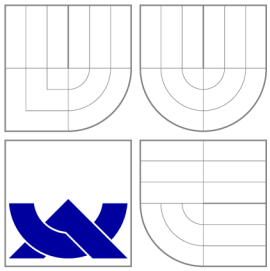
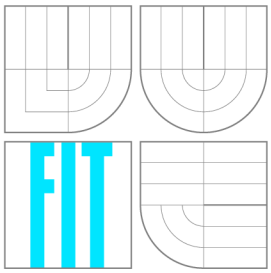


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



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

SIMULÁTOR PROCESORU S OPERACÍ NÁSOBENÍ

MULTIPLE OPERATION SIMULATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

VLADISLAV ZÁVADA

VEDOUCÍ PRÁCE
SUPERVISOR

Doc. Ing. JIŘÍ KUNOVSKÝ, CSc.

BRNO 2016

Zadání bakalářské práce/18534/2015/xzavad12

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav inteligentních systémů Akademický rok 2015/2016

Zadání bakalářské práce

Řešitel: **Závada Vladislav**
Obor: Informační technologie
Téma: **Simulátor procesoru s operací násobení**
Multiple Operation Simulation
Kategorie: Modelování a simulace

Pokyny:

1. Seznamte se s numerickou integrací a principem násobícího numerického integrátoru pro přímé využití Taylorovy řady.
2. Seznamte se s variantami paralelních, sériových a sériově-paralelních integrátorů (PPI, SPI, SSI)
3. Navrhněte a implementujte v provedení FPGA řídící systém uvedených variant integrátorů pro zadaný řád integrační metody, pro zvolenou délku slova a pro zadaný krok výpočtu.
4. Vytvořte simulátor řídícího systému.

Literatura:

- Kraus,M.: Paralelní výpočetní architektury založené na numerické integraci, Dizertační práce, VUT FIT 2013
- podle pokynů vedoucího

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 a 2

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Kunovský Jiří, doc. Ing., CSc.**, UITS FIT VUT
Datum zadání: 1. listopadu 2015
Datum odevzdání: 18. května 2016

L.S.

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav inteligentních systémů
612 66 Brno, Božetěchova 2

doc. Dr. Ing. Petr Hanáček
vedoucí ústavu

Abstrakt

Tato práce se zabývá numerickou integrací. Nejprve je popsáno numerické řešení diferenciálních rovnic použitím metody Taylorovy řady. Poté jsou popsány jednotlivé varianty integrátorů. V praktické části je popsán návrh dvouvstupého integrátoru násobení a dále jeho realizace v prostředí FPGA. Pro tento integrátor je také vytvořen simulátor znázorňující jeho funkci.

Abstract

This work deals with the numeric integration. The reader is acquainted with the numerical solution of differential equations using Taylor series method. Then describes the different variants integrators. The practical part describes the design double-input integrator with multiplication and its implementation in an FPGA. For this integrator is also provided a simulator demonstrate its function.

Klíčová slova

diferenciální rovnice, numerická integrace, Taylorova řada, integrátor, řadič, násobení, simulace, VHDL, FPGA

Keywords

diferential equation, numeric integration, Taylor series, integrator, controller, multiplication, simulation, VHDL, FPGA

Citace

ZÁVADA, Vladislav. *Simulátor procesoru s operací násobení.* Brno, 2016. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Kunovský Jiří.

Simulátor procesoru s operací násobení

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Doc. Ing. Jiřího Kunovského, CSc. Další informace mi poskytli Jan Opálka, František Matečný. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Vladislav Závada
16. května 2016

Poděkování

Tímto bych chtěl poděkovat svému školiteli Doc. Ing. Jiřímu Kunovskému, CSc. za podporu, odborné vedení, cenné rady, náměty a připomínky a hlavně přátelský přístup. Taky bych chtěl poděkovat Janu Opálkovi za pomoc a předání zkušeností, které mi při vypracování této práce velmi pomohly. Mé poděkování patří také rodině, skautům, přátelům a všem, kteří mě během studia podporovali.

Obsah

Kapitola 1

Úvod

Hlavním pilířem této práce je Numerická integrace. Numerická integrace se využívá při numerickém řešení diferenciálních rovnic. Hardwarové bloky realizující tuto integraci se nazývají integrátory. Cílem této práce je navrhnout a implementovat v prostředí FPGA integrátor, který realizuje výpočet diferenciální rovnice se součinem dvou funkcí.
Nejprve jsou kapitole v (2) shrnuty nejpoužívanější jednokrokové metody numerické integrace a jejich vlastnosti. Podrobně je zde popsána metoda použití Taylorovy řady (2.5). Nejdůležitější vlastností této metody je možnost transformovat veškeré výpočty derivací na operace sčítání a násobení. Tyto operace realizují integrátory, které jsou popsány v kapitole (3). V této kapitole je nejdříve popsán obecný numerický integrátor. Dále je zde popsána struktura a funkce tří typů integrátorů, popsaných v disertaci [9]. Jádrem této kapitoly je návrh paralelně-paralelního integrátoru se dvěma vstupy, který realizuje operaci násobení. Ten je popsán v podkapitole (3.4). Nejdůležitější vlastností výpočtu, realizovaného integrátory, je možnost jejich zapojení do sítě integrátorů (3.6), kdy integrátory realizují výpočet paralelně.
V kapitole (4) je popsána analýza potřebných řídících signálů a návrh řadiče pro paralelně-paralelní integrátor násobení. Je zde také podrobně rozepsán mikroprogram, podle kterého tento řadič řídí výpočet integrátoru. V kapitole (5) je popsána implementace integrátoru v jazyku VHDL a její použití v FPGA. Poslední kapitola (6) je věnována popisu simulátoru, který slouží k názorné prezentaci funkce paralelně-paralelního integrátoru. Tento simulátor byl navržen tak, aby mohl sloužit při výuce numerické integrace na naší fakultě.

Kapitola 2

Numerické řešení diferenciálních rovnic

Tato kapitola popisuje řešení diferenciálních rovnic pomocí numerických metod (tzv. Numerickou integrací). Výpočet pomocí těchto metod může být realizován elementárním procesorem, který bude popsán v dalších kapitolách.
Při použití těchto metod jsou nejdůležitějšími parametry *přesnost* a *rychlost*. Vždy se podle potřeby upřednostňuje jedno na úkor druhého nebo se hledá kompromis. Výsledky získané použitím metod numerické integrace jsou tedy počítány pouze s určitou odchylkou.
Numerická integrace se používá hlavně v případech, kdy je analytické řešení velmi složité nebo nemožné. Více informací o numerické integraci lze nalézt v [10], [4], [9] odkud byly čerpány i informace, které jsou popsány dále v této kapitole.

2.1 Diferenciální rovnice

Diferenciální rovnice jsou matematické rovnice, ve kterých se vyskytují funkce společně s jejich derivacemi. U diferenciálních rovnic určujeme řád. Řádem rovnice rozumíme nejvyšší derivaci, která se v rovnici vyskytuje. Těmito rovnicemi popisujeme mnoho fyzikálních dějů. Jejich aplikace najdeme ve většině oblastí lidského vědění. Existují dva způsoby řešení diferenciální rovnice: analytické řešení a numerické řešení. Více o diferenciálních rovnicích lze nalézt v [6], odkud byly také čerpány tyto informace.

2.1.1 Analytické řešení

Při analytickém, neboli matematickém řešení Diferenciálních rovnic, se hledá řešení pomocí známých vztahů a postupných úprav. Tento postup je někdy velmi obtížný nebo nemožný, protože jsou například vzorce příliš složité nebo neexistují vhodné úpravy. V těchto případech je výhodné použít numerické řešení.

2.1.2 Numerické řešení

Numerické řešení funguje na principu tzv. aproximace řešení. Výpočet pomocí numerických metod se skládá z jednotlivých iterací. Po jednotlivých krocích se každým výpočtem přibližujeme ke správnému řešení, dokud není změna hodnoty menší než určená odchylka. Toto řešení poté považujeme za správné.

Takové řešení se používá, pokud je analytické řešení příliš komplikované nebo pokud není potřeba přesné řešení, kdy se spokojíme s určitou zanedbatelnou odchylkou.

2.1.3 Převod na diferenciální rovnice prvního řádu

Níže popsané metody se používají k řešení diferenciálních rovnic prvního řádu. Diferenciální rovnice vyšších řádů je potřeba nejdříve převést na soustavu rovnic prvního řádu. Na převod rovnice na nižší řád je možné použít metodu postupné integrace nebo metodu snižování řádu derivace viz. [13].

Metoda snižování řádu derivace

Metoda snižování řádu derivace je jednodušší, ale vyžaduje, aby na pravé straně rovnice nebyly derivace vstupu. Postup metody je následující:

- Upravíme rovnici tak, aby byl na levé straně nejvyšší řád derivace.
- Poté lze sérií integrací získat nižší derivace dokud nezískáme y.

$$y^{(n-1)} = \int y^{(n)} dt \tag{2.1}$$

$$y^{(n-2)} = \int y^{(n-1)} dt \tag{2.2}$$

$$\vdots$$

$$y = \int y' dt \tag{2.3}$$

Metoda postupné integrace

Metodou postupné integrace je možné řešit i rovnice s derivacemi vstupu na pravé straně. Postup je následující:

- Upravím rovnici tak, aby na levé straně byl pouze nejvyšší řád derivace.
- Integrujeme obě strany a průběžně zavádíme stavové proměnné w_i pro ty podvýrazy, kde zůstane integrál. Tento krok opakujeme dokud nemáme rovnici pro výpočet y.
- Nakonec provedeme výpočet nových počátečních podmínek pro všechny stavové proměnné, které jsme zavedli.

Metoda je použitelná pouze v případě, že jsou u derivací pouze konstantní koeficienty a nejvyšší řád derivace vstupu x musí být menší, než nejvyšší řád derivace výstupu y.

2.2 Numerické metody pro řešení diferenciálních rovnic

Jednotlivé metody jsou podrobněji popsány v [11], odkud byly čerpány i informace pro tuto práci.
Numerické metody se dělí na jednokrokové a vícekrokové.

2.2.1 Jednokrokové metody

Jednokrokové metody počítají následující krok výpočtu pouze s použitím hodnoty kroku předcházejícího. Výpočet probíhá iteračně. První hodnotou pro výpočet je počáteční podmínka.

2.2.2 Vícekrokové metody

Vícekrokové metody na rozdíl od jednokrokových používají k výpočtu následujícího kroku hodnoty několika předchozích kroků. Na začátku výpočtu těmito metodami je zapotřebí vypočítat několik hodnot pomocí jednokrokové metody, tak aby měla vícekroková metoda dostatek hodnot k výpočtu.

2.2.3 Vlastnosti numerických metod

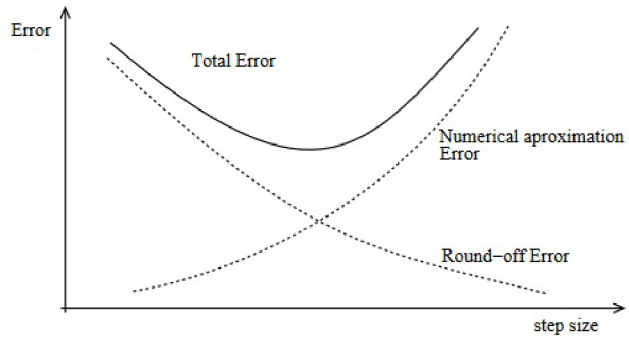
Nejdůležitější vlastnost numerických integračních metod je přesnost, se kterou počítají. Další důležitou vlastností je stabilita numerického řešení při použití dané metody.

Chyba numerických metod

Výsledná chyba numerické metody je dána součtem několika vlivů:

- Lokální chyby - chyby vznikající v jednom kroku
  - Chyba zaokrouhlovací, je závislá na přesnosti aritmetiky počítače a také na velikosti datového typu. Například při počítání s přesností float se tato chyba projeví citelněji než s přesností double.
  - Chyba numerické metody je dána především řádem metody, tedy počtem členů Taylorovy řady které použijeme. Metody vyšších řádů mají obecně vyšší přesnost při stejné délce kroku.
- Akumulovaná chyba, vzniká a roste v průběhu výpočtu, protože se sčítají vlivy lokálních chyb.

