VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

PROPOJOVACÍ SYSTÉM PARALELNÍCH ALU PRO NUMERICKOU INTEGRACI

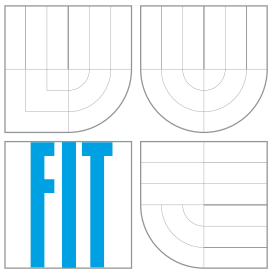
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MICHAELA SEKANINOVÁ

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

PROPOJOVACÍ SYSTÉM PARALELNÍCH ALU PRO NUMERICKOU INTEGRACI

PARALLEL/SERIAL ALU INTERCONECTING SYSTEM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — MICHAELA SEKANINOVÁ
AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. JIŘÍ KUNOVSKÝ, CSc.
SUPERVISOR

BRNO 2015

Abstrakt

Práce se zabývá vytvořením propojovacího systému numerických integrátorů. Obsahem práce je seznámení s propojovacími sítěmi a výpočtem diferenciálních rovnic za použití Taylorovy řady. Součástí praktické části je návrh a realizace propojovacího systému provádějící výpočet numerické integrace pro zvolené diferenciální rovnice. Systém propojuje integrátory automaticky podle zadané úlohy.

Abstract

This thesis deals with creation of parallel/serial ALU interconecting system. The content of the work is to introduce the issue of interconecting systems and numerical solutions of differential equations using Taylor series method. The practical aim of the thesis is to design and implement interconecting system performing calculations of differential equations. The system connects integrators automatically by given problem.

Klíčová slova

propojovací sítě, numerická integrace, diferenciální rovnice, Taylorova řada, integrátor

Keywords

interconecting system, numerical integration, differential equations, Taylor series, integrator

Citace

Michaela Sekaninová: Propojovací systém paralelních ALU pro numerickou integraci, bakalářská práce, Brno, FIT VUT v Brně, 2015

Propojovací systém paralelních ALU pro numerickou integraci

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracovala samostatně pod vedením pana doc. Ing. Jiřího Kunovského, CSc. Uvedla jsem všechny literární prameny a publikace, ze kterých jsem čerpala.

.....
Michaela Sekaninová
13. května 2015

Poděkování

Chtěla bych poděkovat svému školiteli panu doc. Ing. Jiřímu Kunovskému, CSc. za vedení a podporu. Také za veškeré rady, připomínky a náměty, které byly velmi cenné nejen pro samotnou práci. Můj dík patří také rodině, mé florbalové rodině, přátelům a všem, kteří mě podporovali během studia a bez nichž by tato práce nemohla vzniknout.

Obsah

Kapitola 1

Úvod

Numerická integrace využívá numerických metod pro výpočet diferenciálních rovnic. Nejčastěji používané jednokrokové i vícekrokové numerické metody jsou popsány v kapitole 3. Výhodou těchto metod je jednoduchá algoritmizovatelnost, která je vhodná pro výpočet pomocí integrátorů. Jsou představovány aritmeticko logickými jednotkami, které umožňují využití elementárních matematických operací. Integrátory dělíme do tří základních skupin: sériově-sériové, sériově-paralelní a paralelně-paralelní. Bližší seznámení s těmito skupinami je v kapitole 4.

Další výhodou je možnost paralelního výpočtu, který zvyšuje efektivnost řešení pomocí numerických metod, jak je uvedeno v podkapitole 3.3. Paralelní výpočet vede k otázce vytvoření propojovacího systému, kde by se uplatnil vyšší počet integrátorů. Při řešení složitějších diferenciální rovnic nebo jejich soustav je vhodné využití automatické transformace zadání a tvořících diferenciálních rovnic, více v podkapitole 5.1. Počet integrátorů odpovídá počtu diferenciálních rovnic 1. řádu, které vzniknou po transformaci. Tato skutečnost vede k potřebě vytvoření vhodného propojovacího systému.

Cílem této práce je návrh propojovacího systému umožňujícího řešení různých diferenciálních rovnic. Nejprve se seznámíme s propojovacími sítěmi a jejich topologickým rozdělením v kapitole 2. Implementace propojovacího systému je vytvořena pro přípravek FITkit 5.4 s využitím jazyka VHDL. Jazyk je vhodný pro popis hardware a tedy i pro vytvoření integrátorů a propojovacího systému. Na platformě FITkit je hradlové pole FPGA (Field Programmable Gate Array), pro kterou byl propojovací systém navržen. Díky dostupnosti FITkitu může být systém využit pro výukové účely.

Kapitola 2

Propojovací sítě

Propojovací sítě jsou jednou ze základních komponent architektury paralelních systémů a slouží k propojení jednotlivých výpočetních uzlů, viz [4], [15], [3]. Jejich topologie, která může být matematicky popsána pomocí teorie grafů (více v [8]), a vlastnosti do značné míry ovlivňují charakteristiky systému. Návrh sítě splňující specifické požadavky systému je velice důležitý pro jeho výkonnost, zejména při velkém počtu uzlů. Z hlediska propojení rozdělujeme sítě na přímé a nepřímé.

- **přímé propojovací sítě**:
  V tomto typu sítě jsou zprávy zasílány mezi koncovými uzly. Přímé propojení je tvořeno komunikačními linkami propojující směrovače jednotlivých uzlů. Síť je statická a během činnosti již neměnná. Je tedy třeba ji vhodně přizpůsobit požadavkům systému.

- **nepřímé propojovací sítě**:
  Nepřímé propojení je založeno na použití směrovacích uzlů (přepínačů), přes které probíhá zasílání zpráv. Síť je dynamická.

2.1 Přímé propojovací sítě

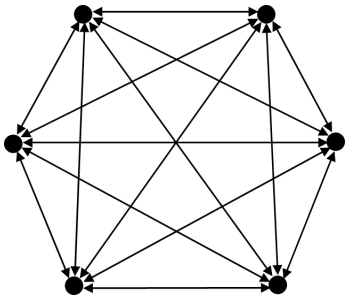
Různá propojení uzlů popisujeme na základě teorie grafů, tedy pomocí uzlů a hran. Uzly dělíme na dva typy: koncové, představující zdroj nebo příjemce, a směrovací neboli přepínače. Hrany grafu reprezentují fyzické linky propojení. U cyklických grafů je výhodou, že při poruše některého z uzlů existuje další možnost propojení. Acyklické grafy naopak v případě poruchy často znemožní spojení. Jejich výhodou je však jednoznačná volba propojení dvou uzlů.

Přímou propojovací síť tvoří uzly, které komunikují zasíláním zpráv přes přímé komunikační linky. Každý uzel se skládá z komponent (procesor, paměť, funkční jednotky), mezi které patří typicky směrovač s propojením na směrovače sousedních uzlů.

Ideální přímá propojovací síť je plně propojená síť, kde je každý uzel propojen s každým. Významnou část tvoří orthogonální topologie, které mají jednotlivé uzly umístěny v souřadnicovém prostoru. Tvoří je dvě skupiny: mřížky a kostky. Přímé propojovací sítě jsou statické, což vedlo ke vzniku různých typů propojení se snahou vyhovět potřebám konkrétního systému (počet uzlů, typ komunikace, aj.)

2.1.1 Plně propojená síť

Plné propojení každého uzlu s každým představuje ideální propojovací síť. Je vhodná pouze pro malý počet uzlů. Je výkonnější, ale dražší ve srovnání s křížovým přepínačem (více v podkapitole 2.2.1). S počtem uzlů roste cena směrovačů, jelikož každý obsahuje n-cestný křížový přepínač. Příklad plně propojené sítě je na obrázku 2.1.

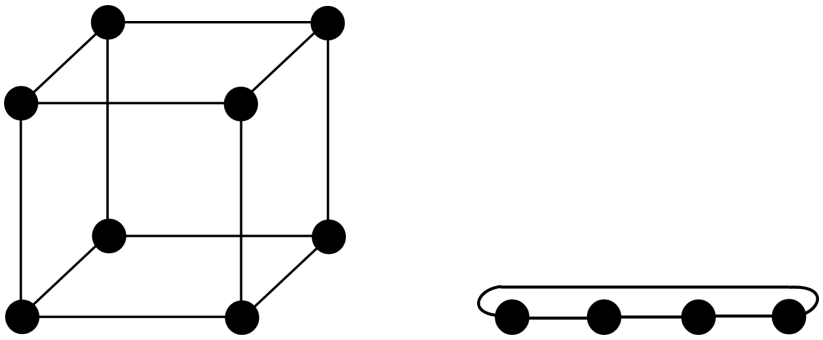


Obrázek 2.1: Topologie sítě plné propojení

2.1.2 Orthogonální topologie

Skupina přímých propojovacích sítí, která využívá řidšího než úplného propojení. Orthogonální topologie jsou tvořeny uzly umístěnými v souřadnicovém prostoru, obecně k uzlů v n dimenzích. Rozlišujeme dva základní typy: kostky a mřížky. Řada je speciální případem a základním prvkem mřížky, viz obrázek 2.3 vpravo. U kostky se jedná o kružnici na obrázku 2.2 vpravo.

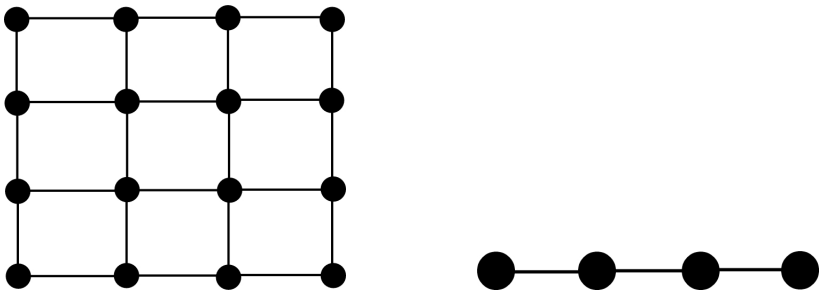
1. **Kostky**:
   Základním typem kostky je kružnice, viz obr. 2.2 vpravo, její dimenze je $n = 1$. Komunikace může být obousměrná, čímž se zkrátí vzdálenost dvou uzlů. Existuje více cest z jednoho uzlu do druhého, tedy v případě poruchy mohou ostatní uzly spolu stále komunikovat. Každý uzel má přiřazenu n-místnou k-ární adresu. Ostatní typy kostek se odvozují pomocí kartézských součinů grafu kružnice. Kostky jsou též nazývány n-dimenzionální toroid o základu k. Příklad kostky je na obrázku 2.2 vlevo.



Obrázek 2.2: Orthogonální topologie: Kostka a kružnice

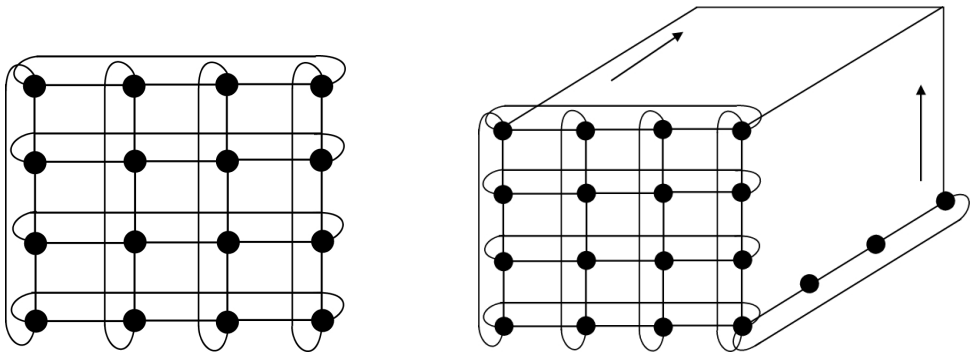
2. **Mřížky**:
   Základním typem mřížky je řada, viz obr. 2.3 vpravo, její dimenze je $n = 1$. Řada

umožňuje průběh několika přenosů dat současně, tím se odlišuje od sběrnice. Vzdálenost uzlů má značný rozptyl a při komunikace vzdálených uzlů jsou blokovány přenosové prostředky ostatních uzlů. Od grafu řady se kartézským součinem odvozují mřížky. Obrázek 2.3 vlevo.



Obrázek 2.3: Orthogonální topologie: Mřížka a řada

3. **Toroidy**:
   2D toroid je tvořen mřížkou, u které je propojen první a poslední prvek každé dimenze. Propojením vznikne kruhová síť a zvýší se počet možných cest pro komunikaci mezi jednotlivými uzly. Přidáním další dimenze vznikne 3D toroid (na obrázku 2.4 vpravo). Příklad 2D toroidu je na obrázku 2.4 vlevo.



Obrázek 2.4: Orthogonální topologie: 2D toroid a 3D toroid

2.2 Nepřímé propojovací sítě

Nepřímé sítě tvoří další hlavní třídu sítí založenou na směrovacím prvku (přepínači). Přes směrovací prvky probíhá komunikace mezi uzly. Přepínače nejsou zdrojem ani cílem dat, mají dvojice vstupních a výstupních kanálů zajišťujících propojení s dalšími prvky sítě. Jako v případě přímých sítí je možné modelovat propojení grafy.

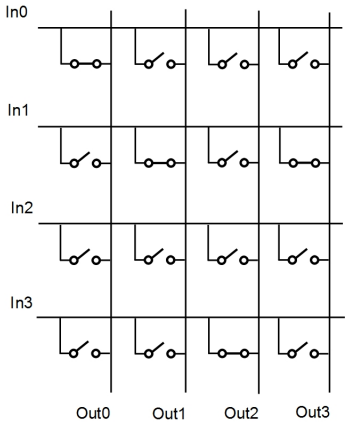
Ideální nepřímá síť je tvořena P-cestným křížovým přepínačem. V porovnání s plným propojením u přímých sítí se jedná o levnější variantu, ale pro velká P stále příliš drahá.

2.2.1 Křížové přepínače

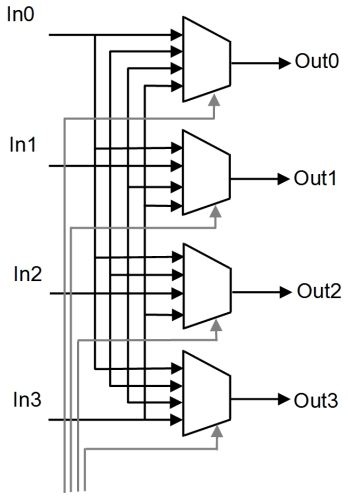
Přepínače jsou základním stavebním prvkem nepřímých propojovacích sítí. Křížový přepínač $m \times n$ přímo propojuje m vstupů na n výstupů. Každý výstup přepínače musí být

připojen maximálně na jeden vstup, naopak jeden vstup může být připojen na více výstupů. V minulosti byly přepínače využívány v telefonii a umožňovaly obousměrné spojení. Model tohoto typu přepínače je znázorněn na obr. 2.5. Naproti tomu spínací prvky, které tvoří přepínače dnes, umožňují pouze jednosměrný přenos signálů. Ukázka implementace jednosměrného křížového přepínače pomocí multiplexoru je na obr. 2.6.

