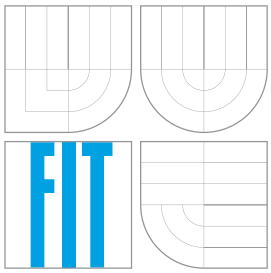
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

AUTOMATICKÉ ŘÍZENÍ VÝPOČTU VE SPECIALIZOVANÉM VÝPOČETNÍM SYSTÉMU

SPECIALIZED COMPUTER SYSTEM AUTOMATIC CONTROL

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. JAN OPÁLKA

VEDOUCÍ PRÁCE
SUPERVISOR

Doc. Ing. JIŘÍ KUNOVSKÝ, CSc.

BRNO 2016

Abstrakt

Práce se zabývá automatizací řízení výpočtů ve specializovaném systému. Nejprve je čtenář seznámen s numerickým řešením diferenciálních rovnic pomocí metody Taylorovy řady a s numerickými integrátory. Praktickým cílem této práce je analýza paralelních vlastností metody Taylorovy řady a specifikace paralelních matematických operací. Je proveden návrh řízení systému pro tyto operace.

Abstract

This work deals with the automatic control of calculations of specialized system. The reader is acquainted with the numerical solution of differential equations by Taylor series method and numerical integrators. The practical aim of this work is to analyze parallel characteristics of Taylor series method, specification of parallel math operations and design of control of this operations.

Klíčová slova

derivace, diferenciální rovnice, Taylorova řada, mikroprocesor, numerická integrace, pevná řádová čárka, pohyblivá řádová čárka, integrátor, řadič, boothův algoritmus, paralelní architektury

Keywords

derivation, diferential equnation, Taylor series, mikroprocessor, numeric integration, fixed point, floating point, integrator, controller, Booth algorithm, parallel architectures

Citace

Jan Opálka: Automatické řízení výpočtu ve specializovaném výpočetním systému, diplomová práce, Brno, FIT VUT v Brně, 2016

Automatické řízení výpočtu ve specializovaném výpočetním systému

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Doc. Ing. Jiřího Kunovského, CSc. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jan Opálka
29. července 2016

Poděkování

Za odborné vedení mé diplomové práce bych chtěl poděkovat svému školiteli Doc. Ing. Jiřímu Kunovskému, CSc., který mi svojí podporou, cennými radami, věcnými připomínkami, náměty a komentáři pomáhal směřovat práci ke zdárnému cíli. Mé poděkování patří také kolegům, rodině a kamarádům, kteří mě po dobu studia jakkoliv podporovali a bez nichž by tato práce nemohla vzniknout.

Obsah

Kapitola 1

Úvod

Tato diplomová práce pojednává o využití numerické integrace pomocí Taylorovy řady. Samotná integrace je řešena třemi variantami numerických integrátorů propojených propojovací sítí. Matematické operace výpočtu rovnic nebo jejich soustav potom mohou být v integrátoru řešeny paralelně.

Metoda Taylorovy řady je jednou z metod, kterou lze provádět numerickou integraci. V kapitolách 2 a 3 je představen výpočet metodou Taylorovy řady pro řešení diferenciálních rovnic za pomoci numerického integrátoru. V kapitole 4 jsou popsány varianty numerických integrátorů.

Možnosti vnitřní reprezentace operandů pro výpočetní operace s pevnou a pohyblivou řádovou čárkou jsou popsány v kapitole 5. Na tuto kapitolu navazuje představení a popis různých možností obecného modelu paralelní architektury a specifikace vybraných paralelních operací pro Taylorovu řadu.

Kapitola 7 obsahuje algoritmy řízení pro specifikované paralelní operace Taylorovy řady a nastiňuje princip změny výpočtů pro pohyblivou řádovou čárku.

Implementaci, simulaci a testování se věnuje kapitola 8. Následně v kapitole 9 je popsána současná realizace a nový návrh propojovací sítě pro numerické integrátory. V poslední kapitole (kapitola 10) je provedena analýza a testování navrženého výpočetního systému.

Kapitola 2

Numerická integrace

Při numerické integraci se využívá postupů, kdy při procesu řešení matematické úlohy, zformulované na základě znalosti problému, lze nalézt řešení úlohy pouze pomocí aritmetických a logických operací. Numerické metody umožňují vyřešit problémy, které nelze řešit analyticky nebo je jejich řešení příliš složité, či časově a ekonomicky náročné. Výsledky výpočtu jsou přibližné. Chyba (nepřesnost) numerického řešení se udává pouze jako (většinou pesimistický) odhad chyby. Odhad chyby je využíván proto, že není nebo nemusí být známo přesné řešení úlohy.

K dispozici je několik možných postupů výpočtu. S ohledem na typ řešené funkce vybereme vhodnou metodu. Každá numerická metoda má své výhody, ale i nevýhody. Popisovaná nepřesnost se vztahuje k použité metodě a jejím vlastnostem. Často lze ovlivnit vhodnou volbou parametrů metody. Někdy je potřeba brát v potaz i chyby vzniklé při matematické formulaci úlohy, které vznikají zanedbáním některých skutečností (používáme zjednodušený matematický model). V současnosti k výpočtům numerické integrace používáme výhradně počítače.

Numerické metody dělíme na dvě hlavní skupiny: jednokrokové a vícekrokové. Základní vlastnosti těchto metod jsou *přesnost* a *rychlost*. Stojí proti sobě a vždy je jedna potlačena na úkor druhé [11].

2.1 Numerické metody pro řešení diferenciálních rovnic

Obecný výpočet diferenciální rovnice *jednokrokovými metodami* se skládá z iterací. Jednokrokové metody využívají hodnotu nalezenou v předchozím kroku t_i pro výpočet následujícího bodu t_{i+1}. Před zahájením výpočtu (rozběh metody) se volí první hodnota tzv. počáteční podmínka. Zmíněné metody jsou vhodné pro řešení obyčejných diferenciálních rovnic prvního řádu a jejich soustav. Pro numerické integrování se parciální diferenciální rovnice a rovnice vyšších řádů musí převést na soustavu obyčejných diferenciálních rovnic prvního řádu.

Pro tento účel slouží např.: *metoda snižování řádu derivace* [10] nebo *metoda postupné integrace*. Jednoduší ze zmíněných je metoda snižování řádu derivace. Podmínkou pro její použití je absence derivace vstupů na pravé straně řešené diferenciální rovnice. Metoda postupné integrace je vhodná pro rovnice s derivacemi vstupů na pravé straně. Pracuje ve třech po sobě následujících etapách. Osamostatní nejvyšší řád derivace. Postupně integruje rovnici a zavádí nové stavové proměnné. Nakonec vypočte nové počáteční podmínky.

Vícekrokové metody k výpočtu aktuálního řešení používají k předcházejících výsledků

(kroků). Mezi vícekrokové metody patří např. Adams–Bashforth nebo metody typu prediktor–korektor. Problémem většiny těchto metod je rozběh samotné integrace. V tomto bodě je vyžadován pro výpočet následujícího členu k předcházejících výsledků. Pro zjištění požadovaného počtu předchozích výsledků se využije některá z metod jednokrokových.

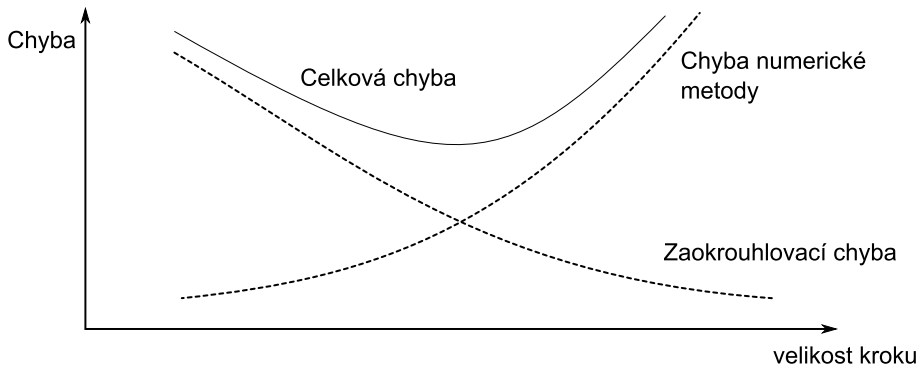
Mezi speciální metody numerické integrace se řadí metody pro rovnice druhého řádu, metody založené na užití vyšších derivací a další.

V následujícím textu popíši nejpoužívanější metody pro řešení diferenciálních rovnic (viz [2], [5], [6]).

2.1.1 Vlastnosti integračních metod

Nejdůležitější vlastností numerických integračních metod je přesnost jejich výpočtů. Při numerických výpočtech se uvažuje velikost chyby řešení, která ovlivňuje výsledek. Velmi důležité vlastnosti jsou i stabilita numerického řešení, konvergence řešení a rychlost metody. Z pragmatického hlediska nesmíme opomíjet i složitost programové implementace numerických metod a také požadovaný počet výchozích kroků pro začátek řešení (více v [1], [8]).

Na chybu numerické metody působí několik faktorů. Prvním z nich je lokální chyba, kterou produkuje každý krok výpočtu. Lokální chyba je tvořena dvěma částmi – zaokrouhlovací chybou (odvíjí se od přesnosti hardwarových komponent počítače a způsobu zaokrouhlování) a chybou numerické metody (určena použitým řádem Taylorovy řady – čím vyšší počet členů Taylorovy řady použijeme, tím vyšší přesnosti výsledku dostaneme). Lokální chybu je možné ovlivnit velikostí kroku výpočtu numerické integrace. Dalším negativní vliv působící na velikost celkové chyby je její akumulace – k té dochází sčítáním chyb jednotlivých kroků.



Obrázek 2.1: Celková chyba numerického výpočtu a její složky

Stabilita numerické metody vyjadřuje vztah mezi chováním metody a délkou integračního kroku. Metoda je stabilní, pokud malé změny vstupních dat ovlivní výsledek jen málo. Špatnou volbou kroku můžeme způsobit skokovou změnu chování metody. Řešení se stane nestabilní a dostáváme zcela chybné výsledky.

2.1.2 Taylorova řada

Taylorova řada je považována za základní jednokrokovou integrační metodu. Taylorova řada funkce y pro bod $i + 1$ vypadá následovně:

$$y_{i+1} = y_i + hy_i' + \frac{h^2}{2!}y_i'' + \frac{h^3}{3!}y_i''' + \cdots + \frac{h^n}{n!}y_i^{(n)} \tag{2.1}$$

Písmenem h označujeme integrační krok. U této metody je možné určit požadovanou přesnost výsledku. Výpočet konkrétní hodnoty y_{i+1} je iteračně prováděn do doby, než je dosažena požadovaná přesnost. Ta se vyhodnocuje z relativního rozdílu dvou následujících členů Taylorovy řady. Nevýhodou použití Taylorovy řady je nutnost vyčíslování vyšších derivací.

Z Taylorovy řady se odvozují některé další integrační metody, přičemž se využívá možnosti vynechat některé členy Taylorova polynomu pro přibližné vyjádření výsledku.

2.1.3 Eulerova metoda

Eulerova metoda vychází z Taylorovy řady. Využívá pouze její první dva členy. Je to nejjednodušší jednokroková metoda pro numerické řešení obyčejných diferenciálních rovnic s počátečními podmínkami.

$$y_{i+1} = y_i + hy_i' \tag{2.2}$$

Pokud vhodně zvolíme integrační krok h, lze při použití této metody dostat relativně přesné výsledky. Čím menší je integrační krok, tím přesnější bude výsledek. Hlavní předností této metody je vynechání výpočtu vyšších derivací Taylorovy řady – jedná se o metodu 1. řádu.

2.1.4 Runge Kutta 4. řádu

Potřebujeme-li vyšší přesnost, než jakou poskytuje Eulerova metoda, můžeme využít některou z metod Runge–Kutta. Tyto metody patří do jednokrokových metod. Pro výpočet y_{i+1} je nutné použít pomocné mezivýpočty. Výsledný přírůstek najdeme jako váhový průměr vypočtených hodnot. Počet pomocných mezivýsledků udává řád metody. Nevýhodou použití této metody je pracnější postup a delší doba trvání výsledku než u Eulerovy metody. V tomto případě neplatí protichůdnost čím pomalejší výpočet tím přesnější výsledek. Při porovnání s metodami prediktor–korektor mají metody Runge–Kutta malou rychlost. Další nevýhodou metod Runge–Kutta je obtížný odhad chyby výpočtu. Velkým přínosem metod Runge–Kutta je využít jejich potenciálu pro získání počátečních hodnot pro vícekrokové metody. Obecný tvar jednokrokových metod Runge–Kutta je:

$$y_{i+1} = y_i + h(\omega_1 k_1 + \cdots + \omega_s k_s) \tag{2.3}$$

$$k_1 = f(t_i, y_i)$$

$$k_1 = f(t_i + \alpha_l h, y_i + h\sum_{j=1}^{l-1}\beta_{lj}k_j), \qquad i = 2, \cdots, s$$

Konstanty ω_n, α_n a β_{lj} se volí podle řádu metody tak, aby získané řešení souhlasilo s Taylorovou řadou v bodě t_{i+1}. Nejčastěji se používá Runge–Kutta 4. řádu. Obecný tvar používaného vzorce je následující:

$$y_{i+1} = y_i + \frac{1}{6}h(k_1 + 2k_2 + 2k_3 + k4) \tag{2.4}$$

$$k_1 = f(t_i, y_i)$$

$$k_2 = f(t_{i+1} + \frac{1}{2}h, y_i + \frac{1}{2}hk_1)$$

$$k_3 = f(t_{i+1} + \frac{1}{2}h, y_i + \frac{1}{2}hk_2)$$

$$k_4 = f(t_{i+1} + h, y_i + hk_3)$$

Podorobné informace o numerické integraci lze nalézt v publikacích věnující se numerickým řešením diferenciálních rovnic ([1], [8]).

Kapitola 3

Metoda Taylorovy řady

Nutnost analytického počítání vyšších derivací pro členy Taylorovy řady bylo považováno v mnoha případech za velmi komplikované. V literatuře o moderní numerické a aplikované matematice se uvádí, že Taylorova řada je z teoretického a koncepčního pohledu velice zajímavá. Použitelnost je ovšem omezena pouze na určité speciální případy. Předkládaná práce se zabývá běžnými „technickými problémy“, pro které lze Taylorovu řadu využít.

Při výpočtu se automaticky nastaví řád metody. To omezí počet použitých členů Taylorovy řady na potřebné množství k tomu tak, aby byla dosažena požadovaná přesnost výsledku. Metoda Taylorovy řady má velice příznivé paralelní vlastnosti. Většina výpočetních operací je vzájemně nezávislá, takže mohou být prováděny paralelně v oddělených procesorech (násobení, dělení, sčítání, odčítání). Důležitou součástí metody Taylorovy řady je automatická transformace zadání na polynomiální tvar. Zadaná nelineární diferenciální rovnice se automaticky transformuje do tvaru, ze kterého lze snadno rekurentně vypočítat jednotlivé členy Taylorovy řady [11].

Z analýz provedených v disertační práci Michala Krause [12] vyplynulo, že při použití stejných řádů pro metody Runge-Kutta a Taylorovy řady, Taylorova řada je výpočetně méně náročná (obsahuje menší počet operací). Dokonce čím vyšší řád metod je použit, tím se stává rozdíl ještě výraznější. Při použití stejného počtu operací Taylorova řada dosahuje vyšší přesnost.

3.1 Jednoduchá diferenciální rovnice - transformace zadání

Tato podkapitola popisuje aplikaci metody Taylorovy řady 2.1 na jednoduchou obyčejnou diferenciální rovnici prvního řádu. Aplikace metody transformujeme zadanou diferenciální rovnici do tvaru použitelného pro blokový výpočet.

$$y^{'} = y_i \qquad y(0) = y_0 \tag{3.1}$$

Rozvinutím Taylorovy řady pro libovolné y_{i+1} získáme:

$$y_{i+1} = y_i + hy_i' + \frac{h^2}{2!}y_i'' + \frac{h^3}{3!}y_i''' + \cdots + \frac{h^p}{p!}y_i^{(p)} \tag{3.2}$$

Dle formule 3.1 ale platí, že $y^{'} = y$ a proto:

$$y = y^{'} = y^{(2)} = y^{(3)} = y^{(4)} = \cdots = y^{(p)} \tag{3.3}$$

Je tedy možné psát přímo:

$$y_{i+1} = y_i + hy_i + \frac{h^2}{2!}y_i + \frac{h^3}{3!}y_i + \cdots + \frac{h^p}{p!}y_i \tag{3.4}$$

Řešení je možné přepsat tímto způsobem:

$$y_{i+1} = y_i + DY1_i + DY2_i + DY3_i + DY4_i + DY5_i + \cdots + DYp_i \tag{3.5}$$

kde význam jednotlivých členů je:

$$DY1_i = hy_i \tag{3.6}$$

$$DY2_i = \frac{h}{2}DY1_i \tag{3.7}$$

$$DY3_i = \frac{h}{3}DY2_i \tag{3.8}$$

$$DY4_i = \frac{h}{4}DY3_i \tag{3.9}$$

$$DY5_i = \frac{h}{5}DY4_i \tag{3.10}$$

$$\vdots$$

$$DYp_i = \frac{h}{p}DY(p-1)_i \tag{3.11}$$

Rovnice představuje základní předpis pro vyřešení problému integrace pomocí Taylorovy řady (pro zadanou homogenní diferenciální rovnici). Zmíněná rovnice nastiňuje první možný koncept pro návrh elementárního integračního procesoru (řešeno v diplomové práci [11]). Výsledkem je výpočetní blok, který je schopen zpracovávat nejjednodušší matematické operace (sčítání, odčítání a násobení). Tento blok bude v následujících kapitolách rozšířen o operaci dělení. Všechny operace budou jak v pevné řádové čárce, tak i v plovoucí řádové čárce.

3.2 Soustava diferenciálních rovnic - transformace zadání

Transformací rovnice 3.12.

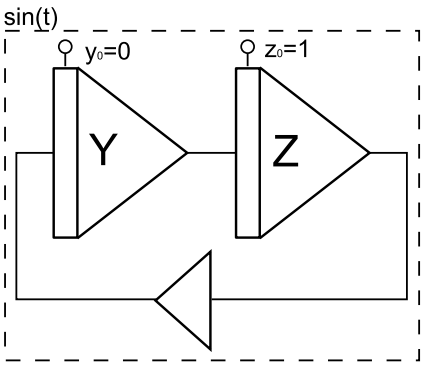
$$y^{'} = sin(t) \qquad y(0) = y_0 \tag{3.12}$$

získáme následujicí systém soustavy rovnic 3.13 - 3.14

$$y^{'} = z \qquad y(0) = 0 \tag{3.13}$$

$$z^{'} = -y \qquad z(0) = 1 \tag{3.14}$$

Tuto soustavu lze blokově vyjádřit jako invertor a dva integrátory. Blokové schéma je naznačené na následujícím obrázku. Invertor převrací polaritu vstupního signálu. Integrátor provádí integraci vstupu. Získaná soustava rovnic 3.13 – 3.14 popisuje goniometrickou funkci sin.