Vliv lokálních chyb na přesnost metody v závislosti na délce kroku znázorňuje obrázek



Obrázek 2.1: Závislost chyby numerických metod na délce kroku [13]

Vliv lokální chyby na přesnost numerických metod v závislosti na délce kroku, znázorňuje obrázek (2.1).

2.3 Eulerova metoda

Speciální jednokrokovou metodou na principu Taylorovy řady je Eulerova metoda, která k řešení diferenciálních rovnic používá pouze první člen Taylorovy řady:

$$y_{i+1} = y_i + h \cdot y_i' \tag{2.4}$$

Touto metodou se dá dosáhnout poměrně přesných výsledků, pokud je vhodně zvolen integrační krok h. Čím je tento krok menší, tím se dosáhne přesnějších výsledků, ale také klesá rychlost výpočtu.

2.4 Metoda Runge-Kutta

Tak jako Eulerova metoda, je i metoda Runge-Kutta jednokroková. Oproti Eulerově metodě je přesnější, ale na druhou stranu zase pomalejší. K výpočtu následujícího členu je nejprve nutno spočítat určité mezi-výpočty. Jejich počet udává řád metody a přírůstek je dán váženým průměrem těchto mezi-výpočtů.

Obecně lze metodu zapsat následovně:

$$y_{n+1} = y_n + h \sum_{i=1}^{p} \omega_i k_i \tag{2.5}$$

$$k_i = f(t + \alpha_i h, y_n + h \sum_{j=1}^{i-1} \beta_{ij} k_j) \tag{2.6}$$

Konstanty $\alpha_i, \beta_{ij}, \omega_i$ jsou vhodně zvoleny, tak aby řád metody odpovídal Taylorovu polynomu funkce stejného řádu a aby získané řešení souhlasilo s Taylorovou řadou.

Nejrozšířenější je metoda Runge-Kutta 4. řádu, která má dobrý poměr mezi přesností a rychlostí. Tvar rovnic je následující:

$$y_{i+1} = y_i + 1\frac{1}{6}h(k_1 + 2k_2 + 2k_3 + k_4) \tag{2.7}$$

$$k_1 = f(t_i, y_i) \tag{2.8}$$

$$k_2 = f(t_{i+1} + \frac{1}{2}h, y_i + \frac{1}{2}hk_1) \tag{2.9}$$

$$k_3 = f(t_{i+1} + \frac{1}{2}h, y_i + \frac{1}{2}hk_2) \tag{2.10}$$

$$k_4 = f(t_{i+1} + h, y_i + hk_3) \tag{2.11}$$

2.5 Taylorova řada

Metoda výpočtu pomocí Taylorovy řady je považována za základní jednokrokovou metodu. Tato metoda se dá vyjádřit vztahem:

$$y_{i+1} = y_i + \frac{h}{1!} \cdot y_i' + \frac{h^2}{2!} \cdot y_i'' + \frac{h^3}{3!} \cdot y_i''' + \cdots + \frac{h^p}{p!} \cdot y_i^{(p)} \tag{2.12}$$

Proměnná h nám určuje velikost kroku výpočtu. Následující krok je vypočítán z součtu hodnoty předchozího kroku a jednotlivých členů Taylorovy řady. Tyto členy jsou iterativně počítány, dokud není rozdíl dvou po sobě jdoucích členů Taylorovy řady, menší než požadovaná odchylka. Nevýhodou tohoto postupu je nutnost počítat vyšší derivace. Tato nevýhoda se stupňuje s počtem členů Taylorovy řady. Tento problém je ale postupem popsaným dále elegantně vyřešen.

2.5.1 Výpočet jednoduché diferenciální rovnice Taylorovou řadou

Postup výpočtu je naznačen na jednoduché lineární diferenciální rovnici 1.řádu se zadanou počáteční podmínkou. Funkce je zadána následovně:

$$y' = y \qquad y(0) = y_0 \tag{2.13}$$

Ze zadané rovnice vyplývá vztah:

$$y = y' = y'' = y''' = \cdots = y^{(n)} \tag{2.14}$$

V tomto případě tedy můžeme Taylorovu řadu (2.12) dosazením upravit na:

$$y_{i+1} = y_i + \frac{h}{1!} \cdot y_i + \frac{h^2}{2!} \cdot y_i + \frac{h^3}{3!} \cdot y_i + \cdots + \frac{h^p}{p!} \cdot y_i \tag{2.15}$$

Tyto členy substituujeme podle vztahu (2.25)

$$DY0_i = y_i \tag{2.16}$$

$$DY1_i = \frac{h}{1!} \cdot y_i \tag{2.17}$$

$$DY2_i = \frac{h^2}{2!} \cdot y_i \tag{2.18}$$

$$\vdots$$

$$DYp_i = \frac{h^p}{p!} \cdot y_i \tag{2.19}$$

A vynikne vztah:

$$y_{i+1} = DY0_i + DY1_i + DY2_i + DY3_i + \cdots + DYp_i \tag{2.20}$$

Jednotlivé členy Taylorovy řady lze vyjádřit tak, že do vztahu dosadíme předchozí člen řady, čímž se vyhneme výpočtu vyšších derivací a značně tím výpočet zjednodušíme. Touto úpravou dostaneme vztahy:

$$\begin{aligned} DY0_i &= y_i && (2.21)\\ DY1_i &= h \cdot DY0_i && (2.22)\\ DY2_i &= \frac{h}{2} \cdot DY1_i && (2.23)\\ &\vdots \\ DYp_i &= \frac{h}{p} \cdot DY(p-1)_i && (2.24) \end{aligned}$$

Ze vztahu (2.24) vyplývá, že následující člen Taylorovy řady se dá vypočítat pomocí předchozího členu Taylorovy řady. Tímto způsobem se dá výpočet velmi zefektivnit, protože se vyhneme počítání vyšších, mnohdy velmi složitých vyšších derivací funkcí, které vyšší členy Taylorovy řady obsahují.

2.5.2 Řešení obecné diferenciální rovnice součinu

Začneme tím že se jednotlivé členy Taylorovy řady dají substituovat:

$$y_{i+1} = DY0_i + DY1_i + DY2_i + DY3_i + \cdots + DYp_i \quad (2.25)$$

Kde význam jednotlivých substitucí je následovný:

$$\begin{aligned} DY0_i &= y_i && (2.26)\\ DY1_i &= \frac{h}{1!} \cdot y_i' && (2.27)\\ DY2_i &= \frac{h^2}{2!} \cdot y_i'' && (2.28)\\ &\vdots \\ DYp_i &= \frac{h^p}{p!} \cdot y_i^{(p)} && (2.29) \end{aligned}$$

Z předchozích rovnic vyplývá obecný vztah:

$$y_i^{(p)} = \frac{DYp_i}{\frac{h^p}{p!}} \quad (2.30)$$

Z tohoto vztahu vyplývá, že lze jakoukoliv derivaci nahradit podílem příslušného členu Taylorovy řady a části kroku výpočtu h.

Předpokládejme zadanou obecnou diferenciální rovnici se zadanou počáteční podmínkou:

$$y' = q \cdot r \qquad y(0) = y_0 \quad (2.31)$$

Pro každou proměnnou v rovnici je potřeba vytvořit Taylorovu řadu. Taylorovy řady pro zadanou rovnici vypadají následovně:

$$Y1_{i+1} = DY0_i + DY1_i + DY2_i + DY3_i + \cdots + DY(n)_i \quad (2.32)$$
$$Q1_{i+1} = DQ0_i + DQ1_i + DQ2_i + DQ3_i + \cdots + QZ(n)_i \quad (2.33)$$
$$R1_{i+1} = DR0_i + DR1_i + DR2_i + DR3_i + \cdots + DR(n)_i \quad (2.34)$$

Výpočet Taylorovy řady pro rovnici součinu se počítá následujícím způsobem:

Nultý člen Taylorovy řady je roven počáteční podmínce:

$$DY0 = Y_0 \tag{2.35}$$

Vyjádření prvního členu Taylorovy řady je následující:

$$DY1 = h \cdot y' \tag{2.36}$$

Podle vztahu (2.31) je y' na hrazen vztahem $q \cdot r$:

$$DY1 = h \cdot q \cdot r \tag{2.37}$$

Zderivujeme počáteční funkci (2.31) a dostaneme vztah:

$$y'' = q' \cdot r + q \cdot r' \tag{2.38}$$

Podle vztahu (2.30), nahradíme derivace pomocí prvků Taylorovy řady.

$$DY2 = \frac{h^2}{2!} \cdot (\frac{DQ1}{h} \cdot r + q \cdot \frac{DR1}{h}) \tag{2.39}$$

Obdobně postupujeme u třetího prvku další derivací funkce:

$$y''' = q'' \cdot r + 2 \cdot q' \cdot r' + q \cdot r'' \tag{2.40}$$

Opět nahradíme derivace členy Taylorovy řady podle vztahu (2.30) a vyjádříme třetí člen:

$$DY3 = \frac{h^3}{3!}(\frac{DQ2}{\frac{h^2}{2!}} \cdot r + 2 \cdot \frac{DQ1}{h} \cdot \frac{DR1}{h} + q \cdot \frac{DR2}{\frac{h^2}{2!}}) \tag{2.41}$$

Obdobně postupujeme při vyjadřování dalších členů. Jednotlivé členy lze následně upravit pokrácením. Zde jsou uvedeny první čtyři členy:

$$DY1 = h \cdot q \cdot r \tag{2.42}$$

$$DY2 = \frac{h}{2} \cdot (DQ1 \cdot DR0 + DQ0 \cdot DR1) \tag{2.43}$$

$$DY3 = \frac{h}{3}(DQ2 \cdot DR0 + DQ1 \cdot DR1 + DQ0 \cdot DR2) \tag{2.44}$$

$$DY4 = \frac{h}{4}(DQ3 \cdot DR0 + DQ2 \cdot DR1 + DQ1 \cdot DR2 + DQ0 \cdot DR3) \tag{2.45}$$

Z těchto vztahů je patrné, že jednotlivé členy se rozvíjejí a tento rozvoj lze algoritmizovat.
Výsledný vztah pro výpočet i-tého členu vypadá následovně

$$\begin{aligned} DY_i = \frac{h}{i}(&DQ_{i-1} \cdot DR_{i-i} + DQ_{i-2} \cdot DR_{i-(i-1)} + \\ &+DQ_{i-(i-1)} \cdot DQ_{i-2} + DQ_{i-i} \cdot DR_{i-1}) \end{aligned} \tag{2.46}$$

Tímto postupem vyjadřujeme další členy Taylorovy řady až do požadované přesnosti.

2.5.3 Řešení konkrétní diferenciální rovnice součinu

Předpokládejme zadanou rovnici součinu se zadanou počáteční podmínkou:

$$y' = e^t \cdot sin(t) \qquad y(0) = 1 \tag{2.47}$$

Před samotným výpočtem je vhodné vytvořit tzv. tvořící rovnice. Které následně umožní celý výpočet provádět paralelně. Zavedeme substituce:

$$q = e^t \qquad r = sin(t) \tag{2.48}$$

Z kterých plyne:

$$y' = q \cdot r \tag{2.49}$$

Derivace e^t je rovna e^t. Z toho plyne q'= q

$$e^{t'} = e^t \tag{2.50}$$
$$q' = q \tag{2.51}$$

Derivace sin(t) je rovna cos(t), což nevyhovuje našemu výpočtu a tak provedeme další substituci:

$$r' = g \tag{2.52}$$

A následnou derivací dostaneme mnohem vhodnější vztah

$$g = cos(t) \tag{2.53}$$
$$g' = -sin(t) \tag{2.54}$$
$$g' = -r \tag{2.55}$$

Tímto způsobem jsme vytvořili čtyři tvořící rovnice. Tyto diferenciální rovnice odpovídající naší zadané diferenciální rovnici. Následně ještě vypočítáme počáteční podmínky dosazením 0 do proměnné t

$$y' = q \cdot r \qquad y(0) = 0 \tag{2.56}$$
$$q' = q \qquad q(0) = 1 \tag{2.57}$$
$$r' = g \qquad r(0) = 1 \tag{2.58}$$
$$g' = -r \qquad g(0) = 1 \tag{2.59}$$

Pro každou proměnnou je zapotřebí vytvořit Taylorovu řadu. Taylorovu řadu pro rovnici součinu (2.46) a Taylorovu řadu pro jednoduchou diferenciální rovnici (2.24) máme vyjádřenu výše . Výpočty těchto řad budou realizované integrátory popsanými v následující kapitole.