Výhodou přepínačů je nezávislost propojovacích cest, ale jejich cena pro systémy s vysokým počtem prvků je vysoká. Cena odpovídá počtu prvků, které propojuje $(m \times n)$. Z těchto důvodů se lépe uplatňují v systémech s menším počtem prvků.

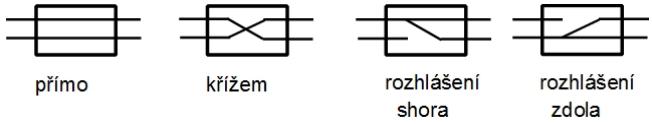


Obrázek 2.5: Model křížové přepínače



Obrázek 2.6: Implementace křížového přepínače multiplexory

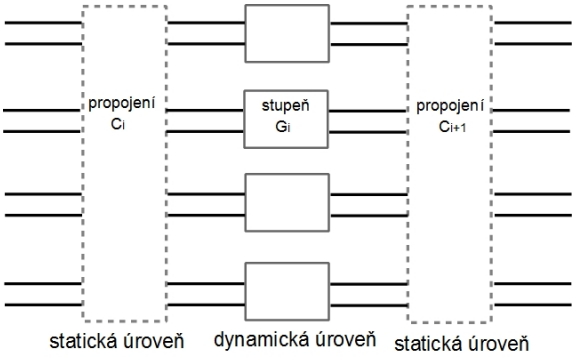
Křížové přepínače, jako základní stavební bloky, určují stupeň celé sítě podle stupně přepínače. Nejčastěji používaný je dvoucestný přepínač, který je základem propojení binárních sítí. Propojení na výstupy pomocí permutací vstupů může být přímo nebo křížem. Další variantou je pak rozhlášení z jednoho vstupu na oba výstupy. Jelikož každý výstup může být připojen maximálně na jeden vstup, jsou poslední dva způsoby propojení nelegální. Typy propojení jsou na následujícím obrázku.



Obrázek 2.7: Dvoucestný přepínač

2.3 Víceúrovňové sítě

V předchozí podkapitole byly popsány křížové přepínače, které jsou základním stavebním prvkem víceúrovňových sítí a jejich stupeň určuje stupeň celé sítě. U rozsáhlejších propojovacích sítí roste cena křížových přepínačů. Z tohoto důvodu vznikla propojení, ve kterých probíhá zasílání zpráv přes více přepínačů uspořádaných do stupňů (úrovní). Tyto sítě často obsahují stejný počet vstupů a výstupů. Jednotlivé úrovně dělíme podle typu, a to na statickou, nebo dynamickou část. Statická část zajišťuje určité propojení (permutaci) vstupů na výstupy dané úrovně. Dynamická část realizuje požadované propojení sítě a je tvořena přepínači. Příklad víceúrovňové sítě je na obrázku 2.8.



Obrázek 2.8: Víceúrovňová obecná síť

U nepřímých propojovacích sítí rozlišujeme dva základní typy sítí: *blokující* a *neblokující*. Křížové přepínače umožňují současné propojení vstupů na výstupy, podle zadané libovolné permutace. V takovém případě nedochází k blokaci jiných propojení, proto se řadí do skupiny *neblokujících sítí*, druhou skupinu pak představují skupiny *blokujících sítí*.

- *blokující sítě*

  Síť se řadí do této skupiny za předpokladu, že některé její párové komunikace vyžadují stejný výstup. V takovém případě musí proběhnout jedna po druhé, jelikož není možný jejich současný průběh.

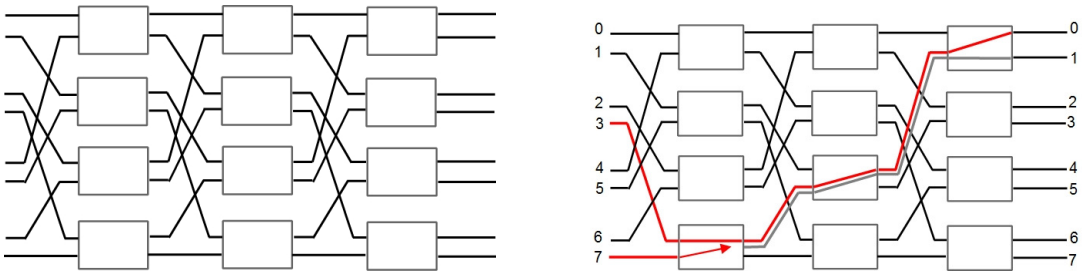
- *neblokující sítě*

  Síť považujeme za neblokující v případě, že můžeme libovolnou permutací propojit její vstupy na výstup a zároveň nedochází k blokování žádného z propojení.

2.3.1 Blokující sítě

Blokující víceúrovňové propojovací sítě jsou levnou a jednoduchou variantou propojení. Jejich nevýhodou je snadné zahlcení, které vzniká z důvodu existence právě jedné cesty z určitého zdroje do cíle. Při určitých permutacích propojení dochází k vyžadování propojení

ze dvou zdrojů na jeden výstup. V takovém případě dojde k blokování a komunikace musí proběhnout jedna po druhé, viz obrázek 2.9 vpravo. Příkladem blokující sítě je síť Omega (obrázek 2.9), motýlková (butterfly) nebo základní (baseline).



Obrázek 2.9: Síť Omega a příklad blokování zpráv v této síti

2.3.2 Neblokující sítě

Neblokující sítě nabízejí různé varianty cest z jednoho zdroje k určenému cíly. V případě blokace jedné z cest se využije jiná cesta. Tímto způsobem je možné řešit zahlcení, ke kterému může dojít u blokujících sítí. Neblokující sítě se dělí na:

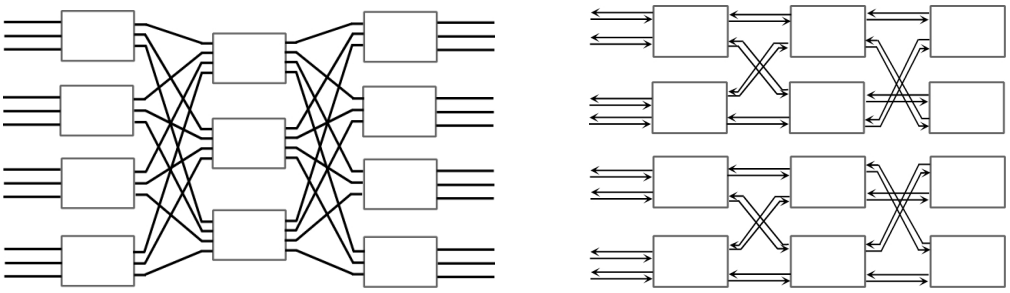
- *přísně neblokující*

  V těchto sítích si žádné párové komunikace nepřekážejí.

- *přestavitelně neblokující*

  V případě, že přídáním další párové komunikace k dalším již běžícím komunikacím dochází ke konfliktu, je síť *přestavitelně neblokující*. Takový konflikt se dá řešit přestavením původních cest.

Příkladem neblokující sítě je Closova síť na obrázku 2.10 vlevo. Tato síť může být přísně neblokující i přestavitelně neblokující, záleží na počtu přepínačů a jejich vstupů a výstupů v určitých stupních. Jednosměrné sítě, které byly popsány v předchozí kapitole, mají nízký počet cest. Pro zvýšení počtu cest mezi zdrojem a cílem zprávy můžeme využít *obousměrné sítě* a tím předejít zahlcení sítě, jak tomu je u blokujích sítí. Na obrázku 2.10 vpravo je ukázka obousměrné motýlkové sítě. Je vidět, že obousměrné propojení je prakticky představováno dvěma jednosmernými sítěmi, které jsou opačného směru a překlopené na sebe.



Obrázek 2.10: Closova síť a obousměrná motýlková síť

Kapitola 3

Numerická integrace

Řešení složitějších soustav diferenciálních rovnic analyticky je složité, mnohdy až nedosažitelné. V těchto případech se využívá numerických metod (více v [6]). Zaměříme se na nejpoužívanější jednokrokové i vícekrokové metody. Výsledek bývá ovlivněn parametry metod a obvykle je pouze přibližný. V práci se budeme zabývat především obyčejnými diferenciálními rovnicemi prvního řádu.

Obecný tvar diferenciální rovnice prvního řádu je

$$g(t, y(t), y'(t)) = 0. \tag{3.1}$$

Pro diferenciální rovnice jsou dále také stanoveny podmínky a věty dokazující např. existenci a jednoznačnost řešení úlohy. Jednou z nich je *Cauchyova úloha*, také známá pod pojmem *počáteční úloha*, tedy rovnice (3.2) spolu s počáteční podmínkou (3.3). Detailní popis podmínek a obyčejných diferenciálních rovnic nalezneme v [2], [13].

$$y' = f(t, y(t)). \tag{3.2}$$

$$y(t_0) = y_0. \tag{3.3}$$

3.1 Numerické metody pro řešení diferenciálních rovnic

Při hledání numerického řešení mohou být použity:

- *jednokrokové metody*:
  Výpočet y_i probíhá iteračně. V každém kroku je hodnota y_{i-1} použita k výpočtu následujího numerického řešení y_i v bodě t_i. První hodnotou pro výpočet je počáteční podmínka. Jednokrokové metody jsou vhodné pro hledání numerického řešení obyčejných diferenciálních rovnic prvního řádu nebo vyšších řádů, které jsou následně převedeny na soustavu rovnic prvního řádu.

  Pro převod na rovnice nižšího řádu je možné použít metodu *snižování řádu derivace* nebo metodu *postupné integrace*. Hlavní rozdíl mezi těmito dvěma metodami spočívá v tom, že u metody *postupné integrace* je možné počítat s derivacemi vstupu na pravé straně rovnice. Toto u metody *snižování řádu derivace*, která je při splnění podmínky pravé strany rovnice bez výskytu derivace vstupu jednodušší, možné není.

- *vícekrokové metody*:
  Vícekrokové metody, narozdíl od předchozí *jednokrokové*, využívají k výpočtu numerického řešení y_i předchozích k výsledků. Na začátek této metody je potřeba využít některou z *jednokrokových metod* pro získání dostatečného počtu hodnot, které jsou nutné ke spuštění metody.

3.1.1 Eulerova metoda

Jednoduchá jednokroková metoda, která vychází z Taylorovy řady. Metoda využívá integrační krok h, který je nutné vhodně zvolit pro získání relativně přesných výsledků. Pracuje se s dostatečně malým kladným číslem integračního kroku.

$$y_{i+1} = y_i + h \cdot f'(t_i, y_i). \tag{3.4}$$

3.1.2 Metoda Runge-Kutta

Stejně jako Eulerova metoda patří mezi jednokrokové metody. Rozdílem však je výpočet i mezi členy y_i a y_{y+1} a vyšší přesnost výsledku. Přírůstek mezi těmito členy získáme jako jejich váhový průměr a jejich počet udává řád metody. I tato metoda, stejně jako ostatní jednokrokové metody, je vhodná pro výpočet prvotních hodnot následně využitých vícekrokovými metodami.

Obecný tvar metody Runge-Kutta je

$$\begin{aligned} y_{n+1} &= y_n + h \cdot (w_1 k_1 + \cdots + w_s k_s) \\ k_1 &= f(t_n, y_n) \\ k_i &= f(t_n + \alpha_i h, y_n + h \sum_{j=1}^{i-1} \beta_{ij} k_j), \quad i = 2, \ldots, s \end{aligned} \tag{3.5}$$

Konstanty $\alpha_i, \beta_{ij}, w_i$ jsou vhodně zvoleny, aby řád metody odpovídal Taylorovu polynomu funkce stejného řádu.

Existují metody Runge-Kutta různých řádů. Například 2. řádu vychází z předchozí metody (3.4) a nazývá se vylepšený Euler. Druhá modifikace Eulerovy metody pak nese název lichoběžníková. Nejužívanější metodou je Runge-Kutta 4. řádu, její obecný tvar vypadá následovně:

$$\begin{aligned} y_{n+1} &= y_n + \frac{1}{6} h (k_1 + 2k_2 + 3k_3 + k_4) \\ k_1 &= f(t_n, y_n) \\ k_2 &= f(t_n + \frac{1}{2} h, y_n + \frac{1}{2} h k_1) \\ k_3 &= f(t_n + \frac{1}{2} h, y_n + \frac{1}{2} h k_2) \\ k_4 &= f(t_n + h, y_n + h k_3) \end{aligned} \tag{3.6}$$

3.2 Metoda Taylorovy řady

Pro řešení diferenciálních rovnic existuje a byla také publikována, kromě výše zmíněných, ještě celá řada dalších metod. Jedním ze základních postupů je Taylorova řada. Řadí se také mezi jednokrokové metody a obecně ji lze vyjádřit jako:

$$y_{i+1} = y_i + \frac{h}{1!} \cdot f'(t_i, y_i) + \frac{h^2}{2!} \cdot f''(t_i, y_i) + \cdots + \frac{h^p}{p!} \cdot f^{(p)}(t_i, y_i), \tag{3.7}$$

kde h představuje integrační krok. Suma jednotlivých členů Taylorovy řady se nazývá Taylorův polynom či mnohočlen. Díky skutečnosti, že se jedná o jednokrokovou metodu a výpočet konkrétní hodnoty pro y_{i+1} je tedy prováděn po jednotlivých krocích, je možné na základě počtu iterací (tzv. řádu) tohoto výpočtu dosahovat jeho požadované a předem stanovené přesnosti. Onu přesnost výpočtu vyjadřuje rozdíl dvou po sobě jdoucích členů Taylorovy řady a k realizaci výpočtu se tak použije pouze tolik členů posloupnosti, kolik je těchto členů zapotřebí k dosažení stanovené přesnosti. Detailní popis Taylorovy řady a jejího použití nalezneme v [1], [10].

Metoda Taylorovy řady má příznivé paralelní vlastnosti. Většina výpočetních operací je vzájemně nezávislých, takže mohou být prováděny paralelně v oddělených procesorech paralelního výpočetního systému. Při použití Taylorovy řady je však nutné brát do úvahy možnou nevýhodu, kterou je zejména závislost na výpočtech derivací vyššího řádu. I přes tuto nevýhodu zůstává Taylorova řada stabilní metodou, která především poskytuje přesnější výsledky než celá řada používaných explicitních metod pro výpočet diferenciálních rovnic. Seznámení s moderní metodou Taylorovy řady a jejím technickém využití je popsáno v [12].