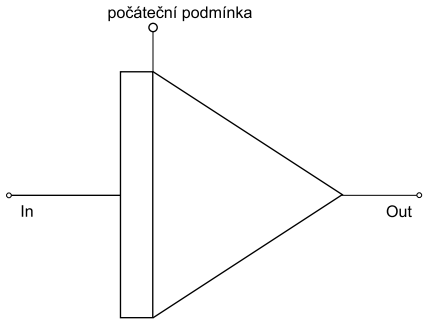
Obrázek 3.1: Řešená soustava rovnic

Pro vstup dalších částí budeme uvažovat již transformované rovnice do tvaru vhodného pro metodu Taylorovy řady.

Kapitola 4

Numerický integrátor a jeho varianty

Numerický integrátor je výpočetní obvod, který provádí integraci v číslicovém prostředí. Vstupem tohoto obvodu jsou počáteční podmínky a hodnoty vstupních proměnných. Známe dva základní typy numerických integrátorů – invertující a neinvertující. Pokud integrátor provádí inverzi znaménka výsledku označuje se jako invertující, jinak je numerický integrátor neinvertující. Blokové schéma numerického integrátoru je na obrázku 4.1. Tohle schéma je často použito pro grafický návrh blokového řešení diferenciálních rovnic.



Obrázek 4.1: Blokové schéma integrátoru

Struktura numerických integrátorů je vytvořena z výpočetních bloků, které provádí základní matematické a logické operace. Matematické operace zastoupené v blokové struktuře integrátorů odpovídají svojí funkcí sčítačkám, násobičkám, odčítačkám nebo děličkám. Dalšími součástmi jsou obvody podpůrného charakteru, jako například multiplexory přepínající kontext, registry tvořící paměť integrátoru, obvod pro řízení negace nebo blokovací obvod.

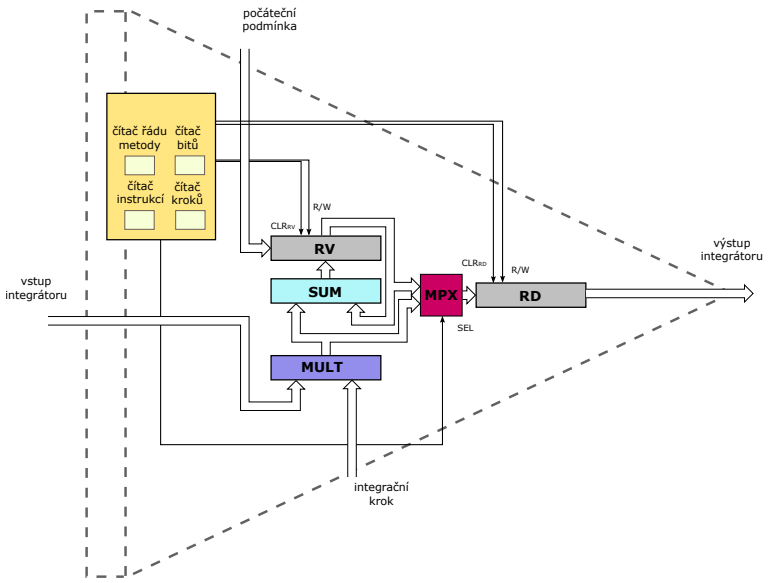
Lze říct, že uvažovaný numerický integrátor je funkčně totožný s mikroprocesorem. Výpočetním základem integrátoru je aritmeticko-logická jednotka (ALU). ALU zodpovídá za provedení numerické integrace. K výpočtu rozsáhlých soustav diferenciálních rovnic mohou být propojeny stovky až tisíce artmeticko-logických jednotek. ALU je koncipována jako specializovaná jednoúčelová jednotka, provádějící základní matematické operace. Zadaný problém je převeden na soustavu homogenních lineárních diferenciálních rovnic s konstantními koeficienty, jak je tomu popsáno v kapitole 3 věnující se metodě Taylorovy řady. Při numerickém výpočtu těchto diferenciálních rovnic Taylorovou řadou se používají dvě základní operace: **násobení** a **sčítání**. Obě z těchto zmíněných operací mohou být prováděny sériově

nebo paralelně. Obdobně se může komunikace mezi procesory řešit sériově nebo paralelně. Z toho vzniklo následující dělení integrátorů:

- **paralelně-paralelní integrátory (PPI)** (paralelní komunikace, paralelní výpočet)
- **sériově-paralelní integrátory (SPI)** (sériová komunikace, paralelní výpočet)
- **sériově-sériové integrátory (SSI)** (sériová komunikace, sériová výpočet)

4.1 Paralelně-paralelní integrátor

U tohoto typu integrátoru je násobení realizováno paralelní násobičkou a sčítání paralelní sčítačkou. Vnitřní bloková struktura je zobrazena na následujícím obrázku.



Obrázek 4.2: Blokové schéma paralelního integrátoru s řadičem

RV	registr výsledku
RD	registr součinu
MPX	multiplexor
SUM	paralelní sčítačka
MULT	paralelní násobička

Paralelně-paralelní integrátor pracuje následovně:

1. Zahájí cyklus tím, že vloží do registrů RD a RV hodnotu y_i.
2. Na vstupu integrátoru se objeví hodnota $f(y_i)$, což je výstupní hodnota prvku připojeného na vstup integrátoru. Je nastaven integrační krok na hodnotu h.
3. Vzniklý součin (z násobičky MULT) integračního kroku h a vstupní hodnoty integrátoru y_i se zapíše do registru RD a současně přičte k registru RV.
4. V RD se nachází hodnota $DY1$ a v RV je mezivýpočet $y_i + DY1$.

5. V následujícím kroku se na vstupu integrátoru objeví $f(DY1)$ a integrační krok $h/2$.

6. Jejich vynásobením vznikne další člen Taylorova polynomu $DY2$. Člen $DY2$ se přičte k RV a vloží se do RD. Celý cyklus se opakuje tak dlouho, dokud není dosaženo požadované přesnosti, příp. maximálního počtu iterací.

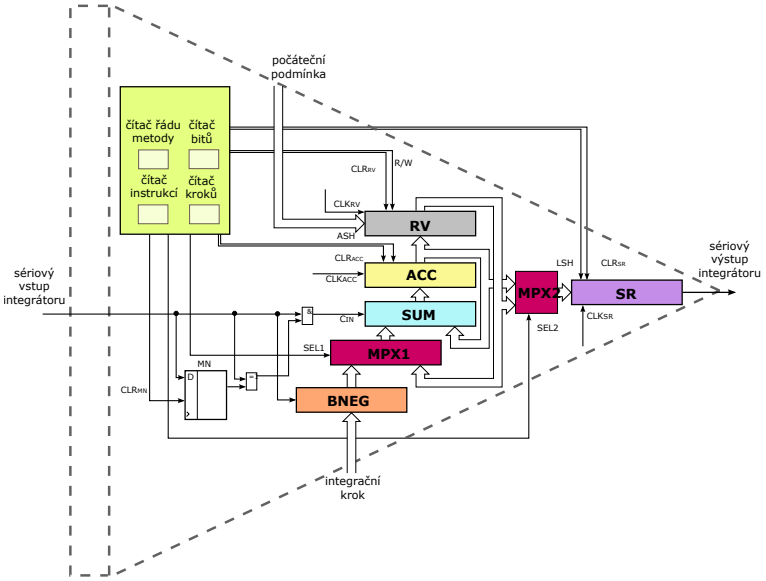
Paralelně-paralelní integrátor je nejrychlejší. Čas pro výpočet jednoho členu Taylorovy řady je:

$$t_{PP} = \tau_{nas} + \tau_{sec} + \tau_{sit} \tag{4.1}$$

tedy je dán součtem času násobení τ_{nas}, sčítání τ_{sec} a případným zpožděním signálů v propojovací síti τ_{sit}. Rychlost tohoto typu integrátoru je vykoupena složitostí zapojení, kterou nejvíce ovlivňuje kombinační násobička. Z toho plyne i větší prostorová složitost této implementace. Nepříznivým kritériem je i počet vstupů a výstupů, který je přímo úměrný šířce paralelní datové sběrnice.

4.2 Sériově-paralelní integrátor

Sériově-paralelní varianta provádí sekvenční násobení s aplikací Boothova algoritmu. Sčítání je provedeno paralelně.



Obrázek 4.3: Blokové schéma sériově-paralelního integrátoru s řadičem

RV	registr výsledku
MPX	multiplexor
SUM	paralelní sčítačka
ACC	akumulátor
SR	posuvný registr
BNEG	obvod řízené negace (pro sekvenční násobení)

Sériově-paralelní integrátor využívá následující princip:

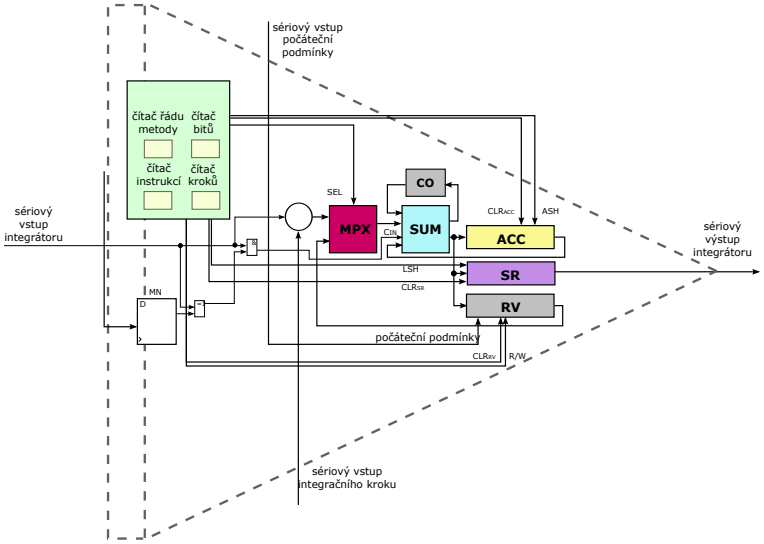
1. Nejprve se vynuluje akumulátor (ACC) a obvod malé nuly (MN). Do posuvného registru (SR) a registru výsledku (RV) je uložena počáteční hodnota y_0. Vstup integračního kroku má hodnotu h.
2. Multiplexor (MPX) je přepnut na cestu od bloku řízení negace (BNEG).
3. Výpočet y_{i+1} probíhá v těchto krocích:
   (a) Na vstupu sério-paralelního integrátoru je připraven nejméně významový bit $f(y_i)$. Tento bit určí, zda se bude násobenec z registru násobence (RN) přičítat nebo odečítat od obsahu akumulátoru (odečítání se provádí náhradou násobence za dvojkový doplněk), nebo zda se přičte nula (násobenec se ignoruje).
   (b) Výsledek ze sčítačky se vloží do akumulátoru.
   (c) Akumulátor a posuvný registr (SR) se následně posune o jedno místo doprava. Tento cyklus se opakuje při zpracování všech bitů z $f(y_i)$. Tím získáme součin integračního krok h a $f(y_i)$.
4. Výsledek násobení se uloží do posuvného registru SR.
5. Multiplexor se přepne na cestu od registru výsledku RV a výsledek mezikroku DYp uložený v akumulátoru se sečte s hodnotou nacházející se v registru výsledku RV.
6. Výsledek sčítání se zapíše do ACC a následně do RV.
7. Celý výpočet se opakuje do té doby, dokud se nedosáhne požadované přesnosti nebo maximálního počtu iterací. Po popsaném ukončení výpočtu získáme y_{i+1}.

Výhodou tohoto přístupu je nízký počet vývodů pro rozhraní zapojení, protože data vstupují i vystupují z integrátoru sériově. Zvýšení přesnosti výpočtu zvětšením počtu bitů operandů nijak nezmění celkové zapojení na rozdíl od předchozí varianty. Změna nastane pouze u šířky registrů, sčítačky a rozšíří se posloupnost řídících signálů z řadiče. Rozhraní integrátoru ani propojovací síť nebudou zvětšením počtu bitů nijak ovlivněny.

Nevýhodou oproti předchozí variantě je delší doba výpočtu z důvodu sekvenčního násobení (násobí se v n krocích, n - počet bitů operandů). Výpočet jednoho členu Taylorovy řady je:

$$\begin{aligned} t_{SP} &= \tau_{nas} + \tau_{sec} + \tau_{sit} \\ t_{SP} &= n{\cdot}\tau_{sec} + \tau_{sec} + \tau_{sit} \end{aligned} \tag{4.2}$$

Tedy je dán součtem času násobení (n krát sčítání $n * \tau_{sec}$), přičtení výsledku násobení k celkovému výsledku τ_{sec} a zpožděním signálů v propojovací síti τ_{sit}. Možná optimalizace této varianty integrátoru je v použití Boothova překódování s radixem 4 a vyšším. Tím se sníží počet kroků potřebných pro násobení. Nevystačíme si jen s kladnou a zápornou hodnotou násobence, ale tato optimalizace vyžaduje také vytváření překódování dvou případně více bitů najednou, což povede k vyšší složitosti zapojení integrátoru.



Obrázek 4.4: Blokové schéma sériového integrátoru s řadičem

4.3 Sériově-sériový integrátor

Sériově sériová varianta integrátoru vychází z předchozího sériově-paralelního integrátoru. Liší se od předchozí varianty tím, že provádí násobení i sčítání sekvenčně.

SUM úplná jednobitová sčítačka
CO klopný obvod pro uchování přenosu
MPX multiplexor
ACC akumulátor
RV posuvný registr výsledku
SR výstupní posuvný registr

Sériový integrátor funguje následovně:

1. Výpočet je zahájen načtením y_i do registru výsledku (RV). Na vstup integrátoru je přivedena hodnota $f(y_i)$. Integrační krok je nastaven na hodnotu h.
2. Vynuluje se akumulátor (ACC) a obvod přenosu CO.
3. Multiplexor se přepne na cestu z RV.
4. Výpočet y_{i+1} probíhá v těchto krocích:
   (a) Na vstupu integrátoru se objeví nejméně významný bit $f(y_i)$. Pomocí sériového vstupu integračního kroku se postupně od nejméně významného bitu objevují jednotlivé bity integračního kroku. Tato posloupnost se v závislosti na hodnotě vstupu integrátoru sériově přičítá k akumulátoru.
   (b) Zpracováním všech bitů integračního kroku se vzniklá hodnota mezivýsledku v závislosti na hodnotě vstupu integrátoru přičte k akumulátoru.
   (c) Po té se na vstupu objeví významnější bit $f(y_i)$ a současně s tím se posune posuvný registr (SR).

5. Zmíněný postup se opakuje do té doby, než se zpracuje ze vstupu nejvýznamnější (poslední) bit $f(y_i)$.

6. Jakmile se ukončí násobení, hodnota z akumulátoru se přepíše do registru SR a sériově se přičte k hodnotě uchované v registru výsledku RV.

Sériově-sériová varianta integrátoru má nejmenší nároky na zapojení ze zmíněných variant. Dochází u ní ovšem k extrémnímu nárůstu cyklů potřebných pro kompletní výpočet.

$$\begin{aligned} t_{SS} &= \tau_{nas} + \tau_{sec} + \tau_{sit} \\ t_{SS} &= n{\cdot}n{\cdot}\tau_{sec} + n{\cdot}\tau_{sec} + \tau_{sit} \\ t_{SS} &= n^2{\cdot}\tau_{sec} + n{\cdot}\tau_{sec} + \tau_{sit} \end{aligned} \tag{4.3}$$

Časová náročnost pro zisk výsledku u SSI je kvadratická v závislosti na počtu bitů násobitele a násobence (integračního kroku a $f(y_i)$). τ_{sec} představuje čas sčítání na jednobitové sčítačce.

4.4 Řízení integrátorů

Integrátory (aritmeticko-logické jednotky) jsou napojeny na řídící obvody (řadiče). Řadiče obsahují mikroprogram, který umožní nahrání počátečních podmínek a integračního kroku h pro jednotlivé integrátory. Také slouží pro poskytnutí výsledku na výstupu při dokončení výpočtu. Mikroprogram dále určuje přesný sled operací, kterým se získá požadovaně přesný výsledek.

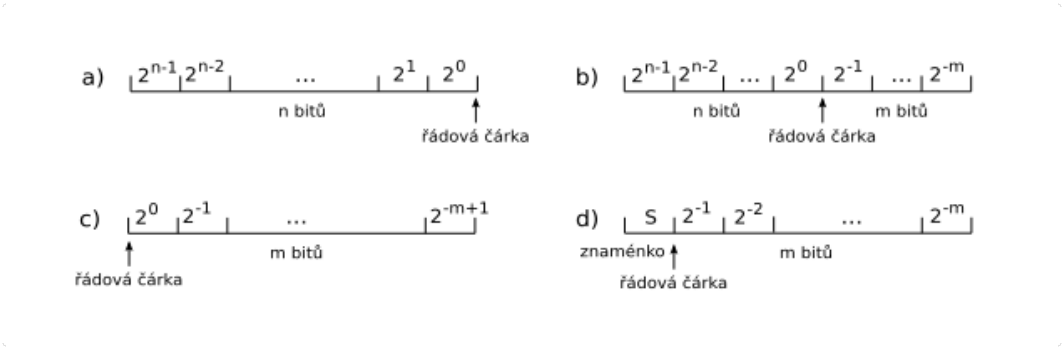
Kapitola 5

Reprezentace operandů

Pro představu o matematické operaci a následném návrhu blokového schématu je nutné mít definován formát operandů, který využíváme pro vnitřní reprezentaci dat používaných v počítačích. Jsou známy dva základní typy formátů – čísla s pevnou a pohyblivou řádovou čárkou.

5.1 Čísla s pevnou řádovou čárkou

Mezi nejčastěji používané formáty pro čísla s pevnou řádovou čárkou patří přímý kód, inverzní a doplňkový kód a kód s lichým případně sudým posunutím. Pro matematické operace v pevné řádové čárce je výhodné použití doplňkového kódu. Na následujícím obrázku je přehled nejčastěji používaných formátů pro pevnou řádovou čárku.



Obrázek 5.1: Formáty čísla s pevnou řádovou čárkou [12]

5.2 Čísla s pohyblivou řádovou čárkou

Čísla s pohyblivou řádovou čárkou (floating point) jsou ve tvaru $M * Z^E$. Číslo se přesně specifikuje počtem bitů, kódem a tvarem mantisy M, základem Z, počtem bitů a kódem exponentu E. Základ Z určuje rozsah zobrazitelných čísel. Pokud zvýšíme základ Z, rozsah zobrazitelnosti se zvětší, ale stává se řidším a přesnost zobrazení klesá. Jedno a totéž číslo je možné zapsat různými způsoby. Vhodnější pro automatické zpracování je jednoznačný zápis, a proto vznikl normalizovaný tvar mantisy. Čísla kromě okrajových a speciálních

hodnot se proto normalizuji na tento tvar. Jedním z nejrozšířenějších formátů zobrazení čísel s pohyblivou řádovou čárkou je standard IEEE 754.