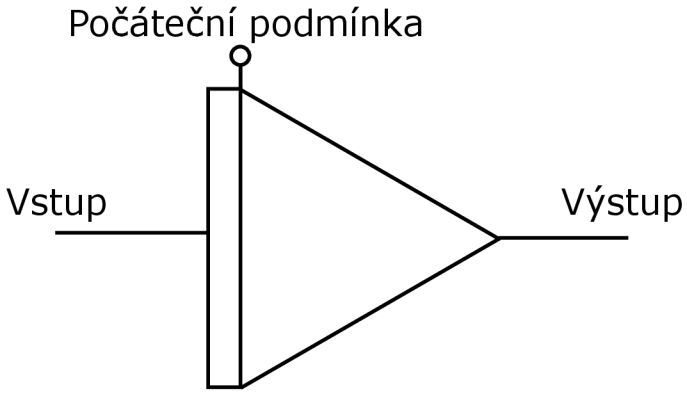
2.6 Shrnutí

Jak je patrné z rovnice (2.46), výpočet diferenciálních rovnic pomocí metody Taylorovy řady se po výše popsaném postupu zjednoduší pouze na operace **násobení** (dvojic předchozích členů a členů s krokem h) a **sčítání** (jak jednotlivých součinů uvnitř členu tak sčítání celých členů Taylorovy řady).
Tyto operace lze efektivně realizovat pomocí Hardwarového vybavení. V následující kapitole bude tato realizace popsána.

Kapitola 3

Numerický integrátor

Numerický integrátor je logický obvod, který provádí výpočet numerické integrace. Integrátor musí mít zadanou počáteční podmínku, z které výpočet vychází. Pomocí operací součtu a součinu provádí integraci vstupní hodnoty. Integrátory lze rozdělit na invertující a neinvertující. Pokud je potřeba změnit znaménko výstupu, lze připojit invertor, který znaménko změní.



Obrázek 3.1: Blokové schéma integrátoru

Na obrázku je zobrazeno blokové schéma numerického integrátoru. Toto schéma se používá při grafickém návrhu sítě integrátorů, které slouží k řešení soustav diferenciálních rovnic. Příklad takového návrhu je uveden níže (3.7).

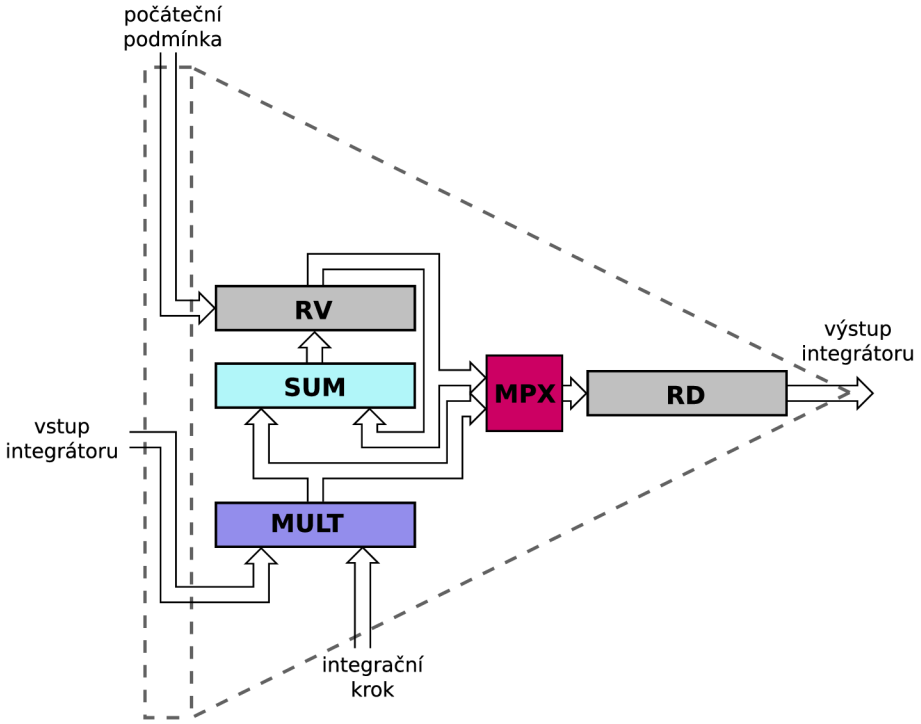
Operace sčítání a násobení lze provádět jak paralelně, tak sériově. Taktéž komunikace mezi integrátory může probíhat jak paralelní, tak sériovou cestou. Na základě těchto parametrů jsou v disertační práci [9] odvozeny tyto typy integrátorů:

- paralelně - paralelní integrátor (paralelní komunikace - paralelní výpočet)
- sériově - paralelní integrátor (sériová komunikace - paralelní výpočet)
- sériově - sériový integrátor (sériová komunikace - sériový výpočet)

V následujících podkapitolách budou tyto integrátory popsány detailně. Obrázky a popisy byly převzaty z dizertační práce [9] a bakalářské práce [12]. V diplomové práci [8] jsou také detailně popsány jednotlivé komponenty integrátorů.

3.1 Paralelně-Paralelní integrátor

Násobení v integrátoru je realizováno paralelní kombinační násobičkou a sčítání paralelní, kombinační sčítačkou. Zkratka integrátoru je PPI. Blokové schéma integrátoru je na následujícím obrázku. Čárkovaně je naznačen symbol integrátoru.



Obrázek 3.2: Blokové schéma paralelně-paralelního integrátoru

Význam jednotlivých bloků:

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

Postup výpočtu realizovaného integrátorem je následující:
Výpočet začíná uložením hodnoty y_i do registrů (RD) a (RV). Integrační krok se nastaví na hodnotu h. Na vstup je přivedena hodnota $f(y_i)$, což je výstupní hodnota prvku, který je připojen na vstup integrátoru. Hodnoty kroku h a vstupů integrátoru jsou vynásobeny násobičkou (MUL). Výsledná hodnota je uložena v registru (RD) a současně sečtena s registrem (RV), kde je i následně uložena. V (RD) je tedy uložena hodnota prvního členu Taylorovy řady $DY1$ a v (RV) je mezisoučet $y_i + DY1$. V dalším kroku je na vstup přivedena hodnota $f(DY1)$ a iterační krok $h/2$. Jejich vynásobením se vypočítá další člen Taylorovy řady $DY2$, který se uloží do (RD) a sečte s (RV). Opakováním cyklu se postupně v (RV) vytváří řada $y_i + DY_P + \cdots + DY_{p+n}$ dokud není dosaženo požadované přesnosti nebo maximálního počtu cyklů. Hodnota (RV) je poté přivedena na výstup a reprezentuje hodnotu y_{i+1}.

Tento typ integrátoru je z uvedených variant nejrychlejší, čas výpočtu jednoho členu Taylorovy řady udává rovnice:

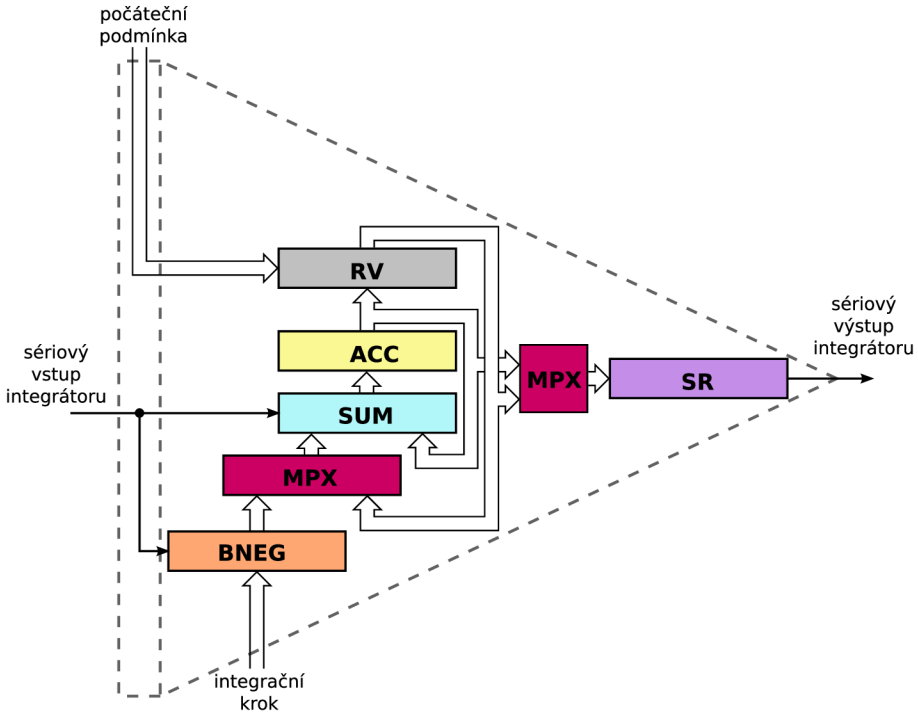
$$t_{PP} = \tau_{mult} + \tau_{sum} + \tau_{sit}$$

Čas výpočtu je tedy složen z času potřebného pro násobení τ_{mult} a času pro sčítání τ_{sum} a dále pak zpožděním signálů v propojovací síti τ_{sit}.

Největší výhodou tohoto integrátoru je tedy rychlost, která je nezávislá na počtu bitů, na kterých jsou čísla zobrazena. Mezi nevýhody patří větší prostorová složitost, zvláště kombinační násobičky. Další nevýhodou je velký počet vstupů a výstupů, který je přímo úměrný šířce použité datové sběrnice.

3.2 Sériově-Paralelní integrátor

Tato varianta se od paralelně-paralelního integrátoru liší tím, že násobení se provádí sekvenční metodou na principu Boothova algoritmu (3.2.1) násobení. Sčítání se stále provádí paralelně v jednom kroku. Zkratka integrátoru je SPI. Blokové schéma integrátoru je na obrázku (3.3). Čárkovaně je naznačen symbol integrátoru.



Obrázek 3.3: Blokové schéma sériově-paralelního integrátoru

Význam jednotlivých bloků:

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

Výpočet začíná uložením hodnoty y_i do registrů (RD) a (RV). Integrační krok se nastaví na hodnotu h. Pomocí multiplexoru je přivedena cesta od (BNEG). Výpočet hodnoty y_{i+1} pak probíhá následovně:
Vynuluje se registr (ACC). Na vstup integrátoru je přiveden nejméně významový bit $f(y_i)$. Tento bit určuje, jestli se bude násobenec uložený v registru (RN) přičítat k (ACC), odečítat(vytvoří se záporný podoba násobence), nebo se bude ignorovat(vynuluje se). Výsledek ze sčítačky se zapíše do (ACC) a celý akumulátor a posuvný registr (RV) se vysune o jeden bit doprava. Tento postup se opakuje, dokud není přijat a zpracován poslední bit na vstupu $f(y_i)$.
Tímto postupem byli tedy vynásobeny hodnoty h a $f(y_i)$. Tato hodnota se uloží do (SR). Multiplexor je přepnut z (BNEG) na (RV) a výsledek násobení DY_p uložený v (ACC) se sečte s hodnotou v (RV) a uloží se do (ACC). Následně se hodnota z (ACC) uloží do (RV). Opakováním cyklu se postupně v (RV) vytváří řada $y_i + DY_P + \cdots + DY_{p+n}$ dokud není dosaženo požadované přesnosti nebo maximálního počtu cyklů, čímž získáme y_{i+1}.
Největší výhodou oproti paralelně-paralelnímu integrátoru je mnohem nižší počet vstupů a výstupů, protože data vstupují a vystupují sériově. Je-li potřeba zvýšit přesnost výpočtu zvýšením počtu bitů na kterých jsou čísla zobrazena, pak je potřeba změnit pouze šířky registrů, sčítačky a změnit řídící signály ve zbytku zapojení se nic nemění. Nevýhodou oproti PPI je zpomalení, které je naopak na počtu bitů n, na kterých jsou čísla zobrazena závislé, protože násobení je prováděno v n krocích. Čas výpočtu jednoho členu Taylorovy řady udává následující rovnice:

$$\begin{aligned} t &= \tau_{mult} + \tau_{sum} + \tau_{net} \\ t &= n{\cdot}\tau_{sum} + \tau_{sum} + \tau_{net} \end{aligned} \tag{3.1}$$

Celkový čas je dán opět součtem času násobení, sčítání a zpožděním signálů v propojovací síti. Čas násobení je v podstatě n kroků sčítání $n{\cdot}\tau_{sum}$.

