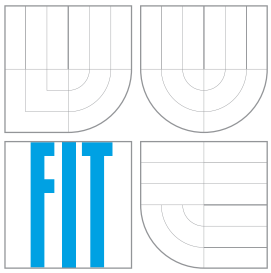
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

SIMULÁTOR PROCESORA S OPERÁCIOU DELENIA

DIVISION OPERATION SIMULATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

FRANTIŠEK MATEČNÝ

VEDOUCÍ PRÁCE
SUPERVISOR

Doc. Ing. JIŘÍ KUNOVSKÝ, CSc.

BRNO 2016

Abstrakt

Práca sa zaoberá numerickou integráciou a operáciou delenia. Najskôr je čitateľ oboznámený s numerickým riešením diferenciálnych rovníc s operáciou delenia pomocou Taylorovej rady. Ďalej je vysvetlený princíp delenia v hardvéri algoritmom SRT a je predstavený návrh sériovo-paralelného a paralelného deliaceho integrátora v pevnej rádovej čiarke. Praktickým cieľom práce je implementácia paralelného deliaceho integrátora a vytvorenie programového simulátora tohoto integrátora.

Abstract

This work deals with numerical integration and division operation. The reader is acquainted with the numerical solution of differential equations using division by the Taylor series. Next is explained the principle of SRT division in hardware and introduction of draft of design series-parallel and parallel division integrator in fixed point arithmetic. The practical aim of this work is implementation parallel division integrator and development of a software simulation of this integrator.

Kľúčové slová

diferenciálna rovnica, Taylorova rada, numerická integrácia, SRT delenie, integrátor, pevná rádová čiarka, kontrolér

Keywords

diferential equnation, Taylor series, numeric integration, SRT division, integrator, fixed point, controller

Citácia

MATEČNÝ, František. *Simulátor procesora s operáciou delenia.* Brno, 2016. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Kunovský Jiří.

Simulátor procesora s operáciou delenia

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Doc. Ing. Jiřího Kunovského Csc. Uviedol som všetky literárne pramene a publikáce, z ktorých som čerpal.

.....
František Matečný
17. mája 2016

Poďakovanie

Chcel by som sa poďakovať svojmu školiteľovi Doc. Ing. Jiřímu Kunovskému Csc., za odborné vedenie práce, podporu, vecné pripomienky, komentáre a rady k mojej práci. Moje poďakovanie taktiež patrí mojej rodine, priateľke Katke, kamarátom a všetkým ostatným, ktorí ma podporovali počas celej doby môjho štúdia a bez ktorých by táto práca nemohla vzniknúť.

Obsah

Zoznam obrázkov

Zoznam tabuliek

Kapitola 1

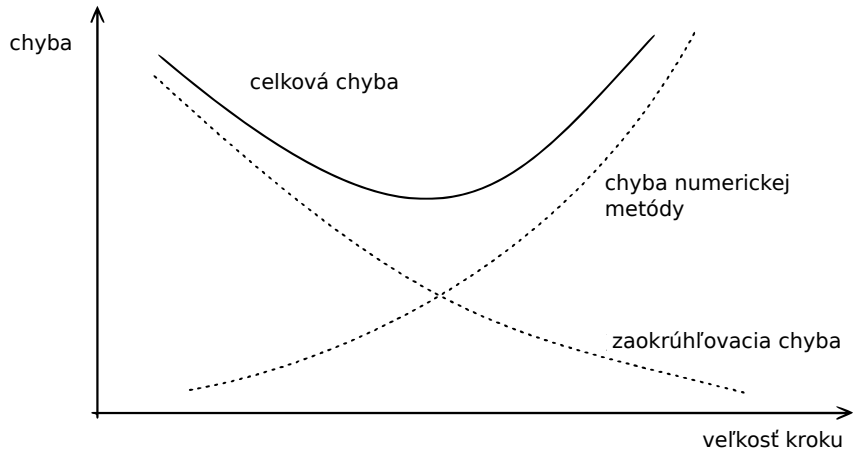
Úvod

Numerické integrátory sú komponenty, ktoré pomocou numerickej integrácie riešia diferenciálne rovnice. Táto práca sa zaoberá konkrétne deliacimi integrátormi, kde hlavným cieľom tejto práce je navrhnúť deliaci integrátor na výpočet diferenciálnych rovníc s operáciou delenie a implementovať tento integrátor pre použitie vo FPGA.
V kapitole 2 sa zoznámime s numerickou integráciou a s vlastnosťami numerických metód. Predstavíme si najčastejšie používané numerické metódy ako sú Taylorova rada, Eulerova metóda a metóda Runge-Kutta. Pomocou Taylorovej rady budeme riešiť diferenciálnu rovnicu s operáciou delenia.
Integrátory vykonávajú numerickú integráciu pomocou základných matematických operácií ako sú sčítanie, odčítanie, násobenie a delenie. Operácia delenia je z vymenovaných najviac časovo náročná a výpočetne zložitá, preto sa používajú rôzne metódy na zjednodušenie jej výpočtu. V kapitole 3 si popíšeme rôzne spôsoby jej výpočtu, kde hlavný dôraz je kladený na vysvetlenie spôsobu delenia pomocou algoritmu SRT s radixom 2.
V ďalšej kapitole 4 popíšeme jednotlivé typy integrátorov a podľa získaných rovníc 2.13 – 2.16 v kapitole 2 navrhneme sériovo-paralelný a paralelno-paralelný deliaci integrátor.
Podľa návrhu implementujeme paralelne-paralelný deliaci integrátor využívajúci SRT algoritmus na výpočet operácie delenia pre použitie do hradlového poľa FPGA. Kapitola 5 je venovaná popisu tejto implementácie.
V kapitole 6 predstavíme jednoduchý simulátor v grafickom prevedení, ktorý názorne a jednoducho znázorňuje spôsob výpočtu paralelne-paralelného integrátora v hardvéri.

Kapitola 2

Numerická integrácia

Na výpočet diferenciálnych rovníc sa čím ďalej tým viac využívajú skôr numerické metódy, než analytické riešenia. Práve prístup analytických riešení je mnohokrát veľmi zložitý, na rozdiel od numerických metód, pretože práve tie možno lepšie algoritmizovať.
Dôležitými kritériami numerických metód sú *presnosť* a *rýchlosť*. Jedno je vždy potlačené na úkor toho druhého. Zvýšenie presnosti (na úkor rýchlosti) numerickej metódy môžeme dosiahnuť zvolením dostatočne malého kroku či zvýšením rádu numerickej metódy. Avšak, tieto hodnoty treba voliť optimálne, aby sa nezvyšovala chyba výpočtu, na ktorú vplýva niekoľko faktorov. Jedným z nich je lokálna chyba, ktorá obsahuje chybu zaokrúhľovaciu (spôsobenú nepresnosťou aritmetiky v hardvéri) a chybu numerickej metódy (čím má metóda vyšší rád, tým je presnejšia). Ďalším faktorom je chyba akumulovaná, ktorá vzniká súčtom lokálnych chýb počas výpočtu (viac na [8], [5], [9]). Obrázok 2.1 znázorňuje závislosť chyby numerickej metódy od dĺžky kroku.



Obr. 2.1: Celková chyba numerických metód v závislosti od dĺžky kroku [9]

V závislosti od počtu krokov sa numerické metódy delia na jednokrokové a viackrokové. Jednokrokové metódy používajú na výpočet ďalšieho člena hodnotu člena predchádzajúceho, vo viackrokových metódach je ďalší člen počítaný z k predchádzajúcich členov. Medzi viackrokové metódy patrí napr. Adams-Bashforthova numerická metóda.
Nasledujúce podkapitoly sa venujú popisu jednokrokovej metódy – Taylorovej rady, a metódam od nej odvodených.

2.1 Taylorova rada

Je jednou zo základných jednokrokových numerických metód. Jej zápis vyzerá nasledovne:

$$y_{i+1} = y_i + hy_i' + \frac{h^2}{2!}y_i'' + \frac{h^3}{3!}y_i''' + \frac{h^4}{4!}y_i^{(4)} + \cdots + \frac{h^n}{n!}y_i^{(n)} \tag{2.1}$$

kde y je funkcia v čase i, pričom h je integračný krok.
Hoci sa jedná o nekonečný rozvoj, pri výpočte sa zanedbávajú veľmi vysoké derivácie, keďže rada rýchlo konverguje k správnemu riešeniu. Je však potrebné zvoliť vhodný integračný krok a rád metódy.
Od Taylorovej rady sú odvodené niektoré ďalšie numerické metódy, ktoré si teraz predstavíme.

2.2 Eulerova metóda

Predstaviteľom najjednoduchšej jednokrokovej metódy je Eulerova metóda. Metóda je odvodená od Taylorovej rady – Taylorova rada 1. rádu.

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

Nasledujúca hodnota sa počíta z aktuálnej. Čím menší integračný krok h zvolíme, tým presnejšie dosiahneme výsledky. Jej výhodou je, že pri výpočte počítame len jednu deriváciu. Výpočet je teda veľmi rýchly.

