



**VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ**

BRNO UNIVERSITY OF TECHNOLOGY

**FAKULTA INFORMAČNÍCH TECHNOLOGIÍ**

FACULTY OF INFORMATION TECHNOLOGY

**ÚSTAV INFORMAČNÍCH SYSTÉMŮ**

DEPARTMENT OF INFORMATION SYSTEMS

**HARDVÉROVÁ REALIZÁCIA NUMERICKÉHO  
INTEGRÁTORA S METÓDOU VYŠŠIEHO RÁDU**

HARDWARE REALIZATION OF HIGHER ORDER NUMERICAL INTEGRATOR

**SEMESTRÁLNÍ PROJEKT**

TERM PROJECT

**AUTOR PRÁCE**

AUTHOR

**FRANTIŠEK MATEČNÝ**

**VEDOUCÍ PRÁCE**

SUPERVISOR

**Ing. VÁCLAV ŠÁTEK, Ph.D.**

**BRNO 2018**

## Abstrakt

Práca popisuje numerickú integráciu a riešenie diferenciálnych rovníc pomocou metódy Taylorovej rady v rôznych typoch integrátorov. Ďalej je popísaná aritmetika pevnej a pohyblivej rádovej čiarky. Následne sú predstavené návrhy a spôsob výpočtu paralelných integrátorov s operáciou násobenia a delania v prevedení pevnej a pohyblivej rádovej čiarky. Súčasťou práce je taktiež implementácia sústavy integrátorov v FPGA

## Abstract

This work deals with numerical integration and solution of differential equations by the Taylor series in many types of integrators. Next is described floating point and fixed point arithmetic. Subsequently are presented designs and method of calculation of parallel multiplication and division integrators in floating point and fixed point arithmetic. TODO - opraviť!!

## Kľúčové slová

diferenciálna rovnica, numerická integrácia, Taylorova rada, pevná rádová čiarka, pohyblivá rádová čiarka, integrátor, FPGA

## Keywords

differential equation, numeric integration, Taylor series, fixed point, floating point, integrator, FPGA

## Citácia

MATEČNÝ, František. *Hardvérová realizácia numerického integrátora s metódou vyššieho rádu*. Brno, 2018. Semestrální projekt. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Václav Šátek, Ph.D.

# Hardvérová realizácia numerického integrátora s metódou vyššieho rádu

## Prehlásenie

Prehlasujem, že som túto diplomovú prácu vypracoval samostatne pod vedením pána Ing. Václava Šátka, Ph.D. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....  
František Matečný  
20. mája 2018

## Podakovanie

Chcel by som sa poďakovať svojmu školiteľovi Ing. Václavovi Šátkovi, Ph.D., 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

|          |  |           |
|----------|--|-----------|
| <b>1</b> | <b>Úvod</b>  | <b>5</b>  |
| <b>2</b> | <b>Numerická integrácia</b>  | <b>6</b>  |
| 2.1      | Taylorova rada . . . . .   | 6         |
| 2.2      | Eulerova metóda . . . . .  | 7         |
| 2.3      | Runge-Kutta . . . . .  | 7         |
| 2.3.1    | Runge-Kutta 2. rádu . . . . .  | 7         |
| 2.3.2    | Runge-Kutta 4. rádu . . . . .  | 8         |
| <b>3</b> | <b>Riešenie diferenciálnych rovníc Taylorovou radou</b>              | <b>9</b>  |
| 3.1      | Riešenie diferenciálnej rovnice s operáciou násobenia . . . . .      | 10        |
| 3.2      | Riešenie diferenciálnej rovnice s operáciou delenia . . . . .        | 11        |
| <b>4</b> | <b>Reprezentácia operandov</b>                                       | <b>13</b> |
| 4.1      | Pevná rádová čiarka . . . . .  | 13        |
| 4.2      | Pohyblivá rádová čiarka . . . . .                                    | 14        |
| 4.2.1    | Súčet a rozdiel . . . . .  | 16        |
| 4.2.2    | Násobenie a delenie . . . . .  | 17        |
| <b>5</b> | <b>Numerické integrátory</b>   | <b>18</b> |
| 5.1      | Návrh násobiaceho integrátora v pevnej rádovej čiarke . . . . .      | 19        |
| 5.2      | Návrh deliaceho integrátora v pevnej rádovej čiarke . . . . .        | 20        |
| 5.3      | Návrh jednovstupového integrátora . . . . .                          | 22        |
| 5.4      | Návrh komponentu na spracovanie exponentov . . . . .                 | 23        |
| 5.5      | Návrh násobiacieho integrátora v pohyblivej rádovej čiarke . . . . . | 25        |
| 5.6      | Deliaci integrátor v pohyblivej rádovej čiarke . . . . .             | 26        |
| 5.7      | Sústava diferenciálnych rovníc . . . . .                             | 27        |
| <b>6</b> | <b>Popísanie integrátorov vo VHDL</b>                                | <b>29</b> |
| 6.1      | Integrátory v pevnej rádovej čiarke . . . . .                        | 30        |
| 6.2      | Integrátory v pohyblivej rádovej čiarke . . . . .                    | 36        |
| 6.3      | Sústava integrátorov . . . . .                                       | 37        |
| <b>7</b> | <b>Analýza</b>   | <b>40</b> |
| <b>8</b> | <b>Záver</b>   | <b>44</b> |
|          | <b>Literatúra</b>  | <b>45</b> |



# Zoznam obrázkov

|     |  |    |
|-----|--|----|
| 4.1 | Rôzne formáty fixed point aritmetiky [2] . . . . .                             | 13 |
| 4.2 | IEEE 754 formát s jednoduchou presnosťou . . . . .                             | 14 |
| 4.3 | IEEE 754 formát s dvojitou presnosťou . . . . .                                | 14 |
| 5.1 | Schéma jednovstupového a dvojevstupového integrátora . . . . .                 | 18 |
| 5.2 | Paralelno-paralelný násobiaci integrátor [14]. . . . .                         | 19 |
| 5.3 | Paralelno-paralelný deliaci integrátor [4]. . . . .                            | 21 |
| 5.4 | Paralelný jednovstupový integrátor v pohyblivej rádovej čiarke. . . . .        | 23 |
| 5.5 | EXP - blok pracujúci s exponentmi. . . . .                                     | 24 |
| 5.6 | Paralelno-paralelný násobiaci integrátor v pohyblivej rádovej čiarke . . . . . | 25 |
| 5.7 | Paralelno-paralelný deliaci integrátor v pohyblivej rádovej čiarke . . . . .   | 27 |
| 5.8 | Schéma zapojenia integrátorov [4] . . . . .                                    | 28 |
| 6.1 | FX aritmetika numerického integrátora. . . . .                                 | 30 |
| 6.2 | FX aritmetika deliaceho integrátora na 32 bitoch. . . . .                      | 31 |
| 6.3 | FX aritmetika deliaceho integrátora na 64 bitoch. . . . .                      | 31 |
| 6.4 | Schéma zapojenia integrátorov . . . . .  | 38 |
| 7.1 | Stavy jednotlivých integrátorov. . . . .                                       | 40 |
| 7.2 | Schéma zapojenia integrátorov . . . . .  | 42 |

# Zoznam tabuliek

|     |  |    |
|-----|--|----|
| 4.1 | Štandard IEEE 754 [11]   | 15 |
| 4.2 | Výsledok operácie sčítania so špeciálnymi hodnotami              | 16 |
| 4.3 | Výsledok operácie násobenia so špeciálnymi hodnotami             | 17 |
| 4.4 | Výsledok operácie delenia so špeciálnymi hodnotami               | 17 |
| 5.1 | Význam skratiek použitých v obrázku 5.2.                         | 20 |
| 5.2 | Význam skratiek v obrázku 5.3.                                   | 22 |
| 5.3 | Význam skratiek použitých v obrázku 5.5.                         | 24 |
| 6.1 | Kontrolér násobiaceho integrátora v FX                           | 33 |
| 6.2 | Kontrolér násobiaceho integrátora v FP                           | 33 |
| 6.3 | Kontrolér deliaceho integrátora v FX                             | 34 |
| 6.4 | Kontrolér deliaceho integrátora v FP.                            | 35 |
| 6.5 | Kontrolér jednovstupového integrátora v FP                       | 36 |
| 6.6 | Kontrolér jednovstupového integrátora v FP v zapojení do sústavy | 37 |
| 7.1 | Porovnanie presnosti a rýchlosti numerických metód.              | 43 |

# Kapitola 1

## Úvod

Hlavným cieľom tejto práce je návrh hardvérových komponentov na riešenie rozsiahlych diferenciálnych rovníc. Diferenciálne rovnice sa väčšinou riešia pomocou numerickej integrácie, a teda s použitím vhodných numerických metód. Hardvérový komponent využívajúci numerickú integráciu sa nazýva numerický integrátor.

V kapitole 2 sú predstavené rôzne numerické metódy - Eulerova metóda, metóda Runge-Kutta a Taylorov rad. Najväčšia pozornosť je venovaná metóde Taylorovej rady, ktorá poskytuje vhodný pomer medzi rýchlosťou a presnosťou [3]. Bližší popis a práca s touto metódou sú uvedené v kapitole 3. Ukážeme si rozdelenie Taylorovej rady na jednotlivé členy a následne úpravu týchto členov tak, aby bolo možné výpočet čo najviac paralelizovať a optimalizovať. Táto úprava bude realizovaná na obyčajných diferenciálnych rovniciach s operáciou násobenia a delenia. Takto upravené a vytvorené rovnice budú následne použité pri návrhu rôznych typov numerických integrátorov.

Kapitola 4 sa zaoberá reprezentáciou operandov v pevnej a v pohyblivej rádovej čiarky. Pri použití pohyblivej rádovej čiarky sú uvedené postupy výpočtu znamienka, exponentu a mantisy na jednoduchých matematických operáciách ako sú sčítanie, odčítanie, násobenie a delenie.

V kapitole 5 sú predstavené a popísané návrhy jednotlivých integrátorov a popis ich činnosti. Podľa rovníc uvedených v kapitole 3, sú navrhnuté paralelné numerické integrátory s operáciou násobenia a delenia. Oba integrátory sú navrhnuté v prevedení pevnej a pohyblivej rádovej čiarky. Operácia delenia je realizovaná pomocou deliacho algoritmu SRT. Bližší popis tohto algoritmu sa nachádza v bakalárskej práci Simulátor procesora s operáciou delenia [4]. Vzájomným zapojením navrhnutých numerických integrátorov je možné riešiť rozsiahle diferenciálne rovnice.



## Kapitola 2

# Numerická integrácia

Diferenciálne rovnice sú matematické rovnice, v ktorých ako premenné vystupujú derivácie funkcií. Najvyššia derivácia v rovnici udáva rád rovnice. Rovnice, ktoré obsahujú derivácie len podľa jednej premennej, sa nazývajú obyčajné diferenciálne rovnice (ODR). Rovnice, ktoré obsahujú derivácie podľa viacerých premenných, sú takzvané parciálne diferenciálne rovnice (PDR). Diferenciálnu rovnicu je možné riešiť analyticky alebo použitím numerickej metódy. Pri väčšine praktických úloh je analytické riešenie veľmi zložité, preto sa používa skôr riešenie numerické. Základným princípom numerického riešenia je diskretizácia premenných, keď spojitú veličinu nahradíme postupnosťou diskrétnych bodov. Pri použití dostatočne hustého rozloženia bodov môžeme približne reprezentovať spojitú veličinu. Vzdialenosť medzi dvoma susednými bodmi sa nazýva krok metódy. Numerické metódy pri svojom výpočte používajú niekoľko predchádzajúcich krokov. Podľa počtu týchto krokov rozdeľujeme numerické metódy na metódy jednokrokové a viackrokové. Jednokrokové metódy pri svojom výpočte používajú len jeden predchádzajúci krok, viackrokové metódy využívajú niekoľko predchádzajúcich krokov. Pri numerických metódach je teda potrebné zvoliť počiatočný stav, t. j. počiatočnú podmienku riešenej úlohy. Od nej sa následne počíta hodnota funkcie v ďalšom bode [5].

Hlavnými kritériami pri numerických metódach sú presnosť a rýchlosť. Tie je možné ovplyvniť veľkosťou integračného kroku a rádom integračnej metódy. Pri počítaní numerickými metódami nedostávame teoreticky presné riešenie, ale výsledok konverguje k správne-  
nemu riešeniu, a teda dostávame výsledok s určitou presnosťou. Výsledná chyba výpočtu je súčet lokálnej a akumulovanej chyby. Lokálna chyba zahŕňa chybu numerickej metódy a zaokrúhľovaciu chybu, ktorá môže byť spôsobená typom hardvérovej architektúry, ako napríklad použitím pevnej alebo pohyblivej rádovej čiarky, ktoré sú bližšie popísané v kapitole 4. Akumulovaná chyba je súčtom lokálnych chýb, čiže sa počas výpočtu zvyšuje. Chyba v jednom kroku teda ovplyvňuje výsledky krokov nasledujúcich.

### 2.1 Taylorova rada

Táto numerická metóda je tvorená nekonečným radom, avšak na výpočet sa používa len niekoľko jej členov. Počet použitých členov udáva rád metódy. Čím väčší počet členov použijeme, tým je výsledok presnejší. Počet použitých členov môže byť zadaný fixne, alebo sa môže dynamicky meniť v závislosti od požadovanej presnosti. Presnosť sa počíta pomocou viacerých najvyšších členov. Po dosiahnutí požadovanej presnosti výpočet končí. Nekonečnú Taylorovu radu môžeme zapísať:

$$y_{i+1} = y_i + hy'_i + \frac{h^2}{2!}y''_i + \frac{h^3}{3!}y'''_i + \frac{h^4}{4!}y^{(4)}_i + \dots + \frac{h^n}{n!}y^{(n)}_i, \quad (2.1)$$

kde  $h$  je veľkosť integračného kroku a  $i$  označuje krok diskretizovanej veličiny. Ďalšie popísané metódy sú odvodené od Taylorovej rady.

## 2.2 Eulerova metóda

Najjednoduchšou jednokrokovou metódou na riešenie obyčajných diferenciálnych rovníc je Eulerova metóda. Je to Taylorova metóda 1. rádu, keďže používa len prvé dva členy Taylorovej rady. Je preto rýchla, no menej presná. Zapisuje sa nasledovne:

$$y_{i+1} = y_i + hy'_i. \quad (2.2)$$

Zvolením dostatočne malého integračného kroku  $h$  môžeme zvýšiť jej presnosť. Čím je však krok menší, tým je výpočet pomalší.

## 2.3 Runge-Kutta

Ďalšou veľmi známou numerickou metódou je Runge-Kutta. Všeobecná schéma tejto metódy má tvar:

$$\begin{aligned} y_{i+1} &= y_i + \sum_{j=1}^r \alpha_j k_j, \quad i = 0, 1, \dots \\ k_1 &= f(x_i, y_i) \\ k_j &= f(x_i + \lambda_j h, y_i + \mu_j h k_{j-1}), \quad j = 2, \dots, r, \end{aligned} \quad (2.3)$$

kde  $\alpha_j$ ,  $\lambda_j$  a  $\mu_j$  sú vhodne zvolené konštanty a  $r$  určuje rád metódy. Ako je v uvedených rovniach vidieť, pri výpočte sa používajú medzivýpočty  $k$ , ktorých počet sa rovná rádu metódy. Najznámejšie a najčastejšie používané varianty sú Runge-Kutta 2. a 4. rádu, ktoré sú popísané nižšie.

### 2.3.1 Runge-Kutta 2. rádu

Táto metóda je oproti Eulerovej metóde presnejšia, ale pri rovnakej veľkosti integračného kroku vyžaduje viac operácií. Na výpočet používa dva medzivýpočty  $k_1$  a  $k_2$ . Má nasledujúci tvar:

$$\begin{aligned} y_{i+1} &= y_i + \frac{1}{2}h(k_1 + k_2) \\ k_1 &= f(t_i, y_i) \\ k_2 &= f(t_{i+1}, y_i + hk_1). \end{aligned} \quad (2.4)$$

### 2.3.2 Runge-Kutta 4. rádu

Runge-Kutta 4. rádu je najpoužívanější tvar tejto metódy. Môžeme ho zapísať nasledovne:

$$\begin{aligned}y_{i+1} &= y_i + \frac{1}{6}h(k_1 + 2k_2 + 2k_3 + k_4) \\k_1 &= f(t_i, y_i) \\k_2 &= f(t_{i+1} + \frac{1}{2}h, y_i + \frac{1}{2}hk_1) \\k_3 &= f(t_{i+1} + \frac{1}{2}h, y_i + \frac{1}{2}hk_2) \\k_4 &= f(t_{i+1} + h, y_i + hk_3).\end{aligned}\tag{2.5}$$

Napriek tomu, že sú pri výpočte použité štyri medzivýpočty, má táto metóda dobrý pomer rýchlosti a presnosti.

## Kapitola 3

# Riešenie diferenciálnych rovníc Taylorovou radou

Na riešenie diferenciálnych rovníc je možné upraviť základný tvar Taylorovej rady (2.1) tak, aby sa dali jednotlivé operácie vykonávať paralelne. Prevod jednoduchej obyčajnej diferenciálnej rovnice je prevzatý z [4]. Ďalšími možnými zdrojmi sú [7], [2].