3.2.1 Boothův algoritmus násobení

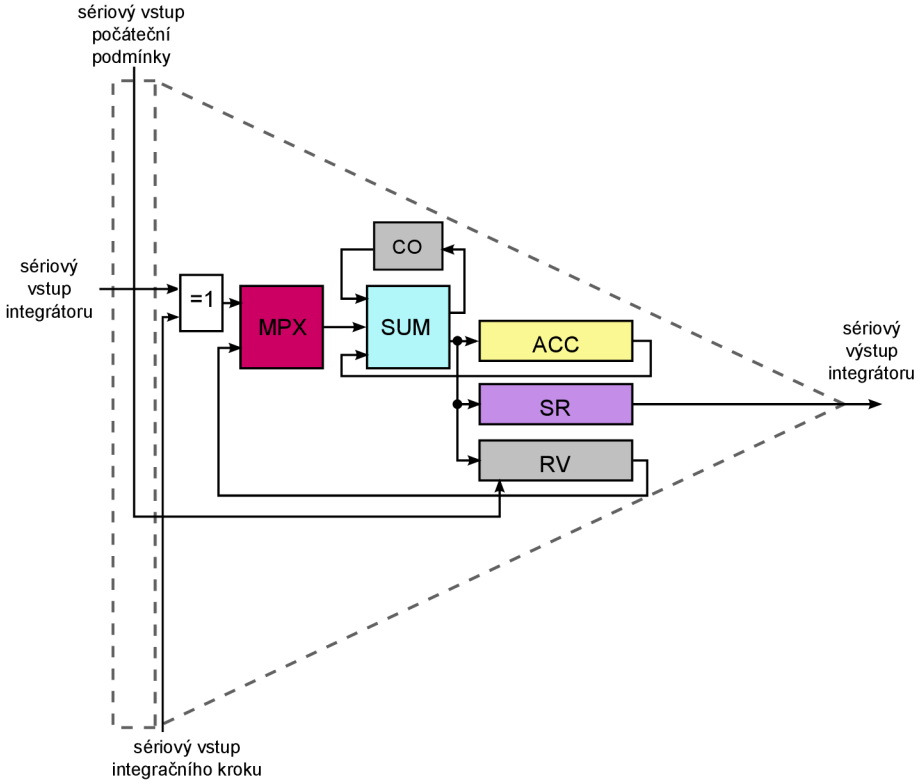
Boothův algoritmus násobení je použit v sériové násobičce. Ta je použita například v sériově-paralelním nebo sériově-sériovém integrátoru. Slouží pro násobení dvou operandů ve dvojkové soustavě. Je založen na operacích dílčích součtů nebo rozdílů a na operacích posuvu. Princip násobení je následující:
Vždy se porovnávají logické hodnoty dvou sousedních bitů. Podle hodnot těchto bitů se pak k mezivýsledku buď, přičte násobenec nebo odečte násobenec, popřípadě se mezivýsledek nezmění (přičtou se pouze nuly). Po této operaci je mezivýsledek posunut o jeden bit v vpravo. Poté porovnáváme další dva sousední bity v násobiteli. Tento postup se opakuje dokud nejsou porovnány všechny bity násobitele.

Tabulka 3.1: Boothův algoritmus s překódováním radix 2 [8]

bity násobitele		
bit_i	bit_{i-1}	akce při výpočtu
0	0	přičtení nul k mezivýsledku
0	1	přičtení násobence k mezivýsledku
1	0	odečtení násobence od mezivýsledku
1	1	přičtení nul k mezivýsledku

3.3 Sériově-sériový integrátor

Na rozdíl od předchozí varianty se v sériově-sériovém integrátoru provádí sekvenčně jak násobení tak i operace sčítání. Zkratka integrátoru je SSI. Blokové schéma integrátoru je na obrázku (3.4). Čárkovaně je naznačen symbol integrátoru.



Obrázek 3.4: Blokové schéma sériově-sériového integrátoru

Zde je popsán význam jednotlivých bloků:

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

Výpočet probíhá následovně: Nejprve je vynulován registr (ACC) a obvod pro uchování přenosu (CO). Do (RV) je uložena hodnota y_i. Multiplexor (MPX) se přepne na cestu z (ACC). Na vstup integrátoru je přiveden nejméně významový bit $f(y_i)$. Na vstup integračního kroku h jsou postupně přivedeny jednotlivé bity integračního kroku, počínaje nejméně významovým bitem. V závislosti na vstupu integrátoru, se tato hodnota může přičíst sčítačkou (SUM) k hodnotě akumulátoru (ACC). Po dokončení výpočtu mezivýsledku se na vstup přivede významnější bit $f(y_i)$ a současně dojde k posunutí výstupního registru doprava. Celý postup probíhá v cyklech dokud se na vstupu integrátoru neobjeví nejvýznamnější bit $f(y_i)$. Po dokončení operace násobení se hodnota uložená v (ACC), tedy člen Taylorovy řady DY_p uloží do posuvného výstupního registru (SR) a dále se sériově přičte k hodnotě v registru(RV).

Tato varianta má opět menší nároky na zapojení, ale počet cyklů který se musí při výpočtu provést, a tím pádem i čas výpočtu, je dán exponenciálním vztahem.

$$\begin{aligned} t &= \tau_{mult} + \tau_{sum} + \tau_{net} \\ t &= n{\cdot}n{\cdot}\tau_{sum} + n{\cdot}\tau_{sum} + \tau_{net} \\ t &= n^2{\cdot}\tau_{sum} + n{\cdot}\tau_{sum} + \tau_{net} \end{aligned} \tag{3.2}$$

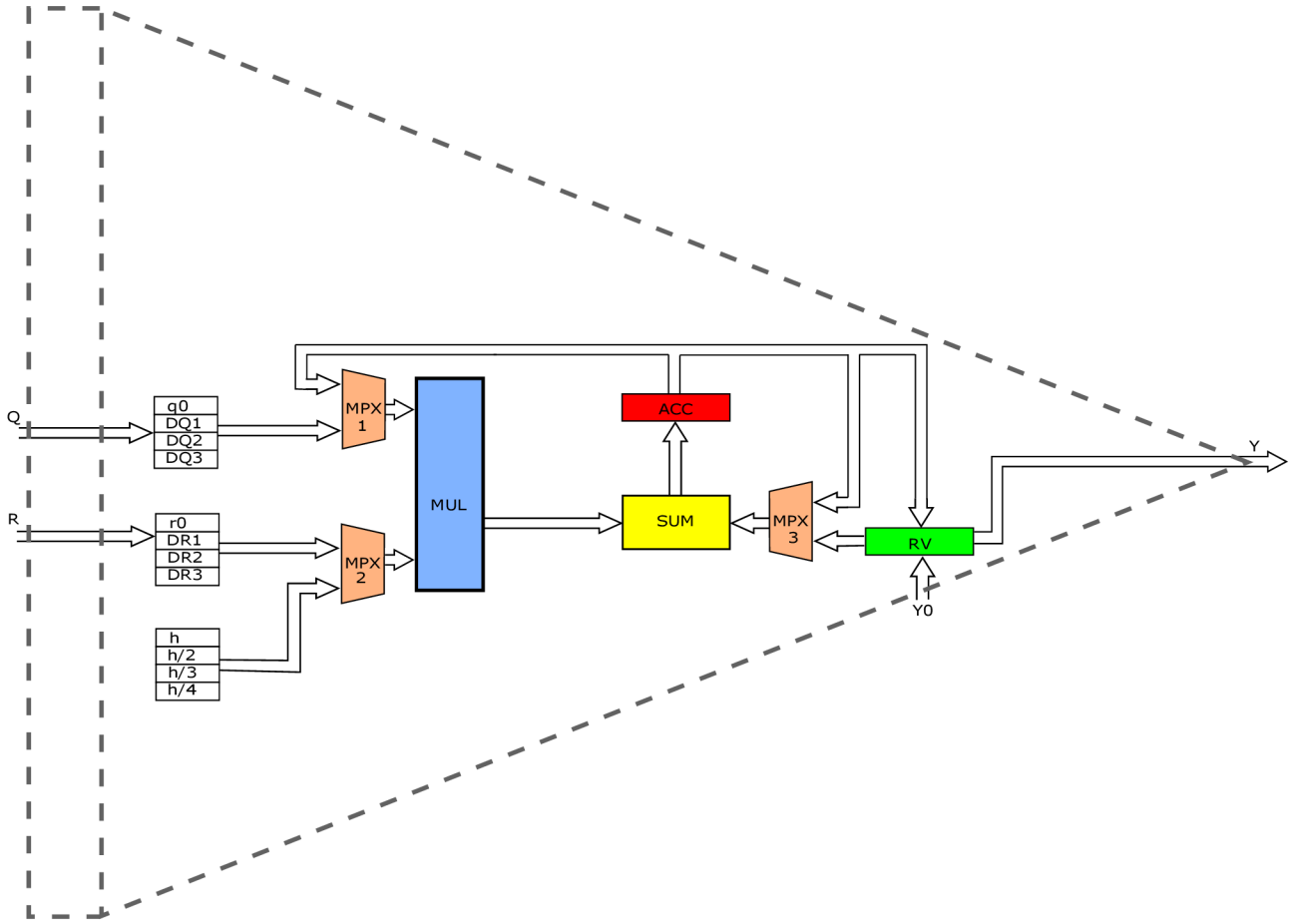
Hodnota τ_{sum} reprezentuje čas sčítání úplné jednobitové sčítačky, n určuje počet bitů potřebných k uložení hodnot násobence a násobitele.

3.4 Paralelně-paralelní dvouvstupý integrátor násobení

Předchozí varianty integrátorů počítali pouze integraci jedné funkce. Následující integrátor provádí násobení a integraci dvou funkcí, kdy na jeho vstupy přicházejí hodnoty členů Taylorovy řady. Tyto členy se počítají v integrátorech, které byly uvedeny výše. Takové propojení se nazývá síť integrátorů. Příklad takové sítě je znázorněn na obrázku (3.7).
Z rovnice (2.46) je patrné, že při výpočtu v jednotlivých krocích potřebuje integrátor všechny předešlé hodnoty DR_i a DQ_i. K ukládání těchto hodnot slouží pole registrů jehož velikost je určena řádem Taylorovy řady kterou počítáme.

V tomto návrhu integrátoru probíhá výpočet prvních čtyř členů Taylorovy řady. Toto řešení bylo zvoleno pouze pro názornost výpočtu. Integrátor lze velmi jednoduše rozšířit přidáním registrů, ve kterých jsou uloženy hodnoty předchozích členů a hodnoty podílů kroku výpočtu.