3.2.1 Výpočet diferenciální rovnice Taylorovou řadou

Příklad využití Taylorovy řady (3.7) uvedeme na obyčejné diferenciální rovnici 1. řádu.

$$y' = y_i \qquad y(0) = y_0 \tag{3.8}$$

Podmínka platnosti Taylorovy řady platí, jelikož ze zadané rovnice (3.8) vyplývá vztah:

$$y' = y \qquad y = y' = y'' = y''' = \cdots = y^{(n)} \tag{3.9}$$

Následným dosazením do Taylorovy řady (3.7) získáme

$$y_{i+1} = y_i + \frac{h}{1!} \cdot y_i + \frac{h^2}{2!} \cdot y_i + \cdots + \frac{h^p}{p!} \cdot y_i, \tag{3.10}$$

které lze zjednodušit na:

$$y_{i+1} = y_i \cdot (1 + \frac{h}{1!} + \frac{h^2}{2!} + \cdots + \frac{h^p}{p!}). \tag{3.11}$$

Zavedeme si členy Taylorovy řady (3.13)–(3.15), které reprezentují jednotlivé dílčí přírůstky. Hodnotu řešení y_i získáme součtem počáteční podmínky a členů Taylorovy řady, viz rovnice (3.12). Pro výpočet nám stačí zadání počáteční podmínky, integračního kroku a požadované přesnosti. Tím se stává výpočet lehce algoritmizovatelným. Jednou z možností

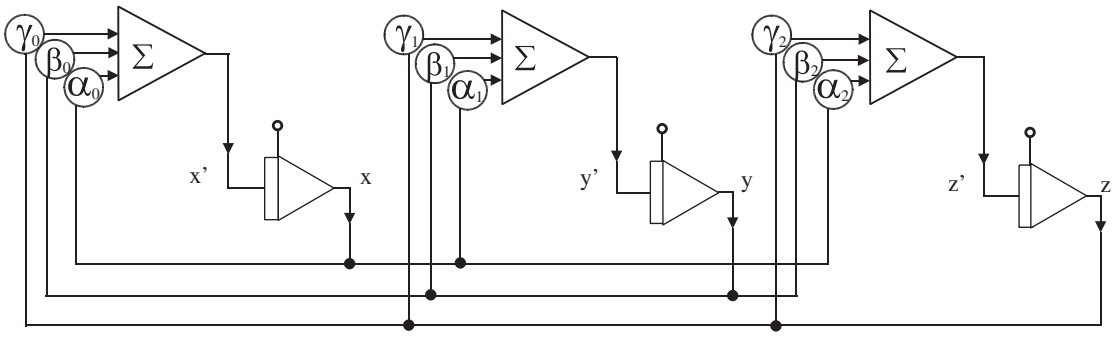
Taylorovy řady je počítat do "plné přesnosti", tedy dokud je člen Taylorovy řady (3.15) uplatněn v součtu rovnice (3.12).

$$
\begin{aligned}
y_i &= y_0 + DY1_i + DY2_i + \cdots + DYp_i && (3.12)\\
DY1_i &= h \cdot y_i && (3.13)\\
DY2_i &= \frac{h}{2} DY1_i && (3.14)\\
&\vdots \\
DYp_i &= \frac{h}{p} DY(p-1)_i && (3.15)
\end{aligned}
$$

3.3 Řešení homogenních diferenciálních rovnic a efektivnost Taylorovy řady

V předchozí podkapitole je uvedeno řešení pro homogenní diferenciální rovnici (3.8), které je aplikovatelné i na soustavu rovnic. Při numerickém řešení této soustavy homogenních diferenciálních rovnic s konstantními koeficienty je možné použít některou z dříve uvedených numerických metod, viz podkapitola 3.1. Provedeme srovnání efektivnosti nejpoužívanějších metod na následující soustavě rovnic.

$$
\begin{aligned}
x' &= \alpha_0 \cdot x + \beta_0 \cdot y + \gamma_0 \cdot z \qquad x(0) = x_0 && (3.16)\\
y' &= \alpha_1 \cdot x + \beta_1 \cdot y + \gamma_1 \cdot z \qquad y(0) = y_0 && (3.17)\\
z' &= \alpha_2 \cdot x + \beta_2 \cdot y + \gamma_2 \cdot z \qquad z(0) = z_0 && (3.18)
\end{aligned}
$$



Obrázek 3.1: Blokové schéma soustavy homogenních diferenciálních rovnic s konstantními koeficienty

Na soustavě rovnic (3.16)–(3.18) uvedeme výpočet pomocí Taylorovy řady a metody Runge-Kutta, nejprve 2. a následně 4. řádu.

Řešení soustavy rovnic metodami 2. řádu

Metoda Runge-Kutta 2. řádu Obecný tvar metody Runge-Kutta aplikovaný na soustavu rovnic (3.16)–(3.18):

$$\begin{aligned} x_1 &= x_0 + \frac{1}{2} \cdot (k_{1(x)} + k_{2(x)}) \\ y_1 &= y_0 + \frac{1}{2} \cdot (k_{1(y)} + k_{2(y)}) \\ z_1 &= z_0 + \frac{1}{2} \cdot (k_{1(z)} + k_{2(z)}) \end{aligned} \tag{3.19}$$

pro koeficienty platí:

$$\begin{aligned} k_{1(x)} &= h \cdot (\alpha_0 \cdot x_0 + \beta_0 \cdot y_0 + \gamma_0 \cdot z_0) \\ k_{1(y)} &= h \cdot (\alpha_1 \cdot x_0 + \beta_1 \cdot y_0 + \gamma_1 \cdot z_0) \\ k_{1(z)} &= h \cdot (\alpha_2 \cdot x_0 + \beta_2 \cdot y_0 + \gamma_2 \cdot z_0) \end{aligned} \tag{3.20}$$

$$\begin{aligned} k_{2(x)} &= h \cdot (\alpha_0 \cdot (x_0 + k_{1(x)}) + \beta_0 \cdot (y_0 + k_{1(y)}) + \gamma_0 \cdot (z_0 + k_{1(z)})) \\ k_{2(y)} &= h \cdot (\alpha_1 \cdot (x_0 + k_{1(x)}) + \beta_1 \cdot (y_0 + k_{1(y)}) + \gamma_1 \cdot (z_0 + k_{1(z)})) \\ k_{2(z)} &= h \cdot (\alpha_2 \cdot (x_0 + k_{1(x)}) + \beta_2 \cdot (y_0 + k_{1(y)}) + \gamma_2 \cdot (z_0 + k_{1(z)})) \end{aligned}$$

Pro výpočet rovnic (3.19)–(3.20) jsou zapotřebí dvě základní matematické operace. Předpokládáme, že procesor provádí sčítání a násobení. V navzájem si odpovídajících rovnicích je posloupnost matematických operací identická, což umožňuje paralelní výpočet v oddělených mikroprocesorech. Posloupností elementárních operací získáme hodnoty pro koeficient k_1. Následně dosazením získané hodnoty koeficientu k_1 získáme hodnoty pro koeficient k_2. Poté získáme celkový výpočet členu.

V tabulce (3.1) jsou popsány operace pro každý mikroprocesor. V kroku 6 získáme hodnotu koeficientu k_1 ($k_{1(x)}$, $k_{1(y)}$, $k_{1(z)}$). Pro hodnoty koeficientu k_2 ($k_{2(x)}$, $k_{2(y)}$, $k_{2(z)}$) získáme výsledek v kroku 15. Celkového řešení soustavy rovnic (3.19) využitím metody Runge-Kutta 2. řádu získáme po 18 krocích paralelních výpočtů ve třech oddělených mikroprocesorech.

Metoda Taylorovy řady 2. řádu Obecný tvar Taylorovy řady aplikovaný na soustavu rovnic (3.16)–(3.18) s využitím značení jako v rovnici (3.12):

$$\begin{aligned} x_1 &= x_0 + DX1_0 + DX2_0 \\ y_1 &= y_0 + DY1_0 + DY2_0 \\ z_1 &= z_0 + DZ1_0 + DZ2_0 \end{aligned} \tag{3.21}$$

pro výpočet jednotlivých členů Taylorovy řady platí vztahy:

$$\begin{aligned} DX1_0 &= h \cdot (\alpha_0 \cdot x_0 + \beta_0 \cdot y_0 + \gamma_0 \cdot z_0) \\ DY1_0 &= h \cdot (\alpha_1 \cdot x_0 + \beta_1 \cdot y_0 + \gamma_1 \cdot z_0) \\ DZ1_0 &= h \cdot (\alpha_2 \cdot x_0 + \beta_2 \cdot y_0 + \gamma_2 \cdot z_0) \end{aligned} \tag{3.22}$$

$$\begin{aligned} DX2_0 &= \frac{h}{2} \cdot (\alpha_0 \cdot DX1_0 + \beta_0 \cdot DY1_0 + \gamma_0 \cdot DZ1_0) \\ DY2_0 &= \frac{h}{2} \cdot (\alpha_1 \cdot DX1_0 + \beta_1 \cdot DY1_0 + \gamma_1 \cdot DZ1_0) \\ DZ2_0 &= \frac{h}{2} \cdot (\alpha_2 \cdot DX1_0 + \beta_2 \cdot DY1_0 + \gamma_2 \cdot DZ1_0) \end{aligned} \tag{3.23}$$

Tabulka 3.1: Posloupnost operací při řešení soustavy metodou Runge-Kutta 2. řádu

Krok	**Procesor X**	**Procesor Y**	**Procesor Z**
1	$X1 = \alpha_0 \cdot x_0$	$Y1 = \alpha_1 \cdot x_0$	$Z1 = \alpha_2 \cdot x_0$
2	$X2 = \beta_0 \cdot y_0$	$Y2 = \beta_1 \cdot y_0$	$Z2 = \beta_2 \cdot y_0$
3	$X3 = \gamma_0 \cdot z_0$	$Y3 = \gamma_1 \cdot z_0$	$Z3 = \gamma_2 \cdot z_0$
4	$X4 = X1 + X2$	$Y4 = Y1 + Y2$	$Z4 = Z1 + Z2$
5	$X5 = X4 + X3$	$Y5 = Y4 + Y3$	$Z5 = Z4 + Z3$
6	$X6 = h \cdot X5$	$Y6 = h \cdot Y5$	$Z6 = h \cdot Z5$
7	$X7 = x_0 + X6$	$Y7 = x_0 + X6$	$Z7 = x_0 + X6$
8	$X8 = y_0 + Y6$	$Y8 = y_0 + Y6$	$Z8 = y_0 + Y6$
9	$X9 = z_0 + Z6$	$Y9 = z_0 + Z6$	$Z9 = z_0 + Z6$
10	$X10 = \alpha_0 \cdot X7$	$Y10 = \alpha_1 \cdot Y7$	$Z10 = \alpha_2 \cdot Z7$
11	$X11 = \beta_0 \cdot X8$	$Y11 = \beta_1 \cdot Y8$	$Z11 = \beta_2 \cdot Z8$
12	$X12 = \gamma_0 \cdot X9$	$Y12 = \gamma_1 \cdot Y9$	$Z12 = \gamma_2 \cdot Z9$
13	$X13 = X10 + X11$	$Y13 = Y10 + Y11$	$Z13 = Z10 + Z11$
14	$X14 = X13 + X12$	$Y14 = Y13 + Y12$	$Z14 = Z13 + Z12$
15	$X15 = h \cdot X14$	$Y15 = h \cdot Y14$	$Z15 = h \cdot Z14$
16	$X16 = X6 + X15$	$Y16 = Y6 + Y15$	$Z16 = Z6 + Z15$
17	$X17 = 1/2 \cdot X16$	$Y17 = 1/2 \cdot Y16$	$Z17 = 1/2 \cdot Z16$
18	$X18 = x_0 + X17$	$Y18 = y_0 + Y17$	$Z18 = z_0 + Z17$

Analogicky k výpočtu soustavy rovnic pomocí metody Runge-Kutta je možno provádět výpočet paralelně ve třech mikroprocesorech. Předpokladem je, že mikroprocesory provádějí základní matematické operace (násobení a sčítání). Obecný tvar Taylorovy řady je aplikován na soustavu diferenciálních rovnic s konstantními koeficienty (3.16)–(3.18). Pro výpočet hodnot x_1, y_1, y_2 a jejich členů Taylorovy řady využíváme stejných operací, které jsou vyobrazeny v tabulce (3.2). V kroku 6 získáme hodnoty prvních členů Taylorovy řady ($DX1_0$, $DY1_0$, $DZ1_0$) a následně v kroku 12 hodnoty druhých členů Taylorovy řady ($DX2_0$, $DY2_0$, $DZ2_0$).

Srovnáním tabulky (3.1) zobrazující posloupnost operací při řešení soustavy metodou Runge-Kutta a tabulky (3.21), která představuje posloupnost řešení stejné soustavy diferenciálních rovnic metodou Taylorovy řady. Z tabulek můžeme vypozorovat, že vyšší efektivnost Taylorovy metody, neboť jejím použitím získáme řešení po 14 krocích, kdežto v případě metody Runge-Kutta získáme řešení pro x_1, y_1, z_1 až po 18 krocích.

Pro další srovnání využijeme výpočet stejné soustavy diferenciálních rovnic stejnými metodami, ale 4. řádu.