2.3 Runge-Kutta

Vychádza tiež z Taylorovej rady, ale zahŕňa aj členy vyšších rádov. Je preto presnejšia a patrí tiež do skupiny jednokrokových numerických metód. Je známych viacero modifikácií podľa počtu členov, a to Runge-Kutta 2., 4. a 8. rádu. Najznámejšia a najpoužívanejšia z nich je Runge-Kutta 4. rádu, ktorú môžeme zapísať nasledovne:

$$\begin{aligned} y_{y+1} &= y_i + \frac{1}{6}h(_1+2k_2 + 2k_3 + k_4) \\ k_1 &= f(t_i, y_i) \\ k_2 &= f(t_{i+1} + \frac{1}{2}h, y_i + \frac{1}{2}hk_1) \\ k_3 &= f(t_{i+1} + \frac{1}{2}h, y_i + \frac{1}{2}hk_2) \\ k_4 &= f(t_{i+1} + h, y_i + hk_3) \end{aligned} \tag{2.3}$$

Pre výpočet y_{y+1} sú potrebné medzivýsledky k_1, k_2, k_3 a k_4, čo výpočet spomaľuje.

2.4 Riešenie diferenciálnych rovníc Taylorovou radou

2.4.1 Riešenie jednoduchej diferenciálnej rovnice

Jednoduchá diferenciálna rovnica:

$$y' = y \quad y(0) = y_0 \tag{2.4}$$

Z tohto vzťahu vyplýva, že:

$$y = y' = y'' = y''' = y^{(4)} = \cdots = y^{(n)} \tag{2.5}$$

Po dosadení do Taylorovej rady 2.1 získame:

$$y_{i+1} = y_i + hy_i + \frac{h^2}{2!}y_i + \frac{h^3}{3!}y_i + \frac{h^4}{4!}y_i + \cdots + \frac{h^n}{n!}y_i \tag{2.6}$$

To je možné prepísať na:

$$y_{i+1} = y_i + DY1_i + DY2_i + DY3_i + DY4_i + \cdots + DY(N)_i \tag{2.7}$$

kde je význam jednotlivých členov nasledujúci:

$$\begin{aligned} DY1_i &= hy_i \\ DY2_i = \frac{h^2}{2!}y_i &= \frac{h}{2}DY1_i \\ DY3_i = \frac{h^3}{3!}y_i &= \frac{h}{3}DY2_i \\ DY4_i = \frac{h^4}{4!}y_i &= \frac{h}{4}DY3_i \\ &\vdots \\ DY(N)_i = \frac{h^n}{n!}y_i &= \frac{h}{n}DY(N)_i \end{aligned} \tag{2.8}$$

Z týchto vzťahov je možné riešiť jednoduché diferenciálne rovnice, ako je tomu [8], [6].

2.4.2 Riešenie diferenciálnej rovnice s operáciou delenia

Diferenciálna rovnica s operáciou delenia:

$$y' = \frac{u}{v} \tag{2.9}$$

Ďalšie derivácie rovnice 2.9 sú:

$$\begin{aligned} y'' = \frac{u'v - uv'}{v^2} &= \frac{1}{v}(u' - y'v') \\ y''' = \left(\frac{1}{v}(u' - y'v')\right)' &= \frac{1}{v}(u'' - 2y''v' - y'v'') \\ y^{(4)} = \left(\frac{1}{v}(u'' - 2y''v' - y'v')\right)' &= \frac{1}{v}(u''' - 3y'''v' - 3y''v'' - y'v''') \\ &\vdots \end{aligned} \tag{2.10}$$

Podobne ako členy 2.8, aj členy $DV(n)_i$ a členy $DU(n)_i$:

$$\begin{aligned} DV1_i &= hv_i & DU1_i &= hu_i \\ DV2_i &= \frac{h}{2}DV1_i & DU2_i &= \frac{h}{2}DU1_i \\ DV3_i &= \frac{h}{3}DV2_i & DU3_i &= \frac{h}{3}DU2_i \\ DV4_i &= \frac{h}{4}DV3_i & DU4_i &= \frac{h}{4}DU3_i \\ &\vdots & &\vdots \\ DV(N)_i &= \frac{h}{n}DV(N)_i & DU(N)_i &= \frac{h}{n}DU(N)_i \end{aligned} \tag{2.11}$$

Po dosadení jednotlivých členov $DV(n)_i$ a $DU(n)_i$ do derivácií 2.10 dostaneme:

$$\begin{aligned} \frac{DY1_i}{h} &= \frac{1}{v}u \\ \frac{DY2_i}{\frac{h^2}{2!}} &= \frac{1}{v}(\frac{DU1_i}{h} - \frac{DY1_i}{h}\frac{DV1_i}{h}) \\ \frac{DY3_i}{\frac{h^3}{3!}} &= \frac{1}{v}(\frac{DU2_i}{\frac{h^2}{2!}} - 2\frac{DY2_i}{\frac{h^2}{2!}}\frac{DV1_i}{h} - \frac{DY1_i}{h}\frac{DV2_i}{\frac{h^2}{2!}}) \\ \frac{DY4_i}{\frac{h^4}{4!}} &= \frac{1}{v}(\frac{DU3_i}{\frac{h^3}{3!}} - 3\frac{DY3_i}{\frac{h^3}{3!}}\frac{DV1_i}{h} - 3\frac{DY2_i}{\frac{h^2}{2!}}\frac{DV2_i}{\frac{h^2}{2!}} - \frac{DY1_i}{h}\frac{DV3_i}{\frac{h^3}{3!}}) \\ &\vdots \end{aligned} \tag{2.12}$$

Po úprave vyzerajú jednotlivé členy Taylorovej rady 2.1 pre riešenie diferenciálnej rovnice 2.10 nasledovne:

$$DY1_i = \frac{1}{v}(hu) \tag{2.13}$$

$$DY2_i = \frac{1}{2v}(DU1_i h - DY1_i DV1_i) \tag{2.14}$$

$$DY3_i = \frac{1}{3v}(DU2_i h - 2DY2_i DV1_i - DY1_i DV2_i) \tag{2.15}$$

$$DY4_i = \frac{1}{4v}(DU3_i h - 3DY3_i DV1_i - 2DY2_i DV2_i - DY1_i DV3_i) \tag{2.16}$$

$$\vdots$$

Formuly jednotlivých členov tvoria Pascalov trojuholník a sú základom pre tvorbu návrhu deliaceho integrátora, ktorému sa venuje kapitola 4.

2.4.3 Riešenie diferenciálnej rovnice substitúciou delenia

Diferenciálnu rovnicu 2.9 upravíme tak, aby sme sa vyhli operácii delenia, ktorá je pre výpočet veľmi náročná a pomalá. Delenie teda nahradíme násobením (viac o operácii delenia v kapitole 3).

Rovnicu 2.9 upravíme na násobenie nasledovne:

$$\begin{aligned} y' &= \frac{u}{v} = \frac{1}{v}u \\ y' &= qu \quad kde\ q = \frac{1}{v} \end{aligned} \tag{2.17}$$

Vyššie derivácie násobenia sú:

$$\begin{aligned} y'' &= q'u - qu' \\ y''' &= q''u + 2q'u' + qu'' \\ y^{(4)} &= q'''u + 3q''u' + 3q'u'' + qu''' \\ &\vdots \end{aligned}$$

Členy $DU(n)_i$ sú rovnaké ako členy 2.11. Členy $DQ(n)_i$ spočítame postupnou deriváciou substitúcie 2.17 takto:

$$\begin{aligned} q' &= -q^2v' \\ q'' &= -2q'qv' - q^2v'' \\ q''' &= -2q''qv' - 2q'q'v' - 4q'qv'' - q^2v''' \\ &\vdots \end{aligned}$$

Výsledné derivácie upravíme do tvaru jednotlivých členov:

$$DQ1_i = -q^2DV1 \tag{2.18}$$

$$DQ2_i = -qDQ1_iDV1_i - q^2DV2_i \tag{2.19}$$

$$DQ3_i = -\frac{1}{3}(DQ2_iDV1_iq + DQ1_i^2DV1_i + 4DQ1_iDV2_iq) - q^2DV3_i \tag{2.20}$$

$$\vdots$$

Jednotlivé členy $DY(n)_i$ potom zapíšeme nasledovne:

$$DY1_i = h(qu) \tag{2.21}$$

$$\begin{aligned} DY2_i &= \frac{h}{2}(DQ1_iu + qDU1_i) \\ DY3_i &= \frac{h}{3}(DQ2_iu + DQ1_iDU1_i + qDU2_i) \\ DY4_i &= \frac{h}{4}(DQ3_iu + DQ2_iDU1_i + DQ1_iDU2_i + qDU3_i) \\ &\vdots \end{aligned}$$

Rovnako ako pri delení i tu tvoria jednotlivé členy Pascalov trojuholník. Tieto členy tvoria základ pre tvorbu násobiaceho integrátora. Avšak ani po substituovaní delenia 2.17 sa operáciám delenia nevyhneme. Jednotlivé členy násobiaceho integrátora 2.22 obsahujú postupné delenie integračného kroku. Rovnako aj člen $DQ3_i$ 2.21 obsahuje delenie. Týmto operáciám sa dá vyhnúť nahradením zlomkov konštantami ešte pred spustením výpočtu. Ďalším nedostatkom tohto prístupu je, že výpočet členov $DQ(n)_i$ 2.19 je oveľa náročnejší než pri pôvodnom prístupe, kde jednotlivé členy $DV(n)_i$ 2.11 obsahujú len jednu operáciu násobenia. Preto je efektívnejšie pri riešení diferenciálnej rovnice s operáciou delenia použiť jeden deliaci integrátor, než použiť dva integrátory – násobiaci integrátor a integrátor, ktorý počíta prevrátenú hodnotu.

