeme uvažovat obvod například s milionem tranzistorů, pak 3 % znamenají třicet tisíc hradel. Pokud bychom pro tato hradla použili multifunkční s průměrným odběrem 100 μA, bude jen těchto třiceti tisíc polymorfních hradel představovat odběr 3A. V případě napájení 3.3V by obvod potřeboval příkon 10W pouze pro polymorfní hradla! Pokud bychom vzali jako příklad obvod s miliardou tranzistorů, činil by příkon jen polymorfních hradel 10kW! Tento příkon je navíc stanoven v režimu na prázdno. Jelikož polymorfní hradla většinou nemají na výstupu přesné napěťové úrovně, po připojení dalších hradel na jejich výstup by odběr proudu a tedy příkon celého obvodu opět znatelně vzrostl. Za tohoto stavu tedy nejsou současná polymorfní hradla vhodným kandidátem pro podobné aplikace. Zlepšení stavu by mohla přinést nová polymorfní hradla, která by podobnými problémy netrpěla.

8.2.3 Grafenová hradla

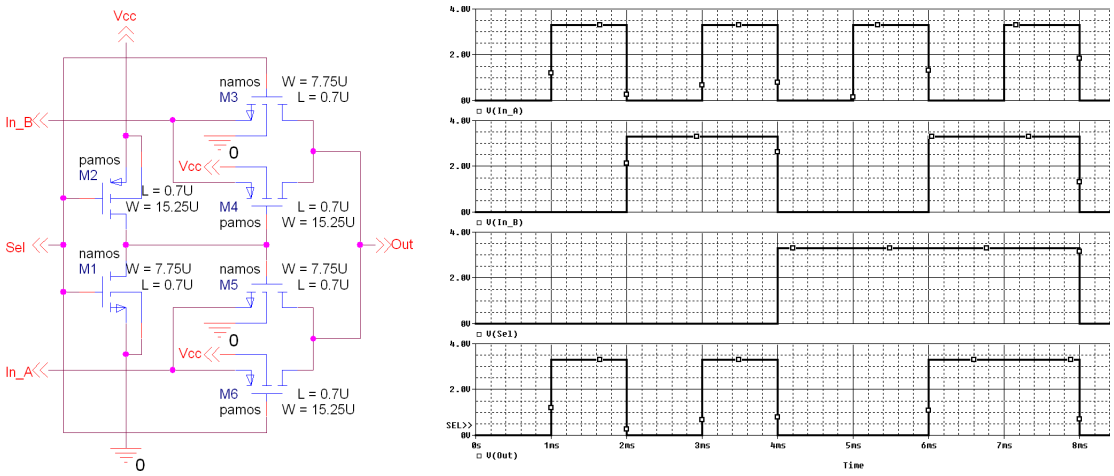
Grafenová multifunkčních hradla se pro optimalizaci testu jeví jako velmi zajímavá. Například multifunkční hradlo prezentované v 3.3 je realizováno pouhými třemi polovodičovými oblastmi v polovodičovém substrátu, jednou grafenovou vrstvou a třemi elektrodami. Jelikož funkce tohoto hradla je shodná s funkcí multiplexeru, lze také jednoduše určit jak by bylo podobné hradlo implementováno v CMOS technologii. Klasická CMOS implementace by sestávala z šesti tranzistorů (viz obrázek 8.2). Jelikož jsou pro každý tranzistor potřeba v substrátu dvě polovodičové oblasti a jedna elektroda pro bránu (viz obrázek 2.1), pro implementaci tohoto hradla by tak bylo potřeba dvanáct polovodičových oblastí a šest elektrod. Oproti grafenovému hradlu to je čtyřnásobek v počtu polovodičových oblastí a dvojnásobek v počtu elektrod. Lze tedy předpokládat, že grafenové multifunkční hradlo spotřebuje nejen menší plochu na výsledném substrátu, ale hlavně bude obsahovat méně potenciálních defektů a tedy možných poruch. Vzhledem ke skutečnostem uvedeným v kapitole 8.1 se podobná hradla jeví jako vhodná pro prezentovanou metodiku.

Na druhou stranu je třeba poznamenat, že technologie grafenu je v elektronice v počátcích a je třeba dalších výzkumů v oblasti jejich vlastností a využitelnosti. V roce 2007 se nepředpokládalo, že by se během následujících dvaceti let objevily složitější číslicové obvody na této technologii [17]. Využitelnost grafenových multifunkčních hradel pro prezentovanou metodiku bude možné s jistotou určit až bude technologie lépe prozkoumána a budou přesnější znalosti o jejich vlastnostech.

8.2.4 Shrnutí

V předchozích kapitolách byla diskutována použitelnost jednotlivých technologií multifunkčních hradel pro optimalizaci testu. Jako velmi zajímavá se zdají být grafenová hradla. Například hradlo prezentované v 3.3 má jednoduchou vnitřní strukturu a zabírá relativně malou plochu na polovodičovém substrátu. Bohužel však tato technologie zatím není v elektronice rozšířena a je třeba dalších výzkumů ohledně jejich vlastností. V dnešní době tak tato technologie není použitelná.

Polymorfní hradla se jeví zajímavá po stránce jejich o něco jednodušší vnitřní struktury oproti hradlům konvenčním a možností zjednodušení způsobu přepínání funkce hradel (například jen změnou velikosti napájecího napětí obvodu). Aktuálně známá polymorfní hradla však trpí několika vážnými problémy, z nichž za největší se dá považovat odběr proudu.



(a) schéma (b) simulace OrCad pSpice

Obrázek 8.2: CMOS Implementace multiplexoru

Ten výrazně zvyšuje odběr proudu celého obvodu a prakticky znemožňuje využití těchto hradel ve větším počtu. Jako další problém lze jmenovat nižší maximální kmitočet oproti hradlům konvenčním. Pokud nebudou vytvořena nová polymorfní hradla, která zmiňované problémy eliminují, nebudou polymorfní hradla pro tento druh úlohy dobře použitelná.