Tabulka 3.2: Posloupnost operací při řešení soustavy metodou Taylorovy řady 2. řádu

Krok	**Procesor X**	**Procesor Y**	**Procesor Z**
1	$X1 = \alpha_0 \cdot x_0$	$Y1 = \alpha_1 \cdot x_0$	$Z1 = \alpha_2 \cdot x_0$
2	$X2 = \beta_0 \cdot y_0$	$Y2 = \beta_1 \cdot y_0$	$Z2 = \beta_2 \cdot y_0$
3	$X3 = \gamma_0 \cdot z_0$	$Y3 = \gamma_1 \cdot z_0$	$Z3 = \gamma_2 \cdot z_0$
4	$X4 = X1 + X2$	$Y4 = Y1 + Y2$	$Z4 = Z1 + Z2$
5	$X5 = X4 + X3$	$Y5 = Y4 + Y3$	$Z5 = Z4 + Z3$
6	$X6 = h \cdot X5$	$Y6 = h \cdot Y5$	$Z6 = h \cdot Z5$
7	$X7 = \alpha_0 \cdot X6$	$Y7 = \alpha_1 \cdot X6$	$Z7 = \alpha_2 \cdot X6$
8	$X8 = \beta_0 \cdot Z6$	$Y8 = \beta_1 \cdot Y6$	$Z8 = \beta_2 \cdot Z6$
9	$X9 = \gamma_0 \cdot Y6$	$Y9 = \gamma_1 \cdot Y6$	$Z9 = \gamma_2 \cdot Z6$
10	$X10 = X7 + X8$	$Y10 = Y7 + Y8$	$Z10 = Z7 + Z8$
11	$X11 = X10 + X9$	$Y11 = Y10 + Y9$	$Z11 = Z10 + Z9$
12	$X12 = h/2 \cdot X11$	$Y12 = h/2 \cdot Y11$	$Z12 = h/2 \cdot Z11$
13	$X13 = X12 + X6$	$Y13 = Y12 + Y6$	$Z13 = Z12 + Z6$
14	$X14 = x_0 + X13$	$Y14 = y_0 + Y13$	$Z14 = z_0 + Z13$

Řešení soustavy rovnic metodami 4. řádu

Metoda Runge-Kutta 4. řádu

Na soustavu rovnic (3.16)–(3.18) aplikujeme metodu Runge-Kutta 4. řádu.

$$\begin{aligned} x_1 &= x_0 + \frac{1}{6} \cdot (k_{1(x)} + 2 \cdot k_{2(x)} + 2 \cdot k_{3(x)} + k_{4(x)}) \\ y_1 &= y_0 + \frac{1}{6} \cdot (k_{1(y)} + 2 \cdot k_{2(y)} + 2 \cdot k_{3(y)} + k_{4(y)}) \\ z_1 &= z_0 + \frac{1}{6} \cdot (k_{1(z)} + 2 \cdot k_{2(z)} + 2 \cdot k_{3(z)} + k_{4(z)}) \end{aligned} \tag{3.24}$$

pro koeficienty k_1 platí:

$$k_{1(x)} = h \cdot (\alpha_0 \cdot x_0 + \beta_0 \cdot y_0 + \gamma_0 \cdot z_0) \tag{3.25}$$

$$\begin{aligned} k_{1(y)} &= h \cdot (\alpha_1 \cdot x_0 + \beta_1 \cdot y_0 + \gamma_1 \cdot z_0) \\ k_{1(z)} &= h \cdot (\alpha_2 \cdot x_0 + \beta_2 \cdot y_0 + \gamma_2 \cdot z_0) \end{aligned} \tag{3.26}$$

pro koeficienty k_2 platí:

$$\begin{aligned} k_{2(x)} &= h \cdot (\alpha_0 \cdot (x_0 + \frac{1}{2} \cdot k_{1(x)}) + \beta_0 \cdot (y_0 + \frac{1}{2} \cdot k_{1(y)}) + \gamma_0 \cdot (z_0 + \frac{1}{2} \cdot k_{1(z)})) \\ k_{2(y)} &= h \cdot (\alpha_1 \cdot (x_0 + \frac{1}{2} \cdot k_{1(x)}) + \beta_1 \cdot (y_0 + \frac{1}{2} \cdot k_{1(y)}) + \gamma_1 \cdot (z_0 + \frac{1}{2} \cdot k_{1(z)})) \\ k_{2(z)} &= h \cdot (\alpha_2 \cdot (x_0 + \frac{1}{2} \cdot k_{1(x)}) + \beta_2 \cdot (y_0 + \frac{1}{2} \cdot k_{1(y)}) + \gamma_2 \cdot (z_0 + \frac{1}{2} \cdot k_{1(z)})) \end{aligned}$$

pro koeficienty k_3 platí:

$$\begin{aligned} k_{2(x)} &= h \cdot (\alpha_0 \cdot (x_0 + \frac{1}{2} \cdot k_{2(x)}) + \beta_0 \cdot (y_0 + \frac{1}{2} \cdot k_{2(y)}) + \gamma_0 \cdot (z_0 + \frac{1}{2} \cdot k_{2(z)})) \\ k_{2(y)} &= h \cdot (\alpha_1 \cdot (x_0 + \frac{1}{2} \cdot k_{2(x)}) + \beta_1 \cdot (y_0 + \frac{1}{2} \cdot k_{2(y)}) + \gamma_1 \cdot (z_0 + \frac{1}{2} \cdot k_{2(z)})) \\ k_{2(z)} &= h \cdot (\alpha_2 \cdot (x_0 + \frac{1}{2} \cdot k_{2(x)}) + \beta_2 \cdot (y_0 + \frac{1}{2} \cdot k_{2(y)}) + \gamma_2 \cdot (z_0 + \frac{1}{2} \cdot k_{2(z)})) \end{aligned}$$

pro koeficienty k_4 platí:

$$\begin{aligned} k_{2(x)} &= h \cdot (\alpha_0 \cdot (x_0 + k_{3(x)}) + \beta_0 \cdot (y_0 + k_{3(y)}) + \gamma_0 \cdot (z_0 + k_{3(z)})) \\ k_{2(y)} &= h \cdot (\alpha_1 \cdot (x_0 + k_{3(x)}) + \beta_1 \cdot (y_0 + k_{3(y)}) + \gamma_1 \cdot (z_0 + k_{3(z)})) \\ k_{2(z)} &= h \cdot (\alpha_2 \cdot (x_0 + k_{3(x)}) + \beta_2 \cdot (y_0 + k_{3(y)}) + \gamma_2 \cdot (z_0 + k_{3(z)})) \end{aligned}$$

Tvorba tabulky posloupnosti operací je identická k tabulce (3.1). Nebude zde uvedena z důvodu její rozsáhlosti. Postup je analogický jako při využití metody druhého řádu ovšem s využitím rovnic (3.24)–(3.27). Počet kroků pro výpočet x_1, y_1, z_1 je 46 operací pro každý mikroprocesor.

Metoda Taylorovy řady 4. řádu

Obecný tvar metody Taylorovy řady 4. řádu pro soustavu rovnic (3.16)–(3.18) je následující:

$$\begin{aligned} x_1 &= x_0 + DX1_0 + DX2_0 + DX3_0 + DX4_0 \\ y_1 &= y_0 + DY1_0 + DY2_0 + DY3_0 + DY4_0 \\ z_1 &= z_0 + DZ1_0 + DZ2_0 + DZ3_0 + DZ4_0 \end{aligned} \tag{3.27}$$

pro výpočet jednotlivých členů Taylorovy řady platí vztahy:

$$\begin{aligned} DX1_0 &= h \cdot (\alpha_0 \cdot x_0 + \beta_0 \cdot y_0 + \gamma_0 \cdot z_0) \\ DY1_0 &= h \cdot (\alpha_1 \cdot x_0 + \beta_1 \cdot y_0 + \gamma_1 \cdot z_0) \\ DZ1_0 &= h \cdot (\alpha_2 \cdot x_0 + \beta_2 \cdot y_0 + \gamma_2 \cdot z_0) \end{aligned} \tag{3.28}$$

$$\begin{aligned} DX2_0 &= \frac{h}{2} \cdot (\alpha_0 \cdot DX1_0 + \beta_0 \cdot DY1_0 + \gamma_0 \cdot DZ1_0) \\ DY2_0 &= \frac{h}{2} \cdot (\alpha_1 \cdot DX1_0 + \beta_1 \cdot DY1_0 + \gamma_1 \cdot DZ1_0) \\ DZ2_0 &= \frac{h}{2} \cdot (\alpha_2 \cdot DX1_0 + \beta_2 \cdot DY1_0 + \gamma_2 \cdot DZ1_0) \end{aligned} \tag{3.29}$$

$$\begin{aligned} DX3_0 &= \frac{h}{3} \cdot (\alpha_0 \cdot DX2_0 + \beta_0 \cdot DY2_0 + \gamma_0 \cdot DZ2_0) \\ DY3_0 &= \frac{h}{3} \cdot (\alpha_1 \cdot DX2_0 + \beta_1 \cdot DY2_0 + \gamma_1 \cdot DZ2_0) \\ DZ3_0 &= \frac{h}{3} \cdot (\alpha_2 \cdot DX2_0 + \beta_2 \cdot DY2_0 + \gamma_2 \cdot DZ2_0) \end{aligned} \tag{3.30}$$

$$\begin{aligned} DX4_0 &= \frac{h}{4} \cdot (\alpha_0 \cdot DX3_0 + \beta_0 \cdot DY3_0 + \gamma_0 \cdot DZ3_0) \\ DY4_0 &= \frac{h}{4} \cdot (\alpha_1 \cdot DX3_0 + \beta_1 \cdot DY3_0 + \gamma_1 \cdot DZ3_0) \\ DZ4_0 &= \frac{h}{4} \cdot (\alpha_2 \cdot DX3_0 + \beta_2 \cdot DY3_0 + \gamma_2 \cdot DZ3_0) \end{aligned} \tag{3.31}$$

Jak již bylo uvedeno v předchozím výpočtu soustavy diferenciálních rovnic Taylorovou metodou 2. řádu, můžeme provádět výpočet paralelně na třech mikroprocesorech. Tabulka posloupnosti elementárních operací (sčítání, násobení) se tvoří analogicky k tabulce 3.2.

Pro její velikost zde také nebude uvedena, ale pro výpočet hodnot x_1, y_1 a z_1 je zapotřebí 18 kroků.

Při výpočtech stejné soustavy diferenciálních rovnic je pro metodu Taylorovy řady zapotřebí menší počet kroků než při použití metody Runge-Kutta. Rozdíl je ještě výraznější u vyšších řádů metod. Porovnáním výsledků rozboru je potvrzena vyšší efektivnost Taylorovy řady, kdy při použití stejného počtu kroků jako u metody Runge-Kutta dosáhneme vyšší přesnosti (dosažením vyššího řádu metody).

Kapitola 4

Numerický integrátor

Pro řešení úlohy vycházející z principu metody Taylorovy řady je nejprve nutné úlohu převést na soustavu homogenních lineárních diferenciálních rovnic s konstantními koeficienty, jak již bylo uvedeno v kapitole 3.1. Následný výpočet záskaných defirenciálních rovnic touto metodou využívá dvou matematických operací, kterými jsou násobení a sčítání. Pomocí násobení získáme hodnoty jednotlivých členů Taylorovy řady, tedy DYp_i a jejich následným iterativním sčítáním jsme schopni určit výsledky této posloupnosti pro libovolný počet členů, tedy hodnoty y_i. Obě tyto operace je možné provádět paralelně nebo také sériově. Na základě této skutečnosti uvádí M. Kraus ve své dizertační práci [11] tři základní skupiny numerických integrátorů, kterými jsou:

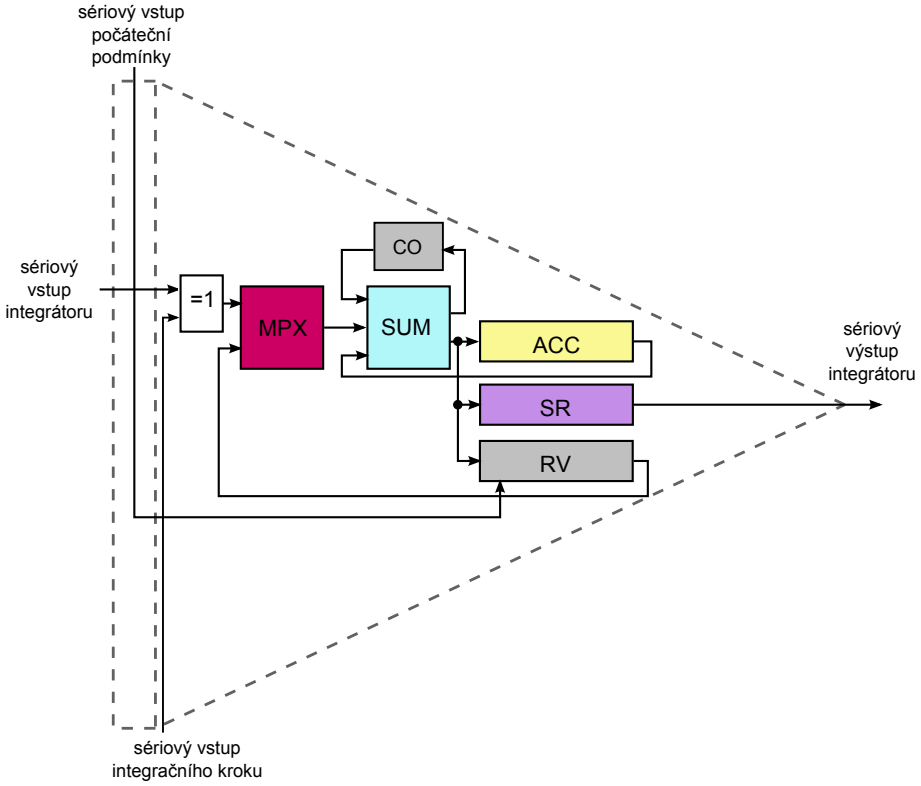
- sériově-sériové integrátory
- sériově-paralelní integrátory
- paralelně-paralelní integrátory

První část názvu skupiny určuje typ komunikace mezi integrátory. Druhá část zastupuje typ výpočtu. Obrázky a popisy v následujících podkapitolách, kde detailněji přiblížíme všechny skupiny zmíněných numerických integrátorů, byly převzaty z dizertační práce [11].

4.1 Sériově-sériový integrátor

První z výše uvedených typů numerických integrátorů je sériově-sériový. U tohoto typu se sekvenčně neprovádí pouze jedna operace, jak tomu bude u sériově-paralelního, nýbrž obě matematické operace. Tedy jak operace násobení, tak také operace sčítání.

Pro výpočet pomocí uvedeného typu integrátoru je využit následující postup. Nejprve inicializujeme potřebné hodnoty vložením hodnoty y_i do registru výsledku. Vynulujume hodnotu akumulárotu a obvodu pro uchování přenosu. Nastavíme řídící signál multiplexoru, který nastaví cestu z akumulátoru. Celý výpočet probíhá sériově, tedy od nejméně významového bitu $f(y_i)$ na vstupu až po ten nejvýznamnější. Stejným způsobem se na sériovém vstupu integračního kroku postupně načítá hodnota integračního kroku, ten se v závislosti na vstupu přičte k hodnotě akumulátoru. Získáme mezivýsledek, dojde k posunu dalšího bitu na vstup a také k posunu výstupního posuvného registru. Po dokončení operace násobení je hodnota členu Taylorovy řady (DYp) přenesena z akumulátoru do posuvného registru a sériově přičtena k hodnotě registru výsledku.