2.4.4 Sústava diferenciálnych rovníc

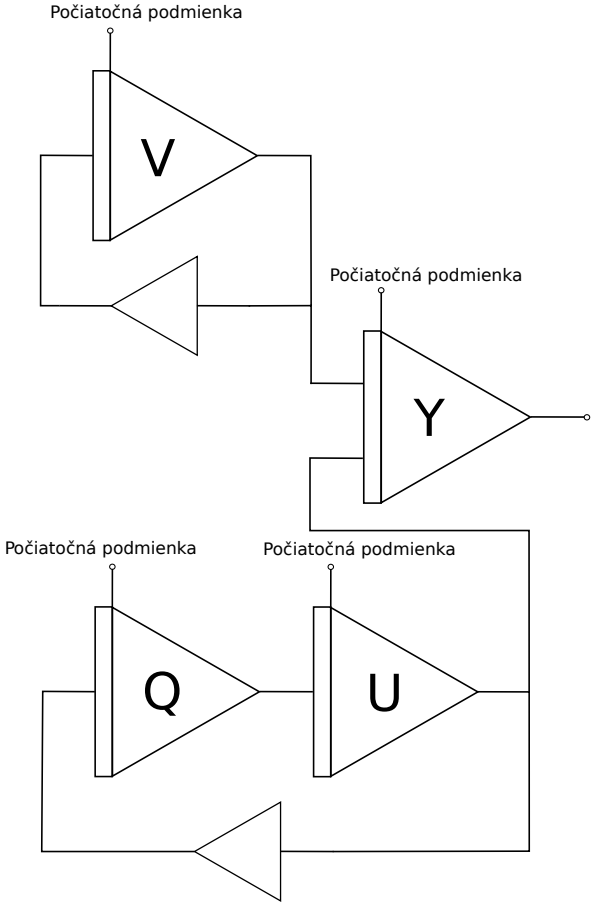
Diferenciálna rovnica s operáciou delenia:

$$y' = \frac{sint}{e^{-t}} \quad y(0) = 1 \tag{2.22}$$

Rovnicu 2.22 popisuje nasledujúca sústava rovníc:

$$\begin{aligned} y' &= \frac{u}{v} \quad y(0) = 1 \\ v' &= -v \quad v(0) = 1 \\ u' &= q \quad u(0) = 0 \\ q' &= -u \quad q(0) = 1 \end{aligned} \tag{2.23}$$

Túto sústavu rovníc je možné prekresliť do schémy 2.2, ktorú tvoria tri jednoduché integrátory, dva invertory a jeden dvojvstupový deliaci integrátor. Integrátor je prvok integrujúci vstupný signál. Invertor mení polaritu vstupného signálu. Deliaci integrátor je prvok, ktorého výsledok je integračný podiel vstupov.



Obr. 2.2: Sústava integrátorov

Integrátory Q, U a invertor tvoria goniometrickú funkciu $sint$. Zapojenie integrátora V s invertorom predstavuje funkciu e^{-t}. Výstup integrátora Y je hodnota rovnice 2.22 v čase $i + 1$.

Kapitola 3

Delenie

Delenie je inverznou operáciou k operácii násobenie. Delenie nulou nie je definované, preto je potrebné tento stav ošetriť výnimkou či chybovou hláškou. Výsledok delenia sa nazýva podiel.
Na rozdiel od násobičiek či sčítačiek, delička nie je preddefinovaným blokom vo FPGA, a preto si ju musíme nadefinovať sami. Algoritmy delenia rozdeľujeme do dvoch skupín na *sekvenčné* a *iteračné* algoritmy. Sekvenčné deliace algoritmy produkujú jednu číslicu podielu počas jednej iterácie výpočtu. Medzi takéto algoritmy patria napríklad algoritmy s návratom k nezápornému zvyšku, algoritmy bez návratu k nezápornému zvyšku, SRT algoritmy a iné. Iteračné deliace algoritmy sú napr. Goldschmidt, Newton–Raphson a ďalšie. Tie v každej iteráci produkujú všetky bity výsledku. Čím viacej iterácií sa prevedie, tým je výsledok presnejší. Ukončovacou podmienkou je stanovený *epsilon* ϵ, ktorý reprezentuje požadovanú presnosť [10].

Hlavná rovnica operácie delenia vyzerá nasledovne:

$$X = Q * D + R \quad kde\ R < D \tag{3.1}$$

kde význam jednotlivých písmen je:
X – delenec
D – deliteľ
R – zvyšok po delení
q – podiel

3.1 Delenie s návratom k nezápornému zvyšku

Každý bit výsledku je počítaný odčítaním deliteľa od priebežného zvyšku. Ak je vypočítaný nový zvyšok kladný (čiže najvyšší bit sa rovná log. 0), tak potom $Q_{n-1} = 1$ a $R_{i+1} = 2R_i$. Ak je však nový zvyšok záporný (čiže najvyšší bit sa rovná log. 1), tak potom $Q_{n-1} = 0$ a $R_{i+1} = 2R_i - d$, čo nie je správne, a preto je nutné vykonať korekciu pričítaním deliteľa – návrat k nezápornému zvyšku. Potom priebežný zvyšok posunieme o 1 bit doľava a znova pokračujeme odčítaním deliteľa.

3.2 Delenie bez návratu k nezápornému zvyšku

Postup algoritmu je podobný ako v predchádzajúcom prípade. Hodnota jednotlivých bitov výsledku je takisto závislá od hodnoty najvyššieho bitu priebežného zvyšku. Rozdiel je v počítaní hodnoty nasledujúceho priebežného zvyšku, a to nasledovne:

$$\text{ak } Q_{n-1} = 1, \text{ tak potom } R_{i+1} = 2R_i - d$$
$$\text{ak } Q_{n-1} = 0, \text{ tak potom } R_{i+1} = 2R_i + d$$

I tu sa vykonáva korekcia, avšak až na konci výpočtu. Ak je priebežný zvyšok menší ako nula, sčítame ho s deliteľom.

Obe tieto metódy opakujú cyklus výpočtu n-krát, kde n určuje počet bitov danej architektúry. Ak predpokladáme, že pravdepodobnosť výskytu log. 1 na najvyššom bite priebežného zvyšku je 50%, tak potom algoritmus delenia s návratom k nezápornému zvyšku vykoná priemerne až o 50% viac aritmetických operácií (sčítania alebo odčítania) než algoritmus delenia bez návratu k nezápornému zvyšku [10].

3.3 SRT delenie

Skratka algoritmu je odvodená od iniciál mien jeho autorov, ktorými sú Sweeney, Robertson a Tocher (1958). Patrí do skupiny deliacich algoritmov bez návratu k nezápornému zvyšku a využíva vyhľadávaciu tabuľku na odhadovanie hodnôt jednotlivých číslic podielu. Tento odhad je realizovaný na základe najvyšších bitov priebežného zvyšku. V niektorých úpravách SRT algoritmu sa na odhad okrem bitov priebežného zvyšku využíva aj niekoľko bitov deliteľa, čím sa zvýši presnosť výpočtu. Algoritmy vyvinuté pred algoritmom SRT vedeli počítať len s absolútnou hodnotou. Znamienko výsledku bolo určené až na konci výpočtu. Algoritmus SRT toto obmedzenie odstránil, keďže je s ním umožnené počítať i so zápornými číslami [10].
Tento algoritmus bol mediálne prezentovaný predovšetkým okolo roku 1995. Dôvodom bola chyba vo vyhľadávacej tabuľke v niektorých procesoroch Intel Pentium, ktorá spôsobila, že pri niektorých matematických operáciách bol výsledok nepresný. Viac informácii nájdeme na [4].