Zde je popsán význam jednotlivých komponent integrátoru:



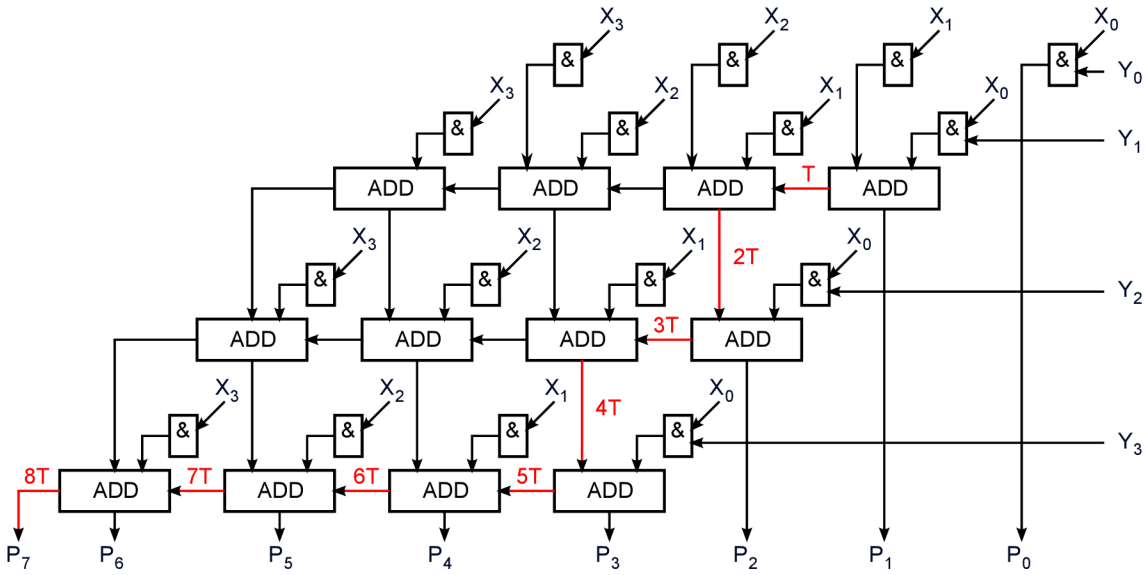
Obrázek 3.5: Paralelně-paralelní integrátor operace násobení

MPX	multiplexor (v závislosti na signálu SEL se na výstup přenese odpovídající vstup)
MUL	paralelní násobička
SUM	paralelní sčítačka
ACC	registr kde se uchovávají mezivýsledky při výpočtu jednoho členu Taylorovy řady
RV	registr výsledku kde se uchovává součet jednotlivých členů Taylorovy řady

Tento integrátor počítá podle rovnice (2.46) první čtyři členy Taylorovi řady. Průběh výpočtu je následující: Předpokládejme že na vstupy integrátoru přichází hodnoty členů Taylorových řad, náležících proměnným P a Q. Tyto hodnoty jsou ukládány v paměti integrátoru.
Na začátku výpočtu se do registru RV uloží hodnota ze vstupu y_0. Registr (ACC) se vynuluje.
Na vstup násobičky (MUL) jsou přivedeny hodnoty r_0 a q_0, výsledek je poté uložen v registru ACC.
Poté se hodnota v registru (ACC) vynásobí s hodnotou h, čímž získáme hodnotu prvního členu Taylorovy řady. Tento člen je sečten s registrem (RV) kde je uložena hodnota y_0. Registr (ACC) je vynulován.
Výpočet následujících členů probíhá následovně:
V registru (ACC) jsou ukládány mezivýsledky při počítání jednotlivých členů. Poté co je aktuální člen spočítán, je přičten k registru výsledku (RV), kde jsou uloženy předchozí členy Taylorovy řady. Poté co je dosaženo požadované přesnosti, je hodnota registru RV přivedena na výstup integrátoru. Hodnota uložená v (RV) zároveň souží jako hodnota Y0, v následujícím kroku výpočtu.

3.5 Paralelní násobička

Na obrázku (3.6) je zobrazena 4-bitová paralelní násobička. Ve schématu je zdůrazněna kritická cesta, která nám zobrazuje šíření přenosu přes celou násobičku a určuje zpoždění celého obvodu. Zpoždění obvodu se zvyšuje s rostoucí délkou operandů.
Jak je z obrázků patrné, délka výsledku je oproti délce operandů dvojnásobná. To v našem řešení není vyhovující a tak se musí spodní polovina bitů oříznout. Tím dojde ke snížení přesnosti výsledků s kterou se musí počítat. Tuto přesnost můžeme zvýšit pouze větší šířkou operandů, s kterými počítáme. To má ale za následek zvýšení doby výpočtu a také nezanedbatelné zvýšení počtu vodičů v celém integrátoru.



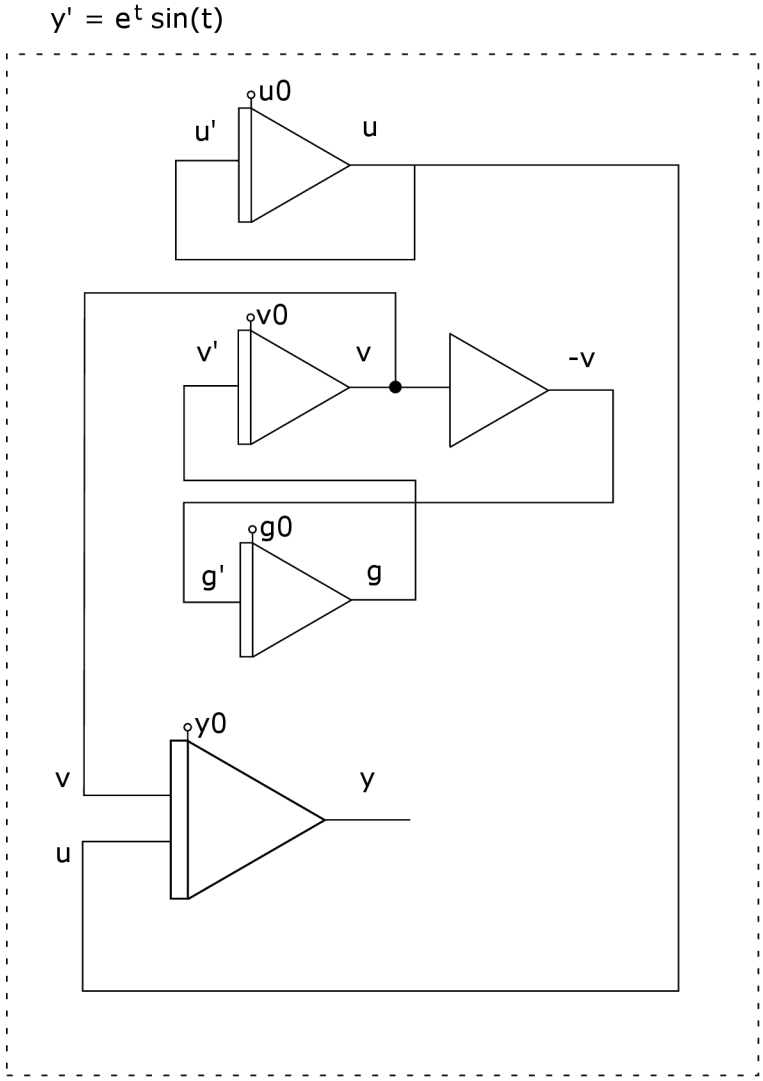
Obrázek 3.6: Symbolické schéma 4-bitové paralelní násobičky

Obrázek a popisky byly převzaty z [9].

3.6 Řešení konkrétní diferenciální rovnice pomocí sítě integrátorů

Jako příklad použití integrátorů, při řešení diferenciální rovnice, je zde uvedeno řešení rovnice z předchozí kapitoly (2.47). Tuto rovnici jsme převedli na soustavu čtyř diferenciálních rovnic (2.56 - 2.59). Každá diferenciální rovnice je reprezentována jedním integrátorem. Tuto konkrétní soustavu je možno překreslit na tři jedno-vstupé integrátory, invertor, který převrací znaménko hodnoty v a jeden dvouvstupý integrátor násobení, jenž bude popsán v následující kapitole.
Celá síť je znázorněna na obrázku 3.7:

Z obrázku (3.7) je patrné, že integrátory počítají proměnné u,v,g které jsou následně použity pro výpočet hodnoty y. Celý výpočet probíhá iteračně, dokud není rozdíl hodnot menší než zadaná odchylka. Poté se výpočet ukončí a hodnotu y považujeme za správný výsledek.



Obrázek 3.7: Síť integrátorů

Nejdůležitějším přínosem takové sítě integrátorů je, že výpočet probíhá ve všech integrátorech paralelně.

3.7 Shrnutí

V této kapitole byl popsán návrh a realizace matematických výpočtů diferenciálních rovnic metodou Taylorovy řady za pomoci jednoduchých hardwarových komponent tzv. integrátorů. Byly popsány jednotlivé varianty integrátorů. Tyto integrátory se dají spojovat do rozsáhlých sítí a tím řešit velmi složité soustavy diferenciálních rovnic. Nejdůležitější vlastností těchto sítí je, že všechny integrátory mohou pracovat paralelně a tím pádem velmi efektivně.

Kapitola 4

Návrh řadiče pro dvouvstupé integrátory násobení

Návrhy integrátorů z předchozí kapitoly obsahovali pouze bloky tvořící aritmeticko-logickou jednotku tzv. ALU. Ke správné funkci této ALU je zapotřebí její řadič.
Řadič je blok, který pomocí signálů, jež jsou rozvedeny do jednotlivých funkčních bloků integrátoru řídí postup výpočtu. Toto řízení zahrnuje přepínání cest jednotlivých multiplexorů, povolování čtení a zápisu registrů a nulování těchto registrů. Řadič pracuje na principu konečného automatu [14], jenž sestává z konečného počtu stavů. Každý tento stav reprezentuje instrukci pro ALU.

4.1 Popis potřebných řídících signálů pro PPI-MUL

Pro návrh řadiče je nejprve nutné analyzovat potřebné signály, které budou řídit funkční bloky. Analýza se týká paralelně-paralelního integrátoru násobení (3.5).

4.1.1 řídící signály pro multiplexory

V integrátoru se vyskytují tři multiplexory. První přepíná mezi polem registrů, kde jsou uloženy hodnoty prvního vstupu a akumulátorem. Druhý přepíná mezi polem registrů, kde jsou uloženy hodnoty druhého vstupu a registry, kde jsou uloženy hodnoty kroku h a jeho podílů. Třetí multiplexor přepíná mezi registrem výsledků a akumulátoru. Všechny tyto multiplexory mají tedy dva vstupy. Tyto vstupy se tedy dají rozeznat za pomoci jednobitových řídících signálů. Tyto signály jsou pojmenovány $(SEL1), (SEL2) a (SEL3)$ a řídí multiplexory v pořadí v němž byli uvedeny výše.

4.1.2 řídící signály pro paralelní sčítačku a násobičku

Paralelní násobička a sčítačka jsou zde implementovány kombinační logikou. Nepotřebují tedy z řadiče žádný řídící signál. Přesto je potřeba jejich činnost synchronizovat s okolními bloky tak, aby byly přijímány pouze platná data.