Obyčajná diferenciálna rovnica:

$$y' = y, \quad y(0) = y_0. \quad (3.1)$$

Z tohoto vzťahu vyplýva, že:

$$y = y' = y'' = y''' = y^{(4)} = \dots = y^{(n)}. \quad (3.2)$$

Po dosadení do Taylorovej rady (2.1) získame:

$$y_{i+1} = y_i + hy_i + \frac{h^2}{2!}y_i + \frac{h^3}{3!}y_i + \frac{h^4}{4!}y_i + \dots + \frac{h^n}{n!}y_i. \quad (3.3)$$

To je možné prepísať na:

$$y_{i+1} = y_i + DY1_i + DY2_i + DY3_i + DY4_i + \dots + DY(N)_i, \quad (3.4)$$

kde je význam jednotlivých členov nasledujúci:

$$\begin{aligned} DY1_i &= hy_i \\ DY2_i &= \frac{h^2}{2!}y_i = \frac{h}{2}DY1_i \\ DY3_i &= \frac{h^3}{3!}y_i = \frac{h}{3}DY2_i \\ DY4_i &= \frac{h^4}{4!}y_i = \frac{h}{4}DY3_i \\ &\vdots \end{aligned} \quad (3.5)$$

Všeobecný zápis:

$$DY(N)_i = \frac{h^n}{n!}y_i = \frac{h}{n}DY(N-1)_i$$

Na základe týchto vzťahov je možné riešiť jednoduché diferenciálne rovnice, ako to je uvedené napríklad v [6] alebo v [2]. Zo vzťahov taktiež vyplýva, že každý ďalší člen Taylorovej rady je počítaný z predchádzajúceho člena, čo vedie k zefektívneniu výpočtu, a to najmä pri vyšších deriváciách, v ktorých je výpočet častokrát zložitý. Podobným postupom je možné upraviť diferenciálne rovnice, ktoré obsahujú operáciu násobenia alebo delenia, a tiež tak zvýšiť efektivitu výpočtu týchto výpočtovo náročnejších operácií.

### 3.1 Riešenie diferenciálnej rovnice s operáciou násobenia

Všeobecný zápis pre diferenciálnu rovnicu s operáciou násobenia je nasledujúci:

$$y' = qr, \quad y(0) = y_0. \quad (3.6)$$

Pre výpočet každej premennej v rovnici je potrebné použiť Taylorovu radu v tvare (3.4). Jednotlivé Taylorove rady vyzerajú nasledovne:

$$y_{i+1} = y_i + DY1_i + DY2_i + DY3_i + DY4_i + \dots + DY(n)_i \quad (3.7)$$

$$q_{i+1} = q_i + DQ1_i + DQ2_i + DQ3_i + DQ4_i + \dots + DQ(n)_i \quad (3.8)$$

$$r_{i+1} = r_i + DR1_i + DR2_i + DR3_i + DR4_i + \dots + DR(n)_i. \quad (3.9)$$

Spočítame jednotlivé derivácie rovnice (3.6):

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

Ako je vidieť, derivácie vytvárajú Pascalov trojuholník a môžeme ich všeobecne zapísať ako

$$y^{(n+1)} = \sum_{k=0}^n \binom{n}{k} q^{(n-k)} r^{(k)}.$$

Z derivácií odvodíme jednotlivé členy Taylorovej rady (2.1), ktoré majú nasledujúci význam:

$$\begin{aligned} \frac{DY1_i}{h} &= DQ0_i DR0_i \\ \frac{DY2_i}{\frac{h^2}{2!}} &= \frac{DQ1_i}{h} DR0_i + DQ0_i \frac{DR1_i}{h} \\ \frac{DY3_i}{\frac{h^3}{3!}} &= \frac{DQ2_i}{\frac{h^2}{2!}} DR0_i + 2 \frac{DQ1_i}{h} \frac{DR1_i}{h} + DQ0_i \frac{DR2_i}{\frac{h^2}{2!}} \\ \frac{DY4_i}{\frac{h^4}{4!}} &= \frac{DQ3_i}{\frac{h^3}{3!}} DR0_i + 3 \frac{DQ2_i}{\frac{h^2}{2!}} \frac{DR1_i}{h} + 3 \frac{DQ1_i}{h} \frac{DR2_i}{\frac{h^2}{2!}} + DQ0_i \frac{DR3_i}{\frac{h^3}{3!}} \\ &\vdots \end{aligned} \quad (3.10)$$

Po úprave dostaneme konečný tvar jednotlivých členov:

$$DY1_i = hq_i r_i \quad (3.11)$$

$$DY2_i = \frac{h}{2}(DQ1_i DR0_i + DQ0_i DR1_i) \quad (3.12)$$

$$DY3_i = \frac{h}{3}(DQ2_i DR0_i + DQ1_i DR1_i + DQ0_i DR2_i) \quad (3.13)$$

$$DY4_i = \frac{h}{4}(DQ3_i DR0_i + DQ2_i DR1_i + DQ1_i DR2_i + DQ0_i DR3_i) \quad (3.14)$$

$\vdots$

Všeobecný zápis:

$$DY(N)_i = \frac{h}{N} \cdot \left( \sum_{k=1}^N DQ(N-k)_i \cdot DR(k-1)_i \right). \quad (3.15)$$

Uvedený prevod vychádza z práce [14]. Jednotlivé rovnice členov Taylorovej rady budú slúžiť pri návrhu a implementácii násobiacich integrátorov.

## 3.2 Riešenie diferenciálnej rovnice s operáciou delenia

Podobne ako v predchádzajúcom prípade, zapíšme diferenciálnu rovnicu s operáciou delenia nasledovne:

$$y' = \frac{u}{v}, \quad y(0) = y_0. \quad (3.16)$$

Jednotlivé Taylorove rady pre premenné v rovnici vyzerajú nasledovne:

$$y_{i+1} = y_i + DY1_i + DY2_i + DY3_i + DY4_i + \dots + DY(n)_i \quad (3.17)$$

$$u_{i+1} = u_i + DU1_i + DU2_i + DU3_i + DU4_i + \dots + DU(n)_i \quad (3.18)$$

$$v_{i+1} = v_i + DV1_i + DV2_i + DV3_i + DV4_i + \dots + DV(n)_i. \quad (3.19)$$

Ďalšie derivácie rovnice (3.16) sú:

$$\begin{aligned} y'' &= \frac{u'v - uv'}{v^2} = \frac{1}{v}(u' - y'v') \\ y''' &= \left( \frac{1}{v}(u' - y'v') \right)' = \frac{1}{v}(u'' - 2y''v' - y'v'') \\ y^{(4)} &= \left( \frac{1}{v}(u'' - 2y''v' - y'v'') \right)' = \frac{1}{v}(u''' - 3y'''v' - 3y''v'' - y'v''') \\ &\vdots \end{aligned} \quad (3.20)$$

Aj tu je vidieť, že jednotlivé derivácie tvoria Pascalov trojuholník. Všeobecný zápis je nasledovný:

$$y^{(n+1)} = \frac{1}{v} \left( u^{(n)} - \left( \sum_{k=1}^n \binom{n}{k} y^{(n-k+1)} v^{(k)} \right) \right).$$

Po dosadení jednotlivých členov  $DV(n)_i$  a  $DU(n)_i$  do derivácií (3.20) dostaneme:

$$\begin{aligned}
\frac{DY1_i}{h} &= \frac{1}{v}u \\
\frac{DY2_i}{\frac{h^2}{2!}} &= \frac{1}{v}\left(\frac{DU1_i}{h} - \frac{DY1_i}{h}\frac{DV1_i}{h}\right) \\
\frac{DY3_i}{\frac{h^3}{3!}} &= \frac{1}{v}\left(\frac{DU2_i}{\frac{h^2}{2!}} - 2\frac{DY2_i}{\frac{h^2}{2!}}\frac{DV1_i}{h} - \frac{DY1_i}{h}\frac{DV2_i}{\frac{h^2}{2!}}\right) \\
\frac{DY4_i}{\frac{h^4}{4!}} &= \frac{1}{v}\left(\frac{DU3_i}{\frac{h^3}{3!}} - 3\frac{DY3_i}{\frac{h^3}{3!}}\frac{DV1_i}{h} - 3\frac{DY2_i}{\frac{h^2}{2!}}\frac{DV2_i}{\frac{h^2}{2!}} - \frac{DY1_i}{h}\frac{DV3_i}{\frac{h^3}{3!}}\right) \\
&\vdots
\end{aligned} \tag{3.21}$$

Po úprave majú jednotlivé členy Taylorovej rady nasledovný tvar:

$$DY1_i = \frac{1}{v_i}(hu_i) \tag{3.22}$$

$$DY2_i = \frac{1}{2v_i}(DU1_i h - DY1_i DV1_i) \tag{3.23}$$

$$DY3_i = \frac{1}{3v_i}(DU2_i h - 2DY2_i DV1_i - DY1_i DV2_i) \tag{3.24}$$

$$DY4_i = \frac{1}{4v_i}(DU3_i h - 3DY3_i DV1_i - 2DY2_i DV2_i - DY1_i DV3_i) \tag{3.25}$$

$\vdots$

Všeobecný zápis:

$$DY(N)_i = \frac{1}{NDV0_i} \cdot \left( DU(N-1)_i \cdot h - \left( \sum_{k=1}^{N-1} (N-k) \cdot DY(N-k)_i \cdot DV(k)_i \right) \right). \tag{3.26}$$

Vyjadrenie jednotlivých členov Taylorovej rady vychádza z bakalárskej práce [4]. Takto upravené členy sú následne použité na návrh jednotlivých typov integrátorov popísaných v kapitole 5.

## Kapitola 4

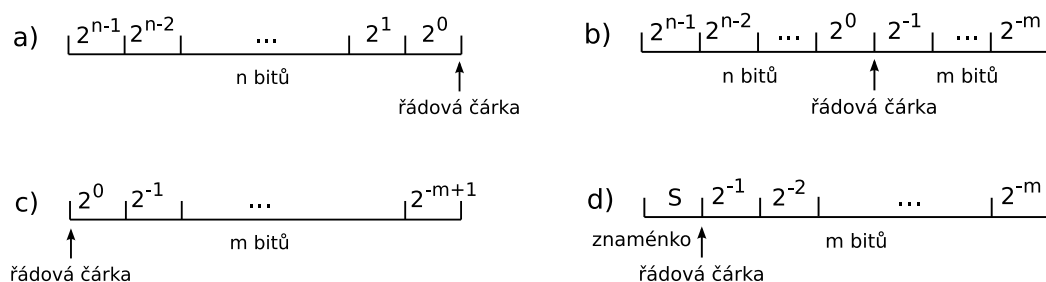
# Reprezentácia operandov

Čísla vo výpočtových systémoch sú reprezentované rôznymi spôsobmi v závislosti od zvolenej aritmetiky. Najpoužívanjšie reprezentácie čísiel sú zapísané v pevnej a v pohyblivej rádovej čiarkke (ang. fixed point (FX) a floating point (FP)). Obe aritmetiky sú popísané v nasledujúcich podkapitolách. Informácie v tejto kapitole sú čerpané z [10], [11], [2], [8], [9].

### 4.1 Pevná rádová čiarka

Pri počítaní v pevnej rádovej čiarkke sú čísla reprezentované na  $k$  bitoch v tvare  $n.m$ , kde prvých  $n$  bitov tvorí časť čísla pred desatinnou čiarkkou a zostávajúcich  $m$  bitov tvorí číslo za desatinnou čiarkkou. Pozícia desatinnej čiarkky je dopredu známa. V závislosti od jej pozície sa používajú rôzne formáty pevnej rádovej čiarkky. Tie najpoužívanjšie sú znázornené na obrázku 4.1.

Obrázok 4.1a zobrazuje aritmetiku s nulovým počtom bitov za desatinnou čiarkkou, a teda ide o celočíselnú aritmetiku na  $n$  bitoch. Vedľajší obrázok 4.1b zobrazuje aritmetiku s  $n$  bitmi pred desatinnou čiarkkou a s  $m$  bitmi za desatinnou čiarkkou. Ďalšia časť obrázku 4.1c je opakom 4.1a, kde je číslo reprezentované len za desatinnou čiarkkou na  $m$  bitoch. Posledný obrázok 4.1d je podobný ako 4.1c, no má pridaný znamienkový bit  $S$ . V tejto práci budeme používať FX aritmetiku v tvare 4.1b s pridaným znamienkovým bitom na pozícii MSB (most significant bit).



Obr. 4.1: Rôzne formáty fixed point aritmetiky [2]

Vo fixed point aritmetike sa používajú rôzne kódy, napr. priamy kód, doplnkový kód či inverzný kód. V nasledujúcich kapitolách a pri návrhu integrátorov v pevnej rádovej čiarkke budeme používať doplnkový kód.



## 4.2 Pohyblivá rádová čiarka

Čísla uložené v pohyblivej rádovej čiarkke sú tvorené exponentom a mantisou. Všeobecný tvar na získanie hodnoty uloženej vo FP je nasledujúci:

$$X = B^E \cdot M \quad (4.1)$$

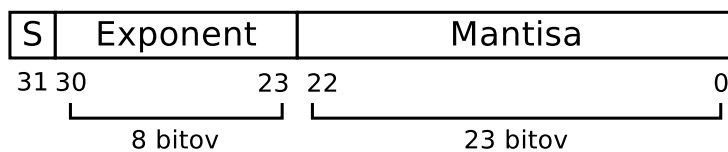
$X$  - výsledná hodnota

$B$  - základ sústavy

$E$  - hodnota exponenta

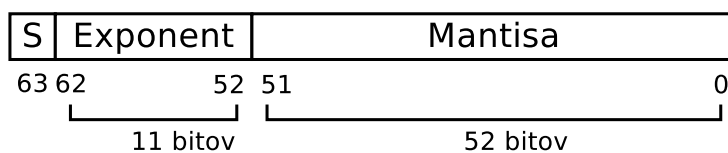
$M$  - mantisa

Zvýšením počtu bitov v exponente  $E$  sa zvýši rozsah hodnôt, ktorý je možné reprezentovať, a zvýšením počtu bitov v mantise  $M$  sa zvýši presnosť uložených čísel. Existuje veľa formátov uloženia čísel v pohyblivej rádovej čiarkke. Najpoužívanější a najrozšírenejší je štandard **IEEE 754**. Ten definuje vlastný formát uloženia čísel a viaceré formáty s rôznou presnosťou. Najpoužívanějšíe z nich sú formáty čísel s jednoduchou (single) a s dvojitou (double) presnosťou. Čísla s jednoduchou presnosťou sú uložené na 32 bitoch, kde MSB je znamienkový bit  $S$ , ďalších 8 bitov tvorí exponent  $E$  a zvyšných 23 bitov tvorí mantisu  $M$ .



Obr. 4.2: IEEE 754 formát s jednoduchou presnosťou

Čísla s dvojitou presnosťou sú uložené v rovnakom formáte ako čísla s jednoduchou presnosťou, avšak počet bitov je zvýšený o dvojnásobok. Dvojitá presnosť je teda tvorená 64 bitmi, kde MSB je znamienkový bit  $S$ , ďalších 11 bitov tvorí exponent  $E$  a zvyšných 52 bitov tvorí mantisu  $M$ .



Obr. 4.3: IEEE 754 formát s dvojitou presnosťou

Znamienko  $S$  nadobúda hodnotu 0, čo znamená kladné číslo; alebo hodnotu 1, čo znamená záporné číslo. Exponent je uložený v kóde s nepárnym posunutím o hodnotu BIAS. Táto hodnota je zvolená tak, aby bol uložený exponent vždy kladný. Pri jednoduchej presnosti má teda BIAS hodnotu 127 a pri dvojitej presnosti hodnotu 2047.