3.3.1 Algoritmus SRT

Základný SRT algoritmus vyhľadáva vo vyhľadávacej tabuľke podľa troch najvyšších bitov priebežného zvyšku. Výsledný podiel je tvorený číslicami relatívnej sústavy, ktorá na rozdiel od binárnej sústavy obsahuje tri číslice: $0, 1, \overline{1}$. To umožňuje efektívnejšie vyjadrenie číslic. Napríklad číslo 7 je v binárnej sústave 0111, po prekódovaní do relatívnej sústavy to môžeme zapísať ako $100\overline{1}$, čo je v dekadickej sústave $8 - 1$. Toto prekódovanie používa aj Boothov algoritmus násobenia, preto sa nazýva Boothovo prekódovanie. Realizuje sa pomocou prevodnej tabuľky 3.1. Tabuľka bola prevzatá zo slajdov k predmetu INP [10]. Ak sa pri prevode používajú dva bity, jedná sa o prekódovanie s *radixom 2*. V praxi sa používajú aj prekódovania s vyšším radixom. Pri použití troch bitov pri prevode hovoríme o prekódovaní s radixom 4, pri využití štyroch bitov ide o radix 8 [11].

prekódovaná číslica	susedný bit vpravo	Boothov kód
0	0	0
0	1	1
1	0	$\overline{1}$
1	1	0

Tabuľka 3.1: Boothovo prekódovanie s radixom 2 [10].

Hlavným cieľom pri tvorbe SRT algoritmu bolo zvýšiť rýchlosť algoritmu delenia bez návratu k nezápornému zvyšku. K tomu bolo potrebných niekoľko úprav tohto algoritmu, ktorého princíp s využitím predkódovania s radixom 2 zapíšeme takto:

$$q_i = \begin{cases} 1 & \text{ak} \quad 2r_{i-1} \geq 0 \\ -1 & \text{ak} \quad 2r_{i-1} < 0 \end{cases}$$

Aby sa odstránilo neustále pričítavanie a odčítavanie deliteľa, do číslic podielu bola pridaná hodnota 0 podľa týchto podmienok:

$$q_i = \begin{cases} 1 & \text{ak} \quad 2r_{i-1} \geq D \\ 0 & \text{ak} \quad -D \leq 2r_{i-1} < D \\ -1 & \text{ak} \quad 2r_{i-1} < D \end{cases}$$

Keďže by bolo porovnávanie všetkých bitov deliteľa neefektívne, pomocou normalizácie deliteľa D podľa vzťahu 3.3 docielime, že na odhad jednotlivých číslic podielu je potrebné porovnanie iba troch najvyšších bitov priebežného zvyšku. Podmienky pre ich odhadnutie vyzerajú nasledovne:

$$q_i = \begin{cases} 1 & ak \quad 2r_{i-1} \geq \frac{1}{2} \\ 0 & ak \quad -\frac{1}{2} \leq 2r_{i-1} < \frac{1}{2} \\ -1 & ak \quad 2r_{i-1} < \frac{1}{2} \end{cases} \tag{3.2}$$

$$\frac{1}{2} \leq |D| < 1 \tag{3.3}$$

Tabuľka 3.2 zobrazuje vzťahy 3.2 pri využití dvojkového doplnku a fix-poind aritmetiky s jedným znamienkovým bitom a s jedným bitom pred desatinnou čiarkou.

znamienko	2^0	2^{-1}	podmienka	podiel
0	0	0	$< \frac{1}{2}$	0
0	0	1	$\geq \frac{1}{2}$	1
0	1	0	$\geq \frac{1}{2}$	1
0	1	1	$\geq \frac{1}{2}$	1
1	0	0	$< -\frac{1}{2}$	$\overline{1}$
1	0	1	$< -\frac{1}{2}$	$\overline{1}$
1	1	0	$< -\frac{1}{2}$	$\overline{1}$
1	1	1	$\geq -\frac{1}{2}$	0

Tabuľka 3.2: Tabuľka odhadov jednotlivých číslic podielu [7]

3.3.2 On-The-Fly prevod

Použitie a reprezentácia relatívnych číslic $(0, 1, \overline{1})$ v hardvéri pracujúcom v binárnej (dvojkovej) sústave nie je možná. Preto je pri použití SRT algoritmu s radixom 2 potrebné zariadiť, aby výpočet podielu prebiehal v binárnej sústave. Prekódovanie z relatívnej sústavy do binárnej sústavy sa nazýva *on-the-fly prevod*.
On-the-fly prevod prebieha pomocou dvoch registrov Q a QM. Na začiatku výpočtu je obsah registrov vynulovaný. V každom kroku výpočtu sa vykoná bitový posun doľava (od najmenej významného bitu k najviac významnému bitu) a najmenej významný bit je nastavený na hodnotu podľa tabuľky 3.3. Týmto postupom sa docieli, že sa bity registrov Q a QM menia od najvýznamnejšieho bitu, a teda výsledok sa v každom cykle výpočtu spresňuje. Posledná hodnota v registri Q je výslednou hodnotou podielu q [7].

Význam jednotlivých označení v tabuľke 3.3 je nasledovný:

- q_{k+1} označuje hodnotu podielu v relatívnej sústave radix 2, kde k je krok výpočtu.
- Q_{in} a QM_{in} určujú, na akú hodnotu je nastavený najmenej významný bit daného registra.
- C_Q a C_{QM} určujú, ktorý register je kopírovaný z predchádzajúceho stavu.

Celý postup prevodu čísla $101\bar{1}01$ je znázornený v tabuľke 3.4.

q_{k+1}	Q_{in}	C_Q	QM_{in}	C_{QM}
1	1	1	0	0
0	0	1	1	1
$\bar{1}$	1	0	0	1

Tabuľka 3.3: Tabuľka On-The-Fly konverzie [7]

k		Q	QM
0		0.000000	0.000000
1	1	0.000001	0.000000
2	0	0.000010	0.000001
3	1	0.000101	0.000100
4	$\bar{1}$	0.001001	0.001000
5	0	0.010010	0.010011
6	1	0.100101	0.100100

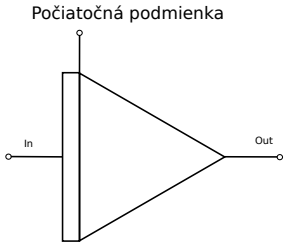
Tabuľka 3.4: Tabuľka zobrazujúca konkrétny príklad On-The-Fly prevodu

Výsledok prevodu čísla $101\bar{1}01$ do binárnej sústavy je 0.100101. V desiatkovej sústave majú tieto čísla hodnotu 0.578125.

Kapitola 4

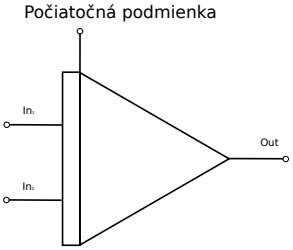
Numerické integrátory

Numerický integrátor je prvok, ktorý integruje vstupný signál a dáva ho na výstup. Na naštartovanie výpočtu je nutné zvoliť počiatočnú podmienku a integračný krok. Schéma takéhoto integrátora je na obrázku 4.1.



Obr. 4.1: Jednoduchý integrátor

V tejto kapitole sa budeme zaoberať návrhom a implementáciou deliaceho integrátora znázorneného na obrázku 4.2. Ten sa líši od jednoduchého integrátora 4.1 počtom vstupov.



Obr. 4.2: Deliaci integrátor

Podľa spôsobu komunikácie a výpočtu sú v dizertačnej práci [6] odvodené názvy jednotlivých integrátorov nasledovné:

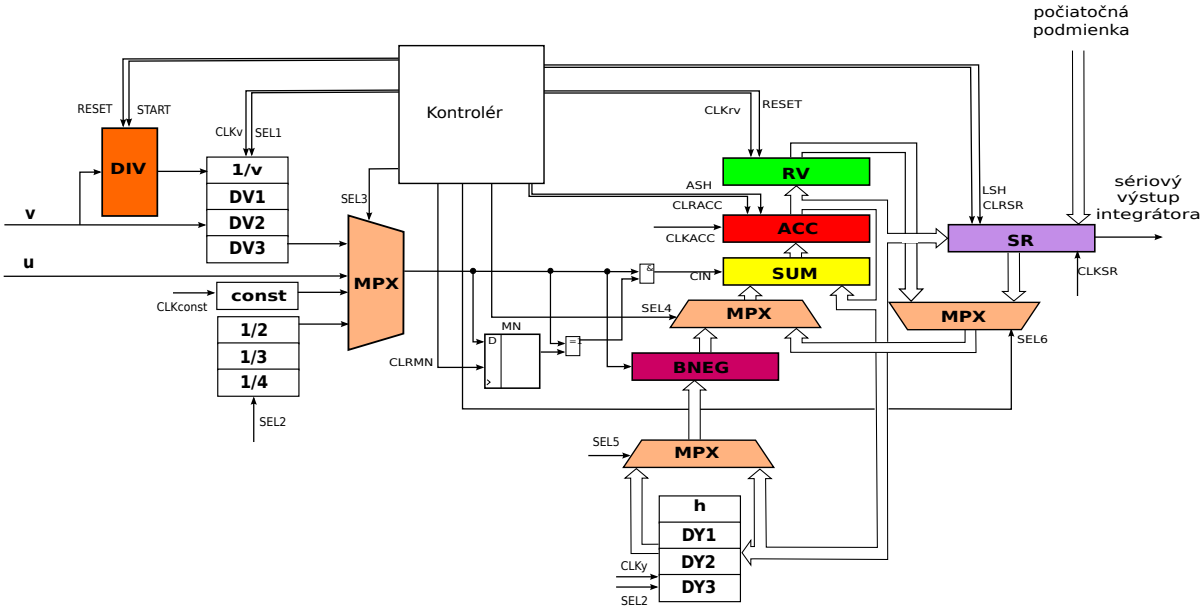
- PPI – paralelno-paralelný integrátor (paralelná komunikácia i výpočet)
- SPI – sériovo-paralelný integrátor (sériová komunikácia a paralelný výpočet)
- SSI – sériovo-sériový integrátor (sériová komunikácia aj výpočet)