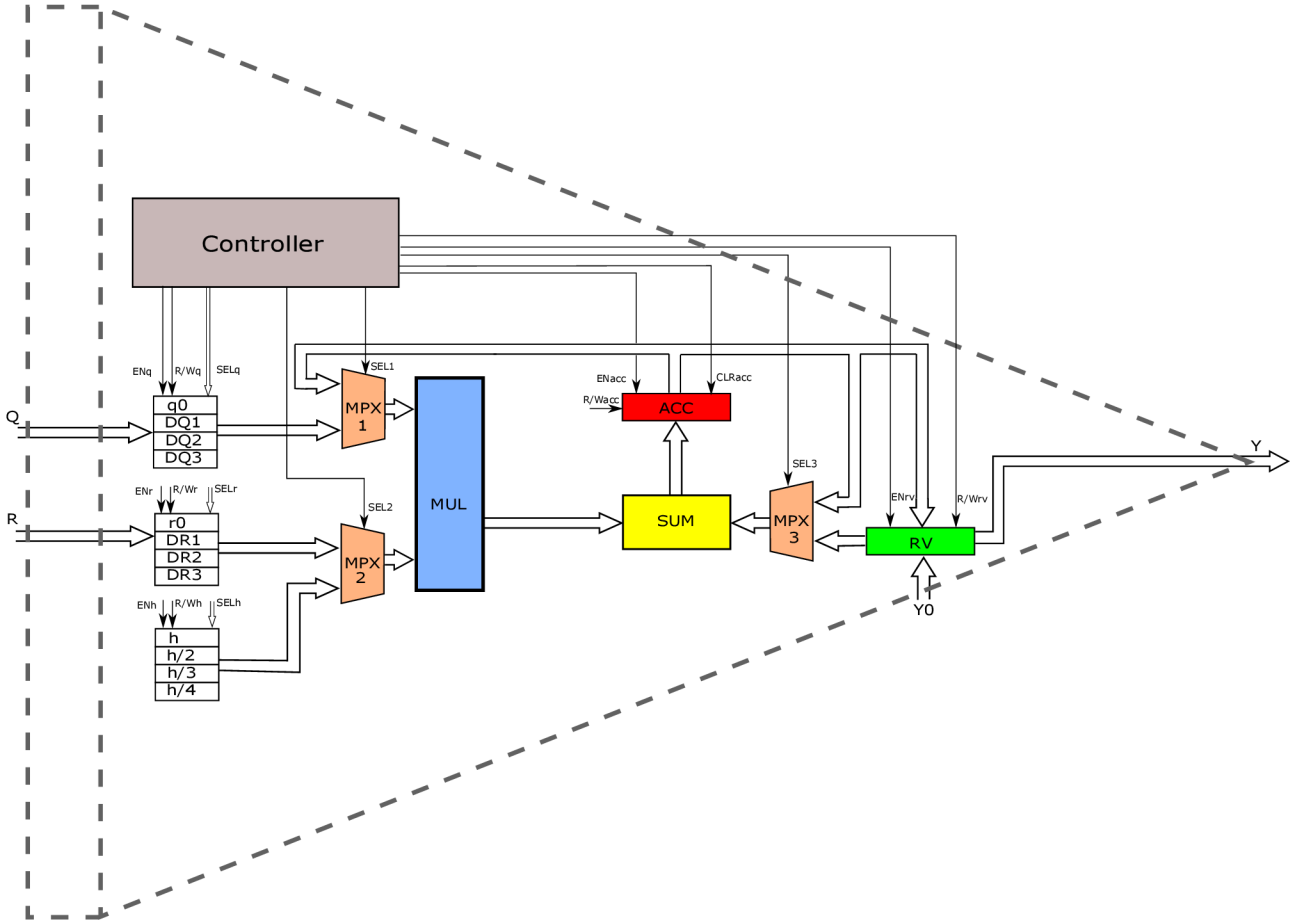
4.1.3 řídící signály pro akumulátor a registr výsledku

Pro akumulátor jsou potřeba tři jednobitové řídící signály. První signál (R/W_{ACC}) vybírá, zda se bude do registru zapisovat nebo se z něj bude číst. Dalším signálem je synchronní

nulování registru (CLR_{ACC}). Třetí signál je povolovací signál (EN_{ACC}), který určuje kdy registr čte, popřípadě odesílá hodnoty.
Registr výsledku potřebuje k řízení pouze signál (R/W_{RV}) a signál (EN_{RV}). V tomto registru totiž k nulování nedochází, hodnota se pouze přepisuje.

4.1.4 řídící signály pro pole registrů

Pole registrů slouží k uchování vstupních hodnot integrátoru a kroku h. Každé pole se skládá ze čtyř registrů. Potřebujeme tedy dvoubitové signály $(SEL_q), (SEL_r), (SEL_h)$, které tyto registry adresují. Dále jsou potřeba signály $(R/W_q), (R/W_r)$, které vybírají zda se bude z registrů číst nebo zapisovat. V poli registrů h není zápis potřeba. Jednobitové signály $(EN_q a EN_r)$ jsou povolovací signály registrů. Signály $(R/W_r, R/W_q, R/W_h)$ jsou povolovací signály registrů.



Obrázek 4.1: PPI integrátor násobení s uvedenými řídícími signály

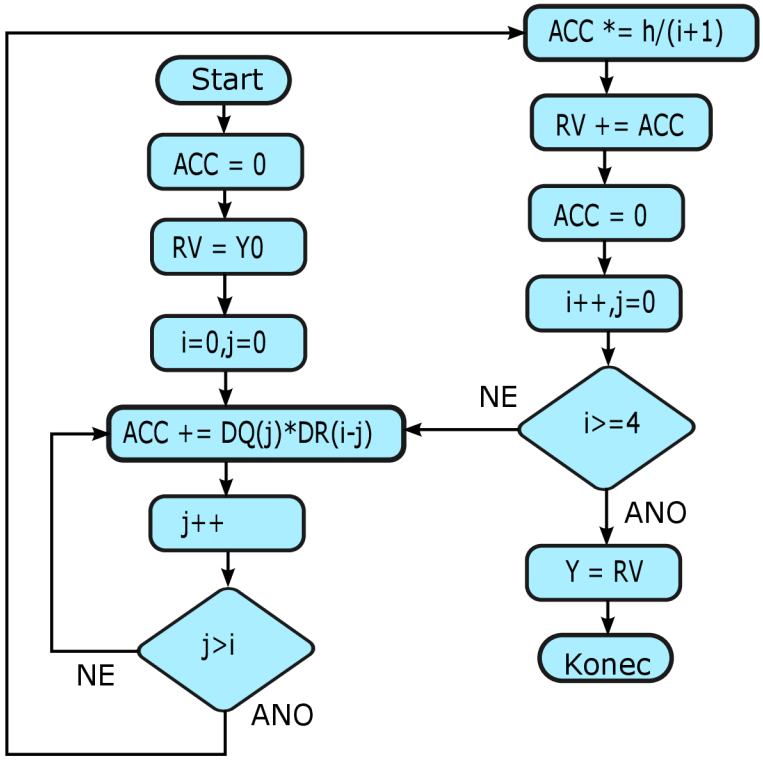
4.2 řadič pro PPI-MUL

4.2.1 Stavy mikroprogramu

1. Do registru RV se uloží počáteční podmínka y_0. Registr ACC se vynuluje.
2. V poli registrů se adresují hodnoty q_0 a r_0. Multiplexory 1 a 2 se přepnou tak aby na vstupu násobičky byly přivedeny hodnoty těchto registrů. Multiplexor 3 se přepne na vstup z ACC. V tomto nastavení se hodnoty q_0 a r_0 vynásobí a je sečtena s registrem ACC
3. Výsledná hodnota z předchozího kroku je uložena do ACC.
4. 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. Toto pole se adresuje na hodnotu h. Toto nastavení zajistí, že násobička vynásobí hodnoty ACC a h a součin se sečte s registrem RV.
5. Výsledná hodnota z předchozího kroku je uložena do ACC.
6. Hodnota z ACC se uloží v RV. Tímto vznikne v RV hodnota $y_0 + DY_1$.
7. ACC se vynuluje.
8. Multiplexory jsou nastaveny posle stavu (2). Adresují se hodnoty DQ_1 a r_0. Ty jsou vynásobeny a přičteny do ACC.
9. Výsledná hodnota z předchozího kroku je uložena do ACC.
10. Multiplexory jsou nastaveny posle stavu (2). Adresují se hodnoty q_0 a DR_1. Ty jsou vynásobeny a sečteny s ACC.
11. Výsledná hodnota z předchozího kroku je uložena do ACC.
12. Multiplexory jsou nastaveny posle stavu (3). Adresuje se hodnota $h/2$, která je vynásobena s obsahem ACC a přičtena k RV.
13. Výsledná hodnota z předchozího kroku je uložena do ACC.
14. Hodnota z ACC se uloží v RV. Tímto vznikne v RV hodnota $y_0 + DY_1 + DY_2$.
15. ACC se vynuluje.
16. Multiplexory jsou nastaveny posle stavu (2). Adresují se hodnoty DQ_2 a r_0. Ty jsou vynásobeny a přičteny do ACC.
17. Výsledná hodnota z předchozího kroku je uložena do ACC.
18. Multiplexory jsou nastaveny posle stavu (2). Adresují se hodnoty DQ_1 a DR_1. Ty jsou vynásobeny a sečteny s ACC.
19. Výsledná hodnota z předchozího kroku je uložena do ACC.
20. Multiplexory jsou nastaveny posle stavu (2). Adresují se hodnoty q_0 a DR_2. Ty jsou vynásobeny a sečteny s ACC.

21. Výsledná hodnota z předchozího kroku je uložena do ACC.
22. Multiplexory jsou nastaveny posle stavu (3). Adresuje se hodnota $h/3$, která je vynásobena s obsahem ACC a přičtena k RV.
23. Výsledná hodnota z předchozího kroku je uložena do ACC.
24. Hodnota z ACC se uloží v RV. Tímto vznikne v RV hodnota $y_0 + DY_1 + DY_2 + DY_3$.
25. ACC se vynuluje.
26. Multiplexory jsou nastaveny posle stavu (2). Adresují se hodnoty DQ_3 a r_0. Ty jsou vynásobeny a přičteny do ACC.
27. Výsledná hodnota z předchozího kroku je uložena do ACC.
28. Multiplexory jsou nastaveny posle stavu (2). Adresují se hodnoty DQ_2 a DR_1. Ty jsou vynásobeny a sečteny s ACC.
29. Výsledná hodnota z předchozího kroku je uložena do ACC.
30. Multiplexory jsou nastaveny posle stavu (2). Adresují se hodnoty DQ_1 a DR_2. Ty jsou vynásobeny a sečteny s ACC.
31. Výsledná hodnota z předchozího kroku je uložena do ACC.
32. Multiplexory jsou nastaveny posle stavu (2). Adresují se hodnoty q_0 a DR_3. Ty jsou vynásobeny a sečteny s ACC.
33. Výsledná hodnota z předchozího kroku je uložena do ACC.
34. Multiplexory jsou nastaveny posle stavu (3). Adresuje se hodnota $h/4$, která je vynásobena s obsahem ACC a přičtena k RV.
35. Výsledná hodnota z předchozího kroku je uložena do ACC.
36. Hodnota z ACC se uloží v RV. Tímto vznikne v RV hodnota $y_0 + DY_1 + DY_2 + DY_3$. Tímto vznikne v RV hodnota $y_0 + DY_1 + DY_2 + DY_3 + DY_4$.

Průběh výpočtu jednoho kroku je zjednodušeně naznačen na vývojovém diagramu (4.2).



Obrázek 4.2: Vývojový diagram výpočtu kroku paralelně-paralelního integrátoru

V diagramu jsou použity pomocné proměnné i a j, kde i reprezentuje kolikátý člen taylorovy řady se počítá a j reprezentuje kolikátý se počítá součin v tomto členu. Z toho tedy vyplývá, že jdou potřeba čítač členu a čítač součinu.

4.2.2 Mikroprogram řadiče pro PPI-MUL

V tabulce (4.1) jsou uvedeny signály, jež definují běh výpočtu integrátoru integrátoru. Nejsou zde uvedeny povolovací signály u registrů a to z důvodu omezení rozsahu tabulky a tedy zlepšení její přehlednosti. Vždy, když je tedy uveden signál R/W určitého bloku, znamená to že je zároveň aktivní signál EN tohoto bloku a naopak. Hodnota X v tabulce znamená, že hodnota signálu není podstatná.

Tabulka 4.1: Tabulka hodnot signálů v jednotlivých stavech

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

Kapitola 5

Implementace integrátoru v provedení FPGA

V této kapitole bude popsána inplementace integrátoru a jeho komponet v jazyce VHDL , tak aby bylo možné nahrát tuto implementaci na hradlových polích FPGA.