Jako nejlépe použitelná se tak jeví hradla konvenční. Jsou v dnešní době bez potíží dostupná, jednoduše implementovatelná a jen minimálně ovlivní funkční parametry obvodu. Při jejich použití je však třeba dbát opatrnosti ohledně jejich složitosti a vlivu na vlastnosti testu.

Kapitola 9

Závěr

Tato práce se zabývala optimalizací parametrů testu číslicového obvodu pomocí multifunkčních hradel. Po prvních kapitolách zabývajících se základními prerekvizitami v oblasti elektroniky, diagnostiky a matematické optimalizace prezentovala různé přístupy a technologie návrhu multifunkčních logických hradel. Tyto přístupy byly v rámci práce analyzovány a vybraná hradla z různých technologií byla poté podrobena simulacím v programech typu SPICE. Simulace se týkaly převážně hradel polymorfních. Konvenční multifunkční hradla obecně dosahují obdobných parametrů jako hradla běžná, a tudíž je nebylo nutné po stránce elektronických vlastností blíže zkoumat. Elektronické vlastnosti grafenového hradla nebylo bohužel možné analyzovat, neboť pro tento princip nejsou dostupné simulační modely a není ani dostupná technologie pro jejich výrobu.

V následující hlavní části práce byl představen samotný princip optimalizace testu pomocí multifunkčních hradel, který byl následně popsán pomocí formálních matematických prostředků. Tento formální popis byl pro práci důležitým předpokladem, neboť na základě matematického popisu řeší několik klíčových problémů metodiky a není tak třeba se jimi dále zabývat. Jedná se například o problematiku platnosti funkce obvodu po náhradě některých jeho hradel za multifunkční. Je-li metodika a její formálni popis dodržen, nemůže náhradami hradel dojít k vytvoření neplatného obvodu. Všechny náhrady hradel, které metodika povolí, vedou vždy k platným obvodům, které umožňují plnit svoji výchozí požadovanou funkci. To zjednodušuje implementaci metodiky a zkracuje její výpočetní čas, neboť velmi mnoho kandidátních řešení je vyloučeno již samotnou metodikou. Tím dojde nejen ke zmenšení prohledávaného prostoru řešení, ale také není třeba výpočetního výkonu na verifikaci kandidátních řešení, která by mohla být neplatná.

V další části práce byla prezentována možnost optimalizace testovatelnosti obvodu. Jako první bylo ukázáno, že je možné pomocí navrhované metodiky dosáhnout podobných vlastností jako při DfT ad-hoc metodě vkládání testovacích řídicích bodů. Uvedený princip má oproti standardní metodě výhodu v podobě neměnné vnitřní struktury na úrovni hradel. To vede nejen ke zjednodušenému návrhu, ale také v případě vhodných multifunkčních hradel tento přístup neovlivní dynamické parametry obvodu.

Jako druhý přístup optimalizace testovatelnosti byl ukázán jednoduchý princip založený na metodě SCOAP. Ten s pomocí multifunkčních hradel použitých ve vybraných místech obvodu dokáže snížit hodnoty řiditelnosti a z části i pozorovatelnosti, tedy celkově zlepšit testovatelnost obvodu. Dále metodika umožňuje i deterministický výběr vhodného hradla pro náhradu. Pro tuto úlohu byly standardní ukazatele metody SCOAP rozšířeny o další ukazatele identifikující místa s největším potenciálem pro optimalizaci. Jedná se však pouze o jednoduchý princip, který uvažuje jen aktuální spoj a nezabývá se následky náhrady

hradla v širším okolí. Domnívám se, že je možné nalézt lepší přístup, který by zohledňoval i vliv na ostatní prvky. Metodiku je tak možné dále rozvíjet a zlepšovat její výsledky.

Hlavní částí práce však bylo vytvoření a implementace metodiky pro optimalizaci parametrů testu prezentované v kapitole 6.3. Na jejím základě byly popsány a vytvořeny dva programové nástroje. Ty měly za úkol snížení počtu testovacích vektorů při zachování pokrytí poruch s co nejmenším nárůstem složitosti obvodu vyjádřeným odhadovaným počtem CMOS tranzistorů. První nástroj implementovaný mnou v jazyce Perl nazvaný Testgen pro optimalizaci umožňoval použít metodu kompletního nebo rekurzivního prohledávání. Druhý nástroj implementovaný v jazyce Python vytvořil Ing. Šimáček pod vedením Prof. Ing. Lukáše Sekaniny Ph.D., a pro optimalizaci využíval evoluční přístup. Nástroje byly následně vyzkoušeny na různých obvodech, kde průnikem byla testovací sada ISCAS 85. U obou nástrojů a všech použitých obvodů vyjma případu nástroje Testgen a obvodu c1355 došlo ke znatelnému snížení počtu testovacích vektorů obvykle v řádu desítek procent. U nástroje Testgen pak došlo k zachování nebo mírnému navýšení pokrytí poruch, u nástroje Ing. Šimáčka došlo většinou k jeho mírnému snížení. S rostoucí složitostí obvodu se nárůst složitosti implementace snižoval a u největších obvodů byl odhadovaný nárůst počtu CMOS tranzistorů pro implementaci v řádu jednotek procent. Nástroje tak byly schopné nalézt kvalitní řešení a uspořit velkou část testovacích vektorů.

Důležitou částí bylo také samotné porovnání výsledků jednotlivých nástrojů z pohledu jejich výkonnosti. Ukázalo se, že je důležitá nejen použitá optimalizační metoda, ale také způsob a kvalita samotné implementace. Implementace zaměřená na výkon dovede pracovat mnohonásobně rychleji a má tak potenciál v rozumném čase nalézat řešení i pro řádově složitější obvody, než ty ze sady ISCAS 85.

