VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV MIKROELEKTRONIKY

DEPARTMENT OF MICROELECTRONICS

NÁVRH ARITMETICKÉ JEDNOTKY V PEVNÉ ŘÁDOVÉ ČÁRCE PRO OBVODY FPGA

IMPLEMENTATION OF FIXED-POINT ARITHMETIC UNIT IN FPGA

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — Vít Kalocsányi

AUTHOR

VEDOUCÍ PRÁCE — Ing. Vojtěch Dvořák

SUPERVISOR

BRNO 2022

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Mikroelektronika a technologie**

Ústav mikroelektroniky

Student: Vít Kalocsányi ***ID:*** 220877

Ročník: 3 ***Akademický rok:*** 2021/22

NÁZEV TÉMATU:

Návrh aritmetické jednotky v pevné řádové čárce pro obvody FPGA

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se se zápisem čísel ve formátu pevné řádové čárky a se způsobem výpočtu aritmetických operací s takovými čísly. Navrhněte bitově přesný model aritmetické jednotky v Matlabu. Aritmetická jednotka musí umožňovat nastavení bitové šířky jednotlivých části čísla a bude implementovat základní operace sčítání, odčítání, násobení a dělení a případně další zvolené dopňující operace. V rámci bakalářské práce popište aritmetickou jednotku v jazyce VHDL a s pomocí simulací ověřte, že výsledky operací se shodují s referenčními modely v Matlabu. Proveďte implementaci pro zvolenou cílovou technologii a uveďte odhad spotřeby zdrojů a maximální pracovní frekvence.

DOPORUČENÁ LITERATURA:

Podle pokynů vedoucího práce

Termín zadání: 7.2.2022 ***Termín odevzdání:*** 2.6.2022

Vedoucí práce: Ing. Vojtěch Dvořák

doc. Ing. Jiří Háze, Ph.D.
předseda rady studijního programu

ABSTRAKT

Tato práce se zabývá návrhem aritmetické jednotky pro práci s čísly v pevné řádové čárce pro obvody FPGA a jejím modelem v Matlabu. V práci je představena reprezentace čísel v digitálních obvodech a základní i vybrané doplňující aritmetické operace s čísly v pevné řádové čárce. Dále je navrhnut model aritmetické jednotky v Matlabu, je popsána realizace této jednotky v jazyce VHDL a provedena její implementace do obvodu FPGA. Na závěr je ukázán konkrétní příklad využití navrhnutého modelu aritmetické jednotky pro simulaci složitých systémů v prostředí Simulink.

KLÍČOVÁ SLOVA

Aritmetická jednotka, pevná řádová čárka, Matlab, VHDL, FPGA

ABSTRACT

This thesis deals with a design of fixed-point arithmetic unit for FPGA circuits and its model in Matlab. The thesis explains a number representation in digital circuits and both basic and selected additional arithmetic operations with fixed-point numbers. The arithmetic unit's model is designed in Matlab, the realization of the unit in VHDL is described and its implementation into FPGA is carried out. A specific example of use of designed arithmetic unit's model for simulation of complex systems in Simulink environment is shown at the end of the thesis.

KEYWORDS

Arithmetic unit, fixed point, Matlab, VHDL, FPGA

BIBLIOGRAFICKÁ CITACE

KALOCSÁNYI, Vít. Návrh aritmetické jednotky v pevné řádové čárce pro obvody FPGA. Brno, 2022. Dostupné také z: https://www.vutbr.cz/studenti/zav-prace/detail/142779. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav mikroelektroniky. Vedoucí práce Vojtěch Dvořák.

Prohlášení autora o původnosti díla

Jméno a příjmení studenta:	Vít Kalocsányi
VUT ID studenta:	220877
Typ práce:	Bakalářská práce
Akademický rok:	2021/22
Téma závěrečné práce:	Návrh aritmetické jednotky v pevné řádové čárce pro obvody FPGA

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne: 2. června 2022 .....

podpis autora

PODĚKOVÁNÍ

Děkuji vedoucímu bakalářské práce Ing. Vojtěchu Dvořákovi za účinnou metodickou, pedagogickou a odbornou pomoc, za výbornou komunikaci a všechny rady při zpracování mé bakalářské práce.

V Brně dne: 2. června 2022 .....

podpis autora

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Aritmetická jednotka je jednou ze základních součástí mikroprocesorů a centrálních procesorových jednotek. Její hlavní funkcí je provádění aritmetických operací, mezi které patří základní operace sčítání, odčítání, násobení a dělení. Dle konkrétního návrhu mohou aritmetické jednotky provádět i další typy výpočtů, jako je například operace modulo, výpočet goniometrických funkcí, odmocňování a jiné. Funkce aritmetické jednotky je řízená pomocí řadiče procesoru a řídících signálů nebo kódů, které dávají aritmetické jednotce pokyn k provedení určitých operací [1].