Hodnota mantisy je uložená v priamom kóde bez znamienka, znížená o hodnotu 1, keďže je tu použitá tzv. normalizácia. Mantis je normalizovaná do tvaru  $1.M$ , kde sa jednotka neukladá – je skrytá, čím sa ušetrí jeden bit. Hodnotu takto uloženého čísla získame zo vzťahu:

$$X_{754} = -1^S \cdot 2^{E-BIAS} \cdot (1, M) \quad (4.2)$$

$X_{754}$  - výsledná hodnota

$BIAS$  - 127 alebo 2047

$E$  - hodnota exponentu

$M$  - mantisa

Štandard IEEE 754 definuje aj špeciálne hodnoty ako kladnú/zápornú nulu, kladné/záporné nekonečno, či hodnotu NaN (not a number). Tieto hodnoty sú uvedené v tabuľke 4.1. Hodnota mantisy v normalizovanom tvare je v intervale  $< 1, 0; 2, 0$ ). Ak tomu tak nie je, ide o tzv. denormalizované číslo, a hodnota exponenta je braná ako  $-126$ . Štandard IEEE 754 definuje aj spôsob vykonávania základných matematických operácií, ktoré sú popísané v sekciách 4.2.1 a 4.2.2.

| S (znamienko) | E (exponent) | M (mantisa)       | význam                    |
|---------------|--------------|-------------------|---------------------------|
| 0/1           | 00000000     | nulová hodnota    | +/- 0                     |
| 0/1           | 00000000     | nenulová hodnota  | +/- denormalizované číslo |
| 0/1           | 1 - 254      | ľubovoľná hodnota | +/- FP číslo              |
| 0/1           | 11111111     | nulová hodnota    | +/- $\infty$              |
| 0/1           | 11111111     | nenulová hodnota  | NaN                       |

Tabuľka 4.1: Štandard IEEE 754 [11]

### 4.2.1 Súčet a rozdiel

Súčet a rozdiel v pohyblivej rádovej čiarke sa počíta podľa vzorcov

$$X + Y = (M_X \cdot 2^{E_X - E_Y} + M_Y) \cdot 2^{E_Y}, \text{ kde } E_X \leq E_Y \quad (4.3)$$

$$X - Y = (M_X \cdot 2^{E_X - E_Y} - M_Y) \cdot 2^{E_Y}, \text{ kde } E_X \leq E_Y. \quad (4.4)$$

Postup výpočtu operácie súčtu alebo rozdielu v pohyblivej rádovej čiarke podľa štandardu IEEE 754 ([12], [15]) je nasledovný:

1. Na začiatku výpočtu sa obe čísla skontrolujú na výskyt špeciálnych hodnôt z tabuľky 4.1. Ak ide o špeciálne číslo, výsledok sa určí podľa tabuľky 4.2. Inak sa pokračuje bodom 2.
2. Vykoná sa porovnanie exponentov. Ak sú exponenty rozdielne, mantisu menšieho čísla posunieme o rozdiel exponentov doprava. Tým docielime rovnosť oboch exponentov. Pri posune je dôležité, aby sme nezabudli na jednotku, ktorá je skrytá kvôli normalizácii. Posun sa vykonáva spolu s toutou jednotkou.
3. Následne sa porovnávajú znamienka a podľa výsledku sa vykoná súčet alebo rozdiel mantisy väčšieho čísla a posunutej mantisy. Ak došlo k pretečeniu, mantisa výsledku sa posunie o jeden bit doprava a hodnota exponentu sa zvýši. Aby bolo možné pretečenie detekovať, je potrebné vykonávať sčítanie alebo odčítanie mantís na sčítačke o jeden bit väčšej než je veľkosť mantisy (veľkosťou mantisy sa tu myslí počet bitov potrebných na uloženie mantisy aj so skrytou 1, čiže  $|1.M| + 1$ ).
4. Ak je to potrebné, vykoná sa normalizácia. Mantisa sa posunie o potrebný počet bitov doprava, resp. doľava, tak, aby bola v tvare  $1.M$ . Exponent sa zvýši, resp. zníži o daný počet bitov.
5. Na koniec výpočtu sa skontroluje hodnota exponentu. Ak je hodnota maximálna, došlo k pretečeniu výsledku. Ten sa nastaví podľa znamienka na kladné alebo záporné nekonečno. V opačnom prípade, ak je hodnota exponentu minimálna (nulová), došlo k podtečeniu, a výsledok je nastavený podľa znamienka na kladnú alebo zápornú nulu.

| Hodnota operandu 1 | Hodnota operandu 2 | Výsledok sčítania |
|--------------------|--------------------|-------------------|
| FP číslo           | +/- $\infty$       | +/- $\infty$      |
| +/- $\infty$       | +/- $\infty$       | +/- $\infty$      |
| + $\infty$         | - $\infty$         | NaN               |
| NaN                | ľubovoľná hodnota  | NaN               |

Tabuľka 4.2: Výsledok operácie sčítania so špeciálnymi hodnotami

### 4.2.2 Násobenie a delenie

Násobenie a delenie v pohyblivej rádovej čiarky sa počíta nasledovne:

$$X \times Y = (M_X \cdot M_Y) \cdot 2^{E_X + E_Y} \quad (4.5)$$

$$X \div Y = (M_X \div M_Y) \cdot 2^{E_X - E_Y}. \quad (4.6)$$

Postup výpočtu operácií násobenia a delenia podľa štandardu IEEE 754 ([12], [15]) je nasledovný:

1. Rovnako ako pri sčítaní, aj teraz sa na začiatku výpočtu skontroluje výskyt špeciálnych hodnôt oboch čísel podľa tabuľky 4.1. Ak ide o špeciálne číslo, výsledok sa určí podľa tabuľky 4.3 alebo 4.4. Pri operácii delenia je potrebné kontrolovať nepovolenú operáciu delenie nulou. Pokračuje sa bodom 2.
2. Pri násobení sa hodnota exponentu vypočíta ako súčet exponentov, od ktorého sa odpočíta hodnota *BIAS*. Pri operácii delenia sa hodnota exponentu vypočíta ako rozdiel exponentov, ku ktorému je pripočítaná hodnota *BIAS*.
3. Výsledná mantisa sa rovná súčinu, resp. podielu mantís. Pri násobení je potrebné použiť násobičku, ktorej bitová šírka sa rovná dvojnásobku počtu bitov mantisy 1.*M*. Ak dôjde k pretečeniu alebo k podtečeniu mantisy, vykoná sa posun mantisy doprava, resp. doľava, a hodnota exponentu sa zvýši, resp. zníži.
4. Pokiaľ je to potrebné, prebehne normalizácia.
5. Na konci výpočtu sa skontroluje hodnota exponentu. Postupuje sa rovnako ako pri operácii súčtu: ak je hodnota exponentu maximálna, nastaví sa podľa znamienka na kladné alebo záporné nekonečno. V opačnom prípade, ak je hodnota exponentu minimálna (nulová), výsledok sa nastaví podľa znamienka na kladnú alebo zápornú nulu.

| Hodnota operandu 1      | Hodnota operandu 2 | Výsledok násobenia |
|-------------------------|--------------------|--------------------|
| kladné/záporné FP číslo | +/- $\infty$       | +/- $\infty$       |
| nula                    | +/- $\infty$       | NaN                |
| +/- $\infty$            | +/- $\infty$       | +/- $\infty$       |
| NaN                     | ľubovoľná hodnota  | NaN                |

Tabuľka 4.3: Výsledok operácie násobenia so špeciálnymi hodnotami

| Hodnota operandu 1      | Hodnota operandu 2 | Výsledok násobenia |
|-------------------------|--------------------|--------------------|
| kladné/záporné FP číslo | +/- 0              | +/- $\infty$       |
| 0                       | 0                  | NaN                |

Tabuľka 4.4: Výsledok operácie delenia so špeciálnymi hodnotami

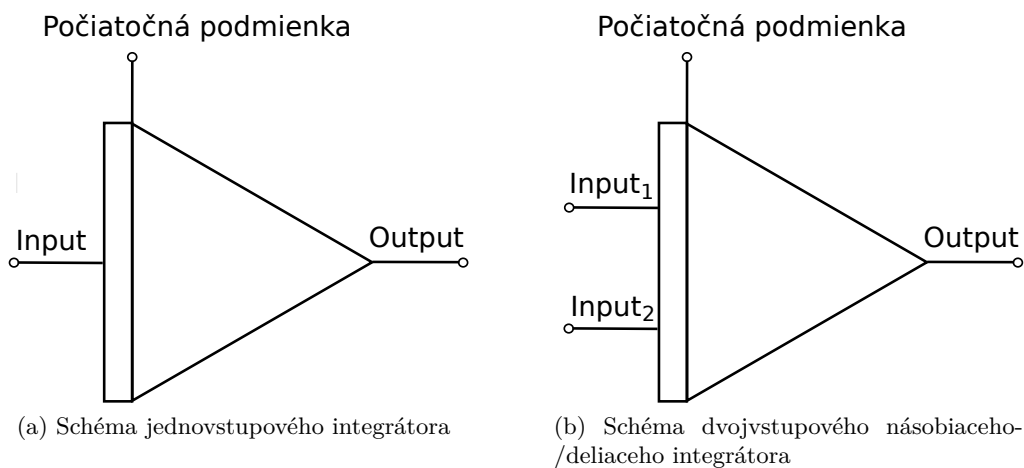
## Kapitola 5

# Numerické integrátory

Numerický integrátor je hardvérový komponent, ktorý slúži na výpočet numerickej integrácie. Podľa spôsobu výpočtu a komunikácie medzi komponentmi integrátora sa numerické integrátory delia na

- sériovo-sériové (sériové) integrátory (sériová komunikácia aj výpočet)
- sériovo-paralelné integrátory (sériová komunikácia a paralelný výpočet)
- paralelno-paralelné (paralelné) integrátory (paralelná komunikácia aj výpočet)

V tejto práci sa budeme zaoberať paralelnými numerickými integrátormi pre jednoduchosť a rýchlosť ich výpočtu. Ďalej môžeme numerické integrátory rozdeliť na jednovstupové a dvojevstupové. Jednovstupový integrátor vykonáva integráciu vstupnej hodnoty a posiela ju na výstup. Schéma tohoto typu integrátora je znázornená na obrázku 5.1a, a ako vyplýva z názvu, obsahuje jeden vstup pre vstupnú hodnotu. Ďalej obsahuje jeden vstup pre počiatočnú podmienku a jeden výstup pre výsledok výpočtu. Dvojevstupové integrátory obsahujú jeden vstup pre počiatočnú podmienku, dva vstupy pre prívod operandov a jeden výstup pre výsledok výpočtu. Schéma integrátora je znázornená na obrázku 5.1b.



Obr. 5.1: Schéma jednovstupového a dvojevstupového integrátora

Tieto integrátory rozdeľujeme podľa použitej operácie na násobiace a na deliace integrátory. Násobiaci integrátor vykonáva deriváciu násobenia dvoch vstupných hodnôt. Deliaci

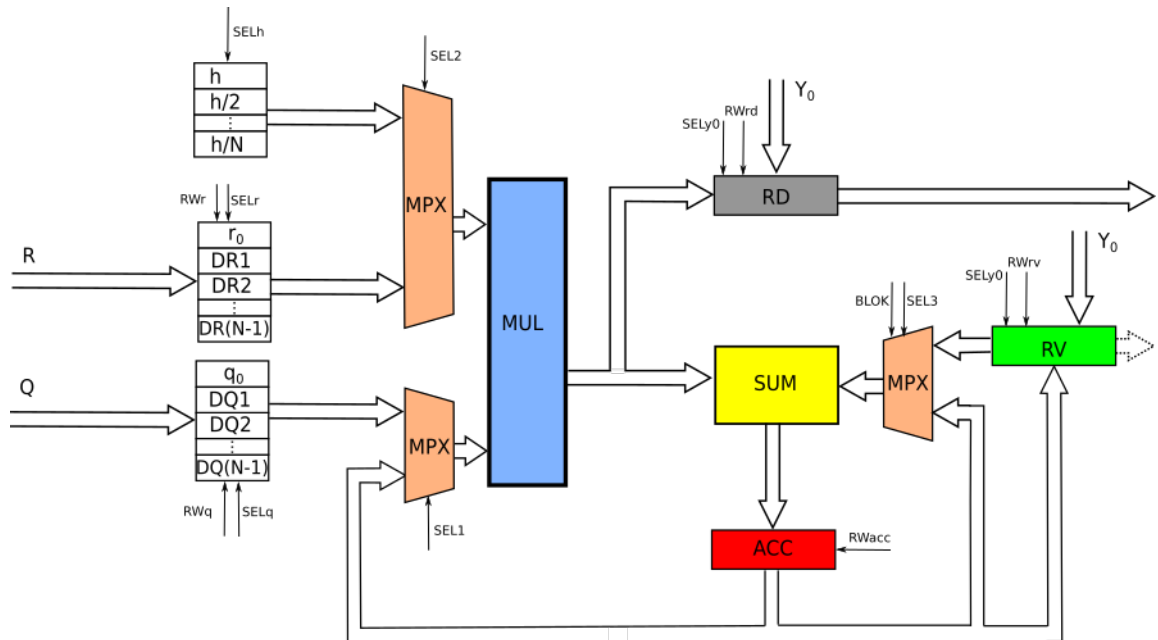
integrátor zasa vykonáva deriváciu delenia dvoch vstupných hodnôt. Výsledná hodnota je poslaná na výstup.

V nasledujúcich podkapitolách predstavíme jednotlivé návrhy paralelných násobiacich a paralelných deliacich integrátorov, oba typy v pevnej a v pohyblivej rádovej čiarke. Predstavíme taktiež jednovstupový paralelný integrátor v pevnej a v pohyblivej rádovej čiarke.

## 5.1 Návrh násobiaceho integrátora v pevnej rádovej čiarke

Násobiaci integrátor počíta rovnicu (3.6) pomocou (3.11) – (3.14). Na základe týchto rovníc bol vytvorený návrh paralelného násobiaceho integrátora, ktorý je na obrázku 5.2. Návrh vychádza z práce V. Závodu [14] a bol upravený a rozšírený o počet registrov  $DR(N - 1)$  a  $DQ(N - 1)$ , ktoré slúžia na ukladanie prichádzajúcich členov. Rozšírený bol aj počet registrov pre uloženie kroku  $h$  integračnej metódy a jeho podielov. Počet týchto registrov je o jeden menší ako rád  $ORD$  použitej Taylorovej metódy ( $N = ORD$ ), keďže pri výpočte sa používajú predchádzajúce členy Taylorovej metódy. Rád metódy je vhodné a potrebné zvoliť vzhľadom na použitú aritmetiku a požadovanú presnosť.

Každý člen  $DY(N - 1)$  obsahuje postupné delenie integračného kroku  $h$ . Tieto hodnoty sú prepočítané a uložené v sade registrov  $h$ . Pre optimalizáciu a pre ušetrenie miesta však nie je potrebné uložiť všetkých  $N$  hodnôt, stačí uložiť len tie hodnoty, ktorých deliteľom je nepárne číslo. Ostatné hodnoty je možné vypočítať jednoduchým posunom registra doprava, čo je vlastne delenie číslom 2. Je síce potrebné pridať riadiace signály na ovládanie posunu jednotlivých registrov, avšak výsledkom je zníženie počtu registrov  $h$  o polovicu. Počet operácií potrebných na výpočet sa nezvyší, keďže posun registrov je možné vykonávať v predstihu a paralelne s inými operáciami.



Obr. 5.2: Paralelno-paralelný násobiaci integrátor [14].

| Skratka                    | Popis   |
|----------------------------|---|
| <b>R, Q</b>                | vstupné operandy integrátora                                |
| <b>Y0</b>                  | počiatočná podmienka  |
| <b>N</b>                   | maximálny rád metódy  |
| <b>h, h/2...h/N</b>        | set registrov s integračným krokom metódy a s jeho podielmi |
| <b>r0, DR1, ...DR(N-1)</b> | set registrov jednotlivých členov vstupného operandu R      |
| <b>q0, DQ1, ...DQ(N-1)</b> | set registrov jednotlivých členov vstupného operandu Q      |
| <b>MUL</b>                 | paralelná násobička   |
| <b>SUM</b>                 | paralelná sčítačka  |
| <b>ACC</b>                 | akumulátor - register na priebežné ukladanie medzivýsledkov |
| <b>RD</b>                  | register na uloženie jednotlivých členov Taylorovej rady    |
| <b>RV</b>                  | register na uloženie celkového výsledku                     |
| <b>MPX</b>                 | multiplexor 2-1   |
| <b>SEL1, SEL2, SEL3</b>    | riadiace signály multiplexorov                              |
| <b>SELh, SELr, SELq</b>    | riadiace signály registrov                                  |
| <b>SELy0</b>               | riadiaci signál na zápis počiatočnej podmienky              |
| <b>RWr, RWq</b>            | povoľovacie signály sád registrov (READ/WRITE)              |
| <b>RWrv, RWrd, RWacc</b>   | povoľovacie signály registrov (READ/WRITE)                  |
| <b>BLOK</b>                | blokový signál multiplexora - na výstupe 0                  |

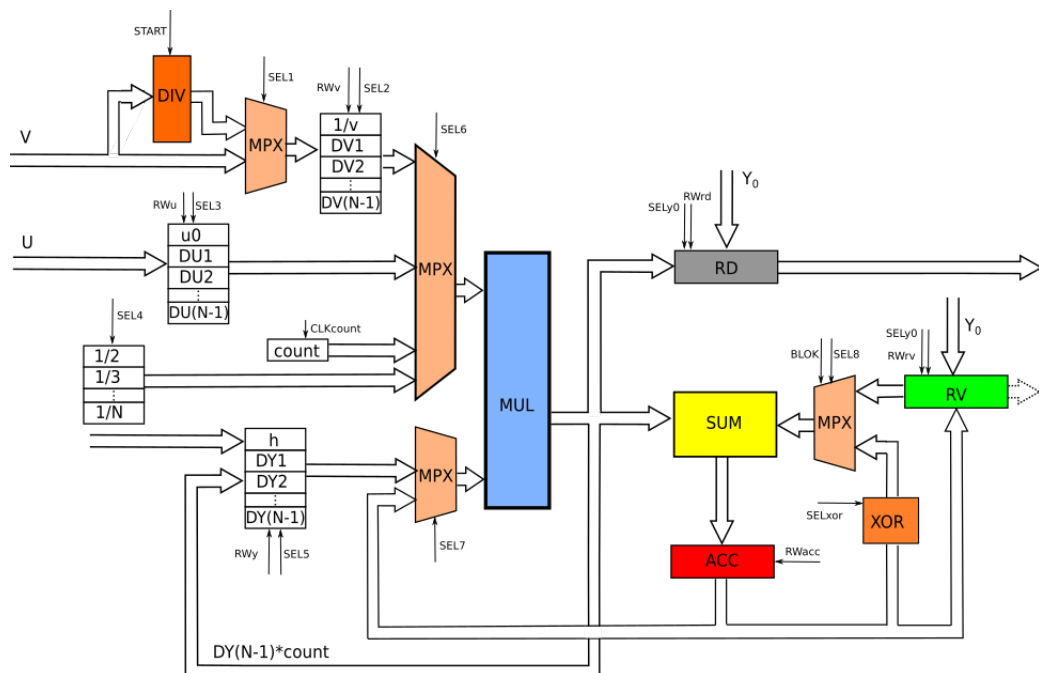
Tabuľka 5.1: Význam skratiek použitých v obrázku 5.2.