5.1 Jazyk VHDL

Jazyk VHDL (zkratka znamená Very High Speed Integrated Circuit Hardware Description Language, tedy jazyk pro popis hardware velmi rychlých integrovaných obvodů) je jazyk, který slouží pro popis hardware. Používá se pro návrh a následnou simulaci digitálních integrovaných obvodů, těmi jsou například programovatelná hradlová pole FPGA.
Hlavní odlišností od klasických programovacích jazyků je možnost modelovat souběžné děje pomocí paralelně pracujících komponent.
Jazyk VHDL popisuje jednotlivé číslicové zařízení pomocí tzv. komponent. Ty mohou být popsány jako entita, kde se definuje rozhraní, tedy vstupy a výstupy komponenty. Dále je definována architektura, která určuje funkci a chování komponenty.

Architekturu je možné popsat třemi různými styly:

- Strukturní popis
  Popisuje architekturu dané entity pomocí komponent a jejich propojení. Lze propojit hradla i komplexní či abstraktní komponenty.

- Behaviorální popis
  Popisuje architekturu dané entity pomocí jejího chování. Popisuje tedy jak se změní výstupy v závislosti na změnách vstupních signálů. Základem behaviorálního popisu je proces. Systém je složen z těchto procesů a ty spolu komunikují pomocí signálů.

- Dataflow popis
  Tento popis modeluje architekturu na základě znalosti toku informací. Popisuje tedy kombinační logiku pomocí souběžně pracujících příkazů. Těmito příkazy jsou například přiřazení a podmíněné přiřazení.

Více o jazyku lze nalézt například v [5], odkud pochází i zde uvedené informace.

5.2 Programovatelné hradlové pole FPGA

Programovatelná hrdlová pole FPGA (Field Programmeble Gate Array) jsou speciální integrované číslicové obvody skládající se z konfigurovatelné matice spojů, která propojuje rozně složité programovatelné bloky. Obvod tedy může bý nakonfigurován k různým účelům.
Rozeznáváme dva typy FPGA podle uložení konfigurace. FPGA s volatilní konfigurací kde se konfigurační informace ukládají do paměťových buněk SRAM a je u nich možná změna konfigurace i za běhu systému. Druhou variantou je FPGA s nevolatilní konfigurací, kde jsou konfigurační informace uloženy ve flash paměti typu EEPROM.
Díky své programovatelnosti a snadnému návrhu a klesajícím cenám jsou dnes FPGA obvody poměrně rozšířené a nacházejí uplatnění v široké škále aplikací. Typycké použití je v oblasti menších sérií navrhovaných zařízení, kdy se nevyplatí návrh zákaznického integrovaného obvodu. Informace byly čerpány z [3], kde je možné nalézt podrobnější popis.

5.3 Implementace Integrátoru

Implementaci paralelně paralelního integrátoru je možné nalézt v přiložených zdrojových souborech. Zde budou shrnuty nejdůležitější vlastnosti implementace.

- Integrátor provádí výpočet s datovou délkou 32 bitů
- První bit je znaménkový
- Druhý bit je jediný bit před desetinou čárkou, musíme tedy volit vhodné vstupy tak aby výsledek byl menší než hodnota dva
- Vstupní hodnoty implementace jsou nastaveny tak, že integrátor počítá jeden krok, o velikosti 0.1, rovnice $e^t * sin(t)$
- Registr ACC není nulován. Nulování probíhá na multiplexoru 3 za pomocí blokovacího signálu. Tím bylo snížen počet stavů a tím pádem urychlen výpočet.

Implementace je tvořena těmito zdrojovými soubory:

- acc.vhd - reprezentuje registry ACC a RV, tyto registry jsou implementovány jako synchronní, provádějí tedy akci vždy na nástupnou hranu hodinového signálu, do registru se zapisuje pokud je hodnota signálu RW rovna 1.
- blockofregister.vhd - reprezentuje bloky registrů kde jsou uloženy vstupní hodnoty. Jednotlivé registry jsou adresovány signálem ARV, do registrů se zapisuje pokud je hodnota signálu RW rovna 1. Tento blok je opět synchronní a reaguje na nástupnou hranu CLK.
- mpx.vhd - reprezentuje dvouvstupý multiplexor. Signál ARV prepíná jednolivé vstupy.
- mpxblok.vhd - rozšiřuje multiplexor o funkci nulování. Pokud je hodnota řídícího signálu BL rovna 1, výstup z integrátoru je nulový.

- mul.vhd - reprezentuje paralelní kombinační násobičku. První bity vstupních signálů jsou pomocí logické funkce xor převedeny na výstup. Toto reprezentuje znaménka vstupních a výstupního signálu. Ostatní bity jsou vynásobeny. Vzhledem k tomu, že výstupem násobení je dvojnásobná datová délka, je spodní polovina bitů ořezána.
- sum.vhd - reprezentuje paralelní kombinační sčítačku. Stejně jako v případě násobičky i zde je potřeba zpracovat znaménko na výstup. Zde je situace složitější. Pokud jsou znaménka obou operandů stejná, znaménko výstupu je rovno znaménkům operandů a ostatní bity se sečtou. Pokud jsou znaménka rozdílná, je potřeba odečíst absolutní hodnoty menšího operandu od většího a na výstup dát znaménko většího operandu.
- controller.vhd - zde je implementován řadič. Řadič pracuje na principu konečného automatu a výstupními signály řídí ostatní komponenty.
- ppi_mul.vhd - spojuje jednotlivé komponenty do jednoho celku pomocí mapování jednotlivých signálů na odpovídající vstupy a výstupy.
- ppi_mul_tb.vhd - Testbench soubor, kde se simulují vstupy komponenty. Slouží pro testování v simulátoru.

Některé zdrojové soubory jsou modifikací souborů z dizertační práce [9] a diplomové práce [12], které popisovali a kde byl implementován odlišný typ integrátoru.

5.4 Simulačí nástroj ModelSim SE 10.4c

Ke kompilaci byl použit simulační program ModelSim SE 10.4c. Tento program umožňuje kompilaci VHDL kódu. Po úspěšné kompilaci je možné přistoupit k simulaci našeho integrátoru. Program ModelSim poskytuje prostředí pro simulaci procesů a signálů tak jak by probíhali přímo v FPGA.
V této simulaci byla ověřena funkce implementovaného integrátoru. Ověřování probíhalo vždy přivedením vstupních dat na vstup simulovaného integrátoru a ověřováním hodnot na sběrnicích mezi jednotlivými bloky a také hodnoty v registrech. Po dokončení výpočtu pak byla porovnávána hodnota registru výsledku s hodnotou, kterou vypočítal simulační program TKSL [2]. Pro všechny testované hodnoty byly výsledky do míry přesnosti, kterou určuje použití 32 bitových registrů, shodné.

Kapitola 6

Simulátor paralelně-paralelního integrátoru

V této kapitole je popsán návrh a realizace simulátoru paralelně-paralelního integrátoru násobení, který je popsán v předchozích kapitolách. Hlavním účelem tohoto simulátoru je názorná prezentace vnitřních výpočtů a průběhu řídících signálů, které se v integrátoru vyskytují. Tento simulátor tedy může být využit pro výukové účely. Hlavní důraz je tedy kladen na srozumitelné zobrazení a jednoduché ovládání. Vzhledem k těmto požadavkům bylo přistoupeno k omezení simulace pouze na vnitřní děje. Simulovaný integrátor tedy nepřímá vstupy. Všechny potřebné vstupy jsou již na začátku simulace zadány uživatelem a uloženy v registrech do kterých by se v průběhu výpočtu reálného integrátoru ukládaly postupně až s příchodem daného signálu. Dále bylo při zobrazování hodnot zvoleno použití dekadické soustavy, která je pro člověka nejlépe srozumitelná.
Simulátor neprovádí pouze před-připravenou posloupnost akcí. Na pozadí probíhá skutečný přesný výpočet. Počítá se na principu Taylorovy řady čtvrtého řádu. Výpočet je počítán na základě hodnot zadaných uživatelem. V grafickém prostředí je tento výsledek zaokrouhlen na osm desetinných míst.

6.1 Grafický návrh simulátoru

Vzhledem k tomu, že simulátor může být použit pro výukové účely vychází grafická koncepce a rozložení ovládacích prvků ze simulátoru sériově-paralelního integrátoru uvedeného v [12]. K tomuto bylo přistoupeno z toho důvodu, aby nedocházelo ke zmatení uživatelů při spouštění různých simulačních programů.

6.2 Realizace

6.2.1 Programovací jazyk Java

K implementaci simulátoru byl zvolen vyšší objektově orientovaný jazyk Java. Jazyk Java je založen na syntaxi jazyků C a C++, nepodporuje ale například ukazatele. Jazyk používá tzv. garbage-collector, který se stará o správu využívané paměti. Ten automaticky vyhledává již nepoužívané části paměti a uvolňuje je pro další použití. Jazyk Java je specifický tím, že místo skutečného strojového kódu se vytváří tzv. bytekód. Tento bytekód je nezá-

vislý na architektuře počítače. Program je tak přenositelný a může pracovat na jakémkoliv zařízení, na kterém je nainstalován interpret Javy tzv. Java Virtual Machine. Všechny potřebné nástroje, které jsou pro spuštění programu implementovaného v Javě zapotřebí jsou zahrnuty v balíku *Java Development kit (JDK)*. Ten je dostupný zdarma na [1]. Existují též mikroprocesory, které dokáží spustit Javu hardwarově namísto softwarové emulace Java Virtual Machine. ARM procesory mohou mít hardwarovou podporu pro spuštění binárního kódu Javy.
Jazyk Java vyvinula firma Sun Microsystem, kterou ale v roce 2010 koupila firma Oracle[1], která tento jazyk aktuálně spravuje. Více o jazyku Java lze naleznout v [7], odkud byly čerpány informace.

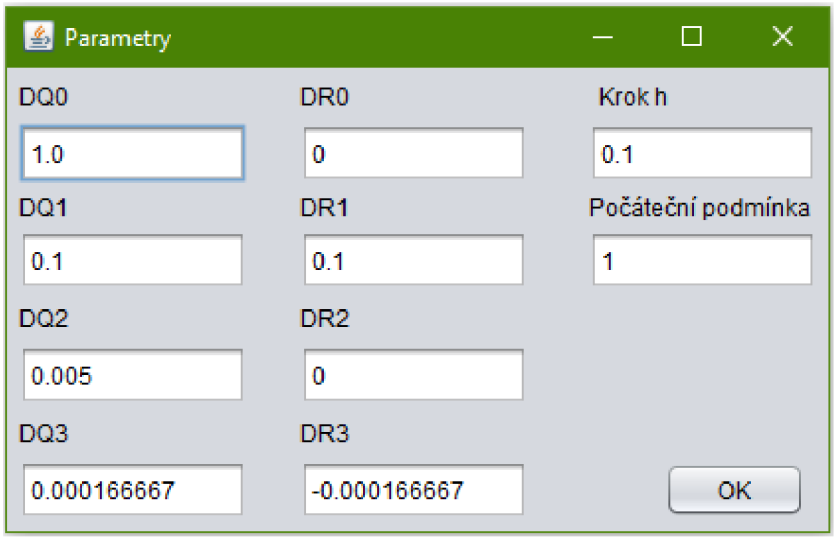
6.2.2 Grafická implementace simulátoru

K implementaci grafického rozhraní byla použita knihovna Swing. Knihovna Swing poskytuje aplikační rozhraní pro tvorbu a obsluhu klasického grafického uživatelského rozhraní. S její pomocí je možno vytvářet okna, dialogy, tlačítka, rámečky a další grafické prvky.