Jako posledním hlavním tématem práce bylo zamyšlení a diskuze nad vypovídající hodnotou testu změněného obvodu a vlastnostech prezentovaných technologií multifunkčních hradel vzhledem k navrhované metodice. Z této části lze vytvořit zjednodušený závěr, že v dnešní době se jako nejvhodnější jeví využití hradel konvenčních. Důraz však musí být kladen na jejich jednoduchou vnitřní strukturu z důvodu snížení potenciálního rizika vnitřní chyby, která se neprojeví na vývodech hradla jako chyba *t0* nebo *t1*. Z hlediska vlastností se jako vhodnější zdají být hradla grafenová. Tato technologie je však ve fázi výzkumu a není v dnešní době použitelná. Její vhodnost tak doopravdy potvrdí až další výzkumy případně praktické zkušenosti. Aktuálně známá polymorfní hradla se pro navrhovanou metodiku nehodí z důvodů jejich neoptimálních elektronických vlastností. Pokud nebudou v budoucnu vytvořena hradla s lepšími vlastnostmi, lze celou technologii polymorfních hradel pro tuto metodiku považovat za nevhodnou. Její použitelnost by byla pouze ve speciálních případech nebo v kombinaci s technologiemi ostatními.

Na závěr lze říci, že práce potvrdila předpoklady. Byla vytvořena, formálně popsána, implementována a na testovacích obvodech ověřena metodika pro optimalizaci parametrů testu obvodu. Současně se podařilo ukázat, že pomocí multifunkčních hradel je možné optimalizovat diagnostické vlastnosti obvodu takovým způsobem, aby došlo k požadovaným úpravám parametrů výsledných testů obvodů při minimálních dopadech na kvalitu a věrohodnost těchto testů.

9.1 Přínos práce

Za hlavní přínos této práce považuji potvrzení předpokladu, že bez změny funkční struktury obvodu na úrovni hradel lze multifunkčními prvky upravit diagnostické vlastnosti obvodu takovým způsobem, aby došlo k požadovaným změnám některých parametrů výsledného

testu, přičemž ostatní parametry mohou zůstat na podobné kvalitativní úrovni. Samotné ověření předpokladu na úloze snížení počtu testovacích vektorů při zachování pokrytí poruch obvodu pak má dobré předpoklady pro snížení nákladů na testování v praxi. Tato optimalizace může zkrátit čas potřebný pro aplikaci testu, zjednodušit testovací zařízení (např. je potřeba méně paměti pro uložení testovacích vektorů), případně zajistit další podobné úspory.

Za další důležitý přínos lze také označit vytvoření základní metodiky prezentované optimalizace (cíl 1), která je vystavěna nad formálně popsaným matematickým základem (cíl 2). Díky tomu je možné s metodikou pracovat pomocí matematických aparátů a těžit z výhod z toho vyplývajících.

Přínosem je také vytvoření konkrétní metodiky pro optimalizaci parametrů testu obvodu, která byla použita na snížení počtu testovacích vektorů a byla ověřena implementací (cíl 3) a výpočty nad různými obvody, včetně testovací sady ISCAS 85 (cíl 4).

Za další přínos lze také určit provedenou analýzu technologií pro tvorbu multifunkčních hradel, včetně nových jako jsou polymorfní hradla nebo hradla založená na grafenu. Důležitou částí je i zhodnocení použitelnosti daných technologií pro navrhované metodiky (cíl 5).

9.2 Možná rozšíření a další práce

Domnívám se, že princip a metodiku prezentovanou v této práci lze chápat jako základní postup, nad kterým je možné stavět další přístupy k optimalizaci diagnostických vlastností obvodů. Plně opomenutou oblastí je například možnost optimalizace testů sekvenčních obvodů. Zde by mohla být zajímavá zejména varianta modifikace sekvenčních částí tak, aby se při testu chovaly jako kombinační, nebo aby došlo alespoň ke snížení sekvenční hloubky testu. Tento přístup by mohl být doplňkem ke stávající metodě částečného scan řetězce. Další prakticky nezmíněnou možností je využití vícefunkční logiky, kdy by hradla mohla mít více než jeden testovací režim.

Z hlediska rozšiřování prezentovaných metodik existuje také velký prostor pro vylepšení. U optimalizace testu například nebyla zkoumána možnost přepínání multifunkčních hradel i v průběhu aplikace testu. To by mohlo řešit problematiku detekce možné poruchy přepínání funkce hradla a současně by to mohlo mít pozitivní vliv na výsledky optimalizací.

U optimalizace testu by také mohla pomoci například metodika pro jednodušší identifikaci hradel, která by bylo vhodné změnit na jinou funkci. Tato metodika by mohla být například založena na některém z principu analýzy testovatelnosti, pomocí které by se identifikovala špatně testovatelná místa a hradla, jejichž změnou by se situaci zlepšila. Pokud by se podobnou metodiku podařilo nalézt, mohl by se prohledávaný prostor řešení značně zmenšit, nebo by bylo dokonce možné metodu kombinatorické optimalizace úplně opustit a náhrady hradel určovat pouze tímto přístupem. Jednou z možností by mohla být právě v této práci diskutovaná metodika založená na SCOAP.

Z hlediska optimalizace testovatelnosti založené na SCOAP zde existuje prostor pro vylepšení, zejména v problematice identifikace hradel pro náhradu. Prezentovaná metodika se při volbě hradla opírá pouze o informace o řiditelnosti a pozorovatelnosti na vývodech daného hradla. Domnívám se však, že v obvodě mohou existovat hradla, jejichž náhradou dojde nejen k optimalizaci parametrů testovatelnosti na jejich vývodech, ale také, že tato změna propaguje na další prvky obvodu. Pro dobrou volbu hradla pro náhradu by tak byla potřeba nejen informace o změně v okolí tohoto hradla, ale také jak tato změna ovlivní celý

obvod. Tento princip by mohl být například založen na identifikaci a analýze kritických cest.