Na začiatku výpočtu sú pomocou signálu *RESET* vynulované všetky registre a nastaví sa potrebné signály na multiplexoroch *MPX*. Následne je do registrov *RV* a *RD* nahraná počiatočná podmienka, do registra *h* je nahraný integračný krok metódy a do registrov  $h/i$ ,  $i=2,3..N$  sú nahrané prepočítané hodnoty podielov. Povoľovacím signálom *EN* sa spustí výpočet. Ako prvé sa do registrov uložia vstupné hodnoty *R* a *Q*. Na začiatku každého cyklu výpočtu sú vstupné hodnoty ukladane postupne do ďalších registrov. Po uložení sa začne samotný výpočet. Hodnota aktuálne počítaného člena Taylorovej rady sa postupne ukladá do registra *RD* a odtiaľ je privedená na výstup integrátora. Do registra *RV* je ukladán súčet jednotlivých členov Taylorovej rady - čiže celkový výsledok derivácie vstupných hodnôt. Táto hodnota nás bude zaujímať napríklad vtedy, keď je integrátor "posledný" v zapojení, čiže jeho výstup nie je nikam pripojený, a teda nie je potrebné použiť hodnoty jednotlivých členov.

## 5.2 Návrh deliaceho integrátora v pevnej rádovej čiarke

Tento integrátor počíta rovnicu (3.16) pomocou členov (3.22) - (3.25). Podobne ako pri násobiacom integrátore bol z týchto rovníc vytvorený návrh paralelného deliaceho integrátora. Návrh vychádza z bakalárskej práce [4] a bol upravený podobne ako predošlý násobiaci integrátor pomocou zvýšenia počtu registrov  $DU(N-1)$  a  $DV(N-1)$ . Taktiež sa zvýšil počet registrov  $DY(N-1)$ , ktoré slúžia na uloženie jednotlivých členov Taylorovej rady, keďže je na výpočet nasledujúceho člena použitý predchádzajúci člen. Set týchto registrov okrem hodnôt  $DY(n-1)$  obsahuje register na uloženie kroku *h* integračnej metódy. Deliaci integrátor, na rozdiel od násobiaceho integrátora, nepotrebuje počas výpočtu hodnoty podielov kroku numerickej metódy, a teda stačí uložiť len jednu hodnotu *h*. Deliaci integrátor však používa set registrov, ktorý obsahuje hodnoty  $1/N$ . Podobne je možné ako pri násobiacom

integrátore, tak aj v deliacom integrátore je možné zmenšiť počet týchto registrov na polovicu s využitím aritmetického posunu (operácia *shift*). Register *count* obsahuje *counter*, ktorý sa postupne inkrementuje v každom cykle výpočtu, a s ktorým sa násobí hodnota  $DY(N-1)$ . Vyplýva to z rovníc (3.22) - (3.25) a (3.26), kde sa pri výpočte každého člena Taylorovej rady hodnota  $DY(N-k)$  násobí hodnotou  $N-k$ . Výsledná vynásobená hodnota je uložená opäť do daného registra  $DY(N-1)$  a znova použitá v ďalšom výpočte. Tým sa ušetrí operácia násobenia v ďalšom cykle výpočtu.



Obr. 5.3: Paralelno-paralelný deliaci integrátor [4].



| Skratka                       | Popis  |
|-------------------------------|--|
| <b>V, U</b>                   | vstupné operandy integrátora                           |
| <b>1/v, DV1, ... DV(N-1)</b>  | set registrov jednotlivých členov vstupného operandu V |
| <b>u0, DU1, ... DU(N-1)</b>   | set registrov jednotlivých členov vstupného operandu U |
| <b>count</b>                  | čítač cyklov výpočtu - N                               |
| <b>1/2, 1/3, ... 1/N</b>      | set registrov s jednotlivými podielmi                  |
| <b>h, DY1, ... DY(N-1)</b>    | set registrov s krokom metódy a s vypočítanými členmi  |
| <b>DIV</b>                    | delička  |
| <b>XOR</b>                    | invertovanie hodnoty v dvojkovom doplnku (xor +1)      |
| <b>MPX</b>                    | multiplexor 2-1, 4-1                                   |
| <b>START</b>                  | povoľovací signál na spustenie výpočtu                 |
| <b>SEL1, SEL6, SEL7, SEL8</b> | riadiace signály multiplexorov                         |
| <b>SEL2, SEL3, SEL4, SEL5</b> | riadiace signály registrov                             |
| <b>SELxor</b>                 | signál na invertovanie vstupnej hodnoty                |
| <b>CLKcount</b>               | signál na zvýšenie hodnoty čítača                      |
| <b>RWv, RWu, RWy</b>          | povoľovacie signály sád registrov (READ/WRITE)         |

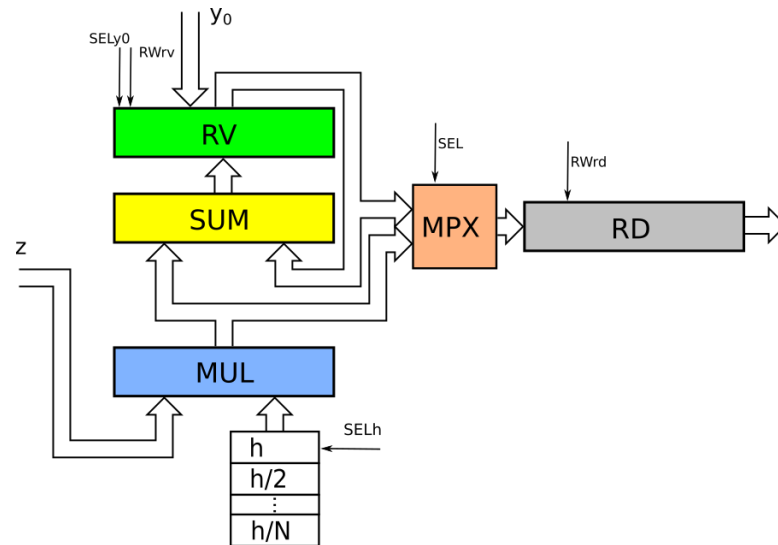
Tabuľka 5.2: Význam skratiek v obrázku 5.3.

Podobne ako pri násobiacom integrátore, aj pri paralelnom deliacom integrátore sú na začiatku výpočtu vynulované registre pomocou signálu *RESET* a nastavené signály multiplexorov *MPX*. Ďalej je do registrov *RD* a *RV* nahraná počiatočná podmienka a do registra *h* je nahraný integračný krok. Do registrov  $1/n$  sú uložené predvypočítané konštanty. Po prijatí hodnôt *U* a *V* sa začne výpočet. Hodnota *V* je privedená do deličky *DIV* a spustí sa výpočet  $1/v$  s použitím deliaceho algoritmu *SRT*. Delenie je realizované len raz počas celého výpočtu, keďže ide o veľmi náročnú operáciu. Po skončení operácie delenia je výsledok uložený do setu registrov *DVN*, a to konkrétne do registra  $1/v$ . V ďalšom cykle výpočtu je vstupná hodnota *V* privedená priamo do setu registrov *DVN* bez delenia. Hodnota *U* je uložená do setu registrov *DUN*. Pre optimalizáciu je paralelne s delením realizovaný výpočet násobenia *uh*. Po skončení výpočtu je výsledok aktuálneho člena uložený do registra *RD* a privedený na výstup deliaceho integrátora. Suma jednotlivých členov je postupne ukladaná v každom cykle do registra *RV*, z ktorého je možné získať hodnotu privedenú na výstup, rovnako ako aj v násobiacom integrátore.

### 5.3 Návrh jednovstupového integrátora

Tento integrátor bol navrhnutý podľa rovníc (3.5), na základe ktorých počíta diferenciálnu rovnicu (3.1). Z toho vyplýva, že vykonáva numerickú integráciu vstupnej hodnoty, a následne výslednú hodnotu posiela na výstup. Schéma integrátora je na obrázku 5.4 a vychádza z práce J. Opálku [6]. Na naštartovanie výpočtu slúži počiatočná podmienka. Tá je nahraná na začiatku výpočtu do registra *RV*. Následne je cez multiplexor *MPX* privedená a uložená do registra *RD*. V sete registrov *h* je uložený integračný krok metódy a jeho podiely. Rovnako ako v predchádzajúcich prípadoch je možné počet registrov znížiť na polovicu. Vstupnú hodnotu *Z* nie je potrebné ukladať do registrov, keďže sa nepoužíva pri výpočte ďalších členov. Jednotlivé hodnoty členov sú ukladané do registra *RV* a poslané na výstup integrátora. Ak chceme získať výslednú hodnotu Taylorovej rady rovnice (3.1),

stačí priviesť hodnotu z registra *RV* cez multiplexor *MPX* do registra *RD*, kde je hodnota sprístupnená na výstupe integrátora.

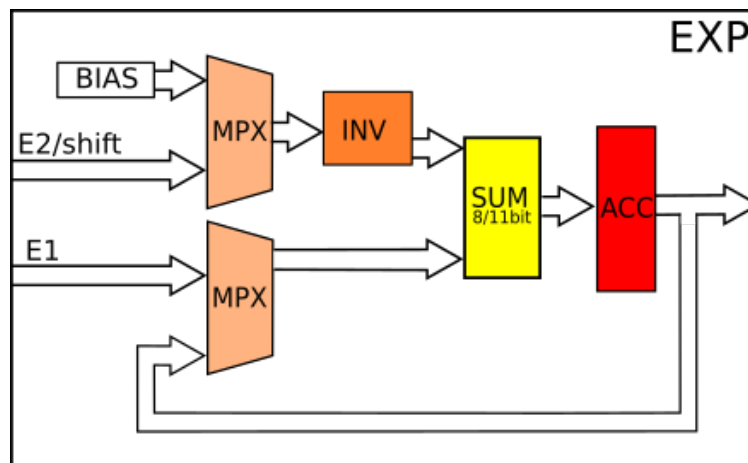


Obr. 5.4: Paralelný jednovstupový integrátor v pohyblivej rádovej čiarke.

## 5.4 Návrh komponentu na spracovanie exponentov

V pohyblivej rádovej čiarke sú operácie násobenia, delenia a sčítania (odčítania) zložitejšie, nakoľko je k týmto operáciám pridaná aj práca s výpočtom znamienka, exponentu a mantisy. Výpočet exponentov je možné vykonávať v každom z komponentov (násobička, delička, sčítačka) samostatne, alebo pomocou jedného komponentu spoločného pre tieto komponentmi. Pri spoločnom komponente je možné znížiť priestorovú zložitosť zapojenia. Tým sa však zvýši náročnosť synchronizácie a riadenia kontrolóra. V deliacom integrátore to môže spôsobiť zdržanie výpočtu vtedy, keď sa operácia delenia a násobenia vykonávajú paralelne. Avšak táto situácia nastane len raz, na začiatku výpočtu. Predošlé popísané návrhy násobiaceho, deliaceho a jednovstupového integrátora je možné použiť ako pri implementácii v pevnej rádovej čiarke, tak aj v pohyblivej rádovej čiarke. Integrátory v pohyblivej rádovej čiarke vychádzajúce z popísaných návrhov pracujú s exponentami v každej operácii samostatne. V nasledujúcich sekciách si popíšeme druhý variant spracovania exponentu so spoločným komponentom.

Blokové schémy zapojenia integrátorov pracujúcich v pohyblivej rádovej čiarke vychádzajú z predchádzajúcich návrhov. Boli však rozšírené o prácu so znamienkami, s mantisou a s exponentmi. Samotný výpočet exponenta sa deje v spoločnom komponente *EXP*, ktorého návrh je zobrazený na obrázku 5.5.



Obr. 5.5: EXP - blok pracujúci s exponentmi.

| Skratka              | Popis   |
|----------------------|---|
| <b>E1</b>            | vstupná hodnota väčšieho exponenta                |
| <b>E2/shift</b>      | vstupná hodnota menšieho exponenta/hodnota posunu |
| <b>BIAS</b>          | hodnota 127 alebo 2047                            |
| <b>SUM (8/11bit)</b> | paralelná 8 alebo 11 bitová sčítačka              |
| <b>INV</b>           | invertor  |

Tabuľka 5.3: Význam skratiek použitých v obrázku 5.5.

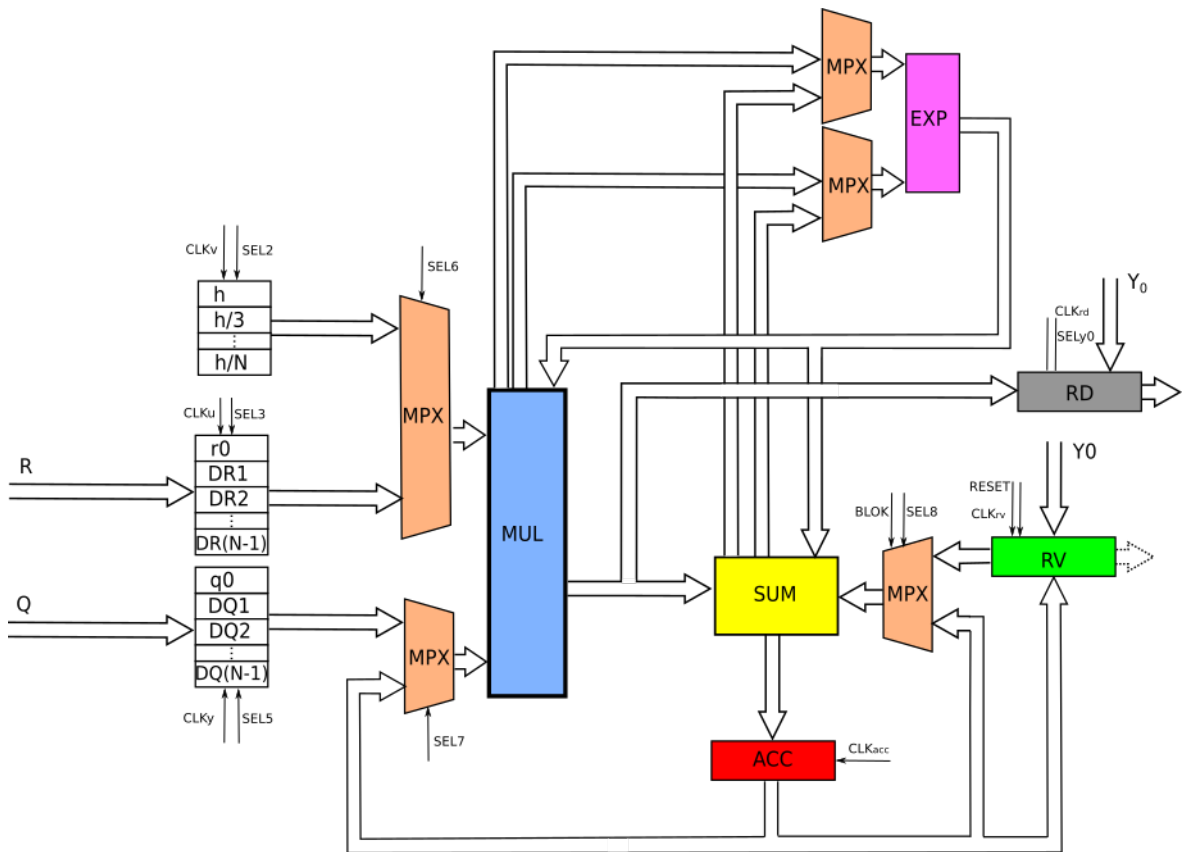
Komponent *EXP* slúži na vykonávanie výpočtov s exponentmi, ktoré sú popísané v sekciách 4.2.1 a 4.2.2. Obsahuje dva vstupy, do ktorých sú privedené jednotlivé exponenty z komponentov *DIV*, *MUL* a *SUM* a jeden výstup. Celý komponent *EXP* pracuje na počte bitov, ktorý sa rovná veľkosti exponenta, ktorá je závislá od použitej aritmetiky. Pri jednoduchšej presnosti (32 bitov) je veľkosť exponenta 8 bitov a pri dvojitej presnosti (64 bitov) na 11 bitov. Komponent teda obsahuje buď 8-bitovú alebo 11-bitovú sčítačku a ostatné komponenty v rovnakej bitovej šírke. Pri vykonávaní operácie súčiny alebo rozdielu slúži komponent *EXP* na výpočet rozdielu exponentov. Výsledná hodnota rozdielu je privedená naspäť do sčítačky *SUM* cez register *ACC*. O túto hodnotu je následne v sčítačke posunutá hodnota mantisy menšieho čísla doprava. Výsledná hodnota mantisy je potom súčet/rozdiel mantisy väčšieho čísla a posunutej mantisy. Znamienko a exponent sa rovnajú hodnote väčšieho čísla z počítaných operandov. Pri operácii delenia alebo násobenia slúži komponent *EXP* na výpočet výsledného exponenta. Podľa typu operácie sa vykoná súčet alebo rozdiel prijatých exponentov. Táto hodnota je uložená v registri *ACC* a privedená naspäť do sčítačky *SUM*<sub>8/11bit</sub> s hodnotou z registra *BIAS*. Vykoná sa súčet alebo rozdiel týchto hodnôt. Získame tak hodnotu nového exponenta, ktorá je privedená naspäť do násobičky *MUL* alebo deličky *DIV*. Ak je však po delení alebo násobení potrebné vykonať posun mantisy, hodnota vypočítaného exponentu je privedená z *ACC* cez multiplexor *MPX* naspäť do sčítačky *SUM*<sub>8/11bit</sub>, a na vstup *E2/Shift* je privedená hodnota posunu. Vykoná sa rozdiel hodnôt, čím vykonáme posun exponenta. Následne je výsledná hodnota privedená do násobičky *MUL* alebo deličky *DIV* a použitá ako nová hodnota exponentu.