Obrázek 4.1: Blokové schéma sériově-sériového integrátoru

Význam jednotlivých bloků:

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

Díky sériové komunikaci je cena zapojení nižší. Naproti tomu se zvyšuje časová náročnost výpočtu (počet iterací) t, která je popsána následujícím vztahem:

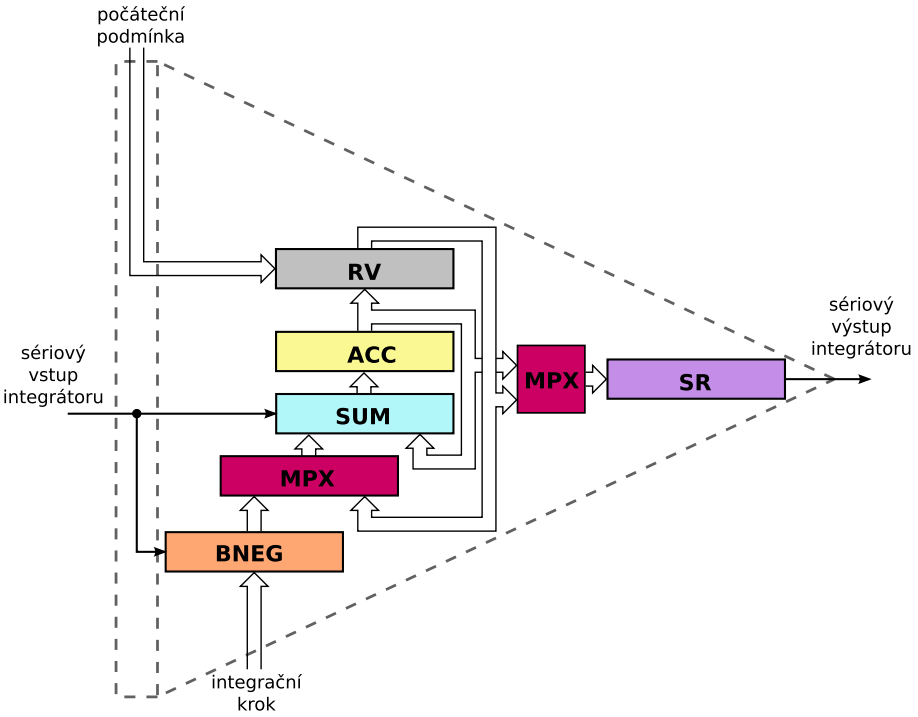
$$\begin{aligned} t &= \tau_{mult} + \tau_{sum} + \tau_{net} \\ t &= n{\cdot}n{\cdot}\tau_{sum} + n{\cdot}\tau_{sum} + \tau_{net} \\ t &= n^2{\cdot}\tau_{sum} + n{\cdot}\tau_{sum} + \tau_{net} \end{aligned} \tag{4.1}$$

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

4.2 Sériově-paralelní integrátor

Sériově-paralelní integrátor je dalším ze skupiny integrátorů umožňujících výpočty stávající ze soustav lineárních diferenciálních rovnic. Stejně jako u předešlého typu integrátoru se

matematická operace násobení vykonává sekvenčně na principu Boothova algoritmu, naproti tomu operace sčítání je prováděna pomocí paralelní sčítačky. Na následujícím obrázku je zachycena bloková struktura tohoto typu integrátoru.



Obrázek 4.2: Blokové schéma sériově-paralelního integrátoru

Význam jednotlivých bloků:

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

Výpočet hodnoty y_{i+1} probíhá postupným načítáním bitů stejně jako u předchozího typu integrátoru. Na vstupu se objevuje nejméně významový bit a kroky výpočtu probíhají iteračně až do načtení posledního bitu vstupní hodnoty. Před samotným výpočtem je do registrů (RV, SR) nahrána hodnota y_i a je vynulován akumulátor. Řídící signál multiplexoru nastaví cestu z obvodu řízené negace. Hodnota integračního kroku odpovídá zadanému h. Bit na vstupu určuje operaci, která bude probíhat mezi akumulátorem a násobencem uloženém v registru výsledku. Může dojít k přičtení či odečtení hodnoty násobence do akumulátoru nebo se vynuluje. Pro odečtení násobence se vytvoří jeho dvojkový doplněk, tedy záporný násobenec. Výsledek ze sčítačky se zapíše do akumulátoru. Následně dojde k posunu doprava o jeden bit u akumulátoru a posuvného registru.

Po zpracování všech vstupních bitů je výsledkem $f(y_i) \cdot h$, který je uložen do posuvného registru. Člen Taylorovy řady (DYp) je uložen v akumulátoru a je k němu přičtena hodnota registru RV, obsah akumulátoru je následně přesunut do RV. V této fázi výpočtu je multiplexor přepnut na vstup z obvodu řízené negace. Výsledek y_{i+1} je získán po dosažení požadované přesnosti nebo maximálního počtu iterací.

Sériová komunikace zvýhodňuje využití popsaného typu integrátoru. Počet vývodů je nízký, jelikož data vstupují a vystupují sériově. Výhodou je také možnost rozšíření registrů a řídících signálů, které umožňuje zvýšení přesnosti výpočtů bez potřeby změnit rozhraní integrátoru či propojovací síť.

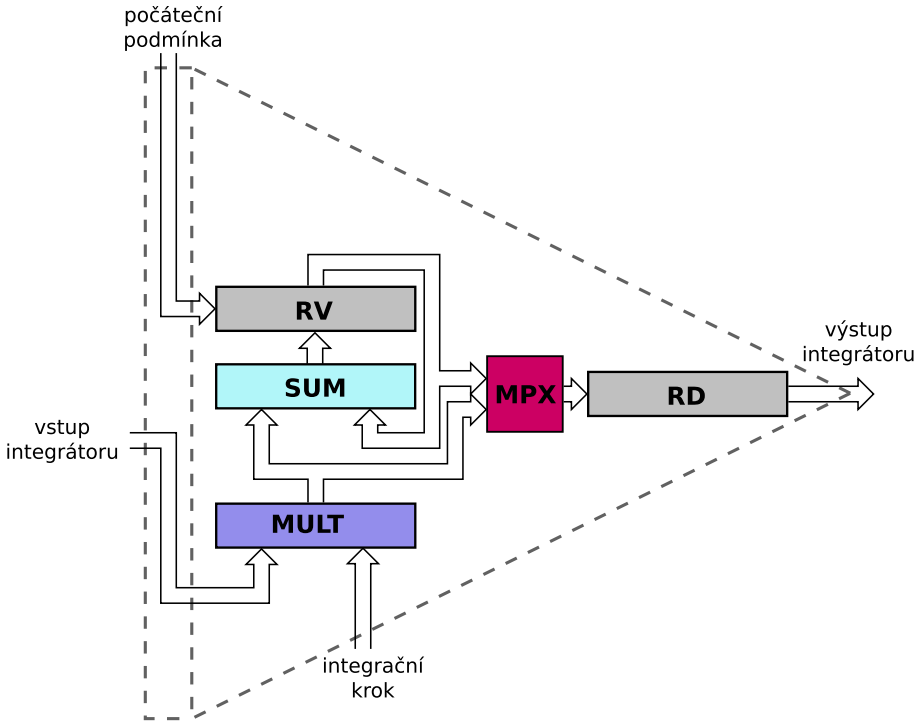
Čas výpočtu t členu Taylorovy řady (DYp) je určen vztahem:

$$\begin{aligned} t &= \tau_{mult} + \tau_{sum} + \tau_{net} \\ t &= n{\cdot}\tau_{sum} + \tau_{sum} + \tau_{net} \end{aligned} \tag{4.2}$$

Hodnota n reprezentuje počet bitů zobrazení čísel. Celkový čas výpočtu je dán zpožděním sítě, časem násobení a součtem předchozího výsledku a násobení. Dobu násobení můžeme převést na n kroků sčítání.

4.3 Paralelně-paralelní integrátor

Poslední z trojice uvedených typů integrátorů je paralelně-paralelní, u kterého operace sčítání a násobení probíhá paralelně, tedy pomocí paralelní sčítačky respektive paralelní násobičky. Blokové schéma integrátoru je uvedeno na následujícím obrázku.



Obrázek 4.3: Blokové schéma paralelně-paralelního integrátoru

Význam jednotlivých bloků:

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

Při inicializaci cyklu se do registru výsledku a součinu vloží hodnota y_i. Je určena hodnota integračního kroku h. Na vstupu integrátoru je hodnota $f(y_i)$, která je vynásobena integračním krokem. Výsledek z násobičky je přenesen do registru součinu (RD obsahuje člen Taylorovy řady $DY1$) a přičten k registru výsledku, který obsahuje mezisoučet $y_i + DY1$. Výpočet probíhá iteračně, tedy na vstupu se objeví hodnota $f(DY1)$ a integrační krok odpovídá $h/2$. Po každé iteraci je v registru součinu hodnota členu Taylorovy řady $DY(p+1)$, registr výsledku obsahuje mezisoučet $y_i + DYp + \cdots + DY(p+n)$. Při dosažení zadané přesnosti nebo počtu cyklů získáme hodnotu y_{i+1}.

Časová náročnost t u paralelně-paralelního integrátoru je nejlepší ze všech typů, viz následující vztah.

$$t = \tau_{mult} + \tau_{sum} + \tau_{net}$$

Kapitola 5

Návrh a implementace propojovací sítě

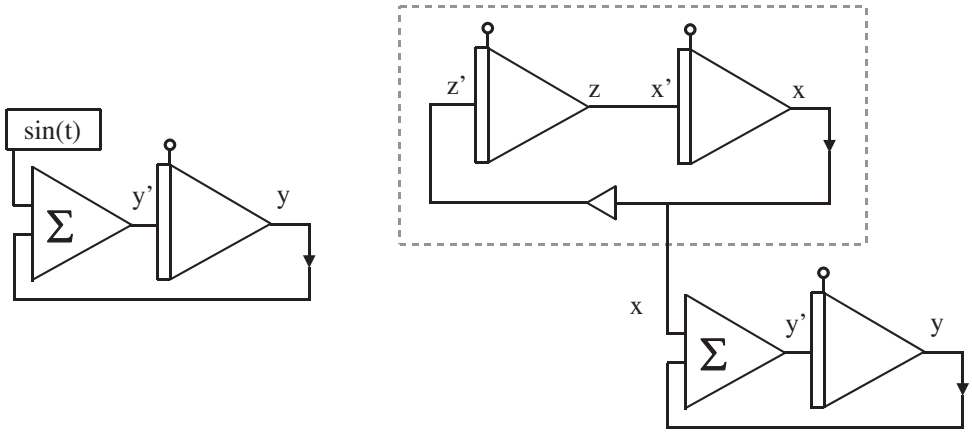
5.1 Automatická transformace zadání

Dosud jsme řešili homogenní diferenciální rovnice s konstantními koeficienty, viz kapitola 3. Jejich řešení je základem pro výpočty další skupiny, a to soustavy nehomogenních lineárních diferenciálních rovnic s konstantními koeficienty. Při řešení této skupiny rovnic se využívá transformace pravé strany rovnice požadované funkce $f(t)$. V závislosti na typu funkce $f(t)$ se původní soustava diferenciálních rovnic rozšíří o diferenciální rovnice 1. řádu. Uvedeme transformaci zadání na následující rovnici.

$$y' = y + sin(t) \qquad y(0) = y_0 \tag{5.1}$$

$$\begin{aligned} y' &= y + x & y(0) &= y_0 \\ x' &= z & x(0) &= 0 \\ z' &= -x & z(0) &= 1 \end{aligned} \tag{5.2}$$

Blokové schéma 5.1 reprezentuje zapojení nehomogenní diferenciální rovnice a její následnou automatickou transformaci pomocí diferenciálních rovnic 1. řádu. Další využití automatické

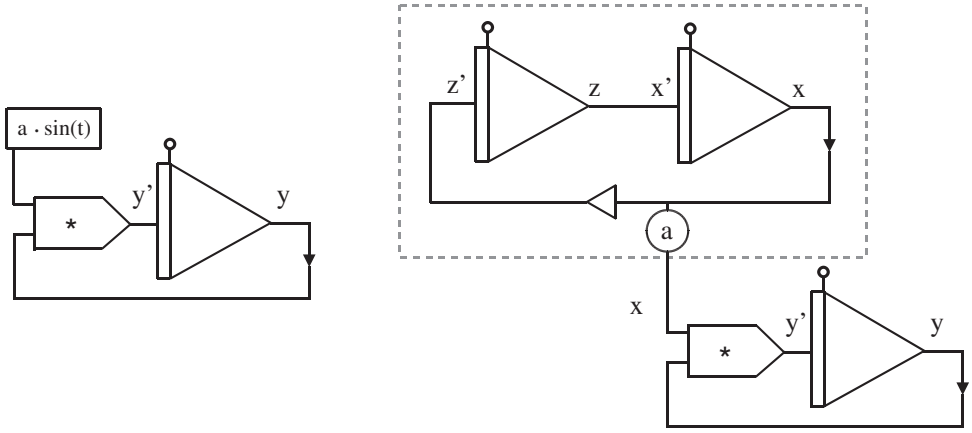


Obrázek 5.1: Blokové schéma rovnice (5.1) před a po transformaci

transformace zadání je vhodné pro řešení diferenciálních rovnic s proměnnými koeficienty. Příklad je v následující rovnici a jejím blokovém schématu.

$$y' = a \cdot y + sin(t) \qquad y(0) = y_0 \tag{5.3}$$

$$\begin{aligned} y' &= a \cdot y + x & y(0) &= y_0 \\ x' &= z & x(0) &= 0 \\ z' &= -x & z(0) &= 1 \end{aligned} \tag{5.4}$$



Obrázek 5.2: Blokové schéma rovnice (5.3) před a po transformaci

Transformace využívá metodiku tvořících diferenciálních rovnic, které jsou popsány v [7]. Využitím tvořících diferenciálních rovnic můžeme generovat všechny elementární funkce. Např. v předchozí rovnici (5.1), kde pro generování funkce *sin* jsou použity dvě diferenciální tvořící rovnice. Následně může být diferenciální rovnice rozložena na sekvenci elementárních operací a jejich kombinací. Díky transformaci můžeme řešit skupiny homogenních diferenciálních rovnic, nehomogenních diferenciálních rovnic, také rovnic s proměnnými koeficienty a skupiny nelineárních diferenciálních rovnic. Soustava je řešena paralelně několika procesory (integrátory). Jejich počet odpovídá počtu diferenciálních rovnic 1. řádu, na které byla soustava převedena. Umožňuje řešit již jednu nelineární rovnici, jak je ukázáno v následujícím příkladu.