5.3 Standard IEEE 754

Ve standardu IEEE 754 se definuje základ $Z = 2$ s jednoduchou a dvojitou přesnost. Jednoduchá přesnost znamená, že číslo je uloženo na 32 bitech, při dvojité je uloženo na 64 bitech. Mimo definici základu Z, mantisy M a exponentu E definuje standard několik výjimečných hodnot čísla. První je nečíselný výsledek NaN (Not a Number). Další je definice nekonečna a to hned ve dvou variantách – kladné a záporné nekonečno. Stejně tak nula je v tomto standardu brána jako speciální hodnota a má svoji zápornou a kladnou verzi. Poslední speciální hodnotou jsou tzv. denormalizovaná čísla, to jsou čísla, která neodpovídají normalizované podobě standardu IEEE 754.

Formát čísla s pohyblivou řádovou čárkou podle standardu IEEE 754 v jednoduché přesnosti (32 bitů):

S	**E**	**E**	**E**	**E**	**E**	**E**	**E**	**M**	**M**	**M**	**M**	**M**	**M**	**M**	**M**	**M**	**M**	**M**	**M**	**M**	**M**	**M**	**M**	**M**
31	30	.	.	.			23	22									.	.	.					0

Nejvíce významový bit reprezentuje znaménko (0 pro kladné, 1 pro záporné). Dalších 8 bitů je vyhrazeno pro exponent a posledních 23 bitů patří mantise. Mantisa je vyjádřena přímým kódem bez znaménka, exponent kódem s lichým posunutím. V poli exponentu je zadáno číslo zvětšeně o hodnotu Bias (127). Normalizovaný tvar čísla standardu IEE 754 předpokládá hodnotu 1 na pozici 2^0, matice pak obsahuje hodnotu za desetinou čárkou. Denormalizovaná čísla se zmíněnou 1 na úrovni 2^0 nejsou. Převodu čísla z tohoto formátu je dosáhnuto tímto vzorcem:

$$X = (-1)^S * 2^{E-Bias} * (1, M)$$

Kapitola 6

Paralelní výpočetní operace pomocí metody Taylorovy řady

6.1 Paralelní architektury

Paralelní systém je definován jako množina počítačů/procesorů/jader. Tyto výpočetní jednotky komunikují a kooperují s cílem rychle vyřešit rozsáhlé problémy. Paralelizace problému rozděluje složitější úlohy na jednoduší. Úsilím paralelizace je především snížení celkové doby výpočtu současným zpracováním většího množství dat.

Pro paralelní systémy se vytváří specifické programy, které se dělí na větve (procesy). Každá větev se může zpracovávat odděleně. Souběžné procesy mezi sebou mohou komunikovat. Komunikace probíhá často skrze paměť a je závislá na architektuře paralelního systému. Známe dvě hlavní třídy těchto systémů - se sdílenou nebo distribuovanou pamětí.

V systémech se **sdílenou pamětí** sdílejí všechny výpočetní jednotky stejný adresový prostor. Komunikace probíhá tak, že procesy zapisují/čtou data do sdílené paměti. U tohoto přístupu je nutná synchronizace čtení/zápisu u sdílené paměti.

Systémy s **distribuovanou pamětí** poskytují každému procesu jeho vlastní paměť. Procesy jsou propojeny propojovací sítí a vyměňují si mezi sebou data. Komunikace u tohoto přístupu probíhá pomocí rozesílání zpráv.

Aktuální trend se snaží umístit větší počet jader (typicky 2, 4, 8 a více, případně u specializovaných čipů i několik tisíc - grafické karty) na jeden čip. Větší počet jader se volí proto, že můžeme zvýšit výkonost čipu při zachování nebo i snížení jeho spotřeby. Více čipů méně výkonných, které jsou zároveň méně náročné na spotřebu, disponuje větší výpočetní silou. Technologicky je nevýhodné navyšovat výkon jader nad určitou hranici, protože nepřiměřeně tomu roste i jejich příkon.

6.1.1 Úrovně paralelismu

Paralelismus rozeznáváme na několika úrovních. Existují minimálně čtyři úrovně paralelismu.

- **uvnitř instrukcí** – nejjemnější typ paralelismu nacházející se na úrovni bitů, příkladem použití jsou 8, 16, 32, 64 bitové procesory
- **mezi instrukcemi** – některé instrukce se provádí paralelně, nebývá to pozorovatelné na úrovni programovacího jazyka (typicky u zřetězeného zpracování, zapojení více funkčních jednotek téhož typu)

- **mezi vlákny** – procesy paralelního systému se rozdělují na výpočetní vlákna, jednoprocesorové střídají běh vláken postupně (případně s optimalizací) na jednom procesoru, u víceprocesorových systémů běží vlákna na všech jádrech současně, vlákna využívají sdílenou paměť
- **mezi procesy** – paralelní program rozložený na množinu procesů, které se vykonávají na jednotlivých procesorech víceprocesorového systému. Tyto paralelní procesy spolupracují na vyřešení úlohy. Každá z procesů disponuje vlastní pamětí, kterou nesdílí s jinými procesy

U paralelizmu na úrovni instrukcí a mezi instrukcemi nemusí mít programátor žádné větší znalosti o dané architektuře paralelního systému, ani nemusí použít specializovaný přístup a konstrukce používané u paralelního programování. O paralelismus na této úrovni se postará kompilátor. Pokud chce programátor využít paralelismus na úrovni vláken nebo procesů, je třeba zohlednit paralelní konstrukci a musí počítat s režií spojenou s paralelním přístupem (vytváření vláken, přepínání kontextů vláken), uvažovat úskalí práce s pamětí v paralelním prostředí (sdílení pamětí), komunikaci a synchronizaci.

Další podrobnosti o architektuře procesorů, využití popisovaných metod paralelního zpracování (typy instrukcí, zřetězené zpracování, vícevláknový provoz) lze nalézt v [14].

6.1.2 Flynnova klasifikace

Víceprocesorové počítačové systémy můžeme klasifikovat z několika hledisek. Častá klasifikace těchto systémů je dělení podle uspořádání operační paměti. Tato klasifikace se nazývá Flynnova. Její autor jako jeden z prvních provedl základní rozdělení podle počtu instrukčních a datových proudů, které se zpracovávají v jednom okamžiku na dané architektuře.

- **SISD** (Single Instruction Single Data) – Nejde o paralelní architekturu. Jeden procesor provádí sekvenční posloupnost instrukcí nad daty uloženými v paměti.
- **SIMD** (Single Instruction Multiple Data) – Několik procesorů se společným řízením pracuje paralelně nad různými daty. Komunikace probíhá pomocí zasílání zpráv nebo přes sdílenou paměť. Díky existenci jedné řídící jednotky jsou procesory implicitně sesynchronizovány. Není tak univerzální jako následující varianty.
- **MISD** (Multiple Instruction Single Data) – Stejná data se zpracovávají v několika jednotkách. Tento typ architektur je využíván jen velmi zřídka.
- **MIMD** (Multiple Instruction Multiple Data) – Paralelní nezávisle řízené procesory, které pracují nad různými daty. Procesory mezi sebou mohou komunikovat pomocí zasílání zpráv nebo přes sdílenou paměť. Univerzálnost této architektury je největší ze zmíněných. Většina dnešních moderních paralelních architektur patří právě do této třídy.

6.1.3 Režie paralelního zpracování

Výpočetní operace v paralelních systémech provádí užitečnou práci stejně tak jako u sekvenčního přístupu. Ovšem u paralelního přístupu je nutné uvažovat režii, která je tvořena synchronizací, správou a interakcí procesů. Čekání na jiné procesory, neboli synchronizace, než dokončí svojí činnost, je nutná z důvodu nerovnoměrného využití procesorů. Správa procesů řídí přepínání a konzistenci přepínaných kontextů. Interakce procesů zahrnuje zejména komunikaci procesů a vláken mezi sebou.

6.2 Paralelní operace pro Taylorovu řadu

Poznatky z kapitoly o Taylorově řadě použijeme k výpočtu šesti různých paralelně spustitelných operací pomocí metody Taylorovy řady. Jedná se o operace sčítání, odčítání, součin, podíl, součin s integrací a podíl s integrací. V následujících podkapitolách budou znázorněny jednotlivé operace.

6.2.1 Součet

Předpokládejme jednoduchou rovnici součtu:

$$y = u + v \tag{6.1}$$

Její první derivace je:

$$y^{'} = u^{'} + v^{'} \qquad y(0) = y_0 \quad u(0) = u_0 \quad v(0) = v_0 \tag{6.2}$$

Taylorovu řadu můžeme vyjádřit jako součet jednotlivých členů dle rovnice 3.5:

$$y_{i+1} = y_i + DY1_i + DY2_i + DY3_i + \cdots + DYp_i \tag{6.3}$$

První člen Taylorovy řady lze vyjádřit následovně:

$$DY1 = h \cdot (u^{'} + v^{'}) \qquad DU1 = h \cdot u^{'} \qquad DV1 = h \cdot v^{'} \tag{6.4}$$

Jednoduchou úpravou členů $DU1$ a $DU2$ získáme vztahy 6.5, kterými dosadíme do rovnice pro první člen Taylorovy řady:

$$u^{'} = \frac{DU1}{h} \qquad v^{'} = \frac{DV1}{h} \tag{6.5}$$

Po úpravě (vykrácení kroku h) vznikne:

$$DY1 = DU1 + DV1 \tag{6.6}$$

Obdobným způsobem vznikají další členy:

$$DY2 = \frac{h^2}{2!} \cdot (u^{''} + v^{''}) \qquad u^{''} = \frac{2! \cdot DU2}{h^2} \qquad v^{''} = \frac{2! \cdot DV2}{h^2} \tag{6.7}$$

$$DY3 = \frac{h^3}{3!} \cdot (u^{'''} + v^{'''}) \qquad u^{'''} = \frac{3! \cdot DU3}{h^3} \qquad v^{'''} = \frac{3! \cdot DV3}{h^3} \tag{6.8}$$

A upravené členy Taylorovy řady:

$$DY2 = DU2 + DV2 \tag{6.9}$$

$$DY3 = DU3 + DV3 \tag{6.10}$$

6.2.2 Rozdíl

Uvažujme jednoduchou rovnici rozdílu:

$$y = u - v \tag{6.11}$$

Její první derivace je:

$$y' = u' - v' \qquad y(0) = y_0 \quad u(0) = u_0 \quad v(0) = v_0 \tag{6.12}$$

Využijeme stejný postupu jako u výpočtu součtu. Jednotlivé členy Taylorovy řady se počítají následovně:

$$DY1 = h \cdot (u' - v') \qquad u' = \frac{DU1}{h} \qquad v' = \frac{DV1}{h} \tag{6.13}$$

A tedy rovnice pro první člen je:

$$DY1 = DU1 - DV1 \tag{6.14}$$

Druhý člen Taylorovy řady pro rozdíl:

$$DY2 = \frac{h^2}{2!} \cdot (u'' - v'') \qquad u'' = \frac{2! \cdot DU2}{h^2} \qquad v'' = \frac{2! \cdot DV2}{h^2} \tag{6.15}$$

Dosazením $DU1$ a $DV1$ získáme vztah:

$$DY2 = DU2 - DV2 \tag{6.16}$$

6.2.3 Součin

Základní rovnicí pro součin je:

$$y = u \cdot v \tag{6.17}$$

První derivace podle vzorce pro derivaci součinu:

$$y' = u' \cdot v + u \cdot v' \qquad y(0) = y_0 \tag{6.18}$$

První člen Taylorovy řady pro součin můžeme zapsat:

$$DY1 = h \cdot (u' \cdot v + u \cdot v') \tag{6.19}$$

A opět dosazením upravených prvních členů $DU1$ a $DV1$ dostaneme vztah:

$$DY1 = h \cdot (\frac{DU1}{h} \cdot v + u \cdot \frac{DV1}{h}) \tag{6.20}$$

Úpravou tohoto vzorce získáme:

$$DY1 = DU1 \cdot v + u \cdot DV1 \tag{6.21}$$

Druhý člen získáme obdobným postupem (2. derivace 6.17 a dosazení do vzorce pro $DY2$):

$$DY2 = \frac{h^2}{2!} \cdot (\frac{DU2}{\frac{h^2}{2!}} \cdot v + 2 \cdot \frac{DU1}{h} \cdot \frac{DV1}{h} + u \cdot \frac{DV2}{\frac{h^2}{2!}}) \tag{6.22}$$

A po zkrácení:

$$DY2 = DU2 \cdot v + DU1 \cdot DV1 + u \cdot DV2 \tag{6.23}$$

6.2.4 Podíl

Základní rovnicí podílu je:

$$y = \frac{v}{u} \tag{6.24}$$

První derivace podle vzorce pro derivaci podílu:

$$y' = \frac{v' \cdot u - v \cdot u'}{u^2} \qquad y(0) = y_0 \tag{6.25}$$

Po provedení úpravy dostaneme:

$$y' = \frac{1}{u} \cdot (v' - y \cdot u') \tag{6.26}$$

První člen Taylorovy řady podílu můžeme zapsat:

$$DY1 = h \cdot \frac{1}{u} \cdot (v' - y \cdot u') \tag{6.27}$$

A dosazením upravených prvních členů Taylorovy řady:

$$DY1 = h \cdot \frac{1}{u} \cdot (\frac{DV1}{h} - y \cdot \frac{DU1}{h}) \tag{6.28}$$

Úpravou tohoto vzorce získáme:

$$DY1 = \frac{1}{u} \cdot (DV1 - y \cdot DU1) \tag{6.29}$$

Druhý člen získáme obdobným postupem (2. derivace 6.24 a dosazení do vzorce pro $DY2$):

$$DY2 = \frac{h^2}{2!} \cdot \frac{1}{u} \cdot (\frac{DV2}{\frac{h^2}{2!}} - 2 \cdot \frac{1}{u} \cdot (\frac{DV1}{h} - y \cdot \frac{DU1}{h}) \cdot \frac{DU1}{h} - y \cdot \frac{DU2}{\frac{h^2}{2!}}) \tag{6.30}$$

Upravíme:

$$DY2 = \frac{1}{u}(DV2 - \frac{1}{u} \cdot (DV1 - y \cdot DU1) \cdot DU1 - y \cdot DU2) \tag{6.31}$$

6.2.5 Integrace součinu

Rovnice integrace součinu je:

$$y' = u \cdot v \tag{6.32}$$

Derivace podle vzorce pro derivaci součinu:

$$y'' = u' \cdot v + u \cdot v' \qquad y(0) = y_0 \tag{6.33}$$

První člen:

$$DY1 = h \cdot (u + v) \tag{6.34}$$

Druhý člen:

$$DY2 = \frac{h^2}{2!} \cdot (\frac{DU1}{h} \cdot v + u \cdot \frac{DV1}{h}) \tag{6.35}$$

Úpravou rovnice dostaneme:

$$DY2 = \frac{h}{2} \cdot (DU1 \cdot v + u \cdot DV1) \tag{6.36}$$

Třetí člen získáme obdobným postupem (2. derivace 6.32 a dosazení do vzorce pro $DY3$):

$$y''' = u'' \cdot v + 2 \cdot u' \cdot v + u \cdot v'' \tag{6.37}$$

$$DY3 = \frac{h^3}{3!} \cdot (\frac{DU2}{\frac{h^2}{2!}} \cdot v + 2 \cdot \frac{DU1}{h} \cdot \frac{DV1}{h} + u \cdot \frac{DV2}{\frac{h^2}{2!}}) \tag{6.38}$$

A po zkrácení:

$$DY3 = \frac{h}{3} \cdot (DU2 \cdot v + DU1 \cdot DV1 + u \cdot DV2) \tag{6.39}$$

6.2.6 Integrace podílu

Rovnici pro integraci podílu odpovídá:

$$y' = s\frac{v}{u} \tag{6.40}$$

Derivace podle vzorce pro derivaci podílu:

$$y'' = \frac{v' \cdot u - v \cdot u'}{u^2} \qquad y(0) = y_0 \tag{6.41}$$

Vytknutím z předchozí rovnice dostaneme:

$$y' = \frac{1}{u} \cdot (v' - y \cdot u') \tag{6.42}$$

První člen Taylorovy řady podílu můžeme zapsat:

$$DY1 = h \cdot \frac{v}{u} \tag{6.43}$$

Další člen:

$$DY2 = \frac{h^2}{2!} \cdot \frac{1}{u} \cdot (\frac{DV1}{h} - y \cdot \frac{DU1}{h}) \tag{6.44}$$

Úpravou získáme:

$$DY1 = \frac{h}{2} \cdot \frac{1}{u} \cdot (DV1 - y \cdot DU1) \tag{6.45}$$

Třetí člen obdobně (3. derivace 6.40 a dosazení do vzorce pro $DY2$):

$$DY3 = \frac{h^3}{3!} \cdot \frac{1}{u} \cdot (\frac{DV2}{\frac{h^2}{2!}} - 2 \cdot \frac{1}{u} \cdot (\frac{DV1}{h} - y \cdot \frac{DU1}{h}) \cdot \frac{DU1}{h} - y \cdot \frac{DU2}{\frac{h^2}{2!}}) \tag{6.46}$$

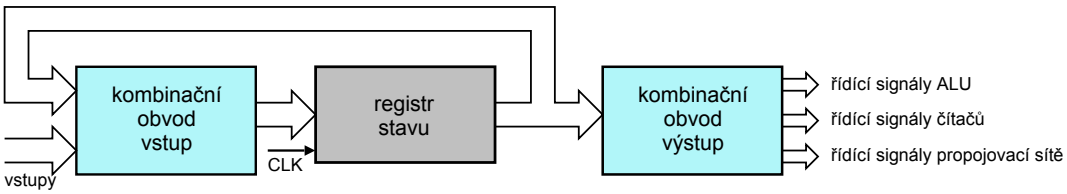
Upravíme:

$$DY3 = \frac{h}{3} \cdot \frac{1}{u}(DV2 - \frac{1}{u} \cdot (DV1 - y \cdot DU1) \cdot DU1 - y \cdot DU2) \tag{6.47}$$

Kapitola 7

Řízení navržené architektury

Základní výpočetní systém tvořený aritmeticko-logickou jednotkou je řízen za pomocí řídící jednotky - řadiče. Ten generuje pevně stanový sled řídících signálů pro ALU (procesor). Jedná se o specializovaný řadič, který využívá instrukcí mikroprogramu. Instrukce jsou tvořeny z adresy a generovaných řídících signálů. Řadič je tvořen Moorovým automatem. Výstup řadiče (řídící signály), který ovládá celý systém, je dán pouze aktuálním stavem.



Obrázek 7.1: Moorův automat [12]

K přechodu z aktuálního do následujícího stavu je potřeba určitý časový okamžik, který je určen zpožděním kombinačních obvodů. Je možné k tomu přistupovat dvěma způsoby: asynchronně nebo synchronně. U asynchronního zpracování hrozí vznik hazardů a parazitních impulsů. Synchronní zpracování využívá hodinového signálu, který řídí zápis do registrů. Tím se vyhneme zmíněným neplatným impulsům, které se vyskytuje u asynchronního přístupu.