Kvôli prehľadnosti schéma neobsahuje znázornenie zapojenia výpočtu znamienka a rozdelenie operandov na znamienko, exponent a na mantisu, a ich opätovné zloženie. Tieto

operácie sa vykonávajú v komponentoch *DIV*, *MUL* a *SUM*.

## 5.5 Návrh násobiaceho integrátora v pohyblivej rádovej čiarke

Návrh paralelného násobiaceho integrátora v pohyblivej rádovej čiarke so spoločným komponentom *EXP* je na obrázku 5.2. Čísla v pohyblivej rádovej čiarke môžeme zobrazit presnejšie ako čísla v pevnej rádovej čiarke, čiže pri uložení malých čísel v pohyblivej rádovej čiarke dochádza k menšej zaokrúhľovacej chybe. Z tohoto dôvodu je možné počítať Taylorovu radu s použitím väčšieho počtu členov a zvýšiť tak presnosť výpočtu. Integrátory v pohyblivej rádovej čiarke teda môžu počítať viac členov Taylorovej rady, a teda počet registrov  $DR(N-1)$  a  $DQ(N-1)$  sa zvýši. Set registrov  $h/i, i = 2, 3..N$  obsahuje  $N-1$  registrov. Počet týchto registrov je možné znížiť rovnako ako pri integrátoroch v pevnej rádovej čiarke.



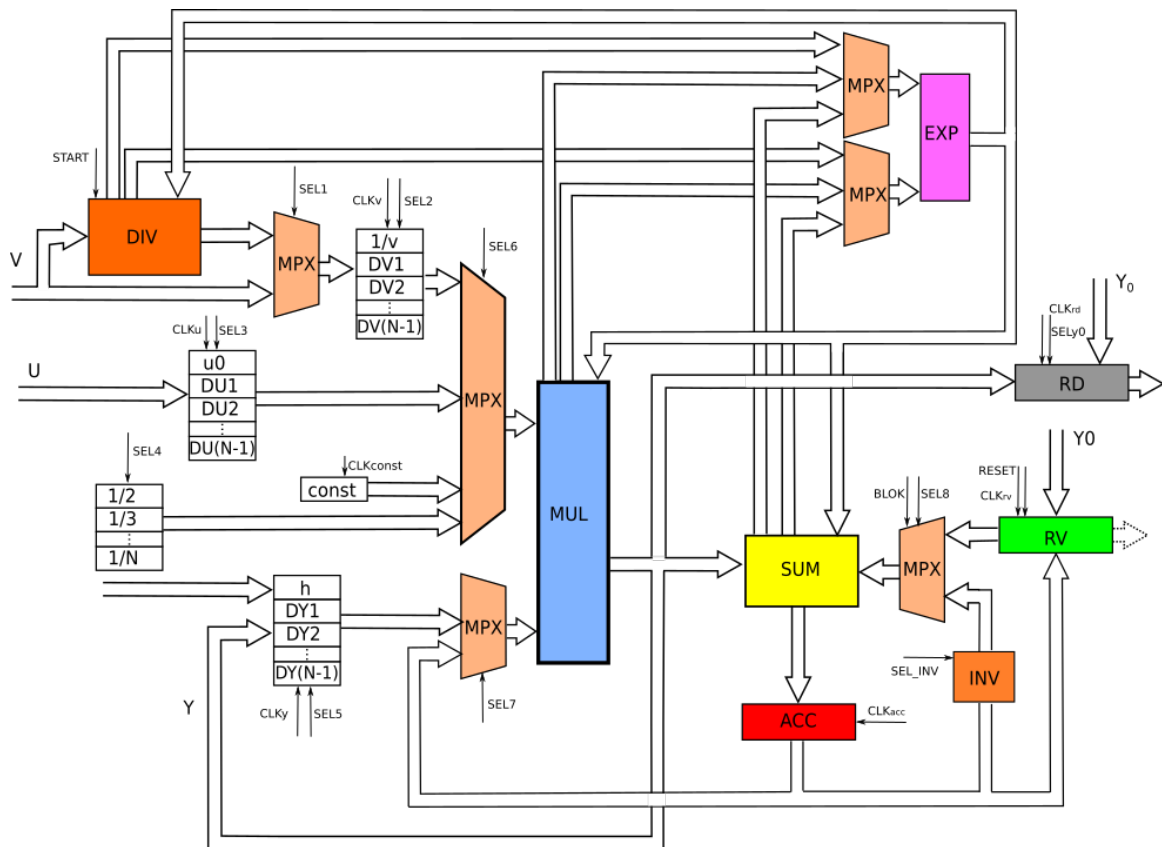
Obr. 5.6: Paralelno-paralelný násobiaci integrátor v pohyblivej rádovej čiarke

Spôsob výpočtu paralelného násobiaceho integrátora v pohyblivej rádovej čiarke so spoločným komponentom *EXP* je takmer rovnaký ako pri násobiacom integrátore v pevnej rádovej čiarke. Rozdielne sa vykonávajú operácie násobenia a sčítania. Po privedení hodnôt na vstupy násobičky *MUL* alebo sčítačky *SUM* sú čísla v FP uložené do pomocných registrov v týchto komponentoch. Z týchto registrov sú jednotlivé časti FP čísla roz distribuované do samostatných výpočtových obvodov. V komponente *MUL* sú znamienka privedené ku

komponentu *XOR*, ktorý vykonáva nonekvivalenciu. V komponente *SUM* sa výsledné znamienko rovná znamienku väčšieho čísla zo vstupných hodnôt. Exponenty, ako je popísané vyššie, spracúva komponent *EXP*. Mantis sú privedené do paralelnej násobičky alebo sčítačky. Veľkosť výsledku násobičky je dvojnásobná veľkosti mantisy so skrytou jednotkou, t. j. 48 bitov pri jednoduchej presnosti a 106 bitov pri dvojitej presnosti. Veľkosť sčítačky sa rovná veľkosti mantisy so skrytou jednotkou, t. j. 24 alebo 53 bitov. Výsledné hodnoty jednotlivých častí čísla v pohyblivej rádovej čiarke sú na výstupe spojené do jedného čísla a sú uložené do výstupného registra daného komponentu (*MUL* alebo *SUM*). Následne sú poskytnuté na jeho výstupe k ďalšiemu výpočtu.

## 5.6 Deliaci integrátor v pohyblivej rádovej čiarke

Paralelný deliaci integrátor v pohyblivej rádovej čiarke so spoločným komponentom *EXP* je najzložitejší z navrhnutých integrátorov. Obsahuje všetky spomínané operácie: sčítanie, odčítanie, násobenie a delenie; a všetky vykonáva v pohyblivej rádovej čiarke, v ktorej je práca s exponentami vykonávaná v jednom komponente *EXP*. Operácie násobenia a sčítania sa vykonávajú rovnako ako v paralelnom násobiacom integrátore v pohyblivej rádovej čiarke. Operácia delenia sa vykonáva iba raz za celý výpočet, a to paralelne s operáciou násobenia, ako aj v deliacom integrátore s FX aritmetikou. Tu však môže dôjsť ku kolízii v použití komponentu *EXP*, keďže delenie je náročnou operáciou, pričom dĺžka výpočtu závisí na použití daného algoritmu. Podľa zvoleného algoritmu je potom potrebné zvoliť, ktorá operácia bude mať prednosť v použití komponentu *EXP*. Po uvoľnení *EXP* násobičkou *MUL* je komponent *EXP* pridelený deličke *DIV* na výpočet exponentov. Ak je použitý zdĺhavejší algoritmus pri výpočte delenia (napr. SRT algoritmus použitý v deliacom integrátore vo FX aritmetike), čiže ak operácia delenia trvá dlhší čas ako operácia násobenia, je vhodnejšie prenechať komponentu *EXP* najprv operácii násobenia. Ak je deliaci algoritmus veľmi rýchly a čas výpočtu je podobný času výpočtu operácie násobenia, tak potom nezáleží, ktorý z komponentov bude ako prvý využívať *EXP*. Po skončení delenia je podiel uložený do registra  $1/v$  a pokračuje sa v ďalšom výpočte. Výpočet prebieha podobne ako pri deliacom integrátore bez komponentu *EXP*. Rozdielne sa vykonávajú operácie delenia, násobenia a sčítania. Ďalší rozdiel je v komponente *XOR*, ktorý bol nahradený komponentom *INV*. Prevrátenie hodnoty čísla sa v pevnej rádovej čiarke použitím doplnkového kódu deje operáciou *XOR* a pričítaním jednotky, ako to je v predchádzajúcom deliacom integrátore. V pohyblivej rádovej čiarke sa však invertovanie hodnoty čísla vykonáva jednoduchým invertovaním znamienkového bitu *S* vstupného operandu. Toto invertovanie sa vykonáva v spomínanom komponente *INV*.



Obr. 5.7: Paralelno-paralelný deliaci integrátor v pohyblivej rádovej čiarke

## 5.7 Sústava diferenciálnych rovníc

Integrátory, ktoré sme si popísali, je možné medzi sebou zapojiť, a riešiť tak diferenciálnu rovnicu. Po zapojení integrátorov nebude nutné predvypočítať vstupné hodnoty, stačí len zadať počiatočné podmienky na naštartovanie výpočtu a všetky ostatné hodnoty sa dopočítajú. Zvoľme konkrétnu diferenciálnu rovnicu s operáciou násobenia v tvare

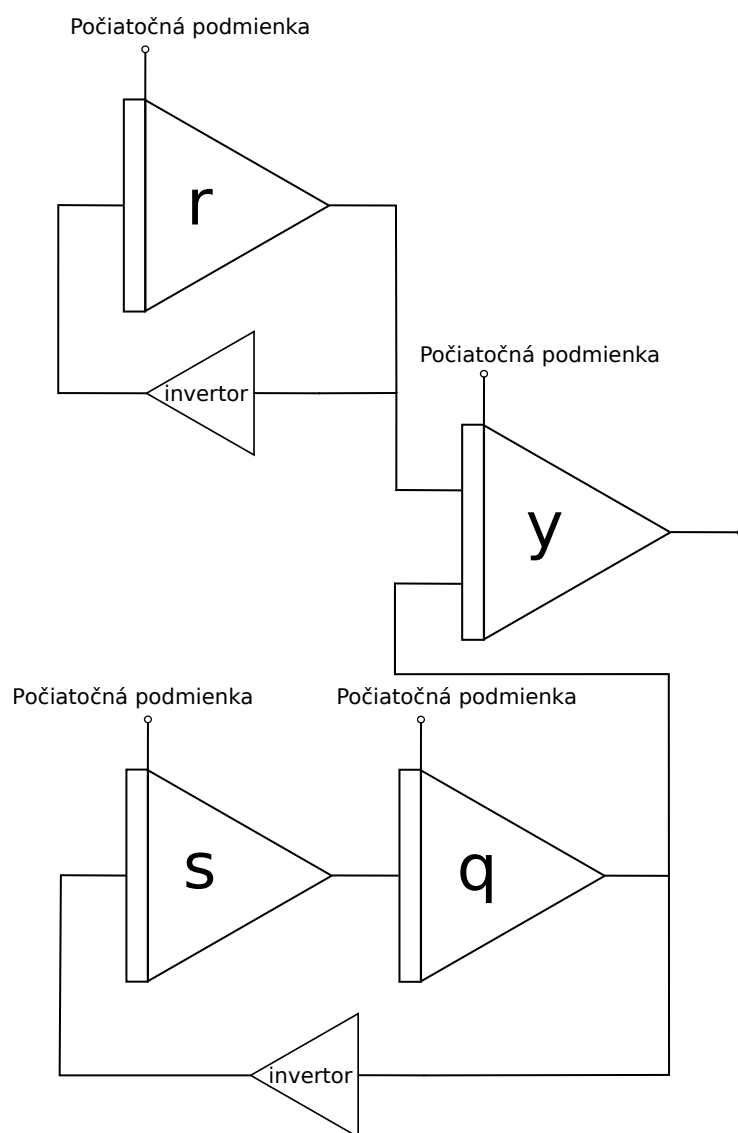
$$y' = \sin(t)e^{-t}, \quad y(0) = 1. \quad (5.1)$$

Rovnicu prevedieme do sústavy diferenciálnych rovníc:

$$\begin{aligned} y' &= qr, & y(0) &= 1 \\ r' &= -r, & r(0) &= 1 \\ q' &= s, & q(0) &= 0 \\ s' &= -q, & s(0) &= 1 \end{aligned} \quad (5.2)$$

Zo sústavy rovníc vytvoríme schému zapojenia integrátorov, znázornenú na obrázku 5.8. Schému tvoria tri jednovstupové integrátory  $r$ ,  $q$  a  $s$ , dvojevstupový násobiaci integrátor  $y$  a dva invertory. Invertor je komponent, ktorý invertuje prichádzajúcu hodnotu. Integrátor  $r$

a invertor tvoria spolu funkciu  $e^{-t}$ . Integrátory  $r$ ,  $s$  a invertor tvoria spolu funkciu  $\sin(t)$ . Nakoniec sú tieto hodnoty privádzané na vstup násobiaceho integrátora  $y$ , ktorý počíta diferenciálnu rovnicu 5.1. TODO rovnicu s deliacim integrátorom. Zo schémy zapojenia je možné následne vytvoriť VHDL kód a vykonať výpočet na FPGA. Takýto postup je vhodný pre rozsiahle diferenciálne rovnice s väčším počtom operácií, ktorých výpočet by bol vykonávaný veľmi často. Ako vhodnejší spôsob použitia sa ponúka prepojavacia sieť medzi rôznymi integrátormi. Pred výpočtom by sa sieť nakonfigurovala podľa danej diferenciálnej rovnice a výpočet by mohol prebiehať bez nutnosti syntézy. Tento variant je síce priestorovo náročný, ale ponúka variabilitu výpočtu. V ďalšej kapitole si predstavíme implementáciu popísaných integrátorov a taktiež implementáciu sústavy diferenciálnych rovníc pomocou metódy bez použitia prepojovacej siete.



Obr. 5.8: Schéma zapojenia integrátorov [4]

## Kapitola 6

# Popísanie integrátorov vo VHDL

Jednotlivé integrátory predstavené v kapitole 5 boli popísané v jazyku VHDL v prostredí Xilinx ISE a následne odsimulované v nástroji Model Sim s pomocou použitia *testbenchu*. Potom boli niektoré z integrátorov otestované na programovateľnom hradlovom poli FPGA. Pre priestorovú zložitosť paralelných integrátorov bola snaha implementovať integrátory na rozsiahlejšom čipe FPGA, a to konkrétne na VIRTEX-5 s čipom XC5VSX50T. Pre prácu s týmto čipom je potrebná plná verzia programu Xilinx ISE s licenciou. Poskytnutie licencie od VUT FIT pre študenta však nie je možné. Riešením nakoniec bolo použitie licencie pomocou pripojenia sa na fakultnú sieť cez VPN. Následne bol integrátor implementovaný na VIRTEX-5, avšak skončenie výpočtu bolo možné zistiť len rozsvietením diódy, čo je nepoužiteľný spôsob na získanie hodnoty výsledku. Preto bolo snahou výsledok zobrazit na LCD displeji, prípadne vytvoriť komunikáciu cez terminál. Sprevádzkovanie displeja podľa rôznych návodov bolo neúspešné a pre komunikáciu s terminálom bolo potrebné doinštalovať ďalšie programy. Vzhľadom na dané komplikácie bolo teda nutné použiť iný čip, a to konkrétne FPGA Spartan 3 XC3S50, ktorý je použitý v študijnej pomôcke FitKit 2.0. Veľkosť tohto FPGA je však veľmi malá, a ani jeden z navrhnutých integrátorov nie je možné naprogramovať na tento čip. Po konzultácii s Ing. Václavom Šimekom sme začali pracovať s prípravkom FitKit 2.0 s vymeneným FPGA za väčší, a to za Spartan 3 XC3S400. Práca s týmto FPGA je kompatibilná s predchádzajúcim čipom, a preto bolo možné na programovanie použiť nástroj QDevKit. Ide o multiplatformový terminálový program pre jednoduchú prácu s FitKitom. Všetky parametre sú už v programe nastavené a komunikácia prebieha cez USB namiesto emulovaného sériového portu, ako je to pri iných terminálových programoch. Informácie o FitKite, o programe QDevKit a taktiež návody na vytvorenie vlastnej aplikácie boli čerpané z webstránky FitKitu [13]. Na FitKite bol odskúšaný jednovstupový integrátor s 32 bitovou floating point aritmetikou. Tento integrátor bol vybraný pre jeho najmenšiu priestorovú zložitosť. Ďalej bol implementovaný integrátor počítajúci rovnicu  $xx$ . Integrátor pracuje na 32 bitoch a používa FX aritmetiku. Rovnica je počítaná Taylorovou radou 4. rádu. Ostatné integrátory a verzie integrátorov nie je možné nahradiť na FPGA, ktoré je použité vo FitKite.