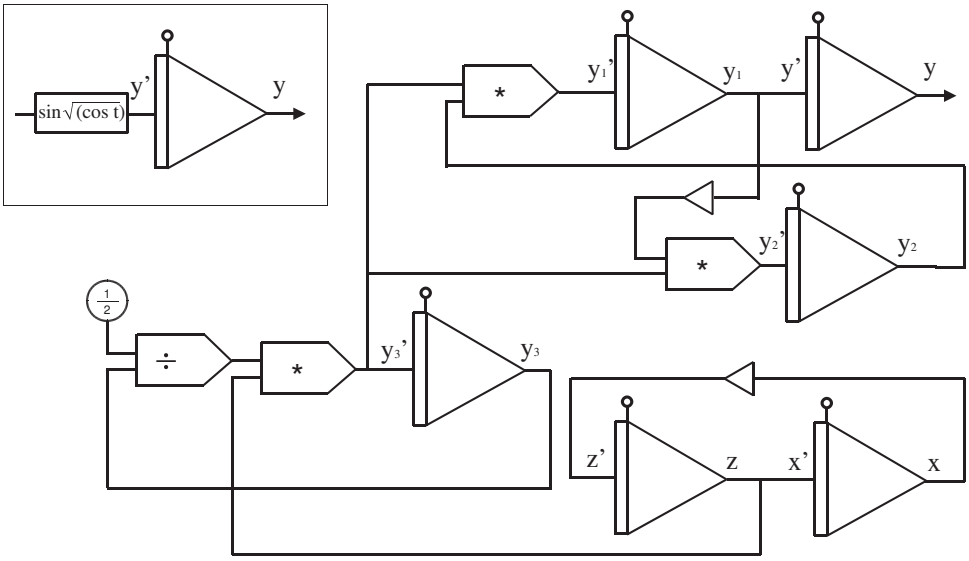
$$y' = sin \cdot (\sqrt{cos \cdot t}) \qquad y(0) = y_0 \tag{5.5}$$

$$\begin{aligned} y' &= y_1 & y(0) &= y_0 \\ y_1 &= sin \cdot y_3 \\ x &= cos(t) \\ y_3 &= \sqrt{x} \end{aligned} \tag{5.6}$$

Pro upravené rovnice platí následující vztahy:

$$\begin{aligned} y_1' &= y_2 \cdot y_3' & y_1(0) &= y_{1(0)} = sin_{y_{3(0)}} \\ y_2' &= -y_1 \cdot y_3' & y_2(0) &= y_{2(0)} = cos_{y_{3(0)}} \\ y_3' &= \frac{1}{2}\frac{1}{y_3} \cdot x' & y_3(0) &= y_{3(0)} = \sqrt{x_0} \\ x' &= z & x(0) &= 1 \\ z' &= -x & z(0) &= 0 \end{aligned} \tag{5.7}$$

Výsledné řešení rovnice (5.5) získáme na výstupu integrátoru Y. Blokové schéma, které



Obrázek 5.3: Blokové schéma rovnice (5.5) před a po transformaci

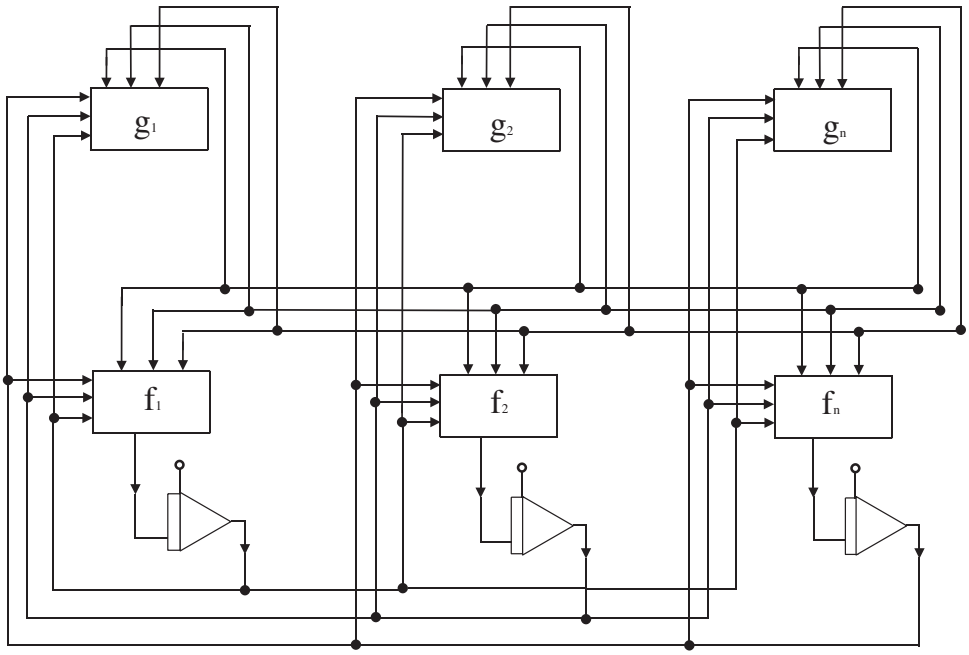
znázorňuje obrázek 5.3, představuje zapojení tvořících diferenciálních rovnic. Počet těchto rovnic je ekvivalentní počtu integrátorů. Metoda Taylorovy řady poskytuje velmi efektivní řešení homogenních diferenciálních rovnic, jak bylo uvedeno v kapitole 3.3. Pomocí tvořících diferenciálních rovnic, které převádějí soustavy na diferenciální rovnice 1. řádu, můžeme řešit soustavy velmi efektivně a paralelně.

5.2 Návrh propojovací sítě

Cílem je vytvoření vhodné propojovací sítě, která nám umožní propojení výstupů integrátorů na různé vstupy podle konkrétního výpočetního schématu (řešené soustavy diferenciálních rovnic). Pro výpočet obecné diferenciální rovnice je nutné použít obecnou propojovací síť, ať statickou či dynamickou, která je na obrázku 5.4. Díky použití FITkitu a jeho rekonfigurovatelného hradlového pole (FPGA) jsme schopni snadno získat řešení obecných diferenciálních rovnic.

Nejvhodnější propojovací sítí (pro nepříliš rozsáhlé systémy) je křížový přepínač, který je tvořen multiplexory, jak bylo popsáno v podkapitole 2.2.1. Ze všech sítí je nejvhodnější, protože má nejmenší zpoždění a umožňuje propojení více vstupů na jeden výstup, což je žádané pro řešení diferenciálních rovnic pomocí integrátorů. Zároveň je využito propojení řídících signálů pro integrátory, čímž se nahraje počáteční podmínka a integrační krok pro jednotlivé integrátory. Výsledek je možné získat z kteréhokoliv výstupu libovolného integrátoru využitím zapojení multiplexoru.

Systémy můžeme z hlediska komunikace rozlišit na sériové a paralelní. Principiálně se od sebe moc neliší, hlavním rozdílem je počet rozvodů, který jsou použity. Pro platformu FITkit je využití paralelně-paralelního integrátoru možné pro propojení pouze v případě snížení počtu bitů registrů. Paralelně-paralelní integrátor pracuje na 16 bitech, protože pro 32 bitovou verzi je FPGA FITkitu malé.

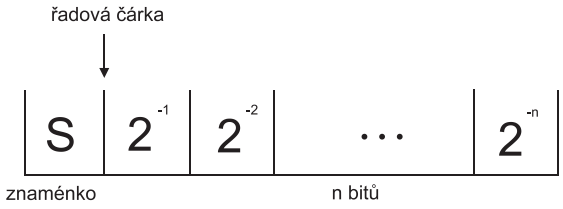


Obrázek 5.4: Obecná propojovací síť

5.3 Implementace propojovací sítě

Tato kapitola se zabývá popisem technické realizace propojovací sítě, jejíž návrh je popsán v kapitolách 2, 4 a 5. Bude popsána implementace propojovací sítě pro tři integrátory ve verzích sériově-paralelní a paralelně-paralelní. Následně uvedeme srovnání obsazení hardware pro obě verze integrátorů.

Koncepce sériově-paralelního a paralelně-paralelního integrátoru je blíže popsána v kapitole 4. Jak bylo uvedeno v dané kapitole, integrátory fungují jako aritmeticko-logické jednotky (ALU). Tyto jednotky umožňují základní matematické operace. Výpočet probíhá v pevné řadové čárce ve formátu, jehož reprezentace je na obrázku 5.5. Nejvyšší významový bit slouží jako znaménko, další je řadová čárka a n bitů reprezentujících desetinnou část zobrazovaného čísla. Vytvořená propojovací síť pracuje s křížovými přepínači, viz podka-

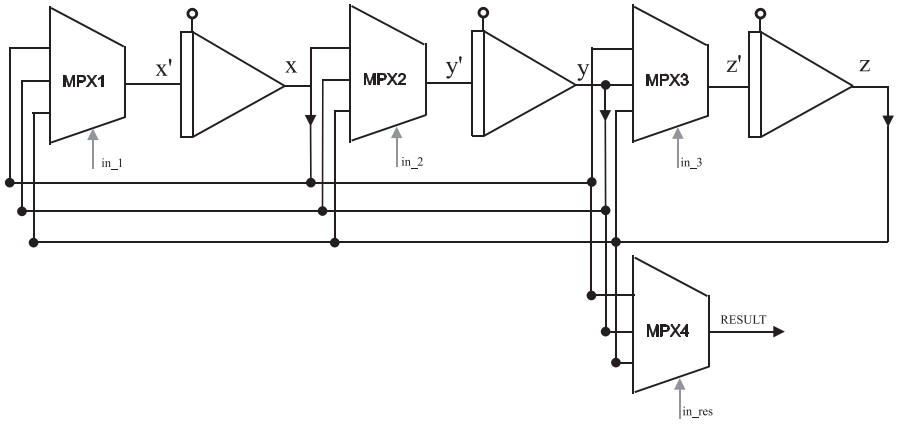


Obrázek 5.5: Formát zobrazovaného čísla

pitola 2.2.1. Křížové přepínače jsou tvořeny třívstupými multiplexory pro každý integrátor a jeden multiplexor pro registr výsledku. Inicializace propojení pro různé diferenciální rovnice je staticky nastaveno před vlastním výpočtem, tedy nastavením cest ve směrovacích prvcích. Multiplexory jsou ovládány jednotlivými řídícími signály (in_1, in_2, in_3, in_{res}), viz obrázek 5.6. Výhodou využití multiplexorů je možnost dynamického propojení sítě i během výpočtu, jak bylo blíže popsáno v podkapitole 2.3.

Jednotlivé aritmericko-logické jednotky jsou napojeny na řídící obvody. Ty umožňují nahrání počáteční podmínky x_0, y_0, z_0 a integračního kroku h pro jednotlivé integrátory. Po provedení výpočtu slouží i pro získání konečného výsledku. Pro řízení výpočtu je využita specializovaná řídící jednotka, která pro každou jednotku vykonává daný sled operací, kterým získáme výpočet. Řídící jednotka je tvořena Moorovým automatem, který přepíná jednotlivé stavy cyklu výpočtu členu Taylorovy řady. Aktuální stav je ovlivněn předchozím stavem automatu a řídícími signály.

Pro řízení výpočtu je velmi důležité využití čítačů. Jsou potřebné pro kontrolu cyklů. Jedná se o čítače pro požadovaný počet výsledků, který v našem případě má hodnotu 1 a pro výpočet následující hodnoty je potřeba vložit získanou hodnotu z výstupu. Pro výpočet, který by nám automaticky zpracovával výsledky jeden po druhém je mikrokontroler (MCU) použitého přípravku pomalé. Dalším čítačem je počítán řád Taylorovy řady DYp_i. U sériových integrátorů je potřebný čítač pro počet součtů sčítačky a posuvů v akumulátoru a posuvném registru.



Obrázek 5.6: Implementace propojovací sítě

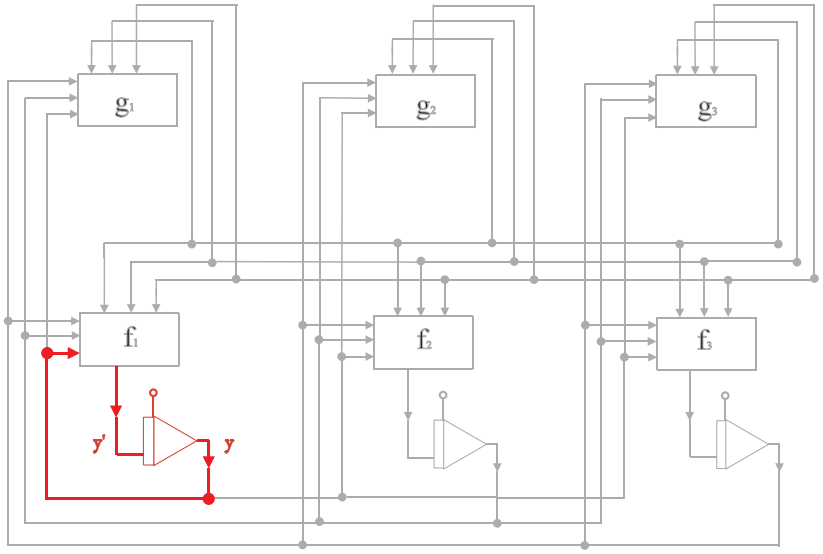
5.3.1 Simulátor propojovacího systému pro sériově-paralelní integrátor

Propojení je realizováno na platformě FITkit s využitím sériově-paralelního integrátoru. Schéma a popis funkčnosti integrátoru jsou uvedeny v kapitole 4.2. Komunikace mezi jednotlivými integrátory probíhá sériově a výpočet diferenciálních rovnic kombinovaně. Základní operace sčítání je realizována paralelní sčítačkou, naproti tomu operace násobení je realizována sériově na principu Boothova algoritmu.

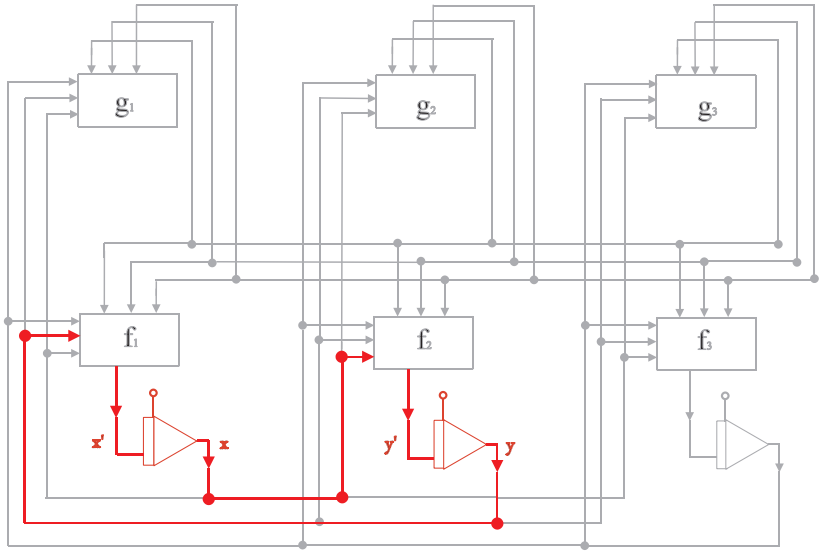
Hodnoty pro počáteční podmínky x_0, y_0, z_0 a integrační krok h jsou nahrány do jednotlivých aritmeticko-logických jednotek. Počáteční podmínky se zadávájí přes terminál FITkitu v 8 místné hexadecimální reprezentaci 32 bitového čísla. Následně dojde k převodu hodnoty a uložení hodnoty vektoru přes sériové rozhraní do řídící komponenty propojení. Koncepce návrhu omezuje vložení hodnot přes terminál, proto je pro tři integrátory uvedena ukázka zapojení bez jednoznačného označení příkazu pouze vložením počátečních podmínek pro každý integrátor.