Rovnaké rozdelenie použijeme aj pre deliace integrátory:

- PPDI – paralelno-paralelný deliaci integrátor (paralelná komunikácia i výpočet)
- SPDI – sériovo-paralelný deliaci integrátor (sériová komunikácia a paralelný výpočet)
- SSDI – sériovo-sériový deliaci integrátor (sériová komunikácia aj výpočet)

4.1 Sériovo-paralelný deliaci integrátor

Na obrázku 4.3 je znázornený návrh sériovo-paralelného deliaceho integrátora. Základom návrhu bol použitý sériovo-paralelný integrátor predstavený v Bakalárskej práci [8]. Z tohto návrhu boli použité komponenty slúžiace na výpočet Boothovho algoritmu násobenia, a to paralelná sčítačka *SUM*, akumulátor *ACC*, register výsledku *RV*, posuvný register *SR*, obvod riadenej negácie *BNEG*, obvod malej nuly *MN* a dva paralelné multiplexory *MPX*. Do návrhu bola pridaná SRT delička *DIV*, registre pre uchovanie hodnôt jednotlivých členov výpočtu, integračného kroku h a pomocných konštánt, sériový a paralelný multiplexor a upravený kontrolér.



Obr. 4.3: Sériovo-paralelný deliaci integrátor

DIV	SRT delička
RV	register výsledku
SUM	sčítačka
ACC	akumulátor
BNEG	obvod riadenej negácie
MN	obvod malej nuly
MPX	multiplexor
const, 1/x	registre konštánt
Dxx	registre jednotlivých členov
h	register integračného kroku

Výpočet je naštartovaný signálom *RESET*, ktorý vynuluje potrebné registre a nastaví signály jednotlivým multiplexorom. Potom je do registra SR uložená počiatočná podmienka y_0 a súčasne do registra h je uložený integračný krok (kvôli prehľadnosti túto zbernicu schéma neobsahuje). Následne je spustený výpočet jednotlivých členov podľa rovníc 2.13 – 2.16. Po ukončení výpočtu je výsledok uložený do posuvného registra SR.

Keďže sa pri výpočte veľakrát používa operácia násobenia, je možné výpočet zrýchliť použitím viacerých násobičiek. Potom by viacej operácií násobenia mohlo prebiehať paralelne (súčasne).

4.2 Sériovo-sériový deliaci integrátor

Návrh tohto integrátora by obsahoval sériovú sčítačku, SRT sériovú deličku, a aj operácia násobenia by musela byť implementovaná sériovo. Takéto riešenie deliaceho integrátora by bolo veľmi pomalé a neefektívne, preto nebol tento návrh realizovaný.

Najviac som sa zameral na tvorbu paralelne-paralelného deliaceho integrátora, ktorého realizácia je z predošlých variánt deliacich integrátorov najjednoduchšia a zároveň aj jeho výpočet prebieha najrýchlejšie. Ďalej si predstavíme túto variantu deliaceho integrátora.

4.3 Paralelno-paralelný deliaci integrátor

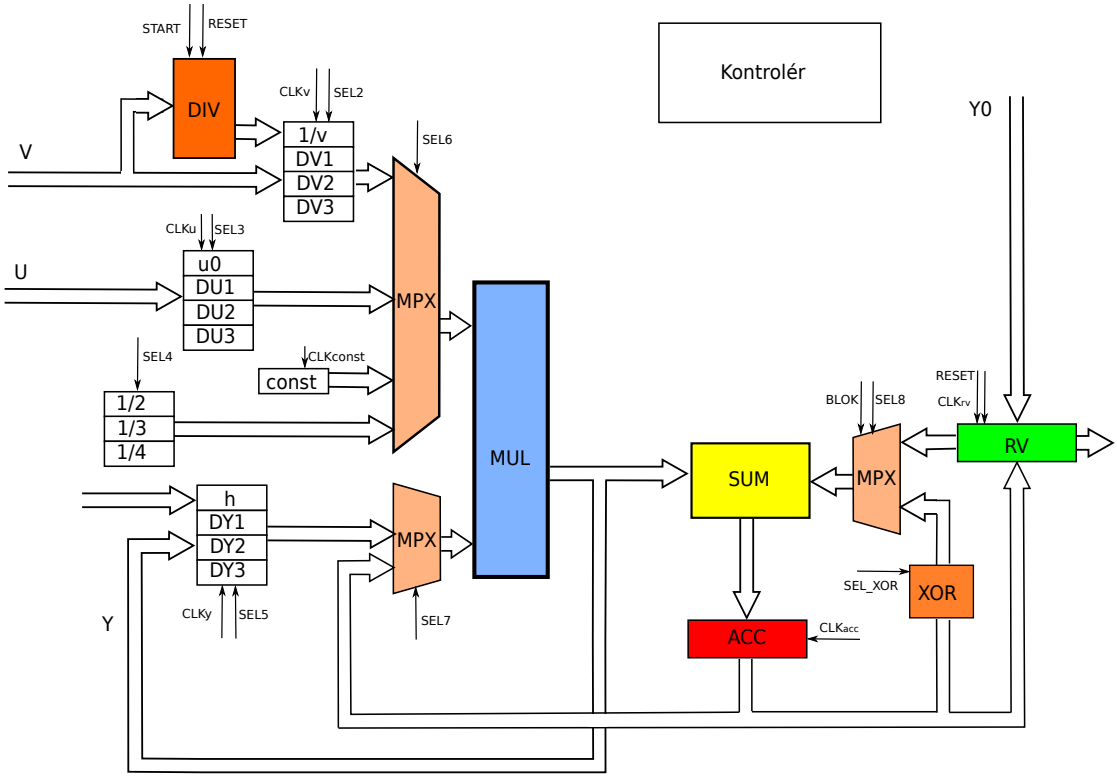
Návrh tohto integrátora obsahuje paralelnú násobičku i sčítačku. Ich výpočet prebieha paralelne, a teda nespomaľujú celkový beh integrátora. Najzložitejšou a najpomalšou operáciou v navrhnutom integrátore je operácia delenia. Delička má síce paralelný vstup aj výstup, ale výpočet v nej prebieha sekvenčne upraveným algoritmom SRT. Aby sa spomalenie neprejavovalo pri výpočte každého člena 2.8 Taylorovej rady, operácia delenia je prevedená len raz, a to na začiatku výpočtu. Pri výpočte vyšších členov Taylorovej rady je zlomok:

$$\frac{1}{jv} \quad kde \quad j \in \{2, 3, 4\} \tag{4.1}$$

prevedený na operáciu násobenia:

$$\frac{1}{v} * \frac{1}{j} \tag{4.2}$$

kde hodnota $\frac{1}{v}$ je uložená v registri rovnakého názvu, a hodnoty $\frac{1}{j}$ sú uložené v pomocných registroch 1/2, 1/3 a 1/4 na začiatku výpočtu. Tento prístup síce zaberá viac miesta v hardvéri, než keby bola použitá operácia delenia pri výpočte každého člena Taylorovej rady, ale veľmi urýchľuje a zefektívňuje výpočet bez nutnosti čakania na dokončenie operácie delenia.
Pred výpočtom je potrebné zvoliť krok h numerickej metódy. Výpočet začína nahraním počiatočnej podmienky do registra *RV*. Následne je hodnota v privedená do SRT deličky *DIV* a začína výpočet $\frac{1}{v}$. Paralelne s ním sú nastavené cesty multiplexorov *MPX* pre vynásobenie u a integračného kroku h. Výsledok z násobičky *MUL* je uložený cez sčítačku *SUM* do akumulátora *ACC*. Aby sa hodnota z násobičky v sčítačke nezmenila, a takto nezmenená sa uložila do akumulátora *ACC*, je na vstup sčítačky privedená hodnota 0 z blokového multiplexora, ktorá je nastavovaná signálom *BLOK* na log. 1. Potom je potrebné počkať na dokončenie delenia, ktorého výsledná hodnota je uložená do registra 1/v, a následne je spolu s hodnotou v *ACC* privedená do násobičky *MUL*. Hodnota *MUL* je uložená do registra *DY1* a privedená do sčítačky spolu s hodnotou počiatočnej podmienky uloženej v RV. Spolu sú uložené do *ACC* , a následne do registra výsledku *RV*. Potom nasleduje výpočet druhého člena Taylorovej metódy.
Rovnice druhého 2.14 a tretieho 2.15 člena Taylorovej rady obsahujú násobenie s rovnakými hodnotami, a to $2DY2_i$. Aby sme tieto rovnaké výpočty nepočítali viackrát, hodnota registra DY2 je po výpočte $2DY2_i$ prepísaná novou hodnotu. Pri výpočte ďalších členov sa použije už nová hodnota, čím sa ušetrí jedna operácia násobenia. Takýmto spôsobom je možné pri ôsmom člene Taylorovej rady ušetriť až päť operácií násobenia.