Jednotlivé časti integrátorov boli popísané samostatne a následne bola vytvorená štruktúra vzájomného prepojenia, ktorá tvorí celý integrátor. Pri každom integrátore sú jednotlivé kroky výpočtu riadené kontrolórom. Ten je tvorený cyklickým konečným automatom. Cyklus vyplýva z rovníc (3.15) a (3.26), keďže výpočet jednotlivých členov je podobný, avšak so zväčšujúcim sa počtom operácií. Samozrejme, po skončení výpočtu je možné pomocou signálu *RESET* prejsť do počiatočného stavu a zadaním novej počiatočnej podmienky začať nový výpočet. Ďalej podrobnejšie popíšeme jednotlivé integrátory.



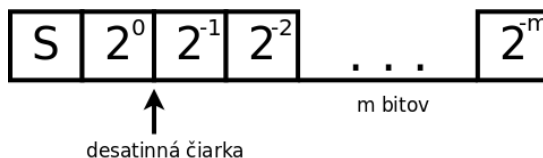
## 6.1 Integrátory v pevnej rádovej čiarke

Všetky integrátory v pevnej rádovej čiarke (t. j. deliaci, násobiaci a jednovstupový integrátor) kódujú čísla v dvojkovom kóde. Použité sčítačky a násobičky sú paralelné a výpočet operácie trvá jeden hodinový takt. Pri výpočte je však potrebné použiť dva takty, aby došlo k ustáleniu signálov: jeden na nastavenie signálov a druhý na uloženie výsledku.

### Násobiaci integrátor

Paralelný dvojjstupový násobiaci integrátor vo fixpoint aritmetike bol implementovaný v dvoch verziách, a to na 32 a 64 bitoch. Veľkosť použitej aritmetiky je však možné jednoducho zväčšiť, keďže celý kód je písaný genericky. Taktiež je možné zmeniť počet cyklov výpočtu, čiže počet počítaných členov Taylorovej rady, avšak v závislosti od počtu cyklov  $N$  je nutné upraviť počet registrov v registrových setoch. Pri násobiacom integrátore sú to sety registrov  $DQ(N-1)$ ,  $DR(N-1)$  a počet registrov  $h/i$ , kde  $i = 2..N$ . Pri deliacom integrátore je potrebné zväčšiť počet registrov  $DV(N-1)$ ,  $DU(N-1)$  a  $1/i$ , kde  $i = 2..N$ . S tým súvisí potreba následného zväčšenia bitovej šírky riadiacich signálov označovaných s prefixom *SEL*. Bitovú šírku signálov je vhodné zvoliť z počtu  $N$  registrov podľa vzťahu *šírka registrov*  $= \log_2 N$  a využiť tak celú šírku a kombinácie binárneho kódovania.

Pre 32 i 64 bitovú verziu bola použitá FX aritmetika, ktorá je názorne zobrazená na obrázku 6.1. Aritmetika je tvorená s jedným znamienkovým bitom, s 1 bitom pred desatinnou čiarkou a s  $m$  bitmi za desatinnou čiarkou. Pri jednoduchšej presnosti sa  $m$  rovná 30 bitom a pri dvojitej presnosti sa  $m$  rovná 62 bitom. Výpočet počítá s Taylorovou metódou 8. rádu, a teda výpočet prebieha až po 8. člen Taylorovej rady, t. j. DY8.



Obr. 6.1: FX aritmetika numerického integrátora.

Násobiaci integrátor je ovládaný pomocou kontrolóra. Jednotlivé stavy kontrolóra a nastavenia signálov sú uvedené v tabuľke 6.1. Kontrolór obsahuje 9 stavov a dva čítače *count* a *countQ*. Čítač *count* udáva číslo aktuálneho cyklu a *countQ* je v každom cykle nastavený na hodnotu *count* a postupne sa dekrementuje. Výpočet začína nastavením signálu *reset* na logickú 1. To spôsobí prepnutie násobiaceho integrátora do stavu s rovnakým názvom, čiže *Reset*. V ňom sa integrátor uvedie do počiatočného stavu. Vynulujú sa počítadlá a čaká sa na povoľovací signál *EN*, ktorým sa prepne integrátor do stavu *Init*. V tomto stave je nahraná počiatočná podmienka a sú uložené vstupné hodnoty. Výpočet prejde do stavu *MulDQxDRx*. V ňom sú vynásobené jednotlivé členy  $DQi$  a  $DRj$ , kde  $i, j$  značí aktuálne počítaný člen. Z rovnice 3.15 vyplýva, že sa vykonáva násobenie kombinácií týchto členov. Na to slúžia spomenuté čítače *count* a *countQ*. Na základe čítačov sa nastaví signály *SELR* a *SELQ*. Po vynásobení členov je výsledok v stave *WriteAcc* uložený do akumulátora. Zo stavu *WriteAcc* je integrátor prepnutý buď do stavu *SetupCounter*, alebo do stavu *MulAccH\_RV*. V stave *SetupCounter* sa dekrementuje počítadlo *countQ* a začína sa násobenie ďalšej kombinácie členov  $DQi$  a  $DQj$ . Ak sú vynásobené všetky kombinácie týchto členov, stačí potom už

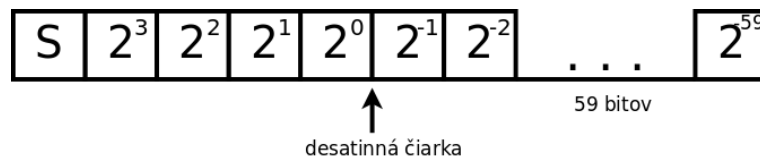
len vynásobiť obsah akumulátora hodnotou  $h/N$ . Integrátor je teda prepnutý do stavu *MulAccH\_RV* a vykoná sa operácia násobenia. Výsledok je uložený do registra *RD* (nejde o variantu "posledného integrátora v zapojení"). Potom je v stave *SaveToAcc* sčítaná hodnota s hodnotou registra *RV* a následne je uložená do akumulátora. V ďalšom stave *SaveToRv* je výsledok v *ACC* uložený do registra *RV*. Výpočet končí v stave *Result*, v ktorom je nastavený signál *OEN* (output enable) na logickú jednotku. Po nastavení povoloacieho signálu *EN* na logickú jednotku je výpočet znova spustený s novými hodnotami.

### Deliaci integrátor

Paralelný deliaci integrátor vo fixpoint aritmetike bol implementovaný, podobne ako násobiaci integrátor, v 32 a 64 bitovej verzii. Pri návrhu integrátorov sa kládol dôraz nielen na rýchlosť výpočtu, ale aj na priestorovú zložitosť. Z rovnice 3.26 vyplýva, že členy *DYN* sú násobené číslom  $N - k$ , kde  $k$  je minimálne 1, a teda najväčšie číslo je  $N - 1$ . Aby bolo možné toto číslo zobraziť, bolo potrebné zvoliť vhodnú aritmetiku, t. j. vhodný počet čísel pred a za desatinnou čiarkou. Čím väčší počet bitov sa nachádza pred desatinnou čiarkou, tým viac sa zväčšuje interval zobraziteľných čísel, avšak znižuje sa počet bitov za desatinnou čiarkou. Pri znižovaní počtu čísel za desatinnou čiarkou sa znižuje hustota zobrazenia čísel a v konečnom dôsledku aj ich presnosť. V dvojkovom doplnkovom kóde pri 1 bite pred desatinnou čiarkou sme schopný vypočítať deliacim integrátorom maximálne dva členy Taylorovej rady. Pri dvoch bitoch sú to 4 členy, pri troch bitoch 8 členov Taylorovej rady a pri štyroch bitoch 16 členov Taylorovej rady. To je možné vyjadriť ako  $N = 2^w$ , kde  $N$  - počet členov,  $w$  - bitová šírka. Keďže pri použití Taylorovej metódy 4. rádu nemusí byť riešenie dostatočne presné, 32 bitová verzia integrátora počíta s Taylorovou metódou 8. rádu, a teda je použitá aritmetika s jedným znamienkovým bitom, s tromi bitmi pred desatinnou čiarkou. Ostatných 28 bitov tvorí časť za desatinnou čiarkou. Vyšší rád metódy nie je použiteľný, keďže hodnoty vyšších členov by boli veľmi malé, a teda nezobraziteľné v danej aritmetike. Táto aritmetika je zobrazená na obrázku 6.2.



Obr. 6.2: FX aritmetika deliaceho integrátora na 32 bitoch.



Obr. 6.3: FX aritmetika deliaceho integrátora na 64 bitoch.

V 64 bitovej verzii deliaceho integrátora ponúka väčšiu bitovú šírku a teda aj presnosť, preto bola veľkosť použitých bitov pred desatinnou čiarkou zväčšená o jeden bit. Za desatinnou čiarkou sa potom nachádza 59 bitov. V tejto verzii deliaceho integrátora sme teda

schopný vypočítať v danej aritmetike až 16 členov Taylorovej rady. Použitá aritmetika je na obrázku 6.3.

Postup výpočtu delaiceho integrátora je podobný ako v násobiacom integrátore. Jendotlivé stavy kontroléra a nastavenie signálov sú uvedené v tabuľke 6.3. Začiatok výpočtu je rovnaký ako u integrátora s operáciou násobenia. Integrátor je v počiatočnom stave t.j. v stave *Reset*. Z neho prejde do stavu *Init*. Následne v stave *MulUH* je násobená hodnota člena  $DU_i$  a s integračným krokom  $h$ . Výsledok je uložený do akumulátora *ACC*. Výsledok násobenia je možné uložiť už v tomto kroku, keďže nastavenie signálov multiplexorov a výpočet násobenia bolo možné začať už v stave *Init*. Paralelne s operáciou násobenia je signálom *START* zahájené delenie  $1/v$ , kde  $v$  je vstupná hodnota operandu. Operácia delenia sa vykonáva pomocou SRT algoritmu s radixom 2. Bližšie informácie o danom algoritme a implementácii môžeme nájsť v [4]. V stave *WaitDiv* je integrátor niekoľko cyklov, až kým sa neskončí operácia delenia. Delička po vykonaní operácie delenia nastaví výstupný signál *done* na 1, čím umožní prechod integrátora do nasledujúceho stavu *SaveDiv*. V ňom je hodnota delenia uložená do registra  $1/v$ . Tým dostaneme hodnotu prvého člena Taylorovej rady. Hodnota je sčítaná s hodnotou registra *RV* a uložená naspäť do registra.

| Stav         | SELy0 | SEL1 | SEL2 | SEL3 | SELQ   | SELR         | SELH  | RWacc | RWrv | BLOK                    |  |
|--------------|-------|------|------|------|--------|--------------|-------|-------|------|-------------------------|--|
| Reset        | 0     | 0    | 0    | 0    | 0      | 0            | 0     | 0     | 0    | 0                       | count = 0;<br>countQ = 0;                    |
| Init         | 1     | X    | X    | X    | X      | X            | X     | 0     | 1    | X                       | count = 0;<br>countQ = 0;                    |
| MulDQxDRx    | X     | 1    | 0    | 0    | countQ | count-countQ | X     | 0     | 0    | If countQ=count ? 1 : 0 | If count <N ?<br>WriteToAcc : Result         |
| WriteToAcc   | X     | 1    | 0    | 0    | countQ | count-countQ | X     | 1     | 0    | If countQ=count ? 1 : 0 | If countQ = 0 ?<br>MulAccH_RV : SetupCounter |
| SetupCounter | X     | X    | X    | X    | X      | X            | X     | 0     | 0    | X                       | countQ-;                                     |
| MulAccH_RV   | X     | 0    | 1    | 1    | X      | X            | count | 0     | 0    | X                       |  |
| SaveToAcc    | X     | 0    | 1    | 1    | X      | X            | count | 1     | 0    | X                       | countQ=count+1;                              |
| SaveToRv     | 0     | 0    | X    | X    | X      | X            | X     | 0     | 1    | X                       | count++;                                     |
| Result       | X     | X    | X    | X    | X      | X            | X     | 0     | 0    | X                       | If en = 0 ?<br>Result : Init                 |

Tabulka 6.1: Kontrolér násobiaceho integrátora v FX

| Stav                                     | SELy0 | SEL1 | SEL2 | SEL3 | SELQ   | SELR         | SELH  | RWacc | RWrv | BLOK                       |   |
|--|-------|------|------|------|--------|--------------|-------|-------|------|----------------------------|---|
| Reset                                    | 0     | 0    | 0    | 0    | 0      | 0            | 0     | 0     | 0    | 0                          |   |
| Init                                     | 1     | X    | X    | X    | X      | X            | X     | 0     | 1    | X                          | count = 0;<br>countQ = 0;                     |
| MulDQxDRx1<br>MulDQxDRx2                 | X     | 1    | 0    | 0    | countQ | count-countQ | X     | 0     | 0    | If countQ=count ?<br>1 : 0 |   |
| MulDQxDRx                                | X     | 1    | 0    | 0    | countQ | count-countQ | X     | 0     | 0    | If countQ=count ?<br>1 : 0 | If count <8 ?<br>WriteToAcc1 : Result         |
| WriteToACC1                              | X     | 1    | 0    | 0    | countQ | count-countQ | X     | 0     | 0    | If countQ=count ?<br>1 : 0 |   |
| WriteToACC                               | X     | 1    | 0    | 0    | countQ | count-countQ | X     | 1     | 0    | If countQ=count ?<br>1 : 0 | If countQ = 0 ?<br>MulAccH_RV1 : SetupCounter |
| SetupCounter                             | X     | X    | X    | X    | X      | X            | X     | 0     | 0    | X                          | countQ-;                                      |
| MulACCh_RV1<br>MulACCh_RV2<br>MulACCh_RV | X     | 0    | 1    | 1    | X      | X            | count | 0     | 0    | X                          |   |
| SaveToAcc1                               | X     | 0    | 1    | 1    | X      | X            | count | 0     | 0    | X                          |   |
| SaveToAcc                                | X     | 0    | 1    | 1    | X      | X            | count | 1     | 0    | X                          | countQ=count+1;                               |
| SaveToRV                                 | 0     | 0    | X    | X    | X      | X            | X     | 0     | 1    | X                          | count++;                                      |
| Result                                   | X     | X    | X    | X    | X      | X            | X     | X     | X    | X                          | If en = 0 ?<br>Result : Init                  |

Tabulka 6.2: Kontrolér násobiaceho integrátora v FP

| Stav           | SEL2  | SEL3      | SEL4        | SEL5              | SEL6                        | SEL7 | SEL8 | ClkAcc | ClkY | BLOK   |   |
|----------------|-------|-----------|-------------|-------------------|-----------------------------|------|------|--------|------|--|---|
| Reset          | 000   | 00        | 000         | 000               | 00                          | 0    | 0    | 0      | 0    | 0  | count = 1;<br>const_int = 1;                              |
| Init           | 000   | 00        | X           | 000               | 01                          | 0    | X    | 0      | 1    | 1  |   |
| MulUH          | 000   | 00        | X           | 000               | 01                          | 0    | X    | 0      | 0    | 1  |   |
| WaitDiv        | 000   | X         | X           | X                 | 00                          | X    | X    | 0      | 0    | X  | done = 1 ?<br>SaveDiv : WaitDiv;                          |
| SaveDiv        | 000   | X         | X           | 001               | 00                          | 1    | 0    | 0      | 0    | 0  |   |
| CalculationDY1 | 000   | X         | X           | 001               | 00                          | 1    | 0    | 0      | 1    | 0  |   |
| WriteToAcc     | 001   | X         | X           | const_int+1       | const_int > 0 ?<br>10 : 00; | X    | X    | 0      | 0    | 1  | RWr = 1;<br>count < 7 ?<br>SetupCounters : Result;        |
| SetupCounters  | 001   | X         | X           | X                 | const_int > 0 ?<br>10 : 00; | 0    | X    | 0      | 0    | 1  | count = 1;<br>const_int > 0 ?<br>MulCountDYx : MulDYxDVx; |
| MulCountDYx    | X     | X         | X           | const_int         | 10                          | 0    | X    | 0      | 1    | 1  |   |
| SetupDYxDVx    | count | X         | X           | const_int-count+1 | 00                          | 0    | 1    | 0      | 0    | count = 1 ?<br>1 : 0;                          |   |
| MulDYxDVx      | count | X         | X           | const_int-count+1 | 00                          | 0    | 1    | 1      | 0    | count < const_int ?<br>SetupCount : SetupDUxH; |   |
| SetupCount     | X     | X         | X           | X                 | X                           | X    | X    | 0      | 0    | X  | count++;  |
| SetupDUxH      | X     | const_int | X           | 000               | 01                          | 0    | 1    | 0      | 0    | 0  | XOR = 1;  |
| MulDUxH        | X     | const_int | X           | 000               | 01                          | 0    | 1    | 1      | 0    | 0  | XOR = 1;  |
| SetupACCdivX   | X     | X         | const_int-1 | X                 | 11                          | 1    | X    | 0      | 0    | 1  |   |
| MulACCdivX     | X     | X         | const_int-1 | X                 | 11                          | 1    | X    | 1      | 0    | 1  |   |
| SetupCalcDYx   | 000   | X         | X           | const_int+1       | 00                          | 1    | 0    | 0      | 0    | 0  |   |
| CalculationDYx | 000   | X         | X           | const_int+1       | 00                          | 1    | 0    | 1      | 1    | 0  |   |
| Result         | X     |           | X           | X                 | X                           | X    | X    | 0      | 0    | X  | en = 0 ? Result : Init;                                   |