Důležitou částí námi navrhovaného řadiče jsou čítače smyček, které udávají informaci pro změnu toku řízení. Jsou použity čtyři čítače: počet výsledků výpočtu, dosažený řád Taylorovy řady (DY_i), údaj o délce slova (dílčí součty násobení) a počet součtů úplné jednobitové sčítačky (u SSI). Řadič může být rozšířen o komparátor, který zastaví výpočet po dosažení požadované přesnosti. Komparátor porovnává dva po sobě jdoucí mezivýsledky Taylorovy řady. Tímto způsobem může urychlit dosažení výsledku zamezením výpočtu dalších řádů.

Mezi úlohy řídící jednotky (řadiče) patří:

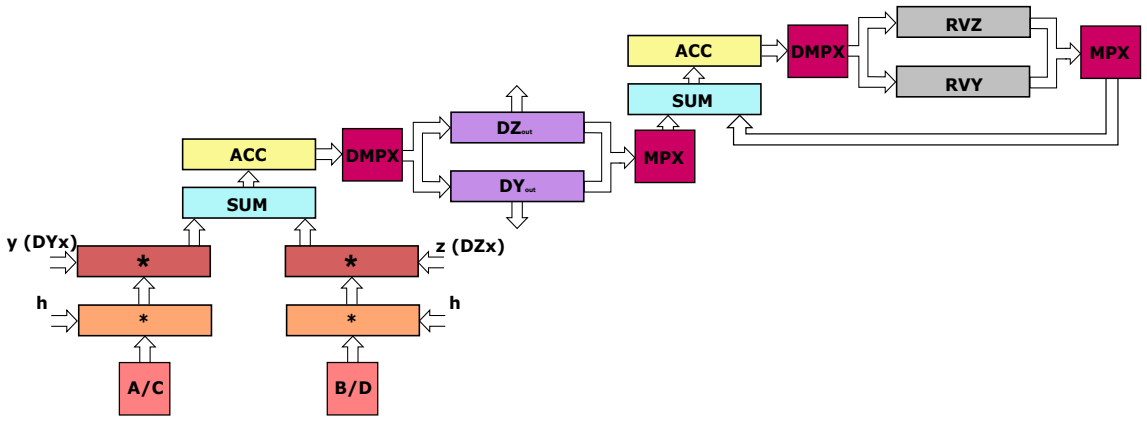
- zápis dat do výpočetní jednotky (počáteční podmínka, krok)
- výpočet paralelního systému (aritmetické operace, zápisy, posuvy)
- distribuce výsledku
- synchronizace rozdílných výpočetních jednotek (čekání na další vstupní data)

Všechny integrátory (ALU) mohou být řízeny buď jednou společnou řídící jednotkou, což odpovídá architektuře SIMD, anebo mohou být řízeny každý vlastním řídícím systémem MIMD. Výhodou použití jedné řídící jednotky pro všechny SIMD je menší prostorová náročnost. Při větších počtech výpočetních jednotek mohou vznikat na rozsáhlých řídících sběrnicích hazardní stavy a přeslechy. U FPGA (programovatelné hradlové pole) mohou být problémem rozsáhle sběrnice řídícího charakteru. Pro rozsáhlejší zapojení je proto vhodnější použít architekturu MIMD.

7.1 Řízení paralelního integrátoru pro součet

7.1.1 Obecný algoritmus řízení součtu

1. Do registru RV se uloží počáteční podmínka y_0. Registry ACC a čítače se vynulují.
2. ACC se vynuluje.
3. Čítač členů Taylorovy řady se inkrementuje o 1.
4. Hodnoty výstupů DY_x a DZ_x se vynásobí s upraveným krokem. Krok je závislý na právě počítaném členu Taylorovy řady. Může být také vynásoben případnou konstantou. Hodnota x je aktuálně počítaný člen Taylorovy řady.
5. Pronásobené hodnoty DY_x a DZ_x s upraveným krokem se sečtou na první sčítačce.
6. Výsledná hodnota z předchozího kroku je uložena do ACC a následně do mezipaměti.
7. Na druhé sčítačce se sečte RV s mezivýsledkem v mezipaměti (aktuálně vypočítaný člen Taylorovy řady).
8. Výsledná hodnota z předchozího kroku je uložena do ACC.
9. Hodnota v ACC se uloží do RV.
10. Pokud není dosažena požadovaná přesnost nebo maximální řád Taylorovy řady, algoritmus pokračuje bodem 2, jinak končí.

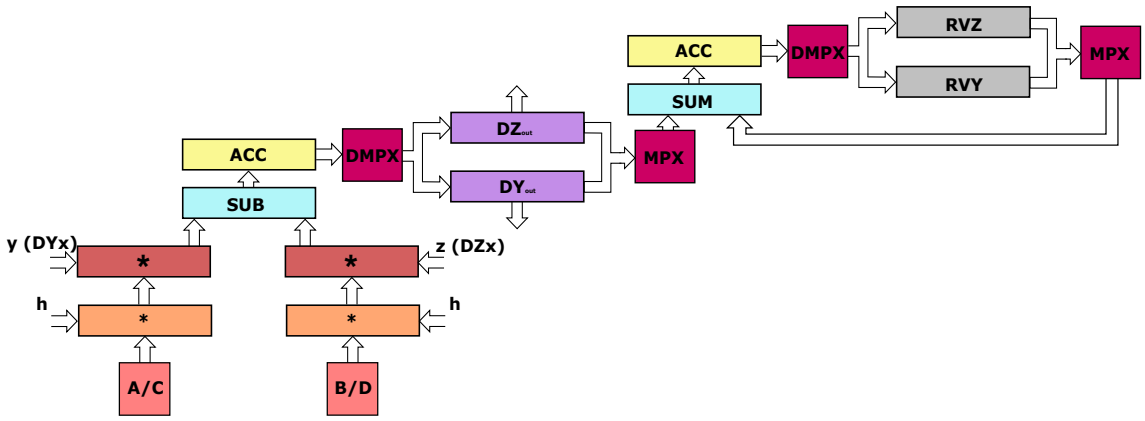


Obrázek 7.2: Schéma rozšířeného součtového integrátoru (obsahuje registry pro výpočet dvou proměnných)

7.2 Řízení paralelního integrátoru pro rozdíl

7.2.1 Obecný algoritmus řízení rozdílu

1. Do registru RV se uloží počáteční podmínka y_0. Registry ACC a čítače se vynulují.
2. ACC se vynuluje.
3. Čítač členů Taylorovy řady se inkrementuje o 1.
4. Hodnoty vystupů DY_x a DZ_x se vynásobí s upraveným krokem. Krok je závislý na právě počítaném členu Taylorovy řady. Může být také vynásoben případnou konstantou. Hodnota x je aktuálně počítaný člen Taylorovy řady.
5. Pronásobené hodnoty DY_x a DZ_x s upraveným krokem se odečtou na paralelní odčítačce.
6. Výsledná hodnota z předchozího kroku je uložena do ACC a následně do mezipaměti.
7. Na paralelní sčítačce se sečte RV s mezivýsledkem v mezipaměti (aktuálně vypočítaný člen Taylorovy řady).
8. Výsledná hodnota z předchozího kroku je uložena do ACC.
9. Hodnota v ACC se uloží do RV.
10. Pokud není dosažena požadovaná přesnost nebo maximální řád Taylorovy řady, algoritmus pokračuje bodem 2, jinak končí.

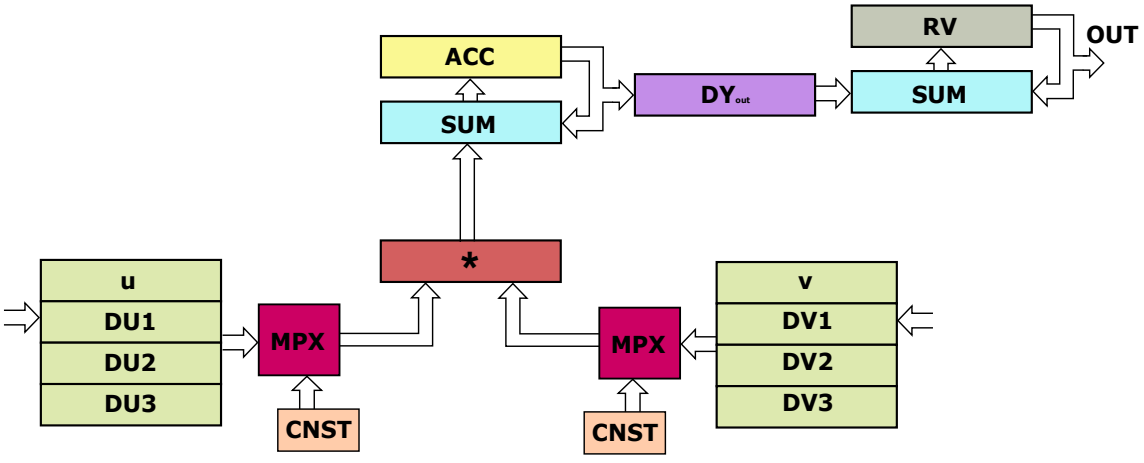


Obrázek 7.3: Schéma rozšířeného rozdílového integrátoru (obsahuje registry pro výpočet dvou proměnných)

7.3 Řízení paralelního integrátoru pro součin

7.3.1 Obecný algoritmus řízení součinu

1. Do registru RV se uloží počáteční podmínka y_0. Registry ACC a čítače se vynulují.
2. ACC se vynuluje.
3. Čítač členů Taylorovy řady se inkrementuje o 1. Pomocný čítač (vzestupný) pro V je omezen na hodnotu čítače členů Taylorvy řady. Druhý pomocný čítač (sestupný) pro U je nastaven na hodnotu čítače členu Taylorovy řady.
4. Pomocí pomocných čítačů se postupně adresují v poli registrů hodnoty v_0 až DV_x a u_0 a DU_x. Multiplexory 1 a 2 se přepnou tak, aby na vstupu násobičky byly přivedeny hodnoty těchto registrů. Multiplexor 3 se přepne na vstup z ACC.
5. Postupně (dokud nedosáhnou pomocné registry hraniční meze) se sčítají vzniklé součiny DU_x a DV_x s registrem ACC.
6. Výsledná hodnota z předchozího kroku je uložena do ACC a následně do mezipaměti DY_{out}.
7. Na druhé paralelní sčítačce se sečte RV s mezivýsledkem v DY_{out} (aktuálně vypočítaný člen Taylorovy řady).
8. Výsledná hodnota z předchozího kroku je uložena zpět do RV.
9. Pokud není dosažena požadovaná přesnost nebo maximální řád Taylorovy řady, algoritmus pokračuje bodem 2, jinak končí.

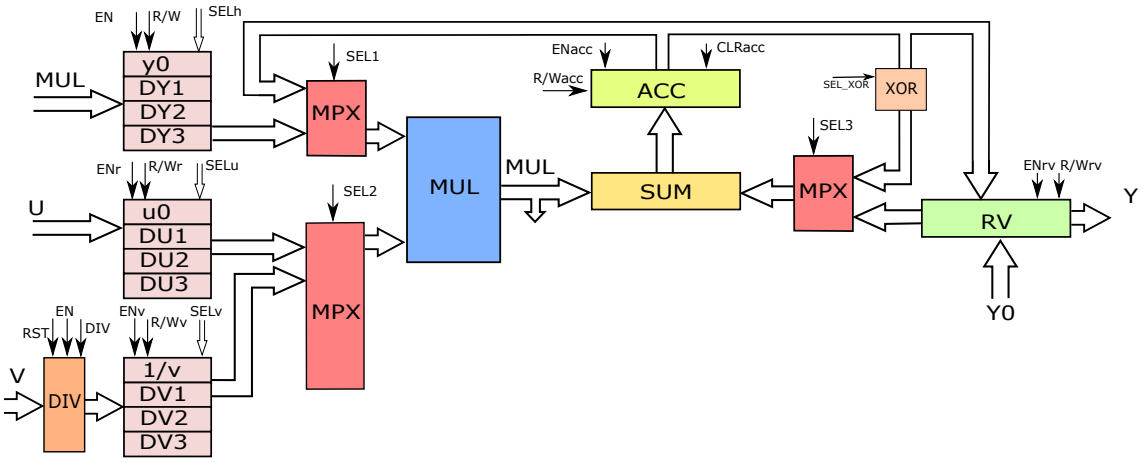


Obrázek 7.4: Schéma násobícího integrátoru

7.4 Řízení paralelního integrátoru pro podíl

7.4.1 Obecný algoritmus řízení podílu

1. Do registru RV se uloží počáteční podmínka y_0. Registr ACC se vynuluje.
2. Provedení SRT dělení pro V (vytvoření hodnoty $\frac{1}{v}$).
3. ACC se vynuluje.
4. Čítač členů Taylorovy řady se inkrementuje o 1. Pomocný čítač (vzestupný) pro V je omezen na hodnotu čítače členů Taylorovy řady. Druhý pomocný čítač (sestupný) pro Y je nastaven na hodnotu čítače členu Taylorovy řady - 1.
5. Hodnota DU_x (kde x je hodnota čítače členu Taylorovy řady - 1) se uloží do ACC.
6. Pomocí pomocných čítačů se postupně adresují v poli registrů hodnoty y_0 až DY_x a DV_1 a DV_x. Multiplexory 1 a 2 se přepnou tak, aby na vstupu násobičky byly přivedeny hodnoty těchto registrů. Multiplexor 3 se přepne na vstup z ACC.
7. Postupně (dokud nedosáhnou pomocné registry hraniční meze) se pronásobují hodnoty DU_x a DV_x a mezivýsledky jsou odečteny od registru ACC.
8. Výsledná hodnota z předchozího kroku je uložena do ACC.
9. Multiplexor 3 je nastaven na vstup z RV. Multiplexor 1 se přepne na vstup z ACC a multiplexor 2 na vstup z pole registrů V. Pole registru DV_x je nastaveno na hodnotu $\frac{1}{v}$. Ta se vynásobí s obsahem ACC. Součin se sečte s registrem RV (obsahující součet předešlých členů Taylorovy řady a počáteční podmínky).
10. Výsledná hodnota z předchozího kroku je uložena do ACC.
11. Hodnota v ACC se uloží do RV.
12. Pokud není dosažena požadovaná přesnost nebo maximální řád Taylorovy řady algoritmus pokračuje bodem 2, jinak končí.

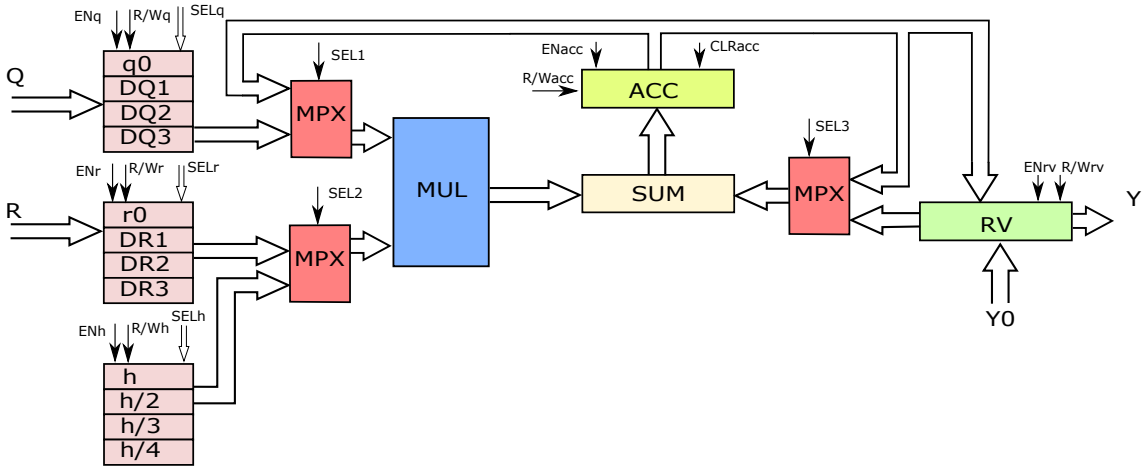


Obrázek 7.5: Schéma dělícího integrátoru

7.5 Řízení paralelního integrátoru pro integraci součinu

7.5.1 Obecný algoritmus řízení integrace součinu

1. Do registru RV se uloží počáteční podmínka y_0. Registr ACC a čítače se vynulují.
2. ACC se vynuluje.
3. Čítač členů Taylorovy řady se inkrementuje o 1. Pomocný čítač (vzestupný) pro V je omezen na hodnotu čítače členů Taylorvy řady. Druhý pomocný čítač (sestupný) pro U je nastaven na hodnotu čítače členu Taylorovy řady.
4. Pomocí pomocných čítačů se postupně adresují v poli registrů hodnoty u_0 až DU_x a v_0 a DV_x. Multiplexory 1 a 2 se přepnou tak, aby na vstupu násobičky byly přivedeny hodnoty těchto registrů. Multiplexor 3 se přepne na vstup z ACC.
5. Postupně (dokud nedosáhnou pomocné registry hraniční meze) se pronásobují hodnoty DU_x a DV_x vynásobí a mezivýsledky jsou sečteny s registrem ACC.
6. Výsledná hodnota z předchozího kroku je uložena do ACC.
7. Multiplexor 3 je nastaven na vstup z RV. Multiplexor 1 se přepne na vstup z ACC a multiplexor 2 na vstup z pole registrů kroku. Pole registru kroků je adresováno čítačem členů Taylorovy řady (výběr vhodné velikosti kroku). Toto nastavení zajistí, že násobička vynásobí hodnoty ACC a krok. Součin se sečte s registrem RV (obsahujícím součet předešlých členů Taylorovy řady a počáteční podmínky).
8. Výsledná hodnota z předchozího kroku je uložena do ACC.
9. Hodnota v ACC se uloží do RV.
10. Pokud není dosažena požadovaná přesnost nebo maximální řád Taylorovy řady algoritmus pokračuje bodem 2, jinak končí.