Literatura

[1] Abramovici, M.; Breuer, M.; Friedman, A.: *Digital systems testing and testable design*. Electrical engineering, communications, and signal processing, IEEE Press, 1994, ISBN 9780780310629, 652 s. 29, 30

[2] Balakrishnan, V.: *Schaum's outline of theory and problems of graph theory*. Schaum's outline series, McGraw-Hill, 1997, ISBN 9780070054899, 293 s. 57

[3] Boole, G.: The Calculus of Logic. *Cambridge and Dublin Mathematical Journal*, ročník III, 1848: s. 183–98. 11

[4] Boyd, S.; Vandenberghe, L.: *Convex optimization*. Cambridge University Press, 2004, ISBN 9780521833783, 716 s. 30

[5] Brglez, F.; Fujiwara, H.: A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Simulator in Fortran. *Proceedings International Symposium on Circuits and Systems (ISCAS)*, 1985: str. 695–698. 25

[6] BSIM Homepage [online]. 2010 [cit. 2011-07-18].
URL http://www-device.eecs.berkeley.edu/~bsimsoi/ 21

[7] Bullis, K.: Graphene Transistors [online]. Leden 2008 [cit. 2011-09-03].
URL http://www.technologyreview.com/Nanotech/20119/ 38

[8] Bushnell, M.; Agrawal, V.: *Essentials of electronic testing for digital, memory, and mixed-signal VLSI circuits*. Frontiers in electronic testing, Kluwer Academic, 2000, ISBN 9780792379911, 690 s. 26, 27, 30, 57, 58, 65

[9] Chen, Z.; Appenzeller, J.; Lin, Y.-M.; aj.: An integrated logic circuit assembled on a single carbon nanotube. *Science*, ročník 311, č. 5768, 2006: str. 1735. 38

[10] CMOS Multifunction Expandable 8-Input Gate Data Sheet. Říjen 2003.
URL http://www.ti.com/lit/ds/symlink/cd4048b.pdf 37

[11] Crouch, A.: *Design for test*. Prentice Hall, USA, 1999, ISBN 0130848271, 349 s. 21, 22, 23, 24, 25, 26, 28, 30

[12] Diestel, R.: *Graph theory*. Graduate texts in mathematics, Springer, třetí vydání, 2006, ISBN 9783540261834, 410 s. 57

[13] Forest, L. D.: Space Telegraphy. U.S. Patent 879,532. 1908-02-18 (vyplněn 1907-01-29). 4, 7

[14] Foty, D.; Foty, D.: *MOSFET modeling with SPICE: principles and practice*. Prentice Hall series in innovative technology, Prentice Hall PTR, 1997, ISBN 9780132279352, 653 s. 21

[15] Gajsky, D.; Kuhn, R. H.: Guest Editors' Introduction: New VLSI Tools. *IEEE Computer*, ročník 6, 1983: s. 11–14. 16

[16] Geiger, R. L.; Allen, P. E.; Strader, N. R.: *VLSI design techniques for analog and digital circuits*. The McGraw-Hill Companies, první vydání, 1989, ISBN 9780070232532, 951 s. 9, 10, 11

[17] Geim, A. K.; Novoselov, K. S.: The rise of graphene. *Nature Materials*, ročník 6, Březen 2007: s. 183–191. 38, 90

[18] Goldstein, L. H.: Controllability/Observability Analysis of Digital Circuits. *IEEE Trans. on Circuits and Systems*, ročník CAS-26, č. 9, 1979: str. 685–693. 26

[19] Goodstein, R.: *Boolean Algebra*. Dover Publications, 2007, ISBN 9780486458946, 140 s. 11

[20] Gregg, J.: *Ones and zeros: understanding Boolean algebra, digital circuits, and the logic of sets*. IEEE Press understanding science & technology series, IEEE Press, 1998, ISBN 9780780334267, 281 s. 11

[21] Heil, O.: Improvements in or relating to electrical amplifiers and other control arrangements and devices. Great Britain GB439457. 1935-12-06 (vyplněn v GB 1934-03-02, původně v DE 1934-03-02). 7

[22] Šimáček, J.; Sekanina, L.; Stareček, L.: Evolutionary Design of Reconfiguration Strategies to Reduce the Test Application Time. In *Evolvable Systems: From Biology to Hardware*, LNCS 6274, Springer Verlag, 2010, ISBN 978-3-642-15322-8, s. 214–225. 77

[23] Kanellos, M.: New life for Moore's Law [online]. 2005 [cit. 2011-09-04]. URL http://news.cnet.com/New-life-for-Moores-Law/2009-1006_3-5672485.html 4, 7

[24] Khateb, A.; Biolek, D.; Novacek, K.: On the Design of low-voltage low-power bulk-driven CMOS Current Conveyors. *Electronics Technology*, ročník 29, 2006: s. 318–321, ISSN 10.1109/ISSE.2006.365121. 9

[25] Kohavi, Z.; Jha, N.: *Switching and finite automata theory*. Cambridge University Press, 2010, ISBN 9780521857482, 617 s. 15

[26] Langeheine, J.: *Intrinsic Hardware Evolution on the Transistor Level*. Dizertační práce, Rupertus Carola University, Heidelberg, 2005. 9

[27] Falling into the Gap - Berkeley Lab Researchers Take a Critical First Step Toward Graphene Transistors [online]. press release, Listopad 2007. URL http://www.lbl.gov/Science-Articles/Archive/sabl/2007/Nov/gap.html 38, 40