6.2.3 Zadávání počátečních hodnot

Při spuštění programu je otevřeno hlavní okno samotného simulátoru a dále okno, kde se dají přenastavit parametry programu (6.1). Těmito parametry jsou čtyři hodnoty pro každý vstup, které reprezentují první čtyři členy Taylorových řad vstupních funkcí a dále počáteční podmínku integrátoru a krok výpočtu h. Tyto parametry jsou při spuštění simulace nastaveny na počáteční hodnoty:

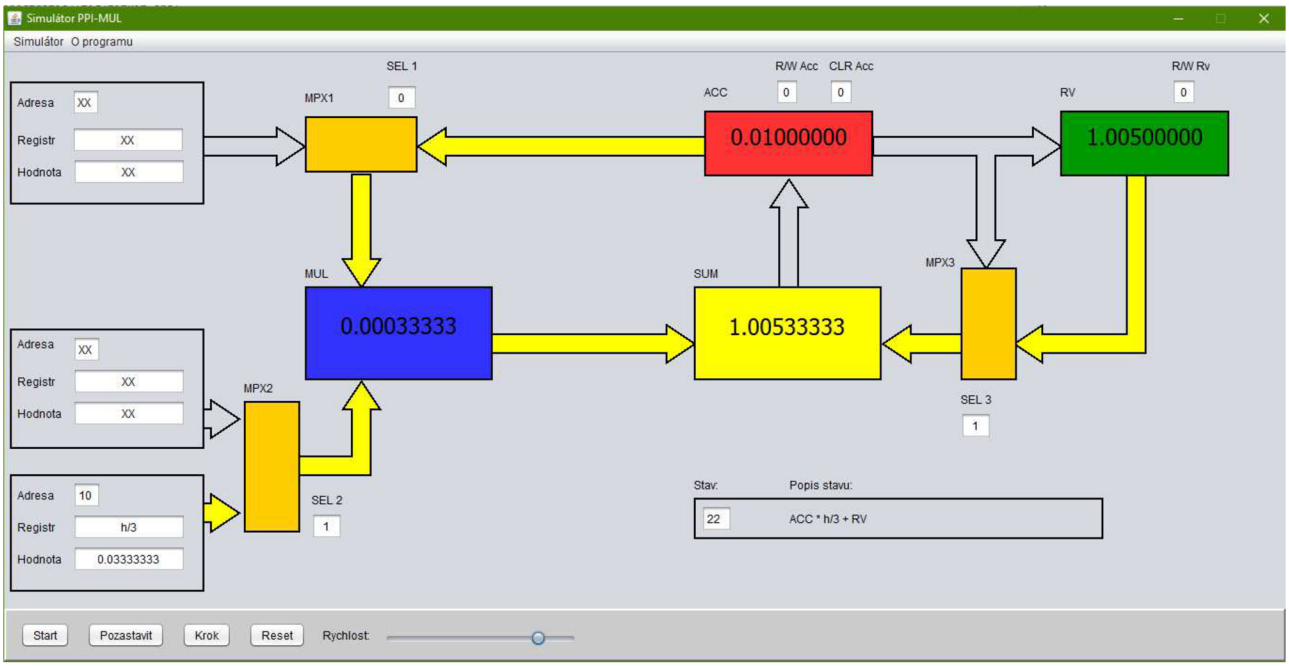
DQ0 = 1.0, DQ1 = 0.1, DQ2 = 0.005, DQ3 = 0.000166667,
DR0 = 0.0, DR1 = 0.1, DR2 = 0.0 DR3 = -0.000166667
h = 0.1 Počáteční podmínka = 1.0



Obrázek 6.1: Okno pro zadání parametrů simulace

Tyto hodnoty reprezentují výpočet rovnice $y' = e^t \cdot sin(t) \quad y(0) = 1$

Hodnoty lze libovolně měnit, pouze je potřeba dodržet formát čísla s desetinnou tečkou. Pokud je hodnota zadána nekorektně, program nedovolí hodnotu potvrdit dokud není chyba opravena. Pokud chce uživatel změnit během výpočtu hodnoty, je možné tak učinit výběrem možnosti Zadat parametry v nabídce simulátor na horním panelu aplikace. Tato možnost resetuje simulaci na počáteční stav.



Obrázek 6.2: Okno simulátoru při probíhající simulaci

6.2.4 Popis programu a práce s ním

Po zadání počátečních hodnot je možné přistoupit k hlavnímu oknu simulace (6.2). V menu na horním okraji aplikace je možné vybrat možnost O programu, která otevře okno se stručným popisem aplikace. Nabídka Simulace otevře možnost Zadat jiné argumenty nebo ukončit program.
Jedinými ovládacími prvky simulace jsou čtyři tlačítka a posuvník, který ovlivňuje rychlost simulace. Posuvník má sedm hodnot, které reprezentují dobu mezi jednotlivými kroky simulace od 500 do 3500 milisekund, v intervalech po 500 milisekundách

Start	zapíná automaticky probíhající simulaci
Pozastavit	pozastaví probíhající simulaci
Krok	provede simulaci jednoho kroku
Reset	navrátí simulátor do výchozího stavu

Po levé straně se nacházejí 3 bloky, které reprezentují registry ve kterých jsou uloženy vstupy integrátoru a podíly kroku h. V těchto blocích je jako Adresa zobrazován signál, který adresuje požadovaný registr. Dále je zobrazen název adresovaného registru a hod-

nota, která je v něm uložena.
Multiplexory jsou reprezentovány oranžovými bloky *MPX1 - MPX3* u těchto bloků jsou uvedeny signály *SEl1 - SEL3*, které přepínají jejich vstupy.
Násobička je reprezentována modrým blokem *MUL*.
Sčítačka je reprezentována žlutým blokem *SUM*
Červený blok *ACC* reprezentuje registr akumulátoru. Tento akumulátor řídí dva signály. Prvním je signál CLR_{acc}, jež při nastavení maže obsah akumulátoru. Druhým signálem je signál R/W_{Acc}, který určuje zda se z registru hodnota čte nebo se do něj zapisuje.
Hodnoty povolovacích signálů registrů nejsou uvedeny. Ty jsou aktivní v okamžiku kdy jsou aktivní vodiče vedoucí do nebo z registru. Zelený blok *RV* reprezentuje Registr výsledku. V tomto registru je po provedení simulace uložen výsledek. Registr je řízen signálem R/W_{RV}, který má stejnou funkci jako u registru akumulátoru. Paralelní vodiče jsou reprezentované šipkami. Žlutá barva znázorňuje, že jsou vodiče aktivní a přenáší hodnoty.

Samotná výpočet se skládá z 32 stavů, z nichž každý reprezentuje určitou operaci integrátoru a s ní spojenou změnu řídících signálů, tak aby mohly jednotlivé komponenty správně pracovat. Každý z těchto 32 stavů se dále pro potřeby simulace dělí na různý počet stavů simulace. Těch je dohromady 50. Po dokončení průchodu těchto stavů je v registru RV výsledek integrace jednoho kroku. Po opětovném spuštění začíná simulace od začátku.

6.2.5 Nepoužité koncepty

Během tvorby simulátoru byly vymyšleny koncepty, které nakonec z určitých důvodů nebyly použity. Jedním takovým konceptem je například již zmíněné použití vstupů simulátoru. Toto nebylo zahrnuto do konečné verze simulátoru kvůli tomu, že na samotný výpočet nemá vliv a jenom by vedlo k vyšší složitosti a tedy i menší přehlednosti simulátoru.
Dále bylo zvažováno použití binární soustavy při zobrazování hodnot. Ve finální verzi byla zvolena dekadická soustava z důvodu lepší čitelnosti.
Posledním nepoužitým konceptem bylo znázornění vodičů, které vedou řídící signály k jednotlivým blokům. Ty nakonec nebyly použity, kvůli vyššímu počtu různých čar, které snižovali přehlednost simulace.

6.2.6 Požadavky ke spuštění

Pro spuštění a správnou funkci aplikace vyžaduje instalaci aktuální verze JRE (Java Runtime Environment), která je dostupná na [1].

Kapitola 7

Závěr

V této práci proběhlo základní seznámení s řešením obecných diferenciálních rovnic za použití numerické integrace. Byly popsány nejčastěji používané metody pro numerickou integraci a byly shrnuty jejich nejdůležitější vlastnosti. Podrobně bylo popsáno řešení metodou Taylorovy řady, jejíž princip byl použit při návrhu dále zmíněných integrátorů.
Další částí této práce bylo uvedení jednotlivých variant integrátorů a popis jejich funkce. Těmito variantami jsou sériově-sériový integrátor, paralelně-paralelní integrátor, a sériově sériový integrátor. Tyto integrátory se však dají použít jen k řešení jednoduchých diferenciálních rovnic.
Hlavní částí této práce byl návrh paralelně-paralelního integrátoru s dvěma vstupy, který tyto vstupy násobí. Součástí návrhu byl také návrh jeho řadiče a řídících signálů, které řídí jednotlivé funkční bloky integrátoru.
Tento integrátor byl poté implementován v jazyce VHDL tak, aby mohl být použit v prostředí FPGA. Tato implementace byla otestována simulačním nástrojem ModelSim.
K prezentaci funkce tohoto integrátoru byl také vytvořena aplikace simulátoru, která názorně ukazuje jeho funkci. Vzhledem k tomu, že byla aplikace navržena tak, aby názorně a jednoduše prezentovala princip integrátoru a algoritmu výpočtu Taylorovy řady, může být použita k výukovým účelům v některých hardwarově založených předmětech na naší fakultě.
Možností dalšího výzkumu je zjištění časové a prostorové náročnosti implementace a její následná optimalizace. Dále také návrh dalších typů násobících integrátorů, jejichž parametry se liší. V neposlední řadě je také potřeba zkoumat použití většího množství těchto integrátorů, s mnohem větší délkou slova, ve velkých hradlových polích. Těmito integrátory by bylo možné řešit rozsáhlé soustavy rovnic s vysokou přesností.

Literatura

[1] Web: Oracle, Java [online].
URL https://www.oracle.com/java/index.html

[2] Web: TKSL/386 [online].
URL http://www.fit.vutbr.cz/~kunovsky/TKSL/tksl386.html.cs

[3] ŠŤASTNÝ, Jakub: *FPGA prakticky: realizace číslicových systémů pro programovatelná hradlová pole*. Praha: BEN - technická literatura, 2010, ISBN 978-80-7300-261-9.

[4] ČAMBOR, Michal: *Paralelní řešení parciálních diferenciálních rovnic.* Diplomová práce, FIT VUT v Brně, 2011.

[5] ARMSTRONG, James R a F GRAY: *Structured logic design with VHDL.* Englewood Cliffs, N.J.: PTR Prentice Hall, 1993, ISBN 0-13-855206-1.

[6] DIBLÍK, Josef and Oto PŘIBYL: *Obyčejné diferenciální rovnice*. Vysoké učení technické v Brně, akademické nakladatelství CERM, 2004, ISBN 80-214-2795-7.

[7] HEROUT, Pavel: *Učebnice jazyka Java. 5. rozš. vyd.* České Budějovice: Kopp nakladatelství, 2011, ISBN 978-80-7232-398-2.

[8] KRAUS, Michal: *Elementární procesor specializovaného paralelního systému.* Diplomová práce, FIT VUT v Brně, 2006.

[9] KRAUS, Michal: *Paralelní výpočetní architektury založené na numerické integraci.* Dizertační práce, FIT VUT v Brně, 2013.

[10] KUNOVSKÝ, Jiří: *Modern Taylor series method.* FEI-VUT Brno„ 1994.

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

[12] OPALKA, Jan: *Automaticke řízení výpočtu.* Diplomová práce, FIT VUT v Brně, 2014.

[13] PERINGER, P.: Studijní opora předmětu IMS. Technická zpráva, FIT VUT v Brně, 2007.

[14] STRAUBING, Howard: *Finite automata, formal logic, and circuit complexity.* Boston: Birkhäuser, 1994, Progress in theoretical computer science, ISBN 3-7643-3719-2.

Přílohy

Seznam příloh

Příloha A

Obsah CD

Přiložené CD obsahuje:

- Zdrojové soubory této práce ve formátu LaTeX.
- Text práce ve formátu PDF.
- Zdrojové soubory paralelně-paralelního integrátoru násobení v jazyce VHDL.
- Zdrojové soubory aplikace simulátoru paralelně-paralelního integrátoru násobení v jazyce Java