Propojovací síť realizuje propojení pro tři aritmeticko-logické jednotky. Na základě jednotlivých uživatelských příkazů se generují řídící signály, které vytváří propojení sítě pro řešení požadované úlohy. Pro diferenciální rovnice zadá uživatel počáteční podmínky, proběhne výpočet a na terminál FITkitu se zobrazí výsledek.

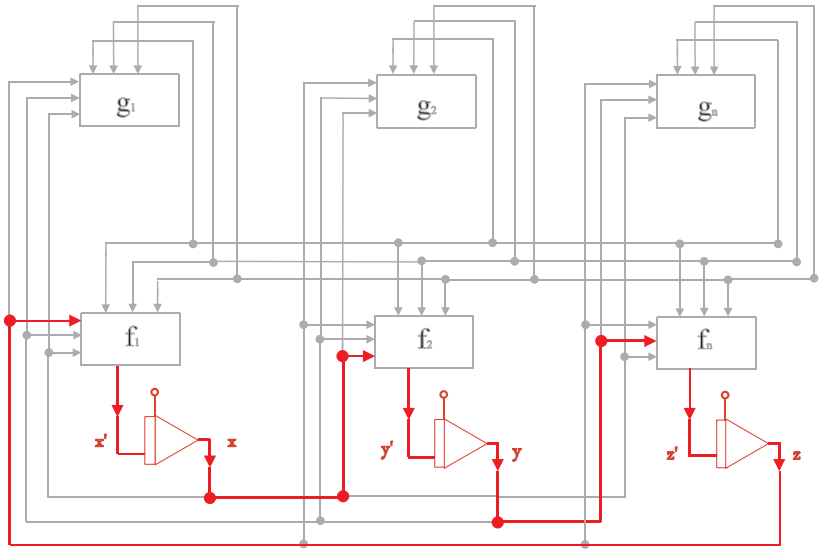
Bloková schémata zapojení diferenciálních rovnic v propojovací síti



Obrázek 5.7: Blokové schéma zapojení diferenciální rovnice $y' = y$



Obrázek 5.8: Blokové schéma propojení soustavy diferenciálních rovnic $y' = x, x' = y$



Obrázek 5.9: Blokové schéma zapojení soustavy diferenciálních rovnic $x' = z, y' = x, z' = y$

Řídící jednotka generuje signály, které jsou identické pro všechny integrátory. Výpočet probíhá ve všech jednotkách identicky v následujícím sledu. Během inicializační fáze vynulujeme čítač výsledků, délky slova a řádu metody. Nahraje se počáteční podmínka a integrační krok h do registru výsledku a posuvného registru. Před započetím výpočtu vynulujeme akumulátor. Následně probíhá sečtení ve sčítačce, zápis mezivýsledku do akumulátoru. Zde je testováno, jestli posuvný registr zpracoval všechny bity.

Pokud nejsou zpracovány všechny bity posuvného registru, uloží se hodnota ze sériového vstupu do klopného obvodu a dojde k posuvu daného registru i akumulátoru. Zvýší se hodnota čítače délky slova, který slouží ke kontrole zpracování všech bitů z registru. Akumulátor i posuvný registr se nastaví zpět do režimu zápisu. Tento cyklus se opakuje do otestování zpracování všech bitů z posuvného registru.

V případě zpracování všech bitů vynulujeme čítač délky slova. Výsledek násobení DYp_i z akumulátoru se uloží do posuvného registru, kde slouží jako podmínka pro výpočet následujícího členu Taylorovy řady. Sečtou se hodnoty registru mezivýsledku a akumulátoru, respektive přičtení hodnoty i-tého členu Taylorovy řady k celkovému výsledku. Hodnota se přesune z akumulátoru do registru výsledku. Pokud není dosaženo požadované přesnosti nastaví se řídící signály pro zopakování celé iterace výpočtu dalšího členu. Při získání požadované přesnosti zvýšíme hodnotu čítače výsledku a výsledek je uložen do posuvného registru jako počáteční podmínka pro výpočet dalšího členu. U požadovaného výsledku je nastaven příznak, který povolí čtení výsledku.

Uživatelské vstupy a výstupy

Tato část je zaměřena na přehled příkazů programu. Také uvedeme bližší popis jednotlivých diferenciálních rovnic, které jsou propojeny v dané implementaci sítě. Odpovídající bloková schémata rovnic jsou na obrázcích v části 5.3.1. Popis použitého programu je uveden v následující kapitole 5.4.2. Vytvořený program je ovládán uživatelskými příkazy, které jsou zadávány přes terminál aplikace pro ovládání FITkitu.

Následuje tabulka jednotlivých příkazů a diferenciálních rovnic, které jim odpovídají.

Tabulka 5.1: Uživatelské příkazy a odpovídající diferenciální rovnice

Úloha	**příkaz terminálu**	**diferenciální rovnice**	**výsledek**
1	$1\ DATA < n >$	$y' = y$	y_{i+1}
2	$2\ DATA < n >< m >$	$y' = x, x' = y$	x_{i+1}
3	$3\ DATA < n >< m >$	$y' = x, x' = y$	y_{i+1}
4	$< n >< m >< o >$	$x' = z, y' = x, z' = y$	x_{i+1}

Pro výpočet uvedených diferenciálních rovnic byla využita metoda Taylorovy řady, viz kapitola 3.2.1. U každé řešené úlohy si uvedeme rovnici pro výpočet členu Taylorovy řady.

- **Úloha 1** reprezentuje zapojení následující rovnice:

$$y' = y \qquad y(0) = y_0 \tag{5.8}$$

Tato úloha odpovídá rovnici (3.8), kde je uvedeno odvození řešení. Výpočet jednoho členu y_{i+1} s využitím 8 členů Taylorovy řady, který je implementován v programu, reprezentuje rovnice:

$$y_{i+1} = y_i + \frac{h}{1!} \cdot y_i + \frac{h^2}{2!} \cdot y_i + \frac{h^3}{3!} \cdot y_i + \frac{h^4}{4!} \cdot y_i + \frac{h^5}{5!} \cdot y_i + \frac{h^6}{6!} \cdot y_i + \frac{h^7}{7!} \cdot y_i + \frac{h^8}{8!} \cdot y_i \tag{5.9}$$

- **Úloha 2 a 3** reprezentuje zapojení následující soustavy diferenciálních rovnic:

$$\begin{aligned} y' &= x & y(0) &= y_0 \\ x' &= y & x(0) &= x_0 \end{aligned} \tag{5.10}$$

Výpočet jednoho členu y_{i+1} s využitím 8 členů Taylorovy řady, který je implementován v programu, reprezentuje rovnice:

$$\begin{aligned} x_{i+1} &= x_i + \frac{h}{1!} \cdot y_i + \frac{h^2}{2!} \cdot x_i + \frac{h^3}{3!} \cdot y_i + \frac{h^4}{4!} \cdot x_i + \frac{h^5}{5!} \cdot y_i + \frac{h^6}{6!} \cdot x_i + \frac{h^7}{7!} \cdot y_i + \frac{h^8}{8!} \cdot x_i \\ y_{i+1} &= y_i + \frac{h}{1!} \cdot x_i + \frac{h^2}{2!} \cdot y_i + \frac{h^3}{3!} \cdot x_i + \frac{h^4}{4!} \cdot y_i + \frac{h^5}{5!} \cdot x_i + \frac{h^6}{6!} \cdot y_i + \frac{h^7}{7!} \cdot x_i + \frac{h^8}{8!} \cdot y_i \end{aligned} \tag{5.11}$$

- **Úloha 4** reprezentuje zapojení následující soustavy diferenciálních rovnic:

$$\begin{aligned} x' &= z & x(0) &= x_0 \\ y' &= x & y(0) &= y_0 \\ z' &= y & z(0) &= z_0 \end{aligned} \tag{5.12}$$

Výpočet jednoho členu y_{i+1} s využitím 8 členů Taylorovy řady, který je implementován v programu, reprezentuje rovnice:

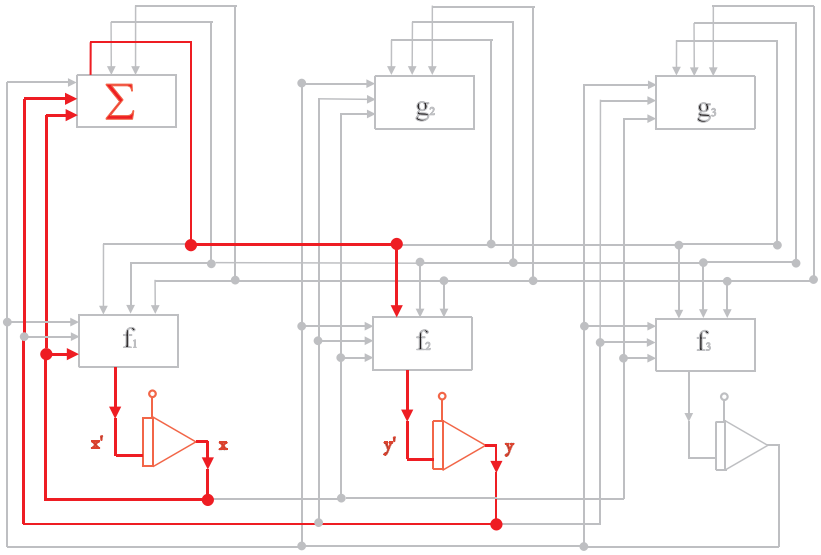
$$x_{i+1} = x_i \cdot (1 + \frac{h^3}{3!} + \frac{h^6}{6!}) + y_i \cdot (h + \frac{h^4}{4!} + \frac{h^7}{7!}) + z_i \cdot (\frac{h^2}{2!} + \frac{h^5}{5!} + \frac{h^8}{8!}) \tag{5.13}$$

5.3.2 Simulátor propojovacího systému pro paralelně-paralelní integrátor

Propojení je realizováno na platformě FITkit s využitím sériově-paralelního integrátoru. Schéma a popis funkčnosti integrátoru jsou uvedeny v kapitole 4.3. Komunikace i výpočet mezi jednotlivými integrátory probíhá paralelně. Základní operace sčítání a násobení jsou realizovány paralelní sčítačkou respektive násobičkou.

Hodnoty pro počáteční podmínky x_0, y_0, z_0 a integrační krok h jsou nahrány do jednotlivých aritmeticko-logických jednotek. Počáteční podmínky se zadávají přes terminál FITkitu ve 4 místné hexadecimální reprezentaci 16 bitového čísla. Následně dojde k převodu hodnoty a uložení hodnoty vektoru přes sériové rozhraní do řídící komponenty propojení.

Realizace zapojení na FITkitu pracuje pro paralelně-paralelní integrátory s 16 bitovými registry hodnot, jelikož pro propojení integrátorů pracujících s 32 bitovými čísly nemá zařízení dostatečnou kapacitu. Propojovací síť je implementována pro tři aritmeticko-logické jednotky. Propojení sítě generujeme na základě řídícího signálu a přepínače (multiplexory) nastaví cesty pro řešení zvolené diferenciální rovnice. Zapojení pro paralelně-paralelní integrátor je uvedeno na obrázcích 5.7, 5.8, 5.10. Implementace řídící jednotky formou koneč-



Obrázek 5.10: Blokové schéma zapojení soustavy diferenciálních rovnic $x' = x, y' = y + x$

ného automatu má následující sled činností. Pro inicializaci aritmeticko-logické jednotky vynulujeme čítač výsledků a členů Taylorovy řady. Dojde k načtení vstupních dat (počáteční podmínky pro integrátory, typ propojené úlohy a podle ní dojde k propojení vstupů na výstupy). Multiplexor se přepne pro vstup dat počáteční podmínky (DATAY0), její hodnota se nahraje do registru výsledku a součinu. Proběhne násobení hodnot v paralelní násobičce a zároveň sečtení hodnot ve sčítačce. Mezivýsledek je zaznamenán do registru výsledku. Hodnota násobení se zapíše do registru součinu, respektive hodnota DYp_i. Zvýšíme hodnotu čítače řádu metody a testujeme počet členů Taylorovy řady. V případě dosažení požadované přesnosti je čítač vynulován a je zvýšena hodnota čítače počtu výsledků. Při dosažení požadovaného výsledku nastavíme příznak, který určuje, že je výsledek připraven ke čtení.

Uživatelské vstupy a výstupy

Tato část je zaměřena na přehled příkazů programu. Také uvedeme bližší popis jednotlivých diferenciálních rovnic, které jsou propojeny v dané implementaci sítě. Popis použitého programu je uveden v následující kapitole 5.4.2.

Vytvořený program je ovládán uživatelskými příkazy, které jsou opět zadávány přes terminál aplikace pro ovládání FITkitu. Následuje tabulka jednotlivých příkazů a diferenciálních rovnic, které jim odpovídají.

Tabulka 5.2: Uživatelské příkazy a odpovídající diferenciální rovnice

Úloha	**příkaz terminálu**	**diferenciální rovnice**	**výsledek**
1	$1\ DATA < n >$	$y' = y$	y_{i+1}
2	$2\ DATA < n >< m >$	$y' = y + x, x' = x$	y_{i+1}
3	$3\ DATA < n >< m >$	$y' = x, x' = y$	x_{i+1}
4	$4\ DATA < n >< m >$	$y' = x, x' = y$	y_{i+1}

Pro výpočet uvedených diferenciálních rovnic byla využita metoda Taylorovy řady, viz kapitola 3.2.1. Úlohy $1, 3$ a 4 odpovídají provedení v sériově-paralelní verzi integrátoru využitých v propojovací síti. Jejich bloková schémata jsou uvedena na obrázcích v části 5.3.1. Výpočet jednotlivých členů je popsán v kapitole 5.3.1. Paralelně-paralelní verze obsahuje úlohu 2 rozdílnou.

- **Úloha 2** reprezentuje zapojení následující rovnice:

$$\begin{aligned} y' &= y + x & y(0) &= y_0 \\ x' &= x & x(0) &= x_0 \end{aligned} \tag{5.14}$$

Tato úloha odpovídá rovnici (3.8), kde je uvedeno odvození řešení. Výpočet jednoho členu y_{i+1} s využitím 8 členů Taylorovy řady, který je implementován v programu, reprezentuje rovnice:

$$y_{i+1} = y_i \cdot t_{mem} + x_i \cdot t_{mem} \cdot h \tag{5.15}$$

$$t_{mem} = (1 + \frac{h}{1!} + \frac{h^2}{2!} + \frac{h^3}{3!} + \frac{h^4}{4!} + \frac{h^5}{5!} + \frac{h^6}{6!} + \frac{h^7}{7!} + \frac{h^8}{8!}), \tag{5.16}$$

kde t_{mem} představuje jednotlivé přírůstky členů Taylorovy řady.