Obrázek 7.6: Schéma integrátoru násobení s integrací [15]

7.5.2 Mikroprogram řadiče pro PPI-MUL

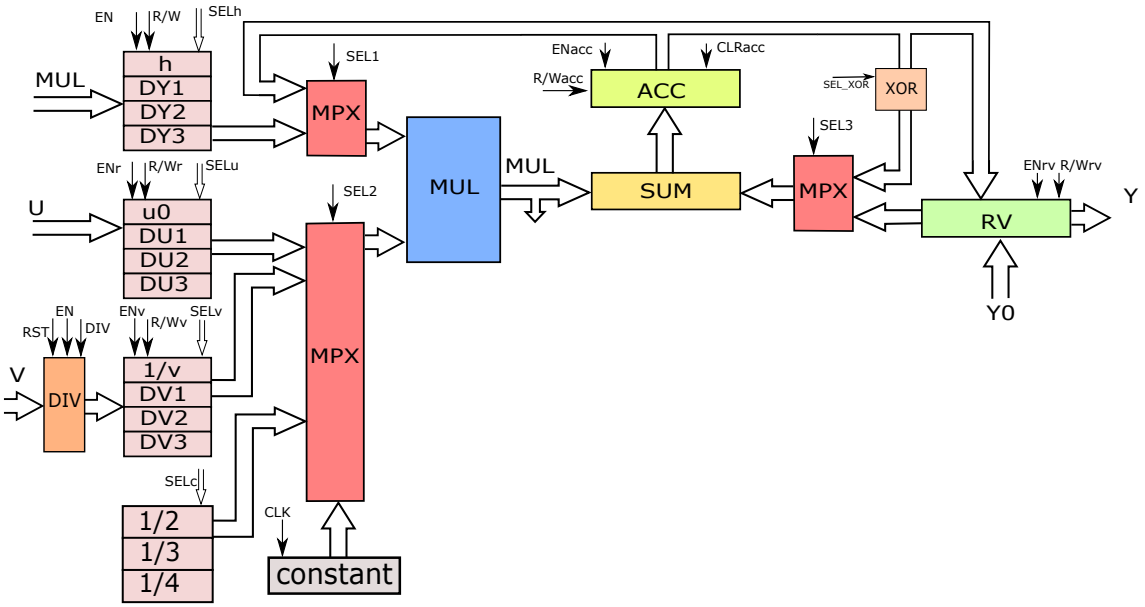
V tabulce (7.1) jsou uvedeny signály, jež definují běh výpočtu integrátoru. Neobsahuje povolovací signály registrů kvůli přehlednosti tabulky. Vždy, když je tedy uveden signál R/W určitého bloku, znamená to, že je zároveň aktivní signál EN tohoto bloku. Hodnota X v tabulce znamená, že hodnota signálu není podstatná. Tato tabulka vznikla za spolupráce s V. Závadou [15].

Tabulka 7.1: Mikroprogram pro násobení s integrací pro 4 členy Taylorovy řady [15]

Stav	SEL_1	SEL_2	SEL_3	SEL_q	SEL_r	SEL_h	CLR_{acc}	R/W_{acc}	R/W_{RV}
1	X	X	X	XX	XX	XX	1	X	0
2	1	0	0	00	00	XX	0	0	X
3	1	0	0	00	00	XX	0	1	X
4	0	1	1	XX	XX	00	0	0	0
5	0	1	1	XX	XX	00	0	1	0
6	X	X	X	XX	XX	XX	0	0	1
7	X	X	X	XX	XX	XX	1	X	X
8	1	0	0	01	00	XX	0	0	X
9	1	0	0	01	00	XX	0	1	X
10	1	0	0	00	01	XX	0	0	X
11	1	0	0	00	01	XX	0	1	X
12	0	1	1	XX	XX	01	0	0	0
13	0	1	1	XX	XX	01	0	1	0
14	X	X	X	XX	XX	XX	0	0	1
15	X	X	X	XX	XX	XX	1	X	X
16	1	0	0	10	00	XX	0	0	X
17	1	0	0	10	00	XX	0	1	X
18	1	0	0	01	01	XX	0	0	X
19	1	0	0	01	01	XX	0	1	X
20	1	0	0	00	10	XX	0	0	X
21	1	0	0	00	10	XX	0	1	X
22	0	1	1	XX	XX	10	0	0	0
23	0	1	1	XX	XX	10	0	1	0
24	X	X	X	XX	XX	XX	0	0	1
25	X	X	X	XX	XX	XX	1	X	X
26	1	0	0	11	00	XX	0	0	X
27	1	0	0	11	00	XX	0	1	X
28	1	0	0	10	01	XX	0	0	X
29	1	0	0	10	01	XX	0	1	X
30	1	0	0	01	10	XX	0	0	X
31	1	0	0	01	10	XX	0	1	X
32	1	0	0	00	11	XX	0	0	X
33	1	0	0	00	11	XX	0	1	X
34	0	1	1	XX	XX	11	0	0	0
35	0	1	1	XX	XX	11	0	1	0
36	X	X	X	XX	XX	XX	0	0	1

7.6 Řízení paralelního integrátoru pro integraci podílu

Řadič pro integraci podílu je vytvořen konečným automatem. Tento stroj nastavuje jednotlivé řídící signály. Je umožněn reset, který uvede automat do počátečního nastavení. Mikroprogram generovaný automatem pro paralelní integrátor integrace podílu pro 4. řád Taylorovy řady je znázorněný v tabulce 7.2. Tato tabulka vznikla za spolupráce s F. Matečným [3].



Obrázek 7.7: Schéma integrátoru podílu s integrací

Po načtení vstupů je připravena převrácená hodnota vstupní proměnné v. V této variantě nejsou použity čítače pro kontrolu toku. Automat může čítače využít pro kontrolu výpočtu, řízení toku a generování konstant pro operace výpočtu Taylorovy řady. Konkrétně se jedná o tyto čítače: 2 čítače konstant (generování konstant pro výpočet), čítač členů Taylorovy řady a čítač zpracovaných výpočtů v rámci jednoho členu Taylorovy řady.

7.6.1 Obecný algoritmus řízení integrace podílu

1. Do registru RV se uloží počáteční podmínka y_0. Registr ACC se vynuluje.
2. Provedení SRT dělení pro V (vytvoření hodnoty $\frac{1}{v}$).
3. ACC se vynuluje.
4. Čítač členů Taylorovy řady se inkrementuje o 1. Pomocný čítač (vzestupný) pro V je omezen na hodnotu čítače členů Taylorovy řady - 1. Druhý pomocný čítač (sestupný) pro Y je nastaven na hodnotu čítače členu Taylorovy řady - 1.
5. Pronásobení hodnoty DU_x (kde x je hodnota čítače členu Taylorovy řady - 1) s krokem h. Hodnota se uloží do ACC.
6. Pomocí pomocných čítačů se postupně adresují v poli registrů hodnoty DY_1 až DY_x a DV_1 a DV_x. Multiplexory 1 a 2 se přepnou tak, aby na vstupu násobičky byly přivedeny hodnoty těchto registrů. Multiplexor 3 se přepne na vstup z ACC.

7. Postupně (dokud nedosáhnou pomocné registry hraniční meze) se vynásobí hodnoty DU_x a DV_x (případná konstanta) a mezivýsledky jsou odečteny od registru ACC.
8. Výsledná hodnota z předchozího kroku je uložena do ACC.
9. Multiplexor 3 je nastaven na vstup z RV. Multiplexor 1 se přepne na vstup z ACC a multiplexor 2 na vstup z pole registrů kroku. Pole registru DV_x je nastaveno na hodnotu $\frac{1}{v}$. Ta se vynásobí s danou konstantou a následně s obsahem ACC. Součin se sečte s registrem RV (obsahující součet předešlých členů Taylorovy řady a počáteční podmínky).
10. Výsledná hodnota z předchozího kroku je uložena do ACC.
11. Hodnota v ACC se uloží do RV.
12. Pokud není dosažena požadovaná přesnost nebo maximální řád Taylorovy řady algoritmus pokračuje bodem 2, jinak končí.

Tabulka 7.2: Mikroprogram pro dělení s integrací pro 4 členy Taylorovy řady [3]

CLKrv	CLKacc	CLKv	CLKu	CLKy	SEL2	SEL3	SEL4	SEL5	SEL6	SEL7	SEL8	BLOK	popis funkcie
0	0	0	0	0	00	00	00	00	00	0	0	0	Init
1	0	0	0	1	00	00	X	00	01	0	X	1	Y0 -> RV, dělení 1/v
0	1	1	0	0	00	00	X	00	01	0	X	1	u * h -> ACC
IF ($done == 1$) pokračuj													
0	0	1	0	0	00	X	X	01	00	1	0	0	DIV -> 1/v
0	1	0	0	1	00	X	X	01	00	1	0	0	ACC * 1/v -> DY1 ACC * 1/v + RV -> ACC
1	0	0	0	0	01	X	X	01	00	0	X	1	ACC -> RV
0	1	0	0	0	01	X	X	01	00	0	X	1	DY1 * DV1 -> ACC
0	0	0	0	0	X	01	X	00	01	0	1	0	nastavení MPX
0	1	0	0	0	X	01	X	00	01	0	1	0	DU1 * h – ACC -> ACC
0	0	0	0	0	X	X	00	X	11	1	X	1	nastavení MPX
0	1	0	0	0	X	X	00	X	11	1	X	1	ACC * 1/2 -> ACC
0	0	0	0	0	00	X	X	10	00	1	0	0	nastavení MPX
0	1	0	0	1	00	X	X	10	00	1	0	0	ACC * 1/v -> DY2 ACC * 1/v + RV -> ACC
1	0	0	0	0	X	X	X	10	10	0	X	1	ACC -> RV
0	0	0	0	1	X	X	X	10	10	0	X	1	2 * DY2 -> DY2
0	0	0	0	0	01	X	X	10	00	0	X	1	nastavení MPX
0	1	0	0	0	01	X	X	10	00	0	X	1	DY2 * DV1 -> ACC
0	0	0	0	0	10	X	X	01	00	0	1	0	nastavení MPX
0	1	0	0	0	10	X	X	01	00	0	1	0	DY1*DV2 + ACC -> ACC
0	0	0	0	0	X	10	X	00	01	0	1	0	nastavení MPX
0	1	0	0	0	X	10	X	00	01	0	1	0	DU2 * h – ACC -> ACC
0	0	0	0	0	X	X	01	X	11	1	X	1	nastavení MPX
0	1	0	0	0	X	X	01	X	11	1	X	1	ACC * 1/3 -> ACC
0	0	0	0	0	00	X	X	11	00	1	0	0	nastavení MPX
0	1	0	0	1	00	X	X	11	00	1	0	0	ACC * 1/v -> DY3 ACC * 1/v + RV -> ACC
1	0	0	0	0	X	X	X	11	10	0	X	1	ACC -> RV
0	0	0	0	1	X	X	X	11	10	0	X	1	3 * DY3 -> DY3
0	0	0	0	0	01	X	X	11	00	0	X	1	nastavení MPX
0	1	0	0	0	01	X	X	11	00	0	X	1	DY3 * DV1 -> ACC
0	0	0	0	0	10	X	X	10	00	0	1	0	nastavení MPX
0	1	0	0	0	10	X	X	10	00	0	1	0	DY2*DV2 + ACC -> ACC
0	0	0	0	0	11	X	X	01	00	0	1	0	nastavení MPX
0	1	0	0	0	11	X	X	01	00	0	1	0	DY1*DV3 + ACC -> ACC
0	0	0	0	0	X	11	X	00	01	0	1	0	nastavení MPX
0	1	0	0	0	X	11	X	00	01	0	1	0	DU3 * h – ACC -> ACC
0	0	0	0	0	X	X	10	X	11	1	X	1	nastavení MPX
0	1	0	0	0	X	X	10	X	11	1	X	1	ACC * 1/4 -> ACC
0	0	0	0	0	00	X	X	X	00	1	0	0	nastavení MPX
0	1	0	0	0	00	X	X	X	00	1	0	0	ACC * 1/v + RV -> ACC
1	0	0	0	0	X	X	X	X	X	X	X	1	ACC -> RV

7.7 Čísla s plovoucí řádovou čárkou pro zmíněné operace

Pro výpočty s velkým rozptylem počítaných hodnot je vhodné využít k uložení dat pohyblivou řádovou čárku. Eliminují se tím částečně chyby vznikající omezeným počtem bitů pro uložení čísla. Koncepce aritmeticko-logických jednotek zmíněných v této prací je pak možné využít pro výpočty s pohyblivou řádovou čárkou. Ovšem návrh musí být doplněn o práci s mantisou a exponentem.

Podrobný návrh integrátorů v pohyblivé řádové čárce je obsažen v práci [9]. Integrátor obsahuje stejné prvky jako u verze pro pevnou řádovou čárku. Je rozšířen o obvody pro posuvy mantis, zvyšování a snižování exponentu operandů. Dále je přidán normalizační obvod pro výsledek a několik registrů sloužících k uložení exponentů a mantis mimo hlavní výpočet.

V následujícím textu je nastíněn princip aritmetickologické jednotky paralelního integrátoru pro čísla s pohyblivou řádovou čárkou.

7.7.1 Násobení a dělení FP čísel

1. Init
Inicializační stav.
2. Uložení mantisy a exponentu Y
Uložení mantisy prvního operandu (Y) do registru. Exponent prvního operandu se uloží do akumulátoru pro exponent (ACC_{exp}).
3. Uložení mantisy a exponentu X
Uložení exponentu druhého operandu (X) do porovnávacího registru a sečtení/odečtení s ACC_{exp} (exponent Y). Výsledek je exponent výsledného součinu/podílu, který se uloží do paměti.
3. a) Pouze pro dělení
Vytvoří se převrácená hodnota mantisy X.
4. Provedení operace násobení
Provedení paralelního násobení mantis operandů.
5. Kontrola a normalizace čísla
Převod čísla do normalizovaného tvaru ve speciálním normalizačním obvodu.
6. Uložení výsledku
Výsledek se uloží do paměti. Algoritmus výpočtu v integrátorech pokračuje stejným způsobem jako u pevné řádové čárky.

7.7.2 Sčítání a odčítání FP čísel

1. Init
Inicializační stav.
2. Uložení mantisy a exponentu Y
Uložení mantisy prvního operandu (Y) do registru. Exponent prvního operandu se uloží do akumulátoru pro exponent (ACC_{exp}).
3. Uložení mantisy a exponentu X
Uložení exponentu druhého operandu (X) do porovnávacího registru. Číslo s menším exponentem se upraví (posuvy mantisy a změna exponentu).
4. Provedení operace sečtení/odečtení
Provedení paralelního sečtení/odečtení mantis.

5. Kontrola a normalizace čísla
Převod čísla do normalizovaného tvaru ve speciálním normalizačním obvodu.
6. Uložení výsledku
Výsledek se uloží do paměti. Algoritmus výpočtu v integrátorech pokračuje stejným způsobem jako u pevné řádové čárky.

Kapitola 8

Implementace, simulace a testování navržené architektury

Implementace navržené architektury je provedena v jazyce VHDL. Jako první byla implementována nejsložitější varianta zmíněných integrátorů, a to sériově-paralelní (SPI). Z této implementace vychází všechny ostatní varianty. Paralelní integrátory jsou intuitivnější a mají jednoduší schémata.

Koncept je vytvořen tak, že jsou připraveny základní výpočetní a funkční bloky, které se využijí pro sestavení integrátoru. Jsou vytvořeny VHDL prvky ze schémat výše: ACC – akumulátor, SR – posuvný registr, R – registry, SUM – sčítačka (případně odčítačka), čítače – používané pro řízení toku mikroprogramů, MPX – multiplexor, DIV a MULT – paralelní dělička a násobička a další. Tyto funkční bloky jsou propojeny datovými a řídícími sběrnicemi v souboru integrátoru. Řídící signály jsou generovány řadičem, který obsahuje příslušný mikroprogram pro daný integrátor. Ten obaluje všechny zmíněné funkční jednotky a řadič do sebe a vytváří tím nadřazený výpočetní celek. Tímto způsobem je možné vytvořit jakoukoliv variantu výše zmíněných integrátorů. Všechny prvky jsou implementovány synchronním způsobem jako ochrana proti nekonzistentním stavům.

V implementovaném SPI integrátoru jsou přítomny doplňující funkce a celý návrh je psán genericky. Je tedy možné jednoduchou konfigurací nastavení změnit šířku dat integrátoru z výchozích 32b např. na 128b. V případě potřeby jsou tyto hodnoty pro čítače předpočteny logaritmem před syntézou. Pro délku kroku je vytvořena funkce, která provede normalizaci zadaného kroku a roztažení na příslušnou bitovou délku, kterou používá daný integrátor.

Testování integrátoru bylo nejdříve provedeno v aplikaci ModelSim s testbenchem. Tímto způsobem byl navržen a částečně otestován mikroprogram pro integrátory. V dalším stupni testování bylo aplikováno implementované zapojení na programovatelné hradlové pole (FPGA). K tomu bylo použito platformy FITkitu[16].

8.1 Programovatelné hradlové pole – FPGA

FPGA neboli programovatelné hradlové pole jsou speciální číslicové integrované obvody, které obsahují programovatelné bloky spojené rekonfigurovatelnou maticí spojů a vstupně výstupní rozhraní. Programovatelné bloky jsou různě složité. Tyto obvody jsou často nakonfigurovány až u zákazníka. V současnosti se FPGA nasazují v široké škále aplikací, zejména díky jejich vlastnostem - programovatelnosti, snadnému návrhu, flexibilitě, nízkým cenám

a malé spotřebě energie samotným čipem FPGA. Nejčastější je použití u menších sérií navrhovaných zařízení, kde se nevyplatí zákaznický integrovaný obvod a použití obecného procesoru není vhodné. Větší čipy FPGA umožňují i implementaci komplikovaných nebo speciálních procesorů, čehož bude využito i v mém případě.

Hlavní jazyky pro syntézu hradlových polí jsou dnes VHDL a Verilog. Jednoduché návrhy je možné nakreslit pomocí grafických editorů. Navrhovaný obvod popisujeme na úrovni RTL. U tohoto popisu se využívá sady registrů propojených kombinační logikou realizující logické operace. Výhodou návrhu na RTL úrovní je vysoký stupeň abstrakce. Typicky jeden řádek zdrojového kódu odpovídá desítkám až stovkám hradel, proto je tento přístup velmi produktivní a přehledný. RTL popis systému se konvertuje do konfiguračního souboru pro FPGA obvod v následných krocích: syntéza, mapování, rozmístění a propojení a generování konfiguračního souboru.

FPGA rozdělujeme na dva základní typy podle uložení konfigurace: FPGA s volatilní a FPGA s nevolatilní konfigurací.

FPGA s volatilní konfigurací ukládá konfigurační soubory do paměťových buněk typu SRAM (technologický proces CMOS). Mezi výhody tohoto přístupu je snadná konfigurace a rekonfigurace za běhu systému. Nevýhodami jsou obtížnější zabezpečení intelektuálního vlastnictví (konfigurace lze jednoduše vyčíst z paměti) a nutnost konfigurace systému po startu (vyžaduje externí paměť s řadičem, což zvyšuje potřebný prostor na desce s tištěnými spoji). Doba konfigurace FPGA obvodů může trvat až stovky milisekund, což je pro některé aplikace nezanedbatelná doba.

FPGA s nevolatilní konfigurací ukládá konfiguraci do flash pamětí nebo pamětí EEPROM. U těchto obvodů je obtížnější změna konfigurace, ale zabezpečují lépe intelektuální vlastnictví a poskytují vyšší odolnost proti vnějším vlivům.

8.2 Algoritmus výpočtu na obvodu FPGA

Po zapojení napájení je systém v nedefinovaném stavu. Proto se jako první věc resetují všechny obvody. FPGA se spustí dříve než MCU (Micro-Controller Unit).