Obr. 4.4: Paralelno-paralelný deliaci integrátor

DIV SRT delička
MUL násobička
RV register výsledku
SUM sčítačka
ACC akumulátor
MPX multiplexor

Kapitola 5

Implementácia paralelne-paralelného deliaceho integrátora vo VHDL

V tejto kapitole si popíšeme implementáciu paralelne-paralelného deliaceho integrátora, ktorého návrh je popísaný v predošlej kapitole. Tento integrátor som implementoval v jazyku VHDL za účelom nasadenia do programovateľného hradlového poľa FPGA. VHDL je programovací jazyk slúžiaci na popis hardvéru a digitálnych obvodov. Programovateľné hradlové pole FPGA (Field Programmable Gate Array) je súhrn špeciálnych číslicových integrovaných obvodov, ktoré je možné neobmedzene modifikovať pre rôzne účely bez nutnosti vytvárania nového harvéru [1].

5.1 Implementácia PPDI

Všetky komponenty PPDI sú implementované v jazyku VHDL tak, aby pracovali aj so zápornými číslami. Preto komponenty, akými sú násobička, delička a sčítačka, boli upravené tak, aby pracovali v dvojkovom doplnkovom kóde. Počas výpočtu je využívaná aj operácia odčítania. Tá je realizovaná pomocou prvku *XOR*, ktorý pri nastavení signálu *SEL_XOR* vykonáva zmenu znamienka v dvojkovom doplnkovom kóde, a jeho výstup je privedený na vstup sčítačky *SUM*, ktorá vykoná operáciu sčítania, pričom prakticky dochádza k operácií odčítania.
Násobička *MUL* je implementovaná ako násobička pracujúca len s kladnými číslami. Avšak, pri vstupe do násobičky sú čísla v závislosti od znamienka vstupných operandov prevedené na kladné, a následne je výsledok po dokončení operácie násobenia prevedený na záporné číslo v dvojkovom doplnkovom kóde podľa hodnôt znamienkových bitov násobenca a násobiteľa.
Implementácia SRT deličky je podrobne popísaná nižšie.

5.2 Implementácia SRT algoritmu v PPDI

SRT algoritmus implementovaný v jazyku VHDL bol prevzatý zo stránok [2]. Implementovaný algoritmus pracuje správne pri splnení podmienky:

$$-D \leq X < D \tag{5.1}$$

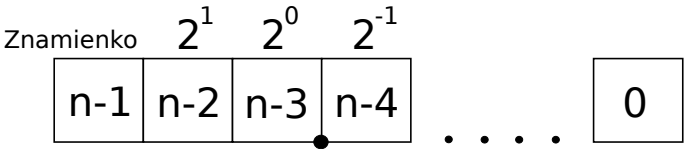
Vzhľadom na to, že delička podľa návrhu PPDI v predošlej kapitole počíta podiel zlomku v tvare $\frac{1}{v}$, vstup delenca X bol odstránený a nahradený konštantnou hodnotou 1. To znamená, že podmienku 5.1 môžeme prepísať na:

$$-D \leq 1 < D \tag{5.2}$$

Táto skutočnosť ale prináša ďalší problém, pretože rovnica 5.2 a rovnica normalizácie 3.3 nemôžu byť súčasne splnené. Táto rovnica však teraz neplatí, keďže použijeme inú fixed-point aritmetiku, než používa základný SRT algoritmus, predstavený v kapitole 3. Aritmetika použitá pri implementácii PPDI je znázornená na obrázku 5.1. Pri výpočte jednotlivých členov Taylorovej rady 2.13 - 2.16 násobíme členy $DY2_i$ a $DY3_i$ konštantami 2 a 3. Na zobrazenie týchto konštánt potrebujeme dva bity, a preto sme museli použiť, namiesto jedného bitu pred desatinnou čiarkou, práve dva bity. Rovnicu 3.3 upravíme teda na:

$$2 \leq |D| < 4 \tag{5.3}$$

Teraz môžu byť rovnice 5.2 a 5.3 súčasne splnené.
Pri zmene aritmetiky musíme zmeniť aj rovnice určujúce odhad jednotlivých číslic podielu. Zmena je minimálna, a to taká, že namiesto porovnávania s hodnotou $\frac{1}{2}$, budeme porovnávať s hodnotou 1. Vyplýva to z toho, že binárna reprezentácia 001 (troch najvyšších bitov) pri fixed-point aritmetike použitej v základnom SRT algoritme znamená v desiatkovej sústave $\frac{1}{2}$. Keďže sme si posunuli desatinnú čiarku doprava, číslo 001 teraz znamená v desiatkovej sústave hodnotu 1. Tabuľka 5.3 túto zmenu zobrazuje.



Obr. 5.1: Fixed point aritmetika

Z rovníc 5.3 a 5.2 vyplýva, že deliteľ D môže nadobúdať len hodnoty v intervale <2, 4). Aby sa tento interval zvýšil, číslo privedené na vstup deličky je normalizované tak, aby platil vzťah 5.3. Normalizácia prebieha prostredníctvom využitia pomocného registra, bitového posunu a čítača. Vstupná hodnota D je uložená do pomocného registra, na ktorý je aplikovaný bitový posun doľava až pokým sa hodnota bitu *n-2* nerovná 1. V každom kroku je inkrementovaný čítač. Po ukončení normalizácie nasleduje samotný výpočet. Na konci výpočtu je výsledná hodnota bitovým posunom posunutá o hodnotu čítača. Týmto spôsobom sme zvýšili interval hodnôt, ktoré môže nadobudnúť deliteľ D, na všetky kladné čísla zobraziteľné s použitím danej aritmetiky na <0, 4).

Aby sme zvýšili tento interval aj na záporné čísla, a teda pokryli celý rozsah hodnôt zobraziteľných v danej fixed-point aritmetike, stačí pomocou operácie *xor* a pričítaním jednotky zmeniť znamienko delenca i deliteľa. Keďže delenec je vždy kladný a deliteľ musí byť normalizovaný ako kladné číslo, pri vstupe sa skontroluje znamienkový bit deliteľa. Ak je znamienkový bit deliteľa záporný, delenec sa prevedie na zápornú hodnotu (-1) a deliteľ na kladnú. Potom nasleduje normalizácia deliteľa.

znamienko	2^1	2^0	podmienka	podiel
0	0	0	< 1	0
0	0	1	≥ 1	1
0	1	0	≥ 1	1
0	1	1	≥ 1	1
1	0	0	< -1	-1
1	0	1	< -1	-1
1	1	0	< -1	-1
1	1	1	≥ -1	0

Tabuľka 5.1: Tabuľka odhadov jednotlivých číslic podielu

Delenie nulou nie je definované, preto je pred výpočtom kontrolované, či sa hodnota deliteľa D nerovná nule. Ak áno, delenie je ukončené a výstupný signál *ERR* je nastavený na log. 1. Konečný automat v kontroléri je na základe tohto signálu prepnutý do počiatočného stavu. Nový výpočet je možné začať nastaveným signálu *RESET* na log. 1.

5.2.1 Kontrolér v PPDI

Kontrolér implementovaný v PPDI obsahuje konečný automat, podľa ktorého nastavuje jednotlivé riadiace signály. Na spustenie kontroléra stačí signál *RESET*, ktorým sa spustí konečný automat. V každej perióde hodinového signálu *CLK* sa PPDI dostáva do nasledujúceho stavu. Celý konečný automat pre výpočet rovnice 2.9 Taylorovou radou 4. rádu je znázornený v tabuľke 5.2 na konci tejto kapitoly. Automat obsahuje 40 stavov a jednu podmienku kontrolujúcu ukončenie výpočtu SRT deličky. Tá po ukončení výpočtu nastaví výstupný signál *done* na log. 1.