Tabulka 6.3: Kontrolér deliaceho integrátora v FX

| Stav           | SEL2  | SEL3      | SEL4        | SEL5              | SEL6                        | SEL7 | SEL8 | ClkAcc | ClkY | BLOK               |  |
|----------------|-------|-----------|-------------|-------------------|-----------------------------|------|------|--------|------|--------------------|--|
| Reset          | 000   | 00        | 000         | 000               | 00                          | 0    | 0    | 0      | 0    | 0                  | count = 1;<br>const_int = 1;                               |
| Init           | 000   | 00        | X           | 000               | 01                          | 0    | X    | 0      | 1    | 1                  |  |
| MulUH          | 000   | 00        | X           | 000               | 01                          | 0    | X    | 0      | 0    | 1                  |  |
| MulUH1         | 000   | 00        | X           | 000               | 01                          | 0    | X    | 0      | 0    | 1                  |  |
| SaveDiv        | 000   | X         | X           | 001               | 00                          | 1    | 0    | 0      | 0    | 0                  | ClkV = 1;  |
| SaveAcc        | 000   | X         | X           | 001               | 00                          | 1    | 0    | 1      | 0    | 0                  |  |
| CalculationDY1 | 000   | X         | X           | 001               | 00                          | 1    | 0    | 0      | 1    | 0                  |  |
| CalculationDY2 | 00    | X         | X           | 001               | 00                          | 1    | 0    | 0      | 0    | 0                  |  |
| CalculationDY3 | 00    | X         | X           | 001               | 00                          | 1    | 0    | 1      | 0    | 0                  |  |
| WriteToAcc     | 001   | X         | X           | const_int+1       | const_int > 0 ?<br>10 : 00; | X    | X    | 0      | 0    | 1                  | RWrV = 1;<br>count < (N-1) ?<br>SetupCounters : Result;    |
| SetupCounters  | 001   | X         | X           | X                 | const_int > 0 ?<br>10 : 00; | 0    | X    | 0      | 0    | 1                  | count = 1;<br>const_int > 0 ?<br>MulCountDYx : MulDYxDVx;  |
| SetupCountDYx  | X     | X         | X           | const_int         | 10                          | 0    | X    | 0      | 0    | 1                  |  |
| MulCountDYx    | X     | X         | X           | const_int         | 10                          | 0    | X    | 0      | 1    | 1                  |  |
| SaveCountDYx   | X     | X         | X           | const_int         | 10                          | 0    | X    | 0      | 1    | 1                  |  |
| SetupDYxDVx    | X     | X         | X           | const_int         | 10                          | 0    | X    | 0      | 1    | 1                  |  |
| SetupDYxDVx1   | count | X         | X           | const_int-count+1 | 00                          | 0    | 1    | 0      | 0    | count = 1 ? 1 : 0; |  |
| MulDYxDVx      | count | X         | X           | const_int-count+1 | 00                          | 0    | 1    | 1      | 0    | count = 1 ? 1 : 0; |  |
| SaveDYxDVx     | count | X         | X           | const_int-count+1 | 00                          | 0    | 1    | 1      | 0    | count = 1 ? 1 : 0; |  |
| SetupCount     | X     | X         | X           | X                 | X                           | X    | X    | 0      | 0    | X                  | count < const_int ?<br>SetupCount : SetupDUxH;<br>count++; |
| SetupDUxH      | X     | const_int | X           | 000               | 01                          | 0    | 1    | 0      | 0    | 0                  | XOR = 1;   |
| MulDUxH        | X     | const_int | X           | 000               | 01                          | 0    | 1    | 0      | 0    | 0                  |  |
| SumMulAcc      | X     | const_int | X           | 000               | 01                          | 0    | 1    | 0      | 0    | 0                  |  |
| SaveMulAcc     | X     | const_int | X           | 000               | 01                          | 0    | 1    | 0      | 0    | 0                  |  |
| SetupACCDivX   | X     | X         | const_int-1 | X                 | 11                          | 1    | X    | 1      | 0    | 1                  |  |
| MulACCDivX     | X     | X         | const_int-1 | X                 | 11                          | 1    | X    | 0      | 0    | 1                  |  |
| MulACCDivX1    | X     | X         | const_int-1 | X                 | 11                          | 1    | X    | 0      | 0    | 1                  |  |
| SumACCDivX     | X     | X         | const_int-1 | X                 | 11                          | 1    | X    | 1      | 0    | 1                  |  |
| SaveACCDivX    | X     | X         | const_int-1 | X                 | 11                          | 1    | X    | 1      | 0    | 1                  |  |
| SetupCalcDYx   | 000   | X         | X           | const_int+1       | 00                          | 1    | 0    | 0      | 0    | 0                  |  |
| SetupCalcDYx1  | 000   | X         | X           | const_int+1       | 00                          | 1    | 0    | 1      | 1    | 0                  |  |
| CalculationDYx | X     | X         | X           | X                 | X                           | X    | X    | 0      | 0    | X                  | en = 0 ?<br>Result : Init;                                 |

Tabulka 6.4: Kontrolér deliaceho integrátora v FP.

## 6.2 Integrátory v pohyblivej rádovej čiarke

Integrátory v pohyblivej rádovej čiarke sú implementované vo verzií bez zdieľanej komponenty *EXP*. Delička *DIV*, násobička *MUL* a sčítačka *SUM* spracovávajú znamienko *S*, exponent *E* a mantisu *M* vnútri seba. Konkrétna implementácia týchto komponent bola prevzatá a upravená z [1]. Komponenty vykonávajú operáciu násobenia v 3 taktoch a operáciu sčítania v dvoch taktoch, keďže pracujeme v pohyblivej rádovej čiarke a výpočet je náročnejší. Podrobný popis prebiehania týchto operácií v pohyblivej rádovej čiarke je popísaný v kapitole 4. Výpočet je teda pomalší oproti variante pracujúcej s pevnou rádovou čiarkou, ale o to presnejší a zároveň sa veľmi zväčšil rozsah zobraziteľných hodnôt a hustota zobrazenia.

### Jednovstupový integrátor

Jednovstupový integrátor v pohyblivej rádovej čiarke bol implementovaný v 32 a v 64 bitovej verzií. Počet cyklov výpočtu je závislý len od počtu registrov v sete  $h/N$ . Na riadenie integrátora rovnako ako v predchádzajúcich variantách sú použité signály *RESET* a *EN*. Ostatné signály a stavy konečného automatu v kontroléry sú zobrazené v tabuľke 6.5. Pri výpočte je použitý jeden čítač. Ten slúži na počítanie jednotlivých cyklov výpočtu a zároveň ako signál pre výber hodnoty zo setu registrov  $h/N$ . Výpočet je naštartovaný signálom *Reset*. V ňom sa uložia počiatočné podmienky do registrov *RD* a *RV*. Následne sa pokračuje nasledujúcimi stavmi *Init*, *MulDYxH*, *SetupToRD* v ktorých prebieha násobenie a začiatok výpočtu sčítania. V stave *SaveToRD* je vypočítaná hodnota nového člena Taylorovej rady uložená do registra *RD*. V ďalšom stave *SaveToRV* je už hodnota člena sčítaná s hodnotou *RV* a naspäť uložená do registra *RV*. Výpočet *N* krát cykli a prejde do stavu *Result*. V registri *RV* sa nachádza výsledná hodnota rovnice 3.1.

| Stav             | SELy0 | SEL | SELh  | RWrđ | RWrv |                                 |
|------------------|-------|-----|-------|------|------|---------------------------------|
| <b>Reset</b>     | 1     | 0   | 0     | 1    | 1    |                                 |
| <b>Init</b>      | X     | X   | count | 0    | 0    |                                 |
| <b>MulDYxH</b>   | X     | X   | count | 0    | 0    |                                 |
| <b>SetupToRD</b> | 0     | 1   | count | 0    | 0    |                                 |
| <b>SaveToRD</b>  | 0     | 1   | count | 1    | 0    | count++                         |
| <b>SaveToRV</b>  | X     | X   | count | 0    | 1    | If count < 8 ?<br>Init : Result |
| <b>Result</b>    | X     | X   | X     | 0    | 0    | If en = 0 ?<br>Result : Init    |

Tabuľka 6.5: Kontrolér jednovstupového integrátora v FP

### Deliaci a násobiaci integrátor

Implementovaný bol aj násobiaci a deliaci integrátor v pohyblivej rádovej čiarke a taktiež oba vo verziách s 32 bitmi a 64 bitmi. Jednotlivé stavy a nastavenia signálov násobiaceho integrátora sú popísané v tabuľke 6.2. Pri porovnaní s tabuľkou násobiaceho integrátora v pevnej rádovej čiarke 6.1 môžeme vidieť, že výpočet oboch integrátorov prebieha rovnako. Rozdiel je v dĺžke vykonávania operácie násobenia, ktorá potrebuje o dva hodinové takty viac ako varianta s FX aritmetikou. Operácia sčítania potrebuje jeden takt navyše.

Popis kontroléru deliaceho integrátora je v tabuľke 6.4. Pri porovnaní s tabuľkou deliaceho integrátora vo FX aritmetike 6.3 je vidieť, že výpočet prebieha podobne v oboch typoch integrátorov. Rozdiel je rovnako ako u násobiacich integrátorov, v dĺžke vykonávania operácií delenia, násobenia a sčítania. Násobenie sa vykonáva v troch taktov hodinového signálu a sčítanie v dvoch taktov (rovnako ako u násobiaceho integrátora). Zaujímavosťou je však operácia delenie. Vo FP aritmetike je použitá paralelná delička používajúca paralelný algoritmus bez návratu k nezápornému zvyšku. Dĺžka výpočtu trvá iba jeden hodinový takt.

### 6.3 Sústava integrátorov

Podľa schémy zapojenia 5.8 v sekcii 5.7 bola implementovaná komponenta počítajúca diferenciálnu rovnicu 5.1. Komponentu tvorí násobiaci a deliaci integrátor v pohyblivej rádovej čiarke so 64 bitmi. Ďalej bola použitá komponenta invertor. Každý integrátor v zapojení používa set registrov  $h/N$ . Pre zníženie priestorovej náročnosti bol použitý len jeden set týchto registrov. Veľkosť setu závisí od rádu metódy, ktorú chceme použiť. Výpočet bol implementovaný v dvoch variantách počítajúcich s Taylorovou metódou 8. rádu a s Taylorovou metódou až 32. rádu. Každý integrátor je riadený svojim vlastným kontrolérom. Celé zapojenie je ešte ovládané hlavným kontrolérom, ktorý ovláda ostatné a to pomocou signálov *Reset* a *EN*. Výpočet prebieha v každom integrátore samostatne spôsobom takmer rovnakým aký bol popísaný v predchádzajúcich sekciiach tejto kapitoly. Rozdiel je že integrátory nepracujú úplne ako samostatné komponenty s predpočítanými hodnotami, ale v zapojení do sústavy integrátorov kde vstupné hodnoty prichádzajú postupne. Upravené boli niektoré kroky kontroléra u použitých integrátorov. Upravený kontrolér jednovstupého integrátora je v tabuľke 6.6. Rozdiel je, že integrátor neobsahuje signál *SELh*, keďže sada registrov bola vytvorená ako zdieľaný komponent medzi všetkými integrátormi. Signál *SELh* je teraz ovládaný pomocou hlavného kontroléra. Ďalej neobsahuje podmienku na ukončenie výpočtu po dosiahnutí stanoveného počtu cyklov. Tá je taktiež v tomto prípade v režii hlavného kontroléra. Rovnakými úpravami prešiel aj kontrolér násobiaceho integrátora. V násobiacom integrátore bol pridaný ešte jeden výstup pre prečítanie výslednej hodnoty z registra *RV*.

| Stav             | SELy0  | SEL | RWrđ | RWrv |
|------------------|--|-----|------|------|
| <b>Reset</b>     | 1  | 0   | 1    | 0    |
| <b>Init</b>      | X  | X   | 0    | 1    |
| <b>MulDYxH</b>   | X  | X   | 0    | 0    |
| <b>SetupToRD</b> | 0  | 1   | 0    | 0    |
| <b>SaveToRD</b>  | 0  | 1   | 1    | 0    |
| <b>SaveToRV</b>  | X  | X   | 0    | 1    |
| <b>Result</b>    | X  | X   | 0    | 0    |
|                  | if ENDyx = 0 ?<br>if en = 0 ? Result : MulDYxh<br>: InitDy |     |      |      |

Tabuľka 6.6: Kontrolér jednovstupového integrátora v FP v zapojení do sústavy

TODO skontrolovať tabuľky a stavy....6.6. savetovv ?? Na začiatku výpočtu je hlavný kontrolér uvedený do stavu *Reset* a signálom privedie ostatné integrátory taktiež do stavu



*Reset.* V tomto stave sú nahrané počiatočné podmienky do integrátorov a privedené na výstup každého integrátora. Hodnota sa tak dostane na potrebné vstupy integrátorov. Následne je pomocou povolovacieho signálu *EN* spustený výpočet na všetkých integrátoroch naraz. Vstupné hodnoty násobačného integrátora sú uložené do sady registrov  $DR(N-1)$  a  $DQ(N-1)$ . Výpočet pokračuje samostatne v každom integrátore. Keďže však výpočet násobiaceho integrátora trvá najdlhšie z použitých integrátorov, ostatné integrátory  $q$ ,  $r$  a  $s$  po dokončení výpočtu čakajú na násobiaci integrátor. Ten po dokončení výpočtu aktivuje signál *OEN* (*output enable*) na 1. Tento signál je privedený do hlavného kontroléru, ktorý následne spustí ďalší cyklus výpočtu nastavením signálu *EN* do 1 v každom integrátore schémy. Toto sa opakuje až kým výpočet jedného kroku metódy neskončí. Pri výpočte ďalšieho kroku výpočtu sú obnovené počiatočné podmienky a výpočet sa znova vykoná.

FPGA na prípravku FitKit 2.0 nie je dostatočne veľké na implementáciu tejto sústavy rovníc predstavenej v sekcii 5.7, preto bola navrhnutá jednoduchšia diferenciálna rovnica v tvare:

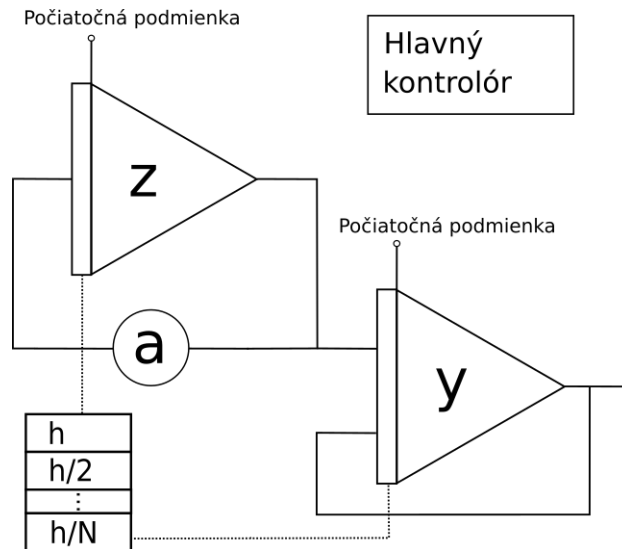
$$y' = y * e^{at} \quad (6.1)$$

Rovniciu môžeme prepísať do sústavy diferenciálnych rovníc:

$$y' = y * z, \quad y(0) = 1 \quad (6.2)$$

$$z' = a * z, \quad z(0) = 1, \text{ kde } a < 0 \quad (6.3)$$

Zo sústavy rovníc vytvoríme schému zapojenia integrátorov, ktorá je na obrázku 6.4. Obsahuje len dva integrátory. Jeden jednovstupý integrátor a jeden násobiaci integrátor. Oba v pevnej rádovej čiarke na 32 bitoch. Pre zníženia miesta je rovnica počítaná s Taylorovou metódou 4. rádu. Komponenta  $a$  predstavuje konštantu, ktorou je násobená výstupná hodnota integrátora  $z$ . Vynásobená hodnota je privedená naspäť na vstup tohto integrátora. Toto zapojenie počíta funkciu  $e^{at}$ . Integrátor  $y$  je zapojený taktiež do smyčky a spolu s integrátorom  $z$  a komponentom  $a$  tvoria schému zapojenia počítajúcu zadanú rovnicu. Do oboch integrátorov je privedená zbernica zo sady registrov  $h$ .