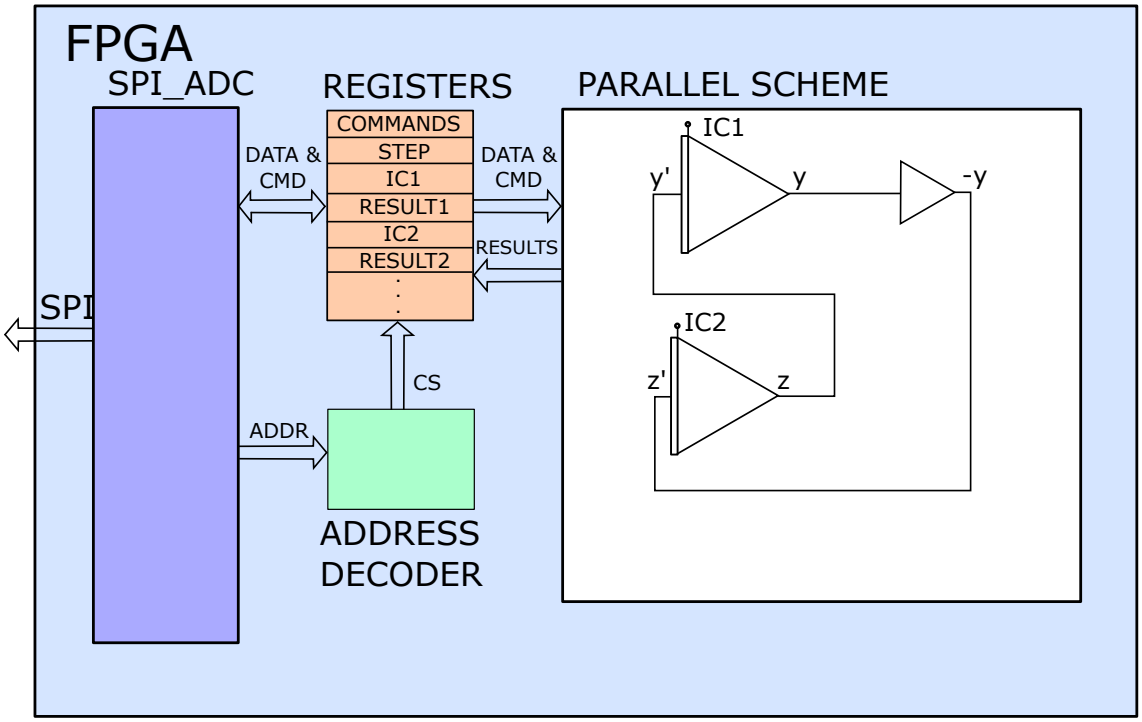
Jakmile se nastartuje MCU, nahrají se počáteční podmínky do registrů pomocí funkce: FPGA_SPI_RW_A8_DN (SPI_FPGA_ENABLE_WRITE, BASE_ADDR, data_xilinx, 4), První parametr této funkce říká, zda se bude číst nebo zapisovat. Druhý určuje adresu v paměti. Třetí a čtvrtý popisují, kam se data ukládají na MCU a jak jsou velká (kolik bytů se přenese).

V obvodu FPGA tato funkce způsobí několik jevů. Nejdříve je vybrán registr, se kterým se bude pracovat. Výběr registru určuje adresový dekodér, který přebírá část adresy vstupující do FPGA přes SPI_ADC (SPI_ADC je převodník pro komunikaci mezi FPGA a MCU). V okamžiku vstupu dat do FPGA je registr již vybraný. Při zápisu dat aktivuje SPI_ADC signál WRITE_EN pro jeden hodinový takt. Tento signál se používá pro aktivaci zápisu do vybraného registru.

V okamžiku, kdy se nahrají na FPGA počáteční podmínky a velikost kroku, dojde k inicializaci schématu integrátorů pomocí resetovacího příkazu v příkazovém registru (COMMAND). Integrátory si načtou počáteční podmínky do sebe a čekají na povolení pro spuštění výpočtu. Dalším příkazem - enable - dojde ke spuštění výpočtu. Jakmile integrátory dokončí svůj výpočet, nahrají výsledek do registrů výsledků (RESULTx). Metodou pooling dochází k odečítání výsledných hodnot do MCU a jejich případné další zpracování.

Převodník přečte z registru data a pošle je do MCU, kde jsou následně dále zpracovány.

Další možné varianty přístupu mohou být: předělat metodu pooling na ověřování output_enable signálu. Propojení registrů výsledku a počáteční hodnoty, což by znamenalo (s aktivním enable/povolovacím signálem) automatický výpočet další hodnoty.



Obrázek 8.1: Schéma zapojení jednoduchého výpočtu na FPGA na FITkitu

Navržená architektura vychází z prací [7], [12], [13]. Využívá poznatky již vzniklých integrátorů pro přepracování a vznik vhodnějšího přístupu pro přenos výpočtů pomocí moderní metody Taylorovy řady z rovnice 2.1.2 na technologii FPGA.

Zlepšeno bylo zapouzdření krabičky do jednoho celku, upraven řídící algoritmus - snížení počtu stavů, zjednodušení a zefektivnění návrhu pro FPGA, přidání adresového dekodéru pro přenos mezi FPGA a MCU. Vytvoření řídících komponent pro celé schéma. Návrh byl přizpůsoben jednoduchému strojovému předzpracování při sestaveném schématu před syntézou.

Testování probíhalo ve dvou úrovních - nejprve se vše testovalo v simulacích ModelSim (testbench), následně se využil FITkit pro otestování na FPGA.

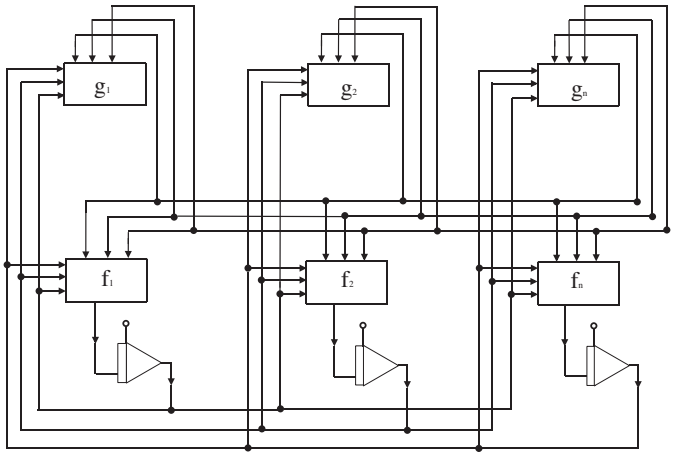
Kapitola 9

Propojovací systém vzniklé architektury

9.1 Propojovací síť

Propojovací sítě jsou jednou z hlavních komunikačních komponent architektury paralelních systémů. Propojují se jimi všechny výpočetní uzly a slouží k nahrání počátečních podmínek. Vzniklá topologie může být popsána pomocí teorie grafů (další informace v [4]). Použitá propojovací síť musí splňovat specifické požadavky systému a je velmi důležitá pro jeho výkonnost. Z hlediska propojení se sítě rozdělují na přímé a nepřímé. Nepřímá sít je dynamická a obsahuje přepínače, které směrují komunikaci mezi určenými uzly. Přímé sítě jsou tvořeny komunikačními propojeními mezi jednotlivými uzly. Takto navržená síť je statická a za běhu již neměnná.

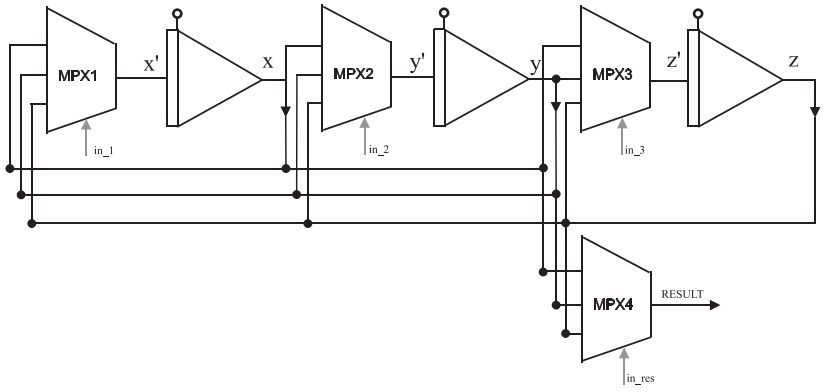
Pro méně rozsáhlé systémy je nejvhodnější křížový přepínač, který je vytvořen pomocí multiplexorů. V bakalářské práci Michaely Sekaninové [13] je navržena propojovací síť, která umožní propojení výstupu integrátorů na různé vstupy podle konkrétního výpočetního schématu (řešených diferenciálních rovnic). Toto řešení je nejvhodnější, protože má nejmenší zpoždění a umožňuje propojení více vstupů na jeden výstup. Také se využívá propojení řídících signálů pro integrátory pro nahrání počáteční podmínky a integračního kroku jednotlivých integrátorů. Výsledek je možné získat z libovolného výstupu zapojených integrátorů využitím zapojení multiplexoru. Obecná propojovací síť je na obrázku 9.1.



Obrázek 9.1: Obecná propojovací síť [13]

9.1.1 Implementace propojovací šítě

Již navržená propojovací síť pracuje s křížovými přepínači. Ty jsou tvořeny třívstupými multiplexory pro každý integrátor a jedním multiplexorem pro registr výsledku. Inicializace nastavení multiplexoru pro zvolenou diferenciální rovnici je staticky nastaveno před vlastním výpočtem. Multiplexory jsou ovládány řídícími signály pro výběr vstupu ($in1$, $in2$, $in3$, $in4$). Výhodou použití multiplexorů je možnost dynamické změny propojení sítě i během výpočtu.



Obrázek 9.2: Implementace propojovací sítě [13]

9.2 Nový návrh propojovací sítě

V novém návrhu propojovací sítě se snažím zaměřit na zjednodušení schématu propojovací sítě a tím snížit prostorovou náročnost vyžadovanou pro umístění na čip FPGA. Tím se zvýší efektivita využívaného místa na čipu a rychlost provedení výpočtu. Zjednodušení spočívá ve vyloučení sítě multiplexoru ze schématu. Toho lze dosáhnout s nově přepracovanou architekturou za pomoci textového předzpracování před samotnou syntézou VHDL. Komponenty integrátoru jsou vytvořeny jako jakési "krabičky", které lze na sebe volně navazovat a vytvořit tak kompletní schéma řešené úlohy.

Tímto způsobem je možné sestavit velmi elegantně celé schéma např do formátu XML. To by obsahovalo pro každou komponentu (integrátor, násobičku, sčítačku) vstupy, výstupy a nezbytné parametry. Tato data se textově předzpracují a nahradí VHDL kódy jednotlivých komponent. Přidají se propojovací signály. Před syntézou je vhodné provést časovou simulaci vzniklého obvodu a ověřit, zda je dodržen potřebný kmitočet hodinového signálu a dosažena potřebná rychlost reakce. Testovat lze i správnost a variantu zvoleného cílového typu obvodu. Výsledek se následně syntetizuje, pro danou technologickou skupinu obvodů, pro kterou se předpokládá implementace. Konfigurační XML soubor by mohl vypadat následovně:

```
<?xml version="1.0" encoding="utf-8"?>
<integrator typ="SPI" id="1">
  <input>
    <in1> id input component 1 </in1>
    <in2> id input component 2 </in2>
  </input>
  <output>
    <out1> id output component 1 </out1>
  </output>
  <icc> 1.0 </icc>
  <step> 0.125 </step>
</integrator>
```

Z počtu vstupů a parametrů vybere preprocesor vhodný integrátor. Tomu nastaví počáteční podmínku a krok. Případně přidá omezení řádu nebo postačující přesnost výpočtu. Po vložení všech integrátorů a komponent vytvoří propojení pomocí signálů a sběrnic. Po té preprocesor připraví komunikační rozhraní - registry, adresový dekodér a přenosové převodníky pro komunikaci s okolím schématu.

Rodiny architektur obvodů mívají stejné rysy alokace zdrojů, atp., čehož využívá kompilátor při syntéze. Po úspěšné simulaci a syntéze schématu řešené úlohy je možné otestovat vzniklý výpočetní systém bez multiplexorové sítě přímo na čipu FPGA.

Vstupem předzpracování jsou transformované vstupní diferenciální rovnice s využitím tvořících diferenciálních rovnic. Tyto tvořící diferenciální rovnice představují předpis propojení paralelního systému. Připojí k tomu počáteční data a sestaví kompletní paralelní schéma pro FPGA. V dalším vývoji může být zajímavé vytvořit nástroj, který zautomatizuje textové předzpracování a případně umožní i grafické vytváření schémat integrátorů, které následně převede do VHDL.

Kapitola 10

Analýza činnosti navrženého výpočetního systému

Postup numerického výpočtu soustavy (homogenních lineárních) diferenciálních rovnic pomocí Taylorovy řady, kterou využívá navržený výpočetní systém, lze analyzovat v několika směrech. Mezi ty nejdůležitější, pro výsledné používání navrženého výpočetního systému, patří přesnost výsledku, počet operací a množství zdrojů, které výpočetní systém využívá. Na první dvě zmíněné oblasti se zaměřím dále v této kapitole. Poslední zmíněnou metrikou, tedy množství využitých zdrojů, se zabývat nebudu.

Pro následující soustavu homogenních lineárních diferenciálních rovnic 1. řádu 10.1 – 10.2 a rovnici 10.3 bude provedeno srovnání efektivnosti výpočtu jednoho kroku metod Taylorovy řady a Runge-Kutta (a to v několika verzích obou metod):

$$y' = a \cdot y + b \cdot z \quad y(0) = y_0 \tag{10.1}$$
$$z' = c \cdot y + d \cdot z \quad z(0) = z_0 \tag{10.2}$$

$$z = e^t \cdot sin(t) \tag{10.3}$$

10.1 Runge-Kutta 2. řádu pro soustavu 10.1 – 10.2

Obecný tvar metody Runge-Kutta 2. řádu lze pro první krok numerického výpočtu soustavy 10.1 – 10.2 přepsat do tvaru:

$$y_1 = y_0 + \frac{1}{2}(k_{1,y} + k_{2,y}) \tag{10.4}$$
$$z_1 = z_0 + \frac{1}{2}(k_{1,z} + k_{2,z}) \tag{10.5}$$

kde pro koeficienty k_1 platí:

$$k_{1,y} = h(a \cdot y_0 + b \cdot z_0) \tag{10.6}$$
$$k_{1,z} = h(c \cdot y_0 + d \cdot z_0) \tag{10.7}$$

a pro koeficienty k_2:

$$k_{2,y} = h(a \cdot (y_0 + k_{1,y}) + b \cdot (z_0 + k_{1,z})) \tag{10.8}$$
$$k_{2,z} = h(c \cdot (y_0 + k_{1,y}) + d \cdot (z_0 + k_{1,z})) \tag{10.9}$$

Ze zápisu rovnic 10.4 – 10.9 vyplývá, že prováděné matematické operace jsou v navzájem si odpovídajících rovnicích stejné (pro výpočet koeficientů k_1, k_2 i celkového kroku). Díky tomu je možné tyto operace (násobení a sčítání) provést paralelně ve dvou oddělených procesorech (navržených integrátorech). Posloupnost provedených operacích při řešení soustavy je uvedena v tabulce 10.1. V jednotlivých procesorech se získají postupně elementárními operacemi hodnoty koeficientů k_1 a k_2, které odpovídají Y_4 a Z_4 a Y_{10} a Z_{10}. Na závěr se z vypočtených hodnot k_1, k_2 a počáteční podmínek dopočítá výsledná hodnota podle metody Runge-Kutta 2. řádu $y_1 = Y_{13}, z_1 = Y_{13}$.

Tabulka 10.1: Posloupnost operací při výpočtu soustavy metodou Runge-Kutta 2. řádu

Pořadí	**Procesor Y**	**Procesor Z**
1	$Y1 = a{\cdot}y_0$	$Z1 = c{\cdot}y_0$
2	$Y2 = b{\cdot}z_0$	$Z2 = d{\cdot}z_0$
3	$Y3 = Y1 + Y2$	$Z3 = Z1 + Z2$
4	$Y4 = h{\cdot}Y3$	$Z4 = h{\cdot}Z3$
5	$Y5 = y_0 + Y4$	$Z5 = y_0 + Y4$
6	$Y6 = z_0 + Z4$	$Z6 = z_0 + Z4$
7	$Y7 = a{\cdot}Y5$	$Z7 = c{\cdot}Z5$
8	$Y8 = b{\cdot}Y6$	$Z8 = d{\cdot}Z6$
9	$Y9 = Y7 + Y8$	$Z9 = Z7 + Z8$
10	$Y10 = h{\cdot}Y9$	$Z10 = h{\cdot}Z9$
11	$Y11 = Y4 + Y10$	$Z11 = Z4 + Z10$
12	$Y12 = 1/2{\cdot}Y11$	$Z12 = 1/2{\cdot}Z11$
13	$Y13 = y_0 + Y12$	$Z12 = z_0 + Z12$

10.2 Taylorova řada 2.řádu pro soustavu 10.1 – 10.2

Při použití zavedeného označení z rovnice 3.5 a Taylorovy metody druhého řádu dostaneme soustavu pro numerický výpočet y_1, z_1 ve tvaru:

$$y_1 = y_0 + DY1_0 + DY2_0 \tag{10.10}$$
$$z_1 = z_0 + DZ1_0 + DZ2_0 \tag{10.11}$$

Význam jednotlivých členů odpovídá:

$$DY1_0 = h \cdot (a \cdot y_0 + b \cdot z_0) \tag{10.12}$$
$$DZ1_0 = h \cdot (c \cdot y_0 + d \cdot z_0) \tag{10.13}$$

$$DY2_0 = \frac{h}{2} \cdot (a \cdot DY1_0 + b \cdot DZ1_0) \tag{10.14}$$
$$DZ2_0 = \frac{h}{2} \cdot (c \cdot DY1_0 + d \cdot DZ1_0) \tag{10.15}$$

Z rovnic 10.10 – 10.15 lze opět vysledovat možnost paralelního výpočtu v oddělených procesorech. Posloupnost operací prováděných metodou Taylorovy řady je rozepsána v tabulce 10.2.

K získání y_1 a z_1 je u Taylorovy řady 2. řádu potřeba 10 matematických operací u Runge-Kutta 2. řádu jich je potřeba 13. Na první pohled je metoda Taylorovy řady o 3 operace rychlejší než Runge-Kutta. Zrychlení je o jednu operaci násobení a dvě operace sčítání. Ze srovnání plyne vyšší efektivnost Taylorovy řady v počtu operací.