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	Reset
1	0	0	0	1	00	00	X	00	01	0	X	1	Y0 -> RV, delenie 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	nastavenie 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	nastavenie 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	nastavenie 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	nastavenie 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	nastavenie 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	nastavenie 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	nastavenie 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	nastavenie 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	nastavenie 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	nastavenie 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	nastavenie 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	nastavenie 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	nastavenie 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	nastavenie 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

Tabuľka 5.2: Riadenie PPDI

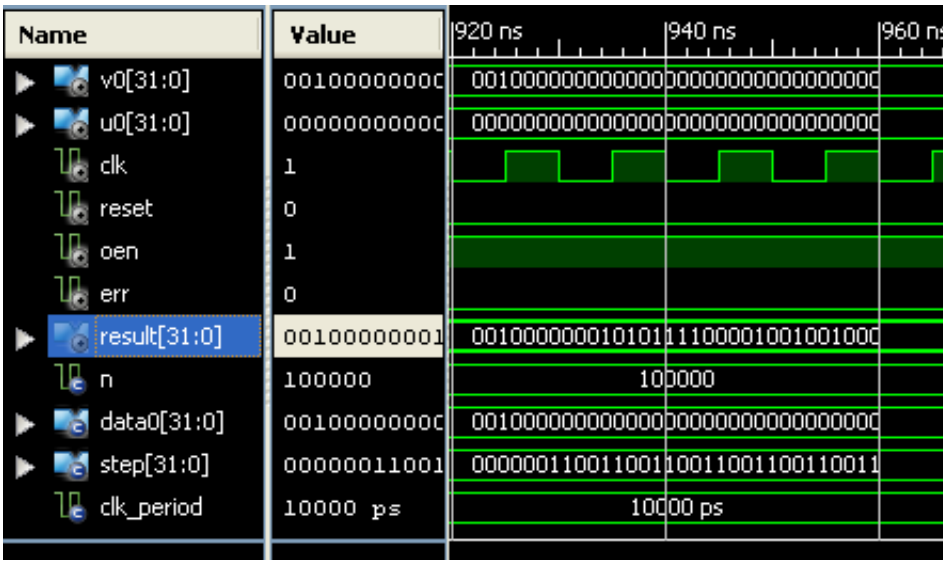
5.2.2 Simulácia

Implementovaný PPDI bol simulovaný a testovaný v programe XilinxISE version 13.1. Simulácia ukázala, že aj pri počítaní Taylorovou radou 4. rádu sú výsledky relatívne presné. Simulácia rovnice 2.22 je zobrazená na obrázku 5.2 na 32 bitoch a na obrázku 5.3 na 64 bitoch.

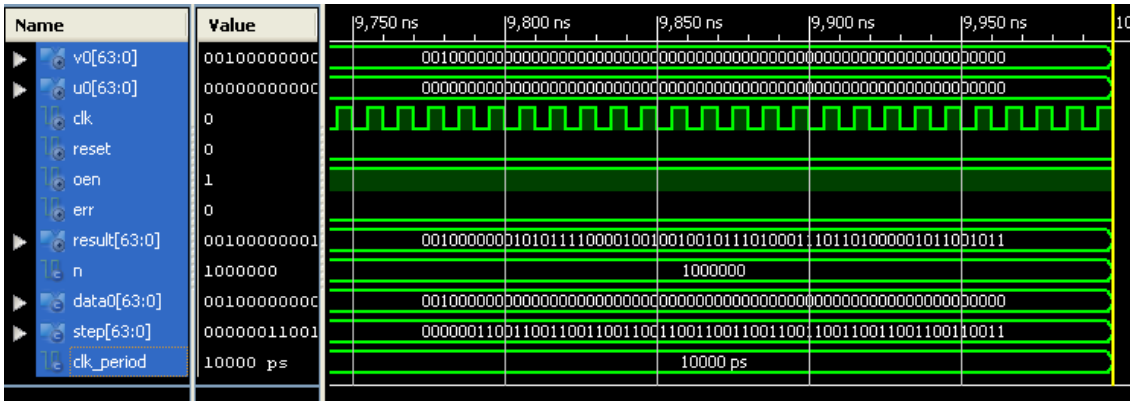
Výsledky simulácie:

$$\begin{aligned} (00100000001010111100001001001000)_2 &= (1.005341663956642)_{10} \\ (00100000001010111100001001001001...)_2 &= (1.005341666666666...)_{10} \end{aligned} \tag{5.4}$$

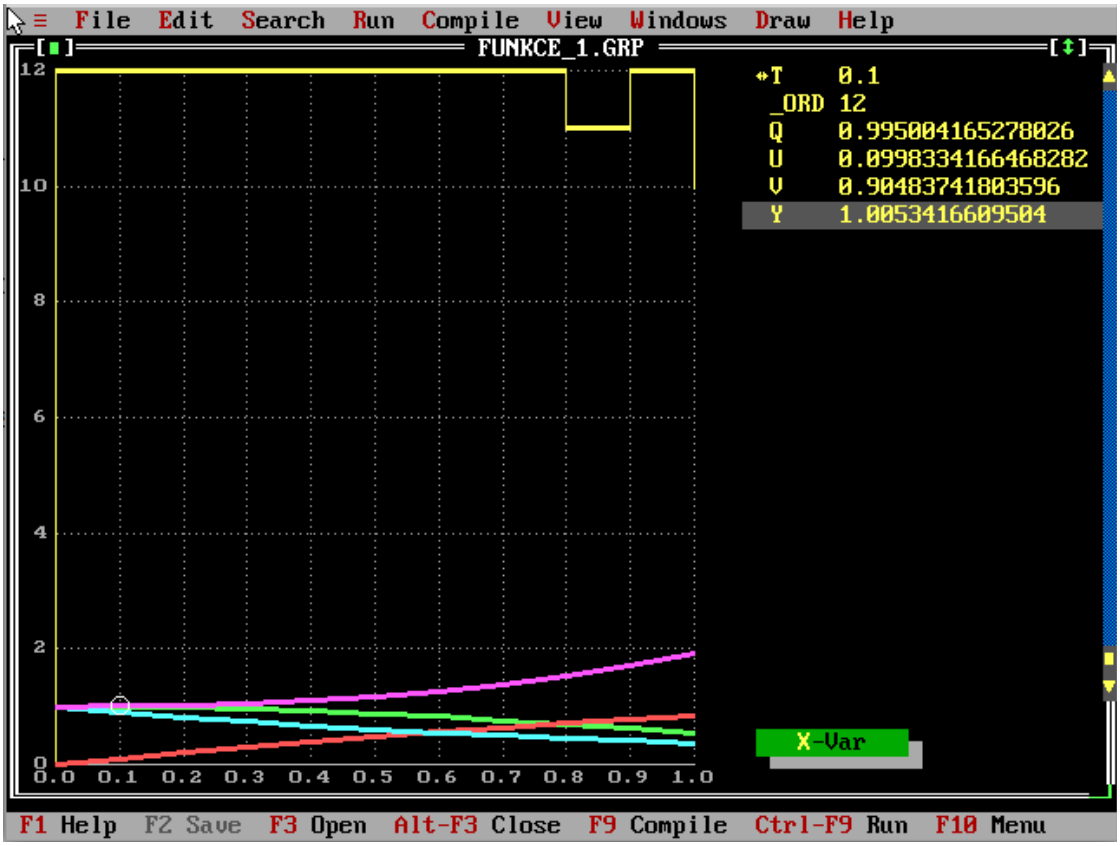
V porovnaní s výsledkami z TKSL 5.4 (viac na [3]) (1.0053416609504) sa výsledky s použitím Taylorovej rady 4. radu líšia po prevode do desiatkovej sústavy až na 9. desatinnom mieste.



Obr. 5.2: Simulácia v programe XilinxISE na 32 bitoch



Obr. 5.3: Simulácia v programe XilinxISE na 64 bitoch



Obr. 5.4: Výpočet v TKSL

Kapitola 6

Simulátor paralelne-paralelného deliaceho integrátora

Simulátor je možné použiť pri výuke numerických metód diferenciálnych rovníc. Pre implementovanie som si zvolil programovací editor Netbeans[12] a jazyk Java. Jedná sa o objektovo orientovaný programovací jazyk, ktorého veľkou výhodou je, že je prenosný medzi rôznymi platformami. Na tvorbu grafického rozhrania som použil knižnicu *Swing*. Ide o Java knižnicu, ktorá umožňuje vytvárať grafické užívateľské rozhranie pomocou aplikačného editoru. Ovládanie editoru je intuitívne, a tak je možné rýchlo a jednoducho vytvárať rôzne okná, dialógy, tlačidlá, panely a iné komponenty, avšak, jednotlivé detaily je potrebné doladiť už priamo v zdrojovom kóde.