5.4 Platforma FITkit

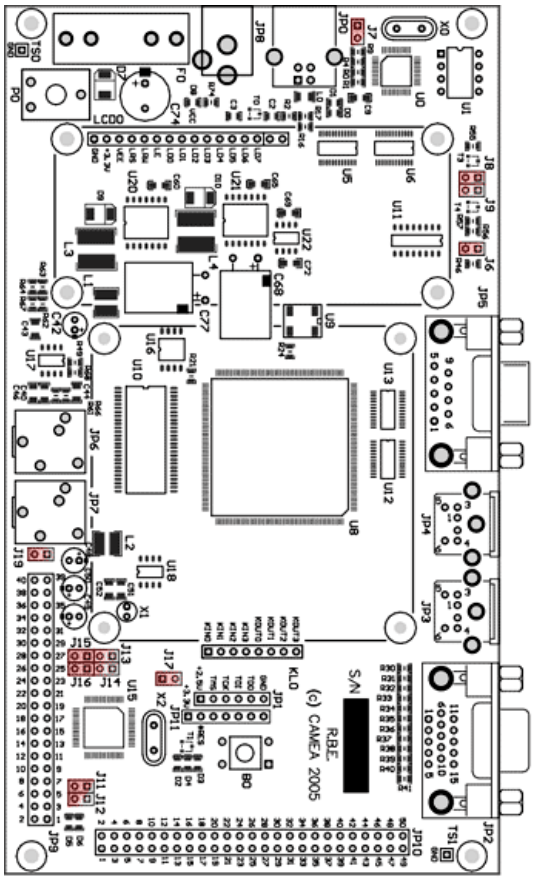
Implementace propojovací sítě je vytvořena pro použití na přípravku FITkit. Pro výpočet jsou důležité propojky JP10(5) (signál z FPGA) a JP9(26) (signál z MCU), které je třeba propojit pro správnou funkci řadiče přerušení. Následující popis a schéma je převzato z [5].

5.4.1 Popis FITkitu

FITkit je samostatný hradware, který obsahuje výkonný mikrokontrolér s nízkým příkonem, hradlové pole FPGA (Field Programmable Gate Array) a řadu periferií. Důležitým aspektem je využití pokročilého reprogramovatelného hardwaru na bázi hradlových polí FPGA jenž lze, podobně jako software na počítači, neomezeně modifikovat pro různé účely dle potřeby. Generování programovacích dat pro FPGA z popisu v jazyce VHDL probíhá zcela automaticky pomocí profesionálních návrhových systémů, které jsou uživateli k dispozici. Software pro ovládání mikrokontroleru (MCU) se tvoří v jazyce C a do spustitelné formy se překládá pomocí GNU překladače.

Schéma a komponenty FITkitu

Současná verze obsahuje tyto komponenty:

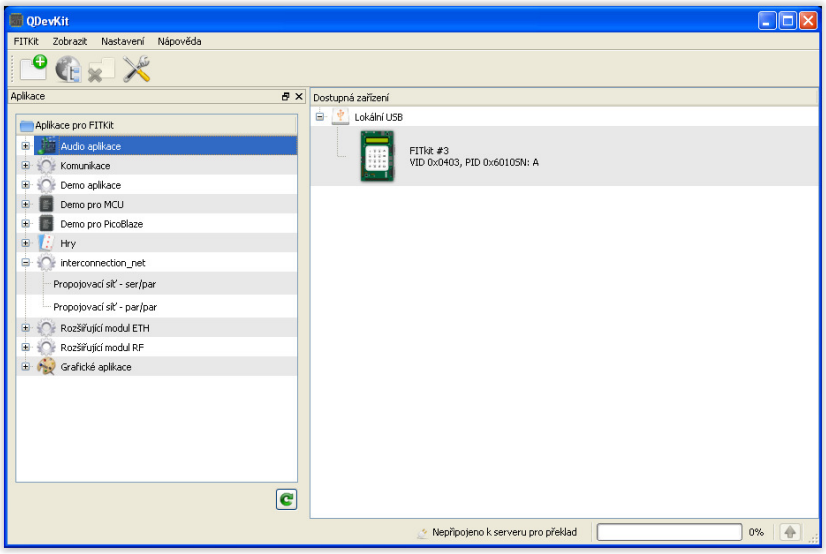


- MCU rodiny MSP430 (Texas Instruments)
- FPGA Spartan 3 XC3S50-4PQ208C nebo XC3S400 - 4PQ208C (Xilinx)
- USB převodník FT2232C
- audio rozhraní
- konektory PS2
- rozhraní VGA
- konektor RS232
- DRAM 8x8Mbit
- klávesnice
- řádkový LCD displej
- rozšiřující konektory

Obrázek 5.11: Schéma přípravku FITkit

5.4.2 Popis práce s programem

Veškerý potřebný software a návody ke zprovoznění FITkitu jsou dostupné v [5]. Základní aplikací pro práci a ovládání FITkitu je QDevKit. Součástí přiloženého CD je složka *interconnection_net*, kterou je potřeba nahrát do adresáře *apps*. Tento adresář je vytvořen při instalaci aplikace QDevKit. Po nahrání složky do adresáře se zobrazí v levé části aplikace, viz obrázek 5.12. V levé části QDevKitu vybereme aplikaci, kterou chceme do FITkitu nahrát. Pravým tlačítkem myši zvolíme možnost *naprogramovat*. Po dokončení zvolené akce je vybraná aplikace nahrána do FITkitu a připravena ke spuštění.

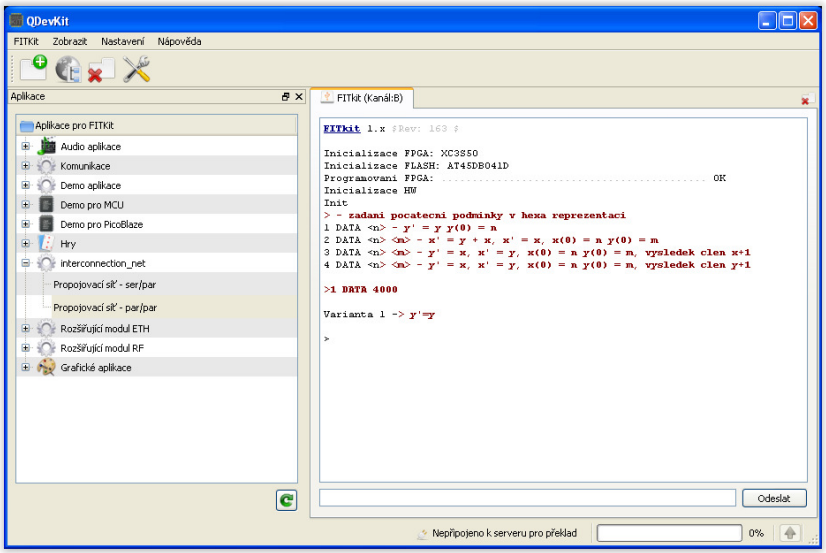


Obrázek 5.12: Aplikace QDevKit

Spuštění programu

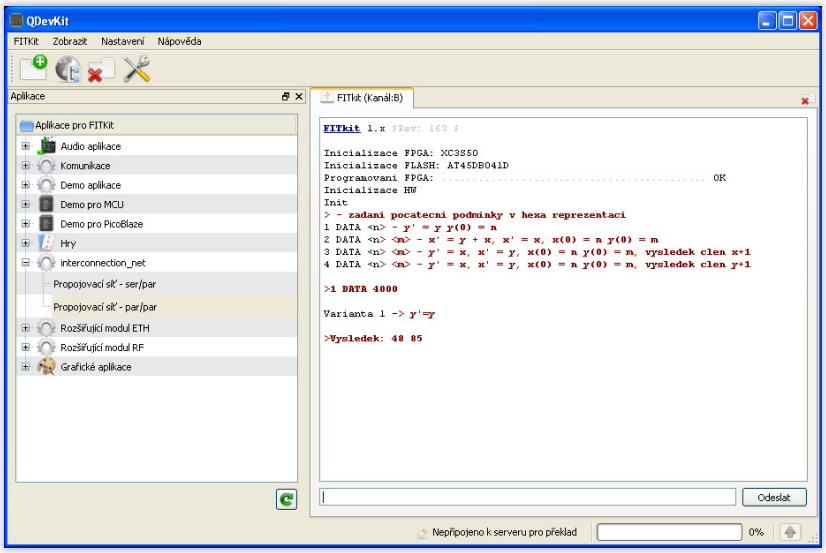
Pro spuštění nahrané aplikace klikneme na ikonu FITkitu v pravé části aplikace. Aplikaci budeme ovládat přes terminál QDevkitu, který je připojen na rozhraní mikrokontroleru. Propojky J8, J9 a J11, J12 musí být pro správnou funkčnost uzavřeny. V případě potřeby je možné pomocí příkazu HELP zobrazit nápovědu. Uživatelské příkazy pro ovládání programu jsou zobrazeny tamtéž. Můžeme je nalézt také přímo v terminálu a v přiloženém popisu programu.

Výpočetní úlohy potřebují zadat počáteční podmínku, která je součástí příkazu pro terminál. Hodnotu zapisujeme v hexadecimální reprezentaci. U sériově-paralelní verze je počáteční podmínka zadávána pro 32 bitové číslo, pro paralelně-paralelní verzi 16 bitové číslo (více viz podkapitoly 5.3.1, 5.3.2). Ukázka zadání příkazu s počáteční hodnotou je zobrazena na následujícím obrázku.



Obrázek 5.13: Aplikace QDevKit: zadání příkazu

K získání vypočtené hodnoty je nutné propojit pin JP10(5) (signál z FPGA) s pinem JP9(26) (signál z MCU). Toto propojení je nezbytné pro správnou činnost přerušení. Po propojení se na terminál vypíše hodnota výsledku (obrázek 5.14). Z důvodu pomalého zpracování mikrokontroleru je potřeba výsledek zadat jako počáteční podmínku pro výpočet následujícího členu.



Obrázek 5.14: Aplikace QDevKit: získání výsledku

Kapitola 6

Závěr

V této práci bylo uvedeno základní seznámení s numerickou integrací a jejím užití pro výpočet diferenciální rovnic. Byly popsány nejčastěji užívané metody pro numerickou integraci s větším zaměřením na metodu Taylorovy řady. Tato metoda byla následně využita pro výpočet, čímž byla také potvrzena její vyšší efektivnost oproti metodě Runge-Kutta.

Teoretický základ této práce spočívá v uvedení topologií propojovacích sítí a jejich využití pro různé systémy. Propojovací síť je určena pro aritmeticko-logické jednotky, tedy integrátory. Proto je v práci uveden bližší popis a funkčnost integrátorů pro tři hlavní skupiny těchto aritmeticko-logických jednotek (sériově-sériové, sériově-paralelní, paralelně-paralelní). Integrátory pracují v pevné řadové čárce a jsou vhodné pro řešení jednoduchých diferenciálních rovnic.

V hlavní část práce jsme se zabývali analýzou propojovacích sítí pro výběr vhodné sítě, která disponuje možností řešit libovolné diferenciální rovnice daným propojením (nastavením cest). Propojovací síť byla navržena pro sériově-paralelní a paralelně-paralelní integrátory. Bylo potvrzeno, že z hlediska prostorové složitosti je vhodnější sériově-paralelní verze i za cenu nižší rychlosti než u druhé uvedené verze. Pro každou síť jsou uvedeny její způsoby zapojení včetně příslušných diferenciálních rovnic, které mohou být daným systémem řešeny.

Propojovací síť je navržena pro využití na platformě FITkit, respektive programovatelném hradlovém poli. Síť propojuje tři integrátory a umožňuje zadáním počátečních hodnot řešit různé diferenciální rovnice či jejich soustavy. Blokové schéma zapojených úloh bylo také popsáno v této práci. Zdrojové kódy k hardwarové realizaci jsou uloženy na přiloženém CD. Řídící obvody i algoritmy je možné nadále optimalizovat. V návaznosti na tuto práci je možné využít větších hradlových polí nebo samostaných chipů, pro které by mohla být současná propojovací síť rozšířena, což by umožnilo řešit složitější soustavy diferenciálních rovnic. Možností dalšího výzkumu je například také porovnání prostorové a časové efektivity důležité zejména z hlediska využití co nejvyššího počtu integrátorů.

Literatura

[1] BRABEC, J.; HRŮZA, B.: *Matematická analýza II.* SNTL/ALFA, 1986.

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

[3] DUATO, J and YALAMANCHILI, S and NI, L: *Interconnection networks.* Morgan Kauffman, první vydání, 2003, Dostupné také z: http://www.openisbn.com/preview/1558608524/.

[4] DVOŘÁK, Václav: *Architektura a programování parelelních systémů.* Vysoké učení technické v Brně, nakladatelství VUTIUM, 2004, ISBN 80-214-2608-X.

[5] Fakulta informačních technologií VUT Brno: Domovská webová stránka FIT-kitu. [online], [cit. 5-5-2015], http://merlin.fit.vutbr.cz/FITkit/.

[6] HALUZÍKOVÁ, Anežka: *Numerické metody.* Vysoké učení technické v Brně, 1989, ISBN 80-214-0039-0.

[7] HOLUB, O.: *Numerické řešení rozsáhlých soustav diferenciálních a algebraických rovnic.* Diplomová práce, FEI, VUT, 1999.

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

[9] HWANG, Enoch O.: *Digital logic and microprocessor design with VHDL.* Thomson, 2006, dostupné také z: http://cec.shfc.edu.cn/download/2c91bf1b-480e-4a62-95f6-a7d1c88a0a8b.pdf.

[10] KNICHAL, V.; BAŠTA, A.; PIŠL, M.; aj.: *Matematika I.* SNTL/ALFA, 1965.

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

[12] KUNOVSKÝ, Jiří: *Modern Taylor series method.* Faculty of Electrical Engineering and Computer Science, 1994.

[13] MUSILOVÁ, Jana and Pavla MUSILOVÁ: *Matematika: pro porozumění i praxi: netradiční výklad tradičních témat vysokoškolské matematiky.* Vysoké učení technické v Brně, nakladatelství VUTIUM, první vydání, 2012, ISBN 978-80-214-4071-5.

[14] SEKANINA, L.: Studijní opora předmětu INP. Technická zpráva, FIT Vysoké učení technické Brno, 2006.

[15] TVRDÍK, Pavel: *Parallel algorithms and computing.* České vysoké učení technické v Praze, vydavatelství ČVUT, třetí vydání, 2003, ISBN 80-01-02824-0.

Příloha A

Obsah CD

Přiložené CD obsahuje:

- Zdrojové texty této práce ve formátu LaTeX
- Text této práce ve formátu PDF
- Zdrojové kódy navržené propojovací sítě pro FITkit
- Vzorové vstupy a výstupy pro výpočty
- Popis činnosti navrženého systému