Obr. 6.4: Schéma zapojenia integrátorov

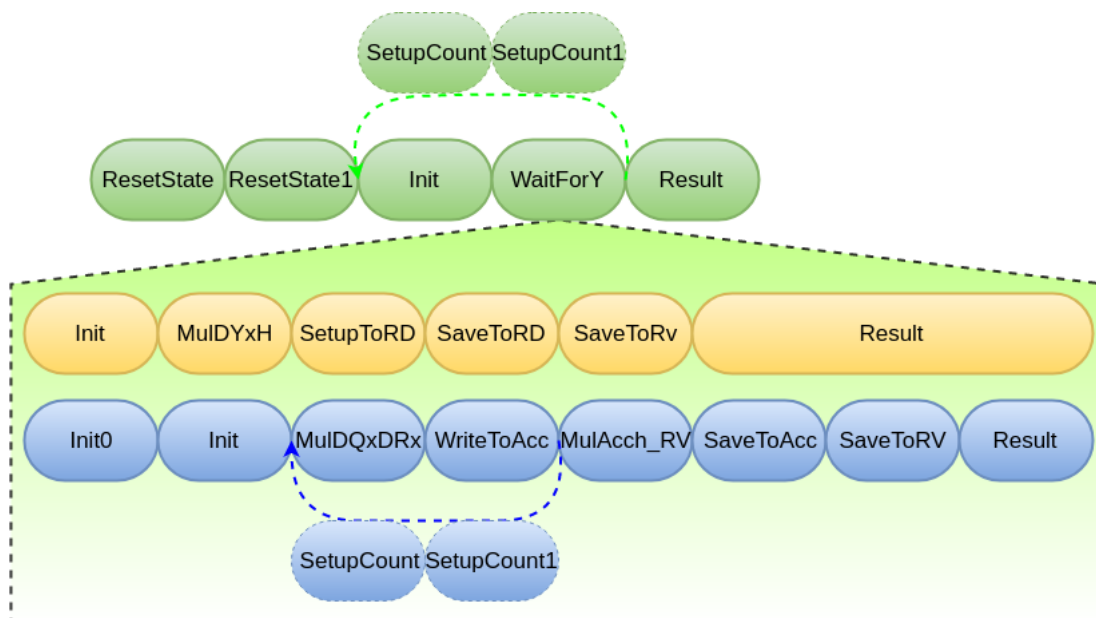
Ovládanie komponentov na prípravku FitKit 2.0 je realizované cez terminál programu QDevKit. Komunikácia medzi MCU (Micro-Controller Unit) a FPGA prebieha pomocou rozhrania *SPI*. Jednotlivé hodnoty zadávané v terminály sú ukladané do pamäti a cez prevodník *SPI\_ADC* privedené do registrov FPGA. Jednotlivé registre sú vybrané adresovým dekoderom. Po inicializácii komunikácie prebieha v nekonečnej smyčke metódou pooling čítanie z riadiaceho registra a z registra výsledku. Hodnoty registrov sú vypisované v terminály v hexadecimálnej forme. Príkazom *help* je možné si zobraziť nápovedu ovládania integrátora. Príkazom *DATAY* je možné nahráť počiatočnú podmienku do integrátora *y* a príkazom *DATAV* do integrátora *v*. Hodnoty počiatočnej podmienky je nutné zadávať v hexadecimálnej forme s veľkými písmenami. Je to kvôli prevodu z textovej formy do číselnej, ktorá je následne uložená do registra. Krok metódy je stanovený na hodnotu *0.1* a nie je možné meniť počas behu aplikácie. Je to z dôvodu nutnosti predpočítania podielov kroku *h*. Tie sú už rpedpočítané v sete registrov. Po zadaní počiatočných podmienok je výpočet možné spustiť príkazom *START*. Tým sa nastaví povolovací signál *EN* a začne výpočet. Následne je výsledok po jednom kroku zobrazený v terminály. Pre výpočet hodnoty v ďalšom kroku je potrebné získaný výsledok zadať ako počiatočnú podmienku násobaiceho integrátora *y*. Novú počiatočnú podmienku je potrebné zadať aj u jednovstupého integrátora, tú však nepoznáme. Jednoduchým a užívateľky príjemným riešením je výsledné hodnoty v oboch integrátorov z registrov *RV* skopírovali do registrov *RD*. Kvôli tomu bol pridaný signál *RvToRd* do jednovstupového integrátora, ktorý umožní presun zjedného registra do druhého bez nutnosti nahrania počiatočnej podmienky príkazom *DATAV*. Presun hodnoty v násobaicom integrátore je realizovaný na úrovni MCU, kde je hodnota získaného výsledku uložená ako počiatočná podmienka. Celý tento postup je skritý za jednoduchým príkazom *NEXTY*.

## Kapitola 7

# Analýza

V tejto kapitole zanalyzujeme dosiahnuté výsledky. Analyzovať môžeme rôzne vlastnosti navrhnutého procesoru a taktiež použitej Taylorovej rady. Najdôležitejšími vlastnosťami sú presnosť výsledku, časová náročnosť a priestorová zložitosť. V tejto analýze sa budeme zaoberať časovou zložitosťou numerických metód a ich presnosťou. Porovnáme Taylorovu radu s metódami nižších rádov, ako sú Eulerova metóda, metóda Runge-Kutta 2. rádu a metóda Runge-Kutta 4. rádu. Tieto metódy sú popísané v kapitole 2.

V prvom rade vypočítame čas procesora, potrebný na výpočet sústavy rovníc 6.2. V každom takte sa procesor nachádza v nejakom stave. Z výpisu jednotlivých stavov kontrolórov tak môžeme vypočítať, koľko stavov je potrebných na výpočet pri použití rôzneho rádu metódy. Výsledný počet stavov potom uvádza koľko taktov procesoru je potrebných na výpočet. Z frekvencie procesora a z počtu taktov následne získame celkový čas výpočtu.



Obr. 7.1: Stavy jednotlivých integrátorov.

Zobrazenie jednotlivých stavov kontrolórov je na obrázku 7.1. Jednotlivé bunky predstavujú stavy kontrolórov. Zelenou farbou je znázornený hlavný kontrolór, ktorý ovláda oba integrátory. Žltou farbou sú znázornené stavy jednovstupového integrátora a modrou

farbou stavy násobiaceho integrátora. Hlavný kontrolór je v stave *WaitForY* počas celého výpočtu integrátorov. Z tohto stavu prejde do ďalšieho až po skončení výpočtu násobiaceho integrátora, keďže jeho výpočet trvá najdlhšie. Ak sa dosiahol maximálny rád metódy, hlavný kontrolór prejde do stavu *Result*. V opačnom prípade výpočet pokračuje stavom *SetupCount* a je začatý nový cyklus výpočtu. Počet taktov  $T_h$  hlavného kontrolóra, a teda celého výpočtu, môžeme vyjadriť vzťahom

$$\begin{aligned} T_h &= 2 + 1 + W + 3(N - 1) + 1 \\ T_h &= 1 + W + 3N, \end{aligned} \quad (7.1)$$

kde  $N$  označuje počet cyklov výpočtu, čiže použitý rád Taylorovej metódy.  $W$  znamená počet stavov (taktov) v stave *WaitForY*. Počet stavov je rovný počtu stavov najdlhšie trvajúcej komponentu v sústave zapojenia. Označme potom  $W$  ako  $W_{jv}$  pre jednovstupový integrátor,  $W_{nas}$  pre násobiaci integrátor a  $W_{div}$  pre deliaci integrátor. Ak je v zapojení len jednovstupový integrátor počet stavov je 5. Celkový čas výpočtu potom vyjadríme nasledovne:

$$\begin{aligned} W_{jv} &= 5N \\ T_{jv} &= (1 + W_{jv} + 3N) * t_{clk} \\ T_{jv} &= (1 + 5N + 3N) * t_{clk} \\ T_{jv} &= 1t_{clk} + 8t_{clk}N. \end{aligned} \quad (7.2)$$

Symbol  $t_{clk}$  udáva čas periódy jedného taktu. Podobný výpočet vykonáme aj pre násobiaci a pre deliaci integrátor. Výpočet násobiaceho integrátora trvá 8 stavov a v každom ďalšom cykle je ich počet zvýšený o štyri. Z toho vyplýva, že

$$\begin{aligned} W_{nas} &= 6N + N(N + 1) + N(N - 1) \\ T_{nas} &= (1 + W_{nas} + 3N) * t_{clk} \\ T_{nas} &= (1 + 6N + N(N + 1) + N(N - 1) + 3N) * t_{clk} \\ T_{nas} &= (1 + 9N + 2N^2) * t_{clk} \\ T_{nas} &= 1t_{clk} + 9t_{clk}N + 2t_{clk}N^2. \end{aligned} \quad (7.3)$$

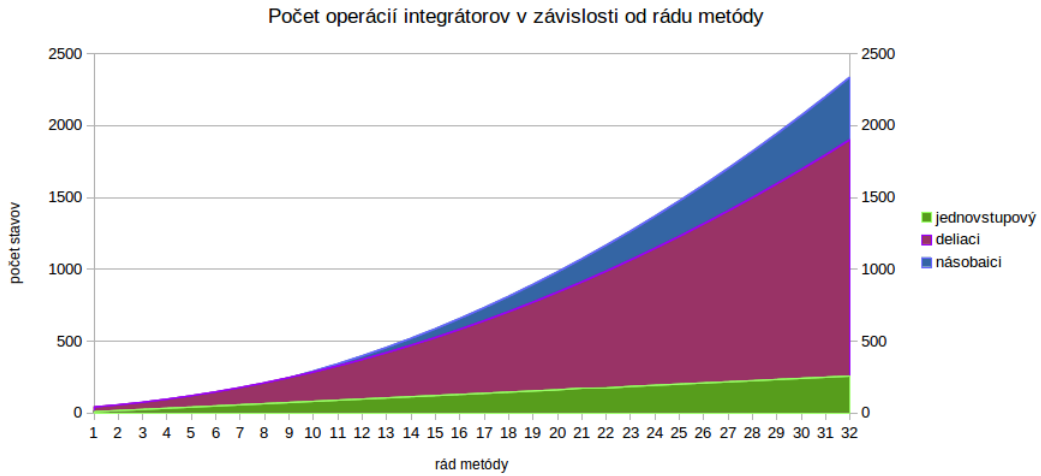
Výpočet počtu krokov deliaceho integrátora je zložitejší. V prvom cykle sa vykonáva delenie a až pri výpočte ďalších členov Taylorovej rady sa násobia predchádzajúce členy  $DVx$  a  $DUx$  vo všetkých kombináciach medzi sebou. Výpočet stavov rozdelíme na počet stavov pre výpočet prvého člena a počet stavov prevýpočet ostatných členov Taylorovej rady. Rovnaké rozdelenie môžeme vidieť aj v konečnom automate deliaceho integrátora z tabuľky xx. Počet stavov  $W_{div_1}$  pre prvý člen Taylorovej rady, počet stavov ostatných členov  $W_{div_x}$  a celkový počet krokov deliaceho integrátora vypočítame nasledovne:

$$\begin{aligned}
W_{div_1} &= 6 + DIV \\
W_{div_x} &= 10(N-1) + N(N-1) + \frac{(N-1)(N-2)}{2} \\
W_{div} &= W_{div_1} + W_{div_x} \\
W_{div} &= 9N - 4 + DIV + N^2 + \frac{N^2 - 3N + 2}{2}
\end{aligned} \tag{7.4}$$

Celkový čas výpočtu je potom

$$\begin{aligned}
T_{div} &= (1 + W_{div} + 3N) * t_{clk} \\
T_{div} &= (1 + 9N - 4 + DIV + N^2 + \frac{N^2 - 3N + 2}{2} + 3N) * t_{clk} \\
T_{div} &= (DIV - 3 + 12N + N^2 + \frac{N^2 - 3N + 2}{2}) * t_{clk}.
\end{aligned} \tag{7.5}$$

Zo vzťahov môžeme vidieť, že časová zložitosť výpočtu pri jednovstupovom integrátore je lineárna, pri násobiacom a deliacom integrátore je kvadratická. Časová zložitosť je znázornená na



Obr. 7.2: Schéma zapojenia integrátorov

Ďalším analyzovaným kritériom je presnosť numerických metód. Na porovnanie presnosti potrebujeme poznať exaktné riešenie problému, ktorý riešime. Rovnica 6.1 má exaktné riešenie nasledovné:

$$y' = e^{\frac{at}{a}} e^{-\frac{1}{a}}, \tag{7.6}$$

Porovnanie jednotlivých metód je zobrazené v tabuľke 7.1. Porovnanie jednotlivých metód spočívalo nájdením maximálneho kroku  $h_{max}$  každej z metód tak, aby rozdiel najvyšších

členov metód v čase  $t_{max}$  bol čo najmenší. Chyba výpočtu Eulerovej metódy bola určená podľa vzťahu:

$$error = |y_{euler}(t_{max}) - y_{exact}(t_{max})| \quad (7.7)$$

Ronvaký postup výpočtu chyby bol použitý aj u ostatných matódach. Výsledný počet operácií (stavov) výpočtu

| metódy          | $h_{max}$ | hodnota najvyššieho člena | $t_{max}$ | výsledok         | error       | počet krokov<br>[ $t_{max}/h_{max}$ ] | počet operácií<br>[ $T * (t_{max}/h_{max})$ ] |
|-----------------|-----------|---------------------------|-----------|------------------|-------------|---------------------------------------|---|
| <b>Euler</b>    | 0.00044   | 3.045E-04                 | 1.00012   | 1.88165390491486 | 25.544E-06  | 2273                                  | 27276   |
| <b>R-K 2</b>    | 0.03750   | -3.075E-04                | 1.01250   | 1.89048695564270 | 272.236E-06 | 27                                    | 729   |
| <b>R-K 4.</b>   | 0.35000   | 3.177E-04                 | 1.05000   | 1.91570568084717 | 45.604E-06  | 3                                     | 168   |
| <b>Taylor 8</b> | 1.00000   | -3.063E-04                | 1.00000   | 1.88219249248505 | 596.105E-06 | 1                                     | 69  |

Tabuľka 7.1: Porovnanie presnosti a rýchlosti numerických metód.

## Kapitola 8

# Záver

V tejto práci sme sa zaoberali numerickou integráciou pomocou metódy Taylorovej rady. Úpravou jej členov sme získali potrebné rovnice, z ktorých boli vytvorené jednotlivé návrhy integrátorov. Rozšírili sme návrhy paralelných integrátorov s operáciou násobenia a delenia v pevnej rádovej čiarky tak, aby s nimi bolo možné počítať diferenciálne rovnice na 20 členov Taylorovej rady. Ďalej sme tieto integrátory navrhli aj v prevedení pohyblivej rádovej čiarky. Navrhli sme komponent slúžiaci na výpočet exponentu a na jeho úpravu pri normalizácii. Keďže výpočet v aritmetike pohyblivej rádovej čiarky je presnejší, integrátory využívajúce túto aritmetiku sú navrhnuté na riešenie diferenciálnych rovníc až na 40 členov Taylorovej rady.

Ďalším pokračovaním práce je popísanie navrhnutých integrátorov vo VHDL, otestovanie ich funkčnosti na VIRTEX5 a následné analyzovanie ich časovej a priestorovej zložitosti v porovnaní s metódou Runge-Kutta.

# Literatúra

- [1] 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](http://www.arithmetic-circuits.org/guide2fpga/vhdl_codes.htm).
- [2] Kraus, M.: *Paralelní výpočetní architektury založené na numerické integraci*. Disertační práce, FIT VUT v Brně, 2013.
- [3] Kunovský, J.: *Modern Taylor Series Method*. Habilitation work, VUT Brno, 1994.
- [4] Matečný, F.: *Simulátor procesora s operáciou delenia*. Bakalářská práce, FIT VUT v Brně, 2016.
- [5] Milan Kubíček, D. J., Miroslava Dubcová: *Numerické metody a algoritmy*. VŠCHT Praha, 2005, ISBN 80-7080-558-7.
- [6] Opálka, J.: *Automatické řízení výpočtu*. Bakalářská práce, FIT VUT v Brně, 2014.
- [7] Opálka, J.: *Automatické řízení výpočtu ve specializovaném výpočetním systému*. Diplomová práce, FIT VUT v Brně, 2016.
- [8] Sekanina, L.: Operace v FP a iterační algoritmy. , 2017, slajdy k predmetu INP - Návrh počítačových systémů.
- [9] Sekanina, L.: Reprezentace dat. , 2017, slajdy k predmetu INP - Návrh počítačových systémů.
- [10] Tišnovský, P.: Fixed point arithmetic [online].  
<http://www.root.cz/clanky/fixed-point-arithmetic/>, 2006-05-24 [cit. 2017-11-22].
- [11] Tišnovský, P.: Norma IEEE 754 a příbuzní: formáty plovoucí řádové tečky [online].  
<https://www.root.cz/clanky/norma-ieee-754-a-pribuzni-formaty-plovouci-radove-tecky/>, 2006-05-31 [cit. 2017-11-22].
- [12] Tišnovský, P.: Aritmetické operace s hodnotami ve formátu plovoucí řádové čárky [online].  
<https://www.root.cz/clanky/aritmeticke-operace-s-hodnotami-ve-formatu-plovouci-radove-carky/>, 2006-06-07 [cit. 2018-01-07].
- [13] Vašíček, Z.: FITkit [online]. <http://merlin.fit.vutbr.cz/FITkit/>, 2006 - 2012 [cit. 2018-05-16].



- [14] Závada, V.: *Simulátor procesoru s operací násobení*. Bakalářská práce, FIT VUT v Brně, 2016.
- [15] Čambor, M.: *Elementární procesor v aritmetice pevné a pohyblivé řádové čárky*. Bakalářská práce, FIT VUT v Brně, 2009.

# Príloha A

## Obsah CD

Priložené CD obsahuje:

- Text tejto práce vo formáte Pdf.
- Zdrojové súbory tejto práce v  $\text{\LaTeX}$ u.
- Zdrojové súbory VHDL paralelného deliaceho integrátora. TODO dopisať