6.1 Počiatočné hodnoty

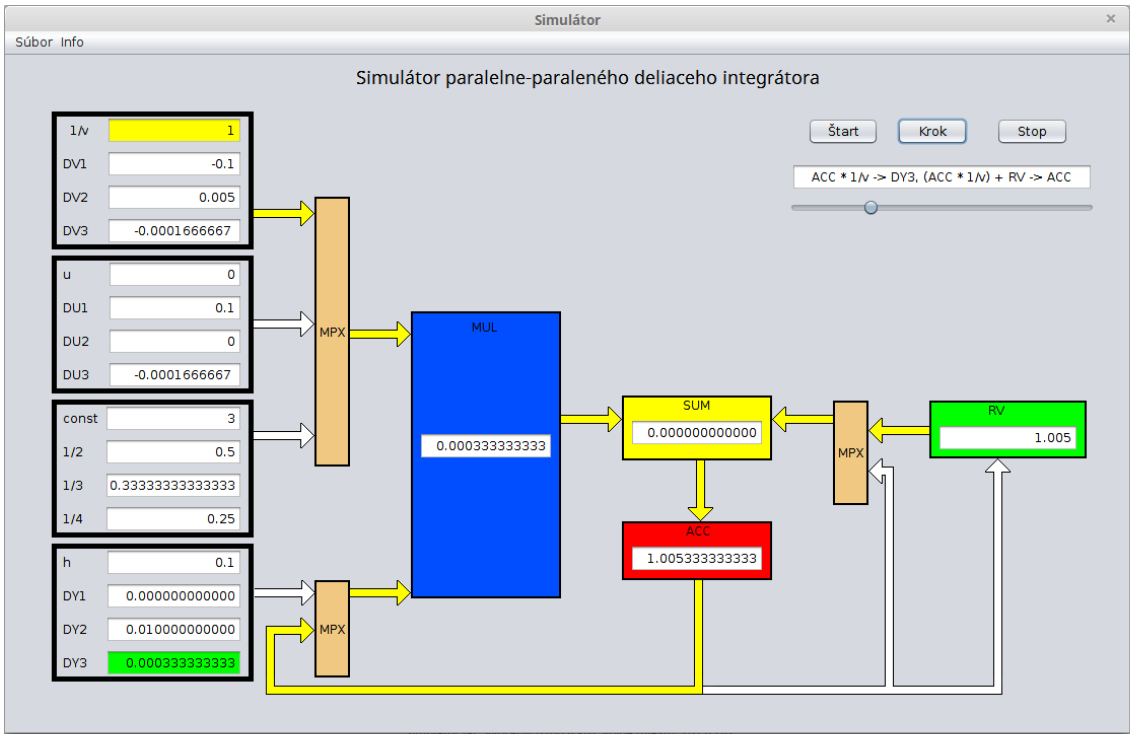
Po spustení programu sa zobrazí zjednodušená schéma PPDI, odvodená od návrhu 4.4 predstaveného v kapitole 4. Snímka 6.1 zachytáva simulátor počas behu. Na ľavej strane okna programu sa nachádzajú štyri bloky s textovými poľami, ktoré predstavujú registre a ich aktuálne hodnoty. Pri spustení sú hodnoty registrov nastavené na počiatočné podmienky tak, aby riešili diferenciálnu rovnicu 2.22 s integračným krokom 0.1. Ak chce užívateľ vložiť vlastné hodnoty, jednoducho stačí, aby aktuálne hodnoty v registroch *1/v*, $DV1 - DV3$, *u*, $DU1 - DU3$, *h* a *RV* prepísal. Registre *1/2*, *1/3*, *1/4* a *const* majú však nastavené pevné hodnoty, ktoré nie je možné meniť. Do registrov $DY1 - DY3$ sú vkladané hodnoty postupne počas výpočtu. Rovnako, aj hodnoty násobičky *MUL*, sčítačky *SUM* a akumulátora *ACC* sa menia počas výpočtu.

6.2 Ovládanie programu

Na ovládanie simulátora slúžia tlačidlá **Štart**, **Krok**, **Stop** a posuvný jazdec. Tlačidlom Štart spustíme simuláciu, ktorá sa prepína do nasledujúceho stavu v časových intervaloch. Pomocou posuvného jazdca môžeme meniť tieto intervaly, čiže rýchlosť simulácie. Tlačidlom Stop simuláciu zastavíme a tlačidlom Krok môžeme simuláciu postupne krokovať.
Horná lišta programu obsahuje menu bar s položkami *Subor* a *Info*. Po otvorení ponuky Súbor sa zobrazí ponuka *Reset* na reštartovanie výpočtu do počiatočného stavu a ponuka *Exit* na ukončenie aplikácie. Položka info obsahuje možnosť zobrazenia nového okna s návodom a okna so základnými informáciami o programe, a meno autora.

6.3 Priebeh simulácie

Po spustení simulácie tlačidlom Štart alebo Krok sú načítané hodnoty z registrov. Následne program skontroluje, či boli hodnoty z registrov zadané ako čísla v správnom formáte, čiže len pomocou použitia numerických znakov. Pri zlom formáte sa v textovom poli pod tlačidlami zobrazí chybová hláška, a výpočet je zastavený. Pri zadaní korektných počiatočných hodnôt (alebo ponechaním hodnôt preddefinovaných) výpočet postupuje stav po stave podľa konečného automatu 5.2. V každom stave sú *žltou* farbou vyznačené zbernice a registre, ktoré sú práve využívane na výpočet. Zbernica z násobičky *MUL* do registrov *DY1 – DY3* nie je zobrazená kvôli prehľadnosti. Ukladanie hodnôt do týchto registrov je užívateľovi znázornené *zelenou* farbou, ako môžeme vidieť na obrázku 6.1. Takto vyznačený je aj register const pri aktualizovaní svojej hodnoty. Po skončení výpočtu je do textového poľa pod tlačidlami zobrazená hláška *Koniec výpočtu*. Konečný výsledok je zobrazený v registri *RV*.



Obr. 6.1: Simulátor paralelne-paralelného deliaceho integrátora

Kapitola 7

Záver

V tejto práci sme sa oboznámili s numerickou integráciou a numerickými metódami, ktoré sa používajú na výpočet diferenciálnych rovníc. S využitím Taylorovej rady sme riešili diferenciálnu rovnicu s operáciou delenie, kde sme sa snažili uľahčiť výpočet dif. rovnice prevedením operácie delenia na operáciu násobenia pomocou substitúcie. Tento prístup sa ukázal ako neefektívny na rozdiel od riešenia bez použitia substitúcie. Na základe týchto výsledkov sme vytvorili návrh sérivo-paralelného a paralelne-paralelného deliaceho integrátora s využitím Taylorovej rady 4. rádu. Návrh sme realizovali tak, aby výpočet prebiehal rýchlo a efektívne. Preto sme výpočet časovo náročnej a zložitej operácie delenia zredukovali len na jeden výpočet počas celého výpočtu.
Podľa tohto návrhu sme implementovali paralelne-paralelený deliaci integrátor pre použitie vo FPGA. Integrátor obsahuje SRT deličku, ktorej princíp sme podrobne popísali a algoritmus delelenia sme upravili podľa požiadaviek výpočtu prebiehajúceho v paralelne-paralelnom deliacom integrátore tak, aby bolo možné ním počítať so všetkými číslami zobraziteľnými na danej architektúre v pevnej rádovej čiarke v dvojkovom doplnkovom kóde. Následne bol vytvorený softvérový simulátor, ktorý zobrazuje priebeh v paralelne-paralelnom deliacom integrátore. Simulátor pre jeho jednoduché grafické prevedenie, intuitívne ovládanie a názornosť je možné využiť na výukové učely v predmetoch zaoberajúcimi sa výpočtom numerickej integrácie alebo programovaním hardvéra.
V tejto práci je možné ďalej pokračovať návrhom a hardverovou implementáciou deliacich integrátorov v pohyblivej rádovej čiarke. Ďalšou možnosťou je použitie SRT algoritmu, ktorý by odhadoval číslice podielu podľa viacerých bitov priebežného zvyšku. Zaujímavou úpravou by mohlo byť použitie vyššieho radixu v násobičke a deličke. Rozsiahlejšou témou by mohlo byť vytvorenie a harverová implementácia prepojovacej siete s viacerými integrátormi.

Literatúra

[1] FIT: Domovská web stránka fit-kitu. `http://merlin.fit.vutbr.cz/FITkit/`, [cit. 10.5.2016].

[2] Jean Pierre Deschamps, Gustavo D. Sutter, Enrique Cantó: *Guide to FPGA Implementation of Arithmetic Functions*. Springer, 2012, ISBN 978-94-007-2986-5, VHDL kódy dostupné z: `http://www.arithmetic-circuits.org/guide2fpga/vhdl_codes.htm`.

[3] Jiří Kunovský: TKSL/386 [online]. `http://www.fit.vutbr.cz/ kunovsky/TKSL/tksl386.html.cs`.

[4] Koncaliev, D.: Bugs in the Intel Microprocessors [online]. `https://www.cs.earlham.edu/ dusko/cs63/fdiv.html`.

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

[6] Kraus, M.: *Paralelní výpočetní architektury založené na numerické integraci*. Disertačná práca, FIT VUT v Brne, Brno, 2013.

[7] Mircea Biagini, A. M.: Project Summary for ECE 429 Introduction to VLSI Design. `https://www.austinandalyssa.com/austin/professional/projects/alu/proj429.pdf`, 2001-06-05, [Online; navštívené 10.4.2016].

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

[9] Peringer, P.: Modelování a simulace IMS. FIT VUT v Brně, 2012-12-11, studíjní opora.

[10] Sekanina, L.: Dělení. , 2015, slajdy k predmetu INP - Návrh počítačových systémů.

[11] Soukup, L.: *Návrh a realizace matematických operací v obvodech FPGA*. Bakalářska práce, FIT VUT v Brně, Brno, 2009.

[12] Web: NetBeans IDE [online]. Dostupné z: https://netbeans.org/.

Prílohy

Zoznam príloh

Príloha A

Obsah CD

Priložené CD obsahuje:

- Text tejto práce vo formáte Pdf.
- Zdrojové súbory tejto práce v LaTeXu.
- Zdrojové súbory VHDL paralelného deliaceho integrátora.
- Zdrojové súbory simulátora v jazyku Java.
- Spustiteľný súbor simulátora – *Simulator.jar*.