Tabulka 10.2: Posloupnost operací při výpočtu soustavy metodou Taylorovy řady 2. řádu

Pořadí	Procesor Y	Procesor Z
1	$Y1 = a \cdot y_0$	$Z1 = c \cdot y_0$
2	$Y2 = b \cdot z_0$	$Z2 = d \cdot z_0$
3	$Y3 = Y1 + Y2$	$Z3 = Z1 + Z2$
4	$Y4 = h \cdot Y3$	$Z4 = h \cdot Z3$
5	$Y5 = a \cdot Y4$	$Z5 = c \cdot Y4$
6	$Y6 = b \cdot Z4$	$Z6 = d \cdot Z4$
7	$Y7 = Y5 + Y6$	$Z7 = Z5 + Z6$
8	$Y8 = h/2 \cdot Y6$	$Z8 = h/2 \cdot Z6$
9	$Y9 = y_0 + Y4$	$Z9 = z_0 + Z4$
10	$Y10 = Y8 + Y9$	$Z10 = Z8 + Z9$

10.3 Runge-Kutta 4.řádu pro soustavu 10.1 – 10.2

Runge-Kutta 4. řádu je jedna z nejrozšířenějších a nejpoužívanějších metod ze tříd Runge-Kutta. Soustavy 10.1 – 10.2 lze přepsat na obecný tvar metody Runge-Kutta 4. řádu:

$$y_1 = y_0 + \frac{1}{6}(k_{1,y} + 2 \cdot k_{2,y} + 2 \cdot k_{3,y} + k_{4,y}) \tag{10.16}$$

$$z_1 = z_0 + \frac{1}{6}(k_{1,z} + 2 \cdot k_{2,z} + 2 \cdot k_{3,z} + k_{4,z}) \tag{10.17}$$

kde pro koeficienty k_1 platí:

$$k_{1,y} = h(a \cdot y_0 + b \cdot z_0) \tag{10.18}$$

$$k_{1,z} = h(c \cdot y_0 + d \cdot z_0) \tag{10.19}$$

pro koeficienty k_2:

$$k_{2,y} = h(a \cdot (y_0 + \frac{1}{2}k_{1,y}) + b \cdot (z_0 + \frac{1}{2}k_{1,z})) \tag{10.20}$$

$$k_{2,z} = h(c \cdot (y_0 + \frac{1}{2}k_{1,y}) + d \cdot (z_0 + \frac{1}{2}k_{1,z})) \tag{10.21}$$

pro koeficienty k_3:

$$k_{3,y} = h(a \cdot (y_0 + \frac{1}{2}k_{2,y}) + b \cdot (z_0 + \frac{1}{2}k_{2,z})) \tag{10.22}$$

$$k_{3,z} = h(c \cdot (y_0 + \frac{1}{2}k_{2,y}) + d \cdot (z_0 + \frac{1}{2}k_{2,z})) \tag{10.23}$$

a pro koeficienty k_4:

$$k_{4,y} = h(a \cdot (y_0 + k_{3,y}) + b \cdot (z_0 + k_{3,z})) \tag{10.24}$$

$$k_{4,z} = h(c \cdot (y_0 + k_{3,y}) + d \cdot (z_0 + k_{3,z})) \tag{10.25}$$

Ze zápisu rovnic 10.16 – 10.25 je tabulka pro paralelně prováděné operace na nezávislých procesorech.

Tabulka 10.3: Posloupnost operací při výpočtu soustavy metodou Runge-Kutta 4. řádu

Pořadí	**Procesor Y**	**Procesor Z**
1	$Y1 = a{\cdot}y_0$	$Z1 = c{\cdot}y_0$
2	$Y2 = b{\cdot}z_0$	$Z2 = d{\cdot}z_0$
3	$Y3 = Y1 + Y2$	$Z3 = Z1 + Z2$
4	$Y4 = h{\cdot}Y3$	$Z4 = h{\cdot}Z3$
5	$Y5 = 1/2 \cdot Y4$	$Z5 = 1/2 * Y4$
6	$Y6 = 1/2 \cdot Z4$	$Z6 = 1/2 * Z4$
7	$Y7 = y_0 + Y5$	$Z7 = y_0 + Z5$
8	$Y8 = z_0 + Y6$	$Z8 = z_0 + Z6$
9	$Y9 = a{\cdot}Y7$	$Z9 = c{\cdot}Z7$
10	$Y10 = b{\cdot}Y8$	$Z10 = d{\cdot}Z8$
11	$Y11 = Y9 + Y10$	$Z11 = Z9 + Z10$
12	$Y12 = h{\cdot}Y11$	$Z12 = h{\cdot}Z11$
13	$Y13 = 1/2 \cdot Y12$	$Z13 = 1/2 * Y12$
14	$Y14 = 1/2 \cdot Z12$	$Z14 = 1/2 * Z12$
15	$Y15 = y_0 + Y13$	$Z15 = y_0 + Z13$
16	$Y16 = z_0 + Y14$	$Z16 = z_0 + Z14$
17	$Y17 = a{\cdot}Y15$	$Z17 = c{\cdot}Z15$
18	$Y18 = b{\cdot}Y16$	$Z18 = d{\cdot}Z16$
19	$Y19 = Y17 + Y18$	$Z19 = Z17 + Z18$
20	$Y20 = h{\cdot}Y19$	$Z20 = h{\cdot}Z19$
21	$Y21 = y_0 + Y20$	$Z21 = y_0 + Y20$
22	$Y22 = z_0 + Z20$	$Z22 = z_0 + Z20$
23	$Y23 = a{\cdot}Y21$	$Z23 = c{\cdot}Z21$
24	$Y24 = b{\cdot}Y22$	$Z24 = d{\cdot}Z22$
25	$Y25 = Y23 + Y24$	$Z25 = Z23 + Z24$
26	$Y26 = h{\cdot}Y25$	$Z26 = h{\cdot}Z25$
27	$Y27 = 2{\cdot}Y12$	$Z27 = 2{\cdot}Z12$
28	$Y28 = 2{\cdot}Y20$	$Z28 = 2{\cdot}Z20$
29	$Y29 = Y4 + Y27$	$Z29 = Z4 + Z27$
30	$Y30 = Y28 + Y29$	$Z30 = Z28 + Z29$
31	$Y31 = Y26 + Y30$	$Z31 = Z26 + Z30$
32	$Y32 = 1/6{\cdot}Y31$	$Z32 = 1/6{\cdot}Z21$
33	$Y33 = y_0 + Y32$	$Z32 = z_0 + Z32$

10.4 Taylorova řada 4.řádu pro soustavu 10.1 – 10.2

Soustava pro numerický výpočet Taylorovou řadou 4. řádu má obecný tvar:

$$y_1 = y_0 + DY1_0 + DY2_0 + DY3_0 + DY4_0 \tag{10.26}$$
$$z_1 = z_0 + DZ1_0 + DZ2_0 + +DZ3_0 + DZ4_0 \tag{10.27}$$

Význam jednotlivých členů odpovídá:

$$DY1_0 = h \cdot (a \cdot y_0 + b \cdot z_0) \tag{10.28}$$
$$DZ1_0 = h \cdot (c \cdot y_0 + d \cdot z_0) \tag{10.29}$$

$$DY2_0 = \frac{h}{2} \cdot (a \cdot DY1_0 + b \cdot DZ1_0) \tag{10.30}$$
$$DZ2_0 = \frac{h}{2} \cdot (c \cdot DY1_0 + d \cdot DZ1_0) \tag{10.31}$$

$$DY3_0 = \frac{h}{3} \cdot (a \cdot DY2_0 + b \cdot DZ2_0) \tag{10.32}$$
$$DZ3_0 = \frac{h}{3} \cdot (c \cdot DY2_0 + d \cdot DZ2_0) \tag{10.33}$$

$$DY4_0 = \frac{h}{4} \cdot (a \cdot DY3_0 + b \cdot DZ3_0) \tag{10.34}$$
$$DZ4_0 = \frac{h}{4} \cdot (c \cdot DY3_0 + d \cdot DZ3_0) \tag{10.35}$$

Z rovnic 10.26 – 10.35 je vytvořena následující tabulka výpočetních operacích vykonávaných na jednotlivých procesorech.

Tabulka 10.4: Posloupnost operací při výpočtu soustavy metodou Taylorovy řady 4. řádu

Pořadí	**Procesor Y**	**Procesor Z**
1	$Y1 = a \cdot y_0$	$Z1 = c \cdot y_0$
2	$Y2 = b \cdot z_0$	$Z2 = d \cdot z_0$
3	$Y3 = Y1 + Y2$	$Z3 = Z1 + Z2$
4	$Y4 = h \cdot Y3$	$Z4 = h \cdot Z3$
5	$Y5 = a \cdot Y4$	$Z5 = c \cdot Y4$
6	$Y6 = b \cdot Z4$	$Z6 = d \cdot Z4$
7	$Y7 = Y5 + Y6$	$Z7 = Z5 + Z6$
8	$Y8 = h/2 \cdot Y6$	$Z8 = h/2 \cdot Z6$
9	$Y9 = a \cdot Y8$	$Z9 = c \cdot Y8$
10	$Y10 = b \cdot Z8$	$Z10 = d \cdot Z8$
11	$Y11 = Y9 + Y10$	$Z11 = Z9 + Z10$
12	$Y12 = h/3 \cdot Y11$	$Z12 = h/3 \cdot Z11$
13	$Y13 = a \cdot Y12$	$Z13 = c \cdot Y12$
14	$Y14 = b \cdot Z12$	$Z14 = d \cdot Z12$
15	$Y15 = Y13 + Y14$	$Z15 = Z13 + Z14$
16	$Y16 = h/4 \cdot Y15$	$Z16 = h/4 \cdot Z15$
17	$Y17 = y_0 + Y4$	$Z17 = z_0 + Z4$
18	$Y18 = Y8 + Y17$	$Z18 = Z8 + Z17$
19	$Y19 = Y12 + Y18$	$Z19 = Z12 + Z18$
20	$Y20 = Y16 + Y19$	$Z20 = Z16 + Z19$

Z tabulek 10.3 a 10.4 vyplývá vyšší efektivnost metody Taylorovy řady. V porovnání při použití Runge-Kutta 4. řádu každý procesor provede pro zadanou soustavu 33 operací, Taylorova řada provede pouze 20 operací. Čím vyšší je řád metody, tím se počet operací stává ještě příznivější pro Taylorovu řadu.

10.5 Analýza numerického výpočtu pro rovnici 10.3

Tuto rovnici je možné v navržené architektuře zpracovat několika způsoby. Prvním a na pohled zřetelným a přímočarým řešením je použití násobícího integrátoru. Taktéž je možné rovnici upravit do následujícího tvaru:

$$z = \frac{sin(t)}{e^{-t}} \tag{10.36}$$

Při této úpravě se nabízí využití navrženého dělícího integrátoru. Poslední možností je použití metodiky tvořících diferenciálních rovnic a přepsat tuto rovnici následovně:

$$z' = e^t \cdot sin(t) + e^t \cdot cos(t) \tag{10.37}$$

kde $e^t \cdot sin(t)$ je z:

$$z' = z + e^t \cdot cos(t) \tag{10.38}$$

vytvoříme novou proměnnou v:

$$v = e^t \cdot cos(t) \tag{10.39}$$

a zderivujeme:

$$v' = e^t \cdot cos(t) \tag{10.40}$$

$$v' = e^t \cdot cos(t) - e^t \cdot sin(t) \tag{10.41}$$

a nahradíme do výsledného tvaru:

$$z' = z + v \quad \&0 \tag{10.42}$$

$$v' = v - z \quad \&1 \tag{10.43}$$

Což vybízí k použití součtového a rozdílového integrátoru. V dalších podkapitolách rozeberu postup výpočtu, počet operací a přesnost výsledku metody Taylorovy řady vůči metodám Runge-Kutta.

10.6 Analýza součtu a rozdílu pro rovnici 10.3

Zpracovávaná rovnice je převedena na soustavu rovnic:

$$z' = z + v \quad \&0 \tag{10.44}$$

$$v' = v - z \quad \&1 \tag{10.45}$$

Vzhledem k tomu, že jsou zpracovávány dvě rovnice, budou použity dva paralelní procesory. Výhodou tohoto řešení je absence Pascalova trojúhelníku při výpočtu jednotlivých členů. Proto počet operací potřebných pro dosažení stejného výsledku, jako u integrace součinu/podílu, je menší. Počet operací na výpočet jednotlivých členu je konstantní. K dosažení výsledku využity následující vzorce:

$$z_1 = z_0 + DZ1 + DZ2 + DZ3 + DZ4 \tag{10.46}$$

$$\begin{aligned} DZ1 &= z_0 + v_0 && (10.47)\\ DZ2 &= DZ1 + DV1 && (10.48)\\ DZ3 &= DZ2 + DV2 && (10.49)\\ DZ4 &= DZ3 + DV3 && (10.50) \end{aligned}$$

$$z_1 = z_0 + DZ1 + DZ2 + DZ3 + DZ4 \tag{10.51}$$

$$\begin{aligned} DZ1 &= v_0 - z_0 && (10.52)\\ DZ2 &= DV1 - DZ1 && (10.53)\\ DZ3 &= DV2 - DZ2 && (10.54)\\ DZ4 &= DV3 - DZ3 && (10.55) \end{aligned}$$

Následuje tabulka s přehledem operací obou procesorů.

Tabulka 10.5: Posloupnost operací při výpočtu rovnice 10.3 pomocí Taylorovy řady 4. řádu a tvořících diferenciálních rovnic

Pořadí	**Procesor Z**	**Procesor V**
1	$Z1 = y_0 + v_0$	$V1 = v_0 - y_0$
2	$Z2 = Z1 + V1$	$V2 = V1 - Z1$
3	$Z3 = Z2 + V2$	$V3 = V2 - Z2$
4	$Z4 = Z3 + V3$	$V4 = V3 - Z3$
5	$Z5 = z_0 + Z1$	$V5 = v_0 + V1$
6	$Z6 = Z2 + Z5$	$V6 = V2 + V5$
7	$Z7 = Z3 + Z6$	$V7 = V3 + V6$
8	$Z8 = Z4 + Z7$	$V8 = V4 + V7$

V tabulce 10.5 jsou operace pro výpočet následující hodnoty rovnice 10.3 pomocí 8 operací pro dva procesory.

10.7 Analýza součinu

Pro operace součinu upravíme rovnici 10.3 následovně:

$$y = u \cdot v \quad \&0 \tag{10.56}$$

opět je použita substituce:

$$v = e^{-t}, \quad u = sin(t) \tag{10.57}$$

Jednotlivé členy DU1 – DUx a DV1 – DVx jsou vypočteny obdobně způsobem, jako v případu integrace součinu 10.80 – 10.65. Pro výpočet 4. řádu Taylorovy se na rozdíl od Integrace součinu používá i čtvrtý řád proměnných (DU4 a DV4).

$$DU1 = u' \cdot h \quad \Rightarrow \quad u' = \frac{DU1}{h} \tag{10.58}$$

$$DU2 = u'' \cdot \frac{h^2}{2!} \quad \Rightarrow \quad u'' = \frac{2! \cdot DU2}{h^2} \tag{10.59}$$

$$DU3 = u''' \cdot \frac{h^3}{3!} \quad \Rightarrow \quad u''' = \frac{3! \cdot DU3}{h^3} \tag{10.60}$$

$$DU4 = u^{(4)} \cdot \frac{h^4}{4!} \quad \Rightarrow \quad u(4) = \frac{4! \cdot DU4}{h^4} \tag{10.61}$$

$$DV1 = v' \cdot h \quad \Rightarrow \quad v' = \frac{DV1}{h} \tag{10.62}$$

$$DV2 = v'' \cdot \frac{h^2}{2!} \quad \Rightarrow \quad v'' = \frac{2 \cdot DV2}{h^2} \tag{10.63}$$

$$DV3 = v''' \cdot \frac{h^3}{3!} \quad \Rightarrow \quad v''' = \frac{6 \cdot DV3}{h^3} \tag{10.64}$$

$$DV4 = v^{(4)} \cdot \frac{h^4}{4!} \quad \Rightarrow \quad v^{(4)} = \frac{4! \cdot DV4}{h^4} \tag{10.65}$$

Výsledný výpočet tedy bude následující:

$$y_1 = y_0 + DY1 + DY2 + DY3 + DY4 \tag{10.66}$$

$$DY1 = DU1 \cdot v + u \cdot DV1 \tag{10.67}$$

$$DY2 = DU2 \cdot v + DU1 \cdot DV1 + u \cdot DV2 \tag{10.68}$$

$$DY3 = DU3 \cdot v + DU2 \cdot DV1 + DV2 \cdot DU1 + u \cdot DV3 \tag{10.69}$$

$$DY4 = DU4 \cdot v + DU3 \cdot DV1 + DU2 \cdot DV2 + DU1 \cdot DV3 + u \cdot DV4 \tag{10.70}$$

Následuje tabulka operací provedených v procesoru pro Taylorovu řadu 4. řádu.

Tabulka 10.6: Posloupnosti operací při výpočtu součinu metodou Taylorovy řady 4. řádu

Pořadí	**Procesor Taylorovy řady**
1	$Y1 = DU1 \cdot v$
2	$Y2 = DV1 \cdot u$
3	$Y3 = Y1 + Y2$
4	$Y4 = DU2 \cdot v$
5	$Y5 = DU1 \cdot DV1$
6	$Y6 = u \cdot DV2$
7	$Y7 = Y4 + Y5$
8	$Y8 = Y6 + Y7$
9	$Y9 = DU3 \cdot v$
10	$Y10 = DU2 \cdot DV1$
11	$Y11 = DV1 \cdot DU2$
12	$Y12 = u \cdot DV3$
13	$Y13 = Y9 + Y10$
14	$Y14 = Y11 + Y13$
15	$Y15 = Y12 + Y14$
16	$Y16 = DU4 \cdot v$
17	$Y17 = DU3 \cdot DV1$
18	$Y18 = DU2 \cdot DV2$
19	$Y19 = DU1 \cdot DV3$
20	$Y20 = u \cdot DV4$
21	$Y21 = Y16 + Y17$
22	$Y22 = Y18 + Y21$
23	$Y23 = Y19 + Y22$
24	$Y24 = Y20 + Y23$
25	$Y25 = y_0 + Y3$
26	$Y26 = Y8 + Y25$
27	$Y27 = Y15 + Y26$
28	$Y28 = Y24 + Y27$

Pro provedení součinu je potřeba 28 operací. Pro výpočet Taylorovou metodou 5. řádu by bylo potřeba o 12 operací více. Lze zobecnit, že pro DYn člen je potřeba právě $2 \cdot n + 2$ operací navíc od Taylorovy metody o jeden řád nižší.

10.8 Analýza podílu

Pro podíl upravíme rovnici 10.3 následovně:

$$y = \frac{u}{v} \quad \&0 \tag{10.71}$$

kde u a v jsou substituované funkce:

$$v = e^{-t}, \quad u = sin(t) \tag{10.72}$$

Jednotlivé členy DU1 – DUx a DV1 – DVx jsou vypočteny jako v případu součinu 10.65 – 10.80. Pro výpočet 4. řádu Taylorovy se na rozdíl od integrace podílu používá i čtvrtý řád proměnných (DU4 a DV4).