Aritmetická jednotka je navrhována pro práci s konkrétním formátem číslic s pevně daným bitovým rozsahem. Z pravidla se využívají dva formáty čísel. První formát je formát pevné řádové čárky, který je vhodný pro práci s celými čísly a racionálními čísly s omezenou přesností. Pro práci s racionálními čísly velkých rozsahů nebo s předem neznámou velikostí čísel se využívá formátu plovoucí řádové čárky.

Hlavním cílem této práce je seznámení s procesem provádění aritmetických operací s čísly ve formátu pevné řádové čárky, navrhnutí bitově přesného modelu aritmetické jednotky pro práci s těmito čísly v prostředí Matlab a samotný návrh této jednotky v jazyce VHDL a jeho implementace do obvodu FPGA.

Text práce je rozdělen do šesti základních částí. V kapitole 1 jsou představeny formáty reprezentace čísel v digitálních obvodech. Kapitola 2 představuje základní aritmetické operace s čísly ve formátu pevné řádové čárky a v kapitole 3 je zpracován model aritmetické jednotky pro práci s těmito čísly v prostředí Matlab a Simulink. Realizace aritmetické jednotky v jazyce VHDL a její implementace je popsána v kapitole 4. Následuje představení praktické ukázky využití navrhnutého modelu jednotky v prostředí Simulink v kapitole 5. Kapitola 6 obsahuje shrnutí celé bakalářské práce.

1 REPREZENTACE ČÍSEL V DIGITÁLNÍCH OBVODECH

V digitálních obvodech jsou využity různé reprezentace čísel v závislosti na tom, s jakým druhem čísel je třeba provádět aritmetické operace. Reprezentace čísel je možná například celočíselně bez znaménka pro obor přirozených čísel nebo celočíselně se znaménkem pro obor celých čísel. Racionální čísla je možné reprezentovat ve formátu pevné nebo plovoucí řádové čárky. Dále je možné reprezentovat množinu komplexních čísel například ve formátu matice dvou racionálních čísel.

V této kapitole jsou blíže představeny reprezentace racionálních čísel, tedy čísel ve formátu pevné řádové čárky označovaného zkratkou FX z anglického výrazu „Fixed-point“ a formátu plovoucí řádové čárky označené jako FP z anglického „Floating-point”.

1.1 Čísla v pevné řádové čárce

Hodnota uložená ve formátu FX je chápana jako podmnožina racionálních čísel a je určena šířkou bitového slova a pozicí řádové čárky. Hodnotu takového čísla pak lze vyjádřit vztahem (1.1).

$$x_{FX} = a \cdot 2^{-k} \tag{1.1}$$

Kde x_{FX} je hodnota uložení ve formátu FX, $a \in Z$, $k \in Z^+$ a zároveň hodnota k určuje polohu řádové čárky [2] a jedná se o konstantní hodnotu, která se nemění ani při výpočtech. Příklad osmibitového čísla ve formátu FX je znázorněn v tabulce 1.1.

Počet bitů I před řádovou čárkou udává celočíselnou část hodnoty a počet bitů F za pozicí řádové čárky udává zlomkovou část. Obecně se počty I a F bitů mohou lišit, pro implementaci v hardwaru je však velmi výhodné pracovat se stejným počtem bitů v každém čísle, a tedy i se stejnou pozicí řádové čárky [2].

Tabulka 1.1 Reprezentace osmibitového čísla ve formátu FX a pozicí řádové čárky mezi 4. a 5. bitem

Bitová pozice	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
Váha bitu	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}

Pro reprezentaci záporných hodnot je nejčastěji využit dvojkový doplněk, což je jednoduché kódování binární numerické hodnoty. Záporné číslo se z kladného získá bitovou inverzí a přičtení čísla 1 k výsledku bitové inverze [3]. Vzhledem k tomu, že u dvojkového doplňku MSB reprezentuje znaménko ('0' pro kladné hodnoty a '1' pro záporné hodnoty), je rozsah hodnot N-bitového slova v intervalu $\langle -(2^{N-1}), 2^{N-1} - 1\rangle$. Příklad konverze kladného čísla na záporné pomocí dvojkového doplňku je uveden v tabulce 1.2.

Tabulka 1.2 Příklad konverze kladného čísla na záporné

Binární číslo (116_{10})	**0**	**1**	**1**	**1**	**0**	**1**	**0**	**0**
Bitová inverze	1	0	0	0	1	0	1	1
Přičtení čísla 1							+	1
Dvojkový doplněk (-116_{10})	1	0	0	0	1	1	0	0

Pro účely této práce je uvažováno s reprezentací všech čísel ve formátu dvojkového doplňku s pevnou řádovou čárkou a čísla pro každou aritmetickou operaci mají stejnou bitovou šířku a stejnou pozici řádové čárky. Tyto hodnoty mohou být nastaveny a upravovány dle potřeby.