[28] Lemme, M. C.; Echtermeyer, T. J.; Baus, M.; aj.: A Graphene Field-Effect Device. *IEEE Electron Device Letters*, ročník 28, Duben 2007: s. 282–284. 38

[29] Lilienfeld, J. E.: Method and apparatus for controlling electric current. U.S. Patent 1,745,175. 1930-01-28 (vyplněn v CA 1925-10-22, v US 1926-10-08). 7

[30] Liou, J.; Ortiz-Conde, A.; García Sánchez, F.: *Analysis and design of MOSFETs: modeling, simulation, and parameter extraction*. Kluwer Academic Publishers, 1998, ISBN 9780412146015, 349 s. 8, 9

[31] Logický člen [online]. 2011 [cit. 2011-07-08].
URL http://cs.wikipedia.org/wiki/Logický_člen 109

[32] McCluskey, E.: A Pseudoexhaustive Test Technique. In *IEEE Transactions on computers*, 1984, ISSN 0018-9340, s. 541–546. 24

[33] Mollick, E.: Establishing Moore's Law. *IEEE Annals of the History of Computing*, ročník 28, 2006: s. 62–75, ISSN 1058-6180. 4, 7

[34] Moore, G. E.: Cramming more components onto integrated circuits. *Electronics*, ročník 38, č. 8, 1965: s. 114–117. 4, 7

[35] Multi-Function Logic Gates [online]. 2011 [cit. 2011-09-01].
URL http://www.semicon.toshiba.co.jp/eng/product/logic/selection/topics/1184109_2333.html 36

[36] Nagel, L. W.: *SPICE2: A Computer Program to Simulate Semiconductor Circuits*. Dizertační práce, EECS Department, University of California, Berkeley, 1975.
URL http://www.eecs.berkeley.edu/Pubs/TechRpts/1975/9602.html 19

[37] Nagel, L. W.; D.O.Pederson: SPICE (Simulation Program with Integrated Circuit Emphasis). Technická Zpráva UCB/ERL M382, EECS Department, University of California, Berkeley, Duben 1973.
URL http://www.eecs.berkeley.edu/Pubs/TechRpts/1973/22871.html 19

[38] The Nobel Prize in Physics 2010 [online]. 2011 [cit. 2011-09-03].
URL http://www.nobelprize.org/nobel_prizes/physics/laureates/2010/ 38

[39] Novoselov, K. S.; Geim, A. K.; Morozov, S. V.; aj.: Electric Field Effect in Atomically Thin Carbon Films. *Science*, ročník 306, Říjen 2004: s. 666–669. 38

[40] Null, L.; Lobur, J.: *The Essentials of Computer Organization and Architecture*. Jones & Bartlett Publishers, Incorporated, třetí vydání, 2010, ISBN 9781449600068, 844 s. 16

[41] NVIDIA's Next Generation CUDA Compute Architecture: Fermi [online]. 2009 [cit. 2010-01-21].
URL http://www.nvidia.com/object/IO_86776.html 4, 7

[42] Papadimitriou, C.; Steiglitz, K.: *Combinatorial optimization: algorithms and complexity*. Dover books on mathematics, Dover Publications, 1998, ISBN 9780486402581, 496 s. 30

[43] Pečenka, T.: *Prostředky a metody pro automatické generování testovacích obvodů.* Dizertační práce, Vysoké učení technické, Brno, 2007. 24

[44] Peirce, C. S.: *1857-1866.* Indiana University Press, 1982, ISBN 9780253372017, 698 s. 11

[45] Price, R. W.: *Roadmap to Entrepreneurial Success*. AMACOM, 2004, ISBN 9780814471906, 292 s. 7

[46] Quarles, T. L.: *Analysis of Performance and Convergence Issues for Circuit Simulation.* Dizertační práce, EECS Department, University of California, Berkeley, 1989.
URL http://www.eecs.berkeley.edu/Pubs/TechRpts/1989/1216.html 19

[47] Růžička, R.; Sekanina, L.; Prokop, R.: Physical Demonstration of Polymorphic Self-checking Circuits. In *Proc. of the 14th IEEE Int. On-Line Testing Symposium*, IEEE Computer Society, 2008, ISBN 978-0-7695-3264-6, s. 31–36. 37, 38, 47

[48] Shannon, C. E.: *A symbolic analysis of relay and switching circuits.* Diplomová práce, Massachusetts Institute of Technology, 1940. 11

[49] Sheffer, H. M.: A set of five independent postulates for Boolean algebras, with application to logical constants. *Transactions of the American Mathematical Society*, ročník 14, 1913: s. 481–488. 11

[50] Sicard, E.: *Microwind & Dsch Users's Manual version 2.7.* National institute & computer engineering, Toulouse, France, 2003, ISBN 2-87649-046-3, 196 s. 9, 20

[51] Siffert, P.; Krimmel, E.: *Silicon: evolution and future of a technology.* Springer, 2004, ISBN 9783540405467, 549 s. 7

[52] Smith, R. D.: Simulation. In *Encyclopedia of Computer Science*, editace A. Ralston; E. D. Reilly; D. Hemmendinger, Chichester, UK: John Wiley and Sons Ltd., Čtvrté vydání, 2000, ISBN 0-470-86412-5, s. 1578–1587. 19

[53] Stoica, A.: Polymorphic electronics – A novel type of circuits with multiple functionality. New Technology Report NPO-21213, NASA, 2000. 37

[54] Stoica, A.; Zebulum, R.: Four-Function Logic Gate Controlled by Analog Voltage. New Technology Report NPO-40772, NASA, 2006. 37, 38