Rovnice počítané v procesoru jsou následující:

$$y_1 = y_0 + DY1 + DY2 + DY3 + DY4 \tag{10.73}$$

$$DY1 = \frac{1}{v} \cdot (DU1 - y_0 \cdot DV1) \tag{10.74}$$

$$DY2 = \frac{1}{v} \cdot (DU2 - DY1 \cdot DV1 - y_0 \cdot DV2) \tag{10.75}$$

$$DY3 = \frac{1}{v} \cdot (DU3 - DY2 \cdot DV1 - DY1 \cdot DV2 - y_0 \cdot DV3) \tag{10.76}$$

$$DY4 = \frac{1}{v} \cdot (DU4 - DY3 \cdot DV1 - DY2 \cdot DV2 - DY1 \cdot DV3 - y_0 \cdot DV4) \tag{10.77}$$

V tabulce 10.9 jsou matematické operace vykonané pro výpočet podílu.

Tabulka 10.7: Posloupnosti operací při výpočtu podílu metodou Taylorovy řady 4. řádu

Pořadí	**Procesor Taylorovy řady**
1	$Y1 = 1/v$
2	$Y2 = DV1 \cdot y_0$
3	$Y3 = DU1 - Y2$
4	$Y4 = Y3 \cdot Y1$
5	$Y5 = DU1 \cdot Y4$
6	$Y6 = y_0 \cdot DV2$
7	$Y7 = DU2 - Y5$
8	$Y8 = Y7 - Y6$
9	$Y9 = Y1 \cdot Y8$
10	$Y10 = Y9 \cdot DV1$
11	$Y11 = Y4 \cdot DV2$
12	$Y12 = y_0 \cdot DV3$
13	$Y13 = DU3 - Y10$
14	$Y14 = Y13 - Y11$
15	$Y15 = Y14 - Y12$
16	$Y16 = Y1 \cdot Y15$
17	$Y17 = Y16 \cdot DV1$
18	$Y18 = Y9 \cdot DV2$
19	$Y19 = Y4 \cdot DV3$
20	$Y20 = y_0 \cdot DV4$
21	$Y21 = DU4 - Y17$
22	$Y22 = Y21 - Y18$
23	$Y23 = Y22 - Y19$
24	$Y24 = Y23 - Y20$
25	$Y25 = Y1 \cdot Y24$
26	$Y26 = y_0 + Y4$
27	$Y27 = Y9 + Y26$
28	$Y28 = Y16 + Y27$
29	$Y29 = Y25 + Y28$

Pro výpočet podílu se provede metodou Taylorovy řady 29 operací. Počet operací pro zvětšení o 1 řád (tzn. přidává se pouze následující člen – DYn) výpočtu je: $2 \cdot n + 2$.

10.9 Analýza integrace součinu

Rovnici 10.3 pro integraci součinu upravíme do tvaru:

$$y' = u \cdot v \quad \&0 \tag{10.78}$$

kde u a v jsou substituované funkce:

$$v = e^t, \quad u = sin(t) \tag{10.79}$$

Členy DU1 – DUx a DV1 – DVx jsou vypočteny v paralelně kooperujícím procesoru s následným přenosem do integrátoru počítajícího integraci součinu. Význam je následující:

$$DU1 = u' \cdot h \quad \Rightarrow \quad u' = \frac{DU1}{h} \tag{10.80}$$

$$DU2 = u'' \cdot \frac{h^2}{2!} \quad \Rightarrow \quad u'' = \frac{2 \cdot DU2}{h^2} \tag{10.81}$$

$$DU3 = u''' \cdot \frac{h^3}{3!} \quad \Rightarrow \quad u''' = \frac{6 \cdot DU3}{h^3} \tag{10.82}$$

$$DV1 = v' \cdot h \quad \Rightarrow \quad v' = \frac{DV1}{h} \tag{10.83}$$

$$DV2 = v'' \cdot \frac{h^2}{2!} \quad \Rightarrow \quad v'' = \frac{2 \cdot DV2}{h^2} \tag{10.84}$$

$$DV3 = v''' \cdot \frac{h^3}{3!} \quad \Rightarrow \quad v''' = \frac{6 \cdot DV3}{h^3} \tag{10.85}$$

Potom upravené (zkrácení po dosazení za derivace u a v) členy včetně výpočtu dalšího kroku metody Taylorovy řady 4. řádu vypadají následovně:

$$y_1 = y_0 + DY1 + DY2 + DY3 + DY4 \tag{10.86}$$

$$DY1 = h \cdot (u \cdot v) \tag{10.87}$$

$$DY2 = \frac{h}{2} \cdot (DU1 \cdot v + u \cdot DV1) \tag{10.88}$$

$$DY3 = \frac{h}{3} \cdot (DU2 \cdot v + DU1 \cdot DV1 + u \cdot DV2) \tag{10.89}$$

$$DY4 = \frac{h}{4} \cdot (DU3 \cdot v + DU2 \cdot DV1 + DV2 \cdot DU1 + u \cdot DV3) \tag{10.90}$$

V tomto procesoru (integrátoru) budou provedeny matematické operace v prvním sloupci tabulky 10.8. Ve druhém sloupci jsou ke srovnání operace metody Runge-Kutta 4. řádu, které vychází z následujících vzorců:

$$y_1 = y_0 + \frac{1}{6}(k_{1,y} + 2 \cdot k_{2,y} + 2 \cdot k_{3,y} + k_{4,y}) \tag{10.91}$$

kde pro koeficient k_1 platí:

$$k_{1,y} = h(u \cdot v) \tag{10.92}$$

pro koeficient k_2:

$$k_{2,y} = h((u + \frac{1}{2} \cdot k_{1,u}) \cdot (v + \frac{1}{2} \cdot k_{1,v}))) \tag{10.93}$$

pro koeficienty k_3:

$$k_{3,y} = h((u + \frac{1}{2} \cdot k_{2,u}) \cdot (v + \frac{1}{2} \cdot k_{2,v}))) \tag{10.94}$$

a pro koeficienty k_4:

$$k_{4,y} = h(u + k_{3,u}) \cdot (v + k_{3,v})) \tag{10.95}$$

Kde koeficienty $k_{1,u}$ až $k_{3,u}$ a $k_{1,v}$ až $k_{3,v}$ jsou přeneseny z procesorů pro výpočet u a v.

Tabulka 10.8: Porovnání posloupnosti operací při výpočtu integrace součinu metodami Taylorovy řady 4. řádu a Runge-Kutta 4. řádu

Pořadí	**Procesor Taylorovy řady**	**Procesor Runge-Kutta**
1	$Y1 = u{\cdot}v$	$R1 = u{\cdot}v$
2	$Y2 = h{\cdot}Y3$	$R2 = h{\cdot}R1$
3	$Y3 = DU1{\cdot}v$	$R3 = 1/2{\cdot}U2$
4	$Y4 = u{\cdot}DV1$	$R4 = 1/2{\cdot}V2$
5	$Y5 = Y3 + Y4$	$R5 = u + R3$
6	$Y6 = h/2{\cdot}Y5$	$R6 = v + R4$
7	$Y7 = DU2{\cdot}v$	$R7 = R5{\cdot}R6$
8	$Y8 = DU1{\cdot}DV1$	$R8 = h{\cdot}R7$
9	$Y9 = u{\cdot}DV2$	$R9 = 1/2{\cdot}U8$
10	$Y10 = Y7 + Y8$	$R10 = 1/2{\cdot}V8$
11	$Y11 = Y9 + Y10$	$R11 = u + R9$
12	$Y12 = h/3{\cdot}Y11$	$R12 = v + R10$
13	$Y13 = DU3{\cdot}v$	$R13 = R11{\cdot}R12$
14	$Y14 = DU2{\cdot}DV1$	$R14 = h{\cdot}R13$
15	$Y15 = DU1{\cdot}DV2$	$R15 = u + U14$
16	$Y16 = u{\cdot}DV3$	$R16 = v + V14$
17	$Y17 = h/4{\cdot}Y16$	$R17 = R15{\cdot}R16$
18	$Y18 = y_0 + Y2$	$R18 = h{\cdot}R17$
19	$Y19 = Y6 + Y18$	$R19 = 2{\cdot}R8$
20	$Y20 = Y12 + Y19$	$R20 = 2{\cdot}R14$
21	$Y21 = Y17 + Y20$	$R21 = R2 + R19$
22	-	$R22 = R20 + R21$
23	-	$R23 = R18 + R22$
24	-	$R24 = 1/6 \cdot R23$
25	-	$R25 = y_0 + R24$

10.10 Analýza integrace podílu

Pro integraci podílu upravíme rovnici 10.3 následovně:

$$y' = \frac{u}{v} \quad \&0 \tag{10.96}$$

kde u a v jsou substituované funkce:

$$v = e^{-t}, \quad u = sin(t) \tag{10.97}$$

Členy DU1 – DUx a DV1 – DVx jsou vypočteny stejným způsobem, jako v případu integrace součinu.

Po upravě členů dostaneme následující rovnice:

$$y_1 = y_0 + DY1 + DY2 + DY3 + DY4 \tag{10.98}$$

$$DY1 = \frac{1}{v} \cdot (u \cdot h) \tag{10.99}$$

$$DY2 = \frac{1}{2v} \cdot (DU1 \cdot h - DY1 \cdot DV1) \tag{10.100}$$

$$DY3 = \frac{1}{3v} \cdot (DU2 \cdot h - 2 \cdot DY2 \cdot DV1 - DY1 \cdot DV2) \tag{10.101}$$

$$DY4 = \frac{1}{4v} \cdot (DU3 \cdot h - 3 \cdot DY3 \cdot DV1 - 2 \cdot DY2 \cdot DV2 - DY1 \cdot DV3) \tag{10.102}$$

V prvním sloupci tabulky 10.9 jsou matematické operace vykonané pro integraci podílu. Ve druhém sloupci jsou ke srovnání operace metody Runge-Kutta 4. řádu, které vychází z následujících vzorců:

$$y_1 = y_0 + \frac{1}{6}(k_{1,y} + 2 \cdot k_{2,y} + 2 \cdot k_{3,y} + k_{4,y}) \tag{10.103}$$

kde pro koeficient k_1 platí:

$$k_{1,y} = h(\frac{u}{v}) \tag{10.104}$$

pro koeficient k_2:

$$k_{2,y} = h\frac{u + \frac{1}{2} \cdot k_{1,u}}{v + \frac{1}{2} \cdot k_{1,v}} \tag{10.105}$$

pro koeficienty k_3:

$$k_{3,y} = h\frac{u + \frac{1}{2} \cdot k_{2,u}}{v + \frac{1}{2} \cdot k_{2,v}} \tag{10.106}$$

a pro koeficienty k_4:

$$k_{4,y} = h\frac{u + k_{3,u}}{v + k_{3,v}} \tag{10.107}$$

Kde koeficienty $k_{1,u}$ až $k_{3,u}$ a $k_{1,v}$ až $k_{3,v}$ jsou přeneseny z procesorů počítajících u a v.

Tabulka 10.9: Porovnání posloupnosti operací při výpočtu integrace součinu metodami Taylorovy řady 4. řádu a Runge-Kutta 4. řádu

Pořadí	**Procesor Taylorovy řady**	**Procesor Runge-Kutta**
1	$Y1 = 1/v$	$R1 = u/v$
2	$Y2 = h{\cdot}u$	$R2 = h{\cdot}R1$
3	$Y3 = Y1{\cdot}Y2$	$R3 = 1/2{\cdot}U2$
4	$Y4 = 1/2{\cdot}Y1$	$R4 = 1/2{\cdot}V2$
5	$Y5 = DU1 \cdot h$	$R5 = u + R3$
6	$Y6 = Y3{\cdot}DV1$	$R6 = v + R4$
7	$Y7 = Y5 - Y6$	$R7 = R5/R6$
8	$Y8 = Y4 \cdot Y7$	$R8 = h{\cdot}R7$
9	$Y9 = 1/3{\cdot}Y1$	$R9 = 1/2{\cdot}U8$
10	$Y10 = DU2{\cdot}h$	$R10 = 1/2{\cdot}V8$
11	$Y11 = 2 \cdot Y8$	$R11 = u + R9$
12	$Y12 = Y11 \cdot DV1$	$R12 = v + R10$
13	$Y13 = Y3{\cdot}DV2$	$R13 = R11/R12$
14	$Y14 = Y10 - Y12$	$R14 = h{\cdot}R13$
15	$Y15 = Y14 - Y13$	$R15 = u + U14$
16	$Y16 = Y9{\cdot}Y15$	$R16 = v + V14$
17	$Y17 = 1/4{\cdot}Y1$	$R17 = R15/R16$
18	$Y18 = DU3{\cdot}h$	$R18 = h{\cdot}R17$
19	$Y19 = 3 \cdot Y15$	$R19 = 2{\cdot}R8$
20	$Y20 = Y19 \cdot DV1$	$R20 = 2{\cdot}R14$
21	$Y21 = 2 \cdot Y8$	$R21 = R2 + R19$
22	$Y22 = Y21 \cdot DV2$	$R22 = R20 + R21$
23	$Y23 = Y3 \cdot DV3$	$R23 = R18 + R22$
24	$Y24 = Y18 - Y20$	$R24 = 1/6 \cdot R23$
25	$Y25 = Y24 - Y22$	$R25 = y_0 + R24$
26	$Y26 = Y25 - Y23$	-
27	$Y27 = Y17 \cdot Y26$	-
28	$Y28 = y_0 + Y3$	-
29	$Y29 = Y8 + Y28$	-
30	$Y30 = Y16 + Y29$	-
31	$Y31 = Y27 + Y30$	-

Z tabulky 10.9 je patrné, že metoda Taylorovy řady má o 6 operací více. Při bližším pohledu zjistíme, že Runge-Kutta využívá operaci dělení, která je pro hardwarové zpracování velmi náročná na zdroje i čas. Experimentálním propočtem jsme zjistili, že výpočet Taylorovou řadou 4. řádu pro dělení je pro tuto rovnici přesnější o 5 desetinných míst než Runge-Kutta 4. řádu.

Kapitola 11

Závěr

V této práci je popsán princip metody Taylorovy řady využívaný pro řešení diferenciálních rovnic. Realizace výpočtu je provedena několika druhy numerických integrátorů. Tyto integrátory jsou propojeny pomocí propojovací sítě. Práce uvádí analýzu paralelních vlastností metody Taylorovy řady a představuje šest základních paralelních výpočetních operací v této metodě. Jedná se o operace součtu, rozdílu, součinu, podílu, integrace součinu a integrace podílu.

Byl proveden návrh specializované výpočetní architektury, která vychází z podobného systému vypracovaného dříve a odstraňuje jeho nedostatky. Návrh byl přizpůsoben cílové technologii FPGA. Při zpracování jsem se zaměřil na jednoduchou rozšiřitelnost, srozumitelnost a na to, aby bylo možné architekturu dále snadno rozvíjet. Během realizace byla také navržena nová efektivnější propojovací síť, která neobsahuje multiplexory pro propojení jednotlivých komponent.

Pro všechny řešené operace byly vytvořeny algoritmy pro řízení výpočtu. Mikroprogramy, které jsou jejich přepisem (do programovacího jazyka VHDL), ovládají mikroprogramový řadič, který řídí výpočetní bloky daných integrátorů. Integrátory byly simulovány v programech ModelSim a následně testovány na FITkitu.

Navržený systém byl ověřen na vhodných příkladech a srovnán s metodami Runge-Kutta. Pro operace součtu a rozdílu se při zvětšujícím se řádu Taylorova metoda stává výhodnější jak v počtu operací, tak i v přesnosti. Počet operací součinu, podílu a součinu s integrací je nižší než zpracování metodami Runge-Kutta, nicméně při zvyšujícím se řádu nenarůstá. Počet operací při zpracování podílu s integrací vypadá na první pohled výhodněji než s metodou Runge-Kutta. Ovšem při bližším zkoumání lze zjistit, že používá několikrát operaci dělení, což je velmi nákladné na zdroje i čas. Navíc je i v tomto případě Taylorova řada přesnější o několik řádů.

Možnost dalšího vývoje spatřuji ve zpracování nového přístupu k propojovací síti nebo ve vytvoření grafického editoru pro vizuální zadávání a zautomatizování řešení soustav diferenciálních rovnic. Zajímavé rozšíření by mohlo být i zavedení zřetězeného zpracování výpočtů (pipeline).

Literatura

[1] A. Ralston: *Základy numerické matematiky*. Academia, 1973.

[2] F. Jirásek a J. Benda: *Matematika: pro bakalářské studium*. Ekopress, první vydání, 2006, ISBN 80-86929-02-7.

[3] F. Matečný: *Simulátor procesoru s operací dělení*. Bakalářská práce, FIT VUT v Brně, Brno, 2016.

[4] HSU, Lih-Hsing and Cheng-Kuan LIN: *Graph theory and interconnection networks*. Boca Raton: CRC Press, 2009, Dostupné také z: http://ksu.edu.sa/sites/py/ar/mpy/departments/math/learnResources/ResourceCenter/Documents/CRC.Graph.Theory.and.Interconnection.Networks.Sep.2008.eBook-DDU.pdf.

[5] J. Brabec a J. Hrůza: *Matematická analýza II*. SNTL - Nakladatelství technické literatury, 1986.

[6] J. Diblík a J. Baštinec: *Matematika III*. VUT v Brně, první vydání, 1991, ISBN 80-214-0315-2.

[7] J. Opálka: *Automatické řízení výpočtu*. Bakalářská práce, FIT VUT v Brně, Brno, 2014.

[8] L. Čermák a R. Hlavička: *Numerické metody*. akademické nakladatelství Cerm, 2005, ISBN 80-214-3071-0.

[9] M. Čambor: *Paralelní řešení parciálních diferenciálních rovnic*. Diplomová práce, FIT VUT v Brně, Brno, 2011.

[10] M. Bobek, J. Haška, I. Serba a M. Lukeš: *Analogové počítače*. SNTL - Nakladatelství technické literatury, 1982.

[11] M. Kraus: *Elementární procesor specializovaného paralelního systému*. Diplomová práce, FIT VUT v Brně, Brno, 2006.

[12] M. Kraus: *Paralelní výpočetní architektury založené na numerické integraci*. Disertační práce, FIT VUT v Brně, Brno, 2013.

[13] M. Sekaninová: *Propojovací systém paralelních ALU pro numerickou integraci*. Bakalářská práce, FIT VUT v Brně, Brno, 2015.

[14] V. Dvořák a V. Drábek: *Architektura procesorů*. nakladatelství VUTIUM, první vydání, 1999, ISBN 80-214-1458-8.

[15] V. Závada: *Simulátor procesoru s operací násobení*. Bakalářská práce, FIT VUT v Brně, Brno, 2016.

[16] Z. Vašíček a kolektiv: FITkit [online].
URL http://merlin.fit.vutbr.cz/FITkit/

Přílohy

Seznam příloh

Příloha A

Obsah CD

Přiložené CD obsahuje:

- Zdrojové soubory této práce ve formátu LaTeX.
- Text práce ve formátu PDF.
- Zdrojové soubory nově navrženého sério-parelelního integrátoru v jazyce VHDL.
- Zdrojové soubory paralelně-paralelního integrátoru násobení, dělení v jazyce VHDL.
- Zdrojové soubory aplikace simulátoru paralelně-paralelního integrátoru násobení a dělení v jazyce Java.