1.2 Čísla v plovoucí řádové čárce

Čísla uložená ve formátu FP jsou dána mantisou, exponentem, znaménkem a bází. Takováto čísla lze vyjádřit vztahem (1.2).

$$x_{FP} = (-1)^s \cdot B^E \cdot M \tag{1.2}$$

Kde x_{FP} je hodnota uložená ve formátu FP, s je znaménko, B je báze, E je exponent a M je mantisa.

Vzhledem k tomu, že se v digitálních obvodech pracuje vždy v binární soustavě, není potřeba ukládat bázi, jelikož má vždy hodnotu 2. Příklad struktury osmibitového čísla ve formátu FP je znázorněn v tabulce 1.3.

Mantisa udává řetězec platných číslic bez řádové čárky, jelikož se předpokládá, že řádová čárka leží v určité pozici této mantisy. Exponent pak udává celočíselnou hodnotu, může být i záporná, na kterou je umocněna báze a tou pak mantisa vynásobena. Prakticky to znamená posunutí řádové čárky v mantise o počet pozic daných exponentem, kdy kladná hodnota exponentu posouvá řádovou čárku doprava a záporná doleva [4].

Způsoby uložení čísel ve formátu FP se pak mohou lišit nejen velikostí exponentu a mantisy, ale také pořadím jednotlivých částí nebo způsobem výpočtu exponentu dle toho, jaký standard je pro uložení hodnoty využit.

V dnešní době se nejčastěji využívá standard IEEE 754, který v základní přesnosti využívá 32 bitů pro uložení číselné hodnoty, kdy na pozici MSB je uložené znaménko, za ním následuje 8 bitů exponentu a 23 bitů mantisy. Exponent je v tomto standardu zakódován pomocí kódu s posunutou nulou, což znamená, že pro získání správné hodnoty exponentu je od něj nutné odečíst číslo 127. To platí ovšem pouze pro hodnoty exponentu od 1 do 254, kdy krajní hodnoty jsou rezervovány pro speciální případy, jako je reprezentace nuly, nekonečna nebo indikace, že se nejedná o číslo. Pozice řádové čárky je v případě IEEE 754 před první pozicí mantisy, a navíc se k mantise vždy přičítá hodnota 1, čímž je efektivní délka mantisy 24 bitů, ale explicitně uloženo jich je pouze 23 [5].

Tabulka 1.3 Příklad reprezentace osmibitového čísla ve formátu FP

Bitová pozice	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
Část formátu	s	E	E	E	E	M	M	M

Formát plovoucí řádové čárky je vhodný zejména pro zpracovávání čísel, mezi kterými jsou velké rozdíly, případně pro práci s čísly, jejichž rozsah hodnot není dopředu znám. Implementace formátu FP je však náročná na složitost struktur hardwaru při operacích s těmito čísly, což může vést k mnohem větší ploše na čipu nebo pomalejším rychlostem provádění operací oproti číslům ve formátu FX. Pro vyšší rychlost a menší využitou plochu na čipu je tedy vhodnější využít formátu pevné řádové čárky.

2 ARITMETICKÉ OPERACE S ČÍSLY VE FORMÁTU PEVNÉ ŘÁDOVÉ ČÁRKY

V této kapitole jsou představeny základní aritmetické operace s binárními čísly ve formátu pevné řádové čárky, metodika jejich výpočtu a limitace dle požadavků na výstupní hodnotu.

Aritmetické operace v FX jsou podobné výpočtům s celými čísly, u operací součinu a podílu je však třeba hlídat pozici řádové čárky a v případě těchto operací pak výsledek bitově posunout, aby byl formát výsledku stejný jako vstupních operandů.

2.1 Součet

Aritmetická operace součtu ve formátu FX je shodná s operací celočíselného součtu. Výsledek součtu obecně však může nabývat o jeden bit většího rozsahu, než byl rozsah vstupních hodnot [6]. Příklad výpočtu lze vidět na obrázku 2.1.

$$\begin{array}{rr} 3.25_{10} & 11.01 \\ 2.50_{10} & +\ 10.10 \\ \hline 5.75_{10} & 101.11 \end{array}$$

Obrázek 2.1 Příklad součtu dvou binárních čísel ve formátu FX

Vzhledem k požadavku této práce je nutné rozsah výsledku aritmetické operace korigovat tak, aby byl výsledek ve stejném formátu jako čísla na vstupu operace. V případě, kdy je hodnota součtu mimo rozsah reprezentace vstupního čísla, dochází k přetečení.