[55] Stoica, A.; Zebulum, R.; Guo, X.; aj.: Taking evolutionary circuit design from experimentation to implementation: some useful techniques and a silicon demonstration. In *IEE Proc.-Comp. Digit. Tech.*, ročník 151(4), 2004, ISSN 1350-2387, s. 295–300. 37, 38, 46

[56] Stoica, A.; Zebulum, R.; Keymeulen, D.: Polymorphic electronics. In *In Proc. of International Conference on Evolvable Systems: From Biology to Hardware, LNCS 2210*, Springer, 2001, s. 291–302. 37, 38

[57] Stoica, A.; Zebulum, R.; Keymeulen, D.; aj.: On polymorphic circuits and their design using evolutionary algorithms. In *Proc. of IASTED International Conference on Applied Informatics AI2002, Insbruck*, 2002. 37

[58] Strnadel, J.: *Analýza a zlepšení testovatelnosti číslicového obvodu na úrovni meziregistrových přenosů*. Dizertační práce, Vysoké učení technické, Brno, 2004. 29

[59] Tanachutiwat, S.; Lee, J. U.; Wang, W.; aj.: Reconfigurable multi-function logic based on graphene P-N junctions. In *Proceedings of the 47th Design Automation Conference*, DAC '10, New York, NY, USA: ACM, 2010, ISBN 978-1-4503-0002-5, s. 883–888. 38, 39, 40

[60] TC7SP57FU, TC7SP58FU Data Sheet.
URL http://www.semicon.toshiba.co.jp/docs/datasheet/en/LogicIC/TC7SP57FU_TC7SP58FU_en_datasheet_081222.pdf 36

[61] TC7SP97TU, TC7SP98TU Data Sheet.
URL http://www.semicon.toshiba.co.jp/docs/datasheet/en/LogicIC/TC7SP97TU_TC7SP98TU_en_datasheet_071019.pdf 36

[62] Tesla, N.: Method of and Apparatus for Controlling Mechanism of Moving Vehicle or Vehicles. U.S. Patent 613,809. 1898-11-08 (vyplněn 1898-07-01). 11

[63] The Spice Page [online]. 2011 [cit. 2011-07-14].
URL http://bwrc.eecs.berkeley.edu/classes/icbook/spice/ 19

[64] Transistor [online]. 2010 [cit. 2010-01-28].
URL http://en.wikipedia.org/wiki/Transistor 9, 10

[65] Vahid, F.: *Digital design, with RTL design, VHDL, and Verilog*. John Wiley & Sons, 2010, ISBN 9780470531082, 592 s. 11

[66] Virius, M.: *Základy algoritmizace*. Vydavatelství ČVUT, 1995, ISBN 9788001013465, 195 s. 32

[67] Wanlass, F. M.: Low stand-by power complementary field effect circuity. U.S. Patent 3,356,858. 1963-06-18. 14

[68] Warwick, C.: Everything you always wanted to know about SPICE (But were afraid to ask). *The EMC journal*, , č. 82, 2009: s. 27–29, ISSN 1748-9253. 19

[69] Wernick, W.: Complete Sets of Logical Functions. *Transactions of the American Mathematical Society*, ročník 51, 1942: s. 117–132. 11

[70] World's Fastest Transistor Approaches Goal Of Terahertz Device [online]. 2006 [cit. 2011-07-29].
URL http://news.illinois.edu/news/06/1211transistor.html 10

[71] Zahradnický, T.: *MOSFET Parameter Extraction*. Dizertační práce, České vysoké učení technické, Praha, 2005. 8, 9

Seznam obrázků

Seznam tabulek

Seznam algoritmů

Seznam použitých zkratek

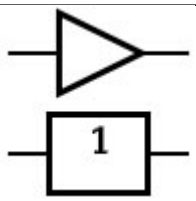
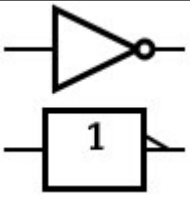
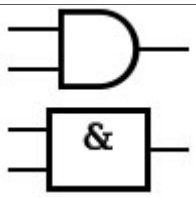
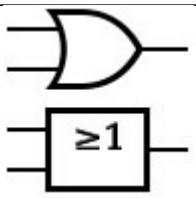
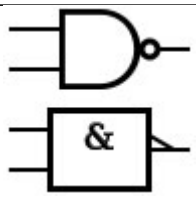
3D-IC	Three-Dimensional Integrate Circuit - obvody implementovány ve více vrstvách, jejichž prvky jsou orientovány jak horizontálně, tak vertikálně
ADK	Asic Design Kit - knihovna logických hradel pro generátor testovacích vektorů
ALU	aritmeticko-logická jednotka
ANSI	American National Standards Institute - americká standadizační organizace
ATPG	Automatic Test Pattern Generator (Generation) - automatický generátor testovacích vektorů nebo automatické generování testovacích vektorů
BJT	Bipolar Junction Tranzistor - bipolární tranzistor
BSIM	Berkeley Short-channel IGFET Model - model tranzistoru MOSFET pro elektronické simulace v SPICE
CANCER	Computer Analysis of Nonlinear Circuits, Excluding Radiation - analogový simulátor elektronických obvodů
CMOS	Complementary Metal-Oxide-Semiconductor - princip návrhu elektronických obvodů složených z komplementárních MOS tranzistorů
DFT	Design For Test - návrh pro snadnou testovatelnost
DL	diodová logika
DTL	diodo-tranzistorová logika
FET	Field-Effect Tranzistor - tranzistor řízený elektrickým polem
GAČR	grantová agentura České republiky
HDL	Hardware Description Language - jazyk pro popis hardware
IEC	International Electrotechnical Commission - mezinárodní elektrotechnická komise
IEEE	Institute of Electrical and Electronics Engineers - mezinárodní nezisková organizace
IGBT	Insulated Gate Bipolar Tranzistor - bipolární tranzistor s izolovaným hradlem

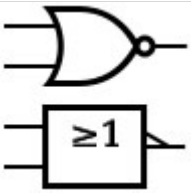
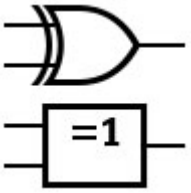
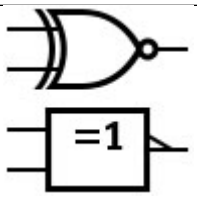
ISCAS	International Symposium on Circuits And Systems - mezinárodní konference o elektronických systémech
JFET	Junction Gate Field-Effect Transistor - tranzistor řízený elektrickým polem s přechodovým hradlem
LSI	Large-Scale Integration - vysoký stupeň integrace
MESFET	Metal Semiconductor Field-Effect Transistor - tranzistor řízený elektrickým polem implementovaný na základě kovu a polovodiče
MIL-STD	Military Standart - standardy amerického ministerstva obrany
MISFET	Metal–Insulator–Semiconductor Field-Effect transistor - tranzistor řízený elektrickým polem implementovaný na základě kovu, izolantu a polovodiče
MOS	Metal-Oxide-Semiconductor - označení způsobu výroby na základě kovu, oxidu a polovodiče
MOSFET	Metal-Oxide–Semiconductor Field-Effect Transistor - tranzistor řízený elektrickým polem implementovaný na základě kovu, oxidu a polovodiče
MSI	Medium-Scale Integration - střední stupeň integrace
NASA	National Aeronautics and Space Administration - americká vládní agentura pro letectví a kosmonautiku
NMOS	princip návrhu elektronických obvodů složených z MOS tranzistorů založených pouze na n nosičích
PODEM	Path-Oriented Decision Making - algoritmus automatického generování testovacích vektorů
RTL	Register transfer level - úroveň meziregistrových přenosů
RTL	rezistor-tranzistorová logika
SCOAP	Sandia Controllability/Observability Analysis Program - algoritmus pro výpočet hodnot řiditelnosti a pozorovatelnosti obvodu.
SoC	System on Chip - obvody obsahující celý systém
SPICE	Simulation Program with Integrated Circuit Emphasis - analogový simulátor elektronických obvodů
SSI	Small-Scale Integration - malý stupeň integrace
TTL	tranzistor-tranzistorová logika
TTM	Time To Market - doba uvedení na trh
ULSI	Ultra-Large-Scale Integration - ultra vysoký stupeň integrace
VHDL	VHSIC Hardware Description Language - VHSIC jazyk pro popis hardware

VHSIC	Very-High-Speed Integrated Circuits - program americké vlády pro vývoj velmi rychlých integrovaných obvodů
VLSI	Very-Large-Scale Integration - velmi vysoký stupeň integrace
WSI	Wafer-Scale Integration - označení obvodů využívajících při výrobě celý plát (wafer)

Příloha A

Základní logické členy

<table>
<tr><th>Název</th><th>Symbol</th><th>Booleovská funkce</th><th>Pravdivostní tabulka</th></tr>
<tr><td>Opakovač (repeater)</td><td> 
</td><td>A</td><td>
<table>
<tr><th>A</th><th>Y</th></tr>
<tr><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td></tr>
</table>
</td></tr>
<tr><td>NOT (invertor)</td><td> 
</td><td>$\overline{A}$</td><td>
<table>
<tr><th>A</th><th>Y</th></tr>
<tr><td>0</td><td>1</td></tr>
<tr><td>1</td><td>0</td></tr>
</table>
</td></tr>
<tr><td>AND</td><td> 
</td><td>$A \cdot B$</td><td>
<table>
<tr><th>A</th><th>B</th><th>Y</th></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>1</td></tr>
</table>
</td></tr>
<tr><td>OR</td><td> 
</td><td>$A + B$</td><td>
<table>
<tr><th>A</th><th>B</th><th>Y</th></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>1</td></tr>
</table>
</td></tr>
<tr><td>NAND</td><td> 
</td><td>$\overline{A \cdot B}$</td><td>
<table>
<tr><th>A</th><th>B</th><th>Y</th></tr>
<tr><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>0</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Název</th><th>Symbol</th><th>Booleovská funkce</th><th>Pravdivostní tabulka</th></tr>
<tr><td>NOR</td><td> 
</td><td>$\overline{A+B}$</td><td><table><tr><th>A</th><th>B</th><th>Y</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table></td></tr>
<tr><td>XOR</td><td> 
</td><td>$A \oplus B$</td><td><table><tr><th>A</th><th>B</th><th>Y</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table></td></tr>
<tr><td>XNOR</td><td> 
</td><td>$\overline{A \oplus B}$</td><td><table><tr><th>A</th><th>B</th><th>Y</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table></td></tr>
</table>

Tabulka A.1: Základní logické členy

Převzato z [31].

Příloha B

Parametry tranzistoru AMI 0.7 μm modelu SPICE level 7

.MODEL namos NMOS LEVEL = 7 ;53

+TNOM = 27	TOX = 1.75E-8	XJ = 2.5E-7
+NCH = 1.7E17	NSUB = 4E16	VTH0 = 0.76
+K1 = 0.8219166	K2 = -8.54312E-3	K3 = 11.1089581
+K3B = -1.9786631	W0 = 1E-6	NLX = 3.751355E-8
+DVT0W = 0	DVT1W = 0	DVT2W = -0.032
+DVT0 = 5.2254747	DVT1 = 0.590721	DVT2 = -0.05
+VBM = -5	U0 = 635.6142994	UA = 1.983902E-9
+UB = 1E-21	UC = 4.667652E-11	VSAT = 9.5E4
+A0 = 0.9331753	AGS = 0.1339124	B0 = 0
+B1 = 0	KETA = -2.746786E-5	A1 = 0
+A2 = 1	RDSW = 1.573286E3	PRWG = 6.719929E-6
+PRWB = -1E-3	WR = 1	WINT = 6.065442E-8
+LINT = 2.87042E-8	DWG = -1.268839E-8	DWB = 1.654199E-8
+VOFF = -0.15	NFACTOR = 0.6887273	CIT = 0
+CDSC = -1E-4	CDSCD = 0	CDSCB = 2E-3
+ETA0 = 0.08	ETAB = -0.07	DSUB = 0.56
+PCLM = 1.0175962	PDIBLC1 = 0.032818	PDIBLC2 = 2.506552E-3
+PDIBLCB = -1E-6	DROUT = 0.6067512	PSCBE1 = 3.356583E8
+PSCBE2 = 5E-5	PVAG = 0.0168906	DELTA = 0.01
+ALPHA0 = 5E-7	BETA0 = 26	RSH = 65
+MOBMOD = 1	PRT = 159.2464225	UTE = -1.9522848
+KT1 = -0.4126334	KT1L = 7.244799E-9	KT2 = 2.671323E-3
+UA1 = 8.353648E-11	UB1 = -2.12098E-19	UC1 = -5.6E-11
+AT = 3.3E4	NQSMOD = 0	WL = 0
+WLN = 1	WW = 0	WWN = 1
+WWL = -5.30182E-20	LL = 0	LLN = 1
+LW = 0	LWN = 1	LWL = 0
+AF = 1	KF = 3E-28	CAPMOD = 2
+CGDO = 4E-10	CGSO = 4E-10	CGBO = 3.35E-10
+CJ = 5E-4	PB = 0.73	MJ = 0.35
+CJSW = 2.8E-10	PBSW = 0.8	MJSW = 0.21
+JS = 1E-03	XPART = 0	ELM = 5

.MODEL pamos PMOS LEVEL = 7 ;53

+TNOM = 27	TOX = 1.75E-8	XJ = 3E-7
+NCH = 1.7E17	NSUB = 4E16	VTH0 = -1.00
+K1 = 0.563991	K2 = 0	K3 = 16.3317811
+K3B = -2.9202228	W0 = 1.23464E-6	NLX = 9.69545E-8
+DVT0W = 0	DVT1W = 0	DVT2W = -0.032
+DVT0 = 3.5648008	DVT1 = 0.3898843	DVT2 = -0.0284121
+VBM = -10	U0 = 235.7724356	UA = 2.964616E-9
+UB = 1.419129E-18	UC = -7.00385E-11	VSAT = 1.1E5
+A0 = 0.4590784	AGS = 0	B0 = 0
+B1 = 1.407805E-9	KETA = -0.047	A1 = 0
+A2 = 1	RDSW = 3E3	PRWG = 2.024978E-3
+RSH = 94	PRWB = 7.428781E-5	WR = 1
+WINT = 10.669321E-8	LINT = 1.9089522E-8	DWG = -1.478082E-8
+DWB = 1.561823E-8	ALPHA0 = 0	BETA0 = 30
+VOFF = -0.1064652	NFACTOR = 0.4324039	CIT = 0
+CDSC = 2.4E-4	CDSCD = 0	CDSCB = 0
+ETA0 = 9.999059E-4	ETAB = -1.999936E-4	DSUB = 0.998946
+PCLM = 2.6025265	PDIBLC1 = 1	PDIBLC2 = 2.853174E-4
+PDIBLCB = 0	DROUT = 0.3837047	PSCBE1 = 4.249266E8
+PSCBE2 = 5E-5	PVAG = 3.8222424	DELTA = 0.01
+MOBMOD = 1	PRT = 216.4347715	UTE = -1.2989809
+KT1 = -0.4521998	KT1L = -2.091783E-8	KT2 = -0.040013
+UA1 = 3.100822E-9	UB1 = -1E-17	UC1 = -8.35439E-11
+AT = 3.289E4	NQSMOD = 0	WL = 0
+WLN = 1	WW = 0	WWN = 1
+WWL = -2.33876E-20	LL = 0	LLN = 1
+LW = 0	LWN = 1	LWL = 0
+CAPMOD = 2	CGDO = 1.0E-10	CGSO = 1.0E-10
+CGBO = 3.35E-10	CJ = 6.0E-4	PB = 0.9
+MJ = 0.51	CJSW = 3.6E-10	MJSW = 0.35
+AF = 1	KF = 5.0E-30	JS = 1E-3
+XPART = 0	ELM = 5	

Příloha C

Počet tranzistorů obsažených v hradlech

and02	6	mux21	10	ao221	12*	oai21	6
and03	8	buf02	4	ao32	12*	oai22	8
and04	10	buf04	6	aoi21	10	oai221	10
nand02	4	buf08	10	aoi22	8	oai222	12
nand03	6	buf12	14	aoi221	10	oai32	10
nand04	8	buf16	20	aoi222	12	oai321	12
nor02	4	inv01	2	aoi32	10	oai322	14
nor03	6	inv02	2	aoi321	12	oai33	12
nor04	8	inv04	4	aoi322	14	oai332	16
or02	6	inv08	8	aoi33	12	oai333	18
or03	8	inv12	12	aoi332	16	oai43	14
or04	10	inv16	16	aoi333	18	oai44	16
fadd1	28	ao21	12*	aoi43	14	xor2	12
hadd1	14	ao22	10*	aoi44	16	xnor2	10

Tabulka C.1: Počet tranzistorů hradel knihovny AMI 0.7 μm

* - odhadovaný počet.

and05	12
and08	18
and09	20
nand05	10
nand08	16
or05	12
nor08	16

Tabulka C.2: Počet tranzistorů hradel přidaných ke knihovně AMI 0.7 μm