Přetečení je nutné ošetřit tak, aby bylo možné získat výsledek ve stejném formátu, jako mají vstupní hodnoty a aby bylo přetečení indikováno, což je možné zajistit zpravidla dvěma způsoby. První možností je výsledek ošetřit odstraněním MSB a toto odstranění indikovat pomocí určitého indikátoru přetečení. Takováto možnost je ekvivalentní k operaci modulo (zbytek po celočíselném dělení) čísla odpovídajícího váze právě odstraněného MSB. Z příkladu na obrázku 2.1 by byla výsledkem hodnota 1.75, což odpovídá zbytku po celočíselném dělení čísla 5.75 hodnotou 4.

Druhou možností je hodnotu výstupního registru saturovat na maximální možné hodnotě, kdy v takovém případě by byl výsledek příkladu z obrázku 2.1 roven hodnotě 3.75, což odpovídá maximální možné hodnotě zapsané ve formátu FX pro danou bitovou šířku a danou pozici řádové čárky.

V případě součtu dvou záporných hodnot může také dojít k podtečení neboli přetečení do záporných hodnot, což je rovněž nutné ošetřit stejným způsobem.

2.2 Rozdíl

Jelikož rozdíl dvou čísel A-B je možné napsat jako součet A+(-B), je možné rozdíl provádět naprosto totožně jako operaci součtu. Pro operaci rozdílu je nutné pouze číslo, které má být odečteno, převést na jeho dvojkový doplněk a poté ho přičíst k číslu, ze kterého se má odečítat.

Jelikož se jedná o totožnou operaci se součtem, platí pro operaci rozdílu stejné požadavky jako pro součet, a tedy stejné nastavení rozsahu a ošetření možnosti přetečení.

2.3 Součin

Operace součinu binárních čísel lze provádět stejně jako součin decimálních čísel, což lze provádět například jako parciální součin pro každý bit jednoho činitele s celým činitelem druhým s tím, že LSB každého výsledku takovéhoto parciálního součinu je na pozici daného bitu činitele, kterým je druhý činitel násoben [7]. Následně jsou tyto parciální součiny sečteny, jak je zobrazeno na obrázku 2.2.

Pro operaci součinu ve formátu FX lze při výpočtu zanedbat řádovou čárku, vypočítat součin stejně jako součin celočíselných binárních hodnot a výsledku přiřadit řádovou čárku na pozici danou součtem zlomkových pozic vstupních hodnot [7].

3.25_{10} 11.01
2.50_{10} x 10.10
00 00
110 1
0000
1101
8.125_{10} 1000.0010

Obrázek 2.2 Příklad násobení dvou binárních čísel ve formátu FX

Vzhledem k nátuře binárního součinu je velikost výsledku dána součtem bitových šířek jednotlivých vstupních hodnot. To znamená, že v případě stejného formátu vstupních čísel je výsledná bitová šířka dvojnásobná, jak je vidět na obrázku 2.2. Z tohoto důvodu je nutné zohlednit kromě přetečení, stejně jako v případě součtu, navíc také zaokrouhlování výsledků, jelikož se zdvojnásobí také zlomková část binární hodnoty. Různé metody zaokrouhlování jsou dále popsány v kapitole 2.5.

2.4 Podíl

Způsob výpočtu podílu binárních čísel je značně komplikovanější než ostatní základní aritmetické operace, pokud se nejedná o dělení mocninou čísla 2, kdy v takovém případě se dělení změní na prostý bitový posuv doprava. V případě dělení obecnou číselnou hodnotou je nutné postup dělení algoritmizovat a provádět dělení v několika krocích.

Existuje několik různých metod algoritmů dělení, mezi dvě hlavní metody patří metoda rekurentních číslic a multiplikativní metoda, dále existují aproximační metody nebo speciální metody např. typu CORDIC [8].

V případě dělení čísel ve formátu FX je výsledná šířka binárního čísla větší, stejně jako v případě součinu, v případě dělení dvou čísel se stejnou bitovou šířkou bude tedy podíl dvojnásobně široký a je tedy nutné výsledek korigovat vůči přetečení a zaokrouhlovat pomocí jedné z metod představených v kapitole 2.5. V dělenci a děliteli musí být řádová čárka na stejné pozici, pro výsledný podíl však není důležité, na které pozici se čárka nachází a v případě výsledku bude řádová čárka vždy uprostřed tak, že celá i zlomková část čísla bude obsahovat stejný počet bitů.

V následující podkapitole jsou obsaženy obrázky znázorňující jednotlivé algoritmy. V těchto obrázcích je použit symbolický zápis: "= =" pro porovnávání hodnot, "←" pro přiřazení hodnoty, kulaté závorky pro zobrazení pozice v registru, hranaté závorky pro šířku registru, složené závorky pro spojení registrů, uvozovky pro binární hodnotu a funkce "inv ()" pro bitovou inverzi.

2.4.1 Rekurentní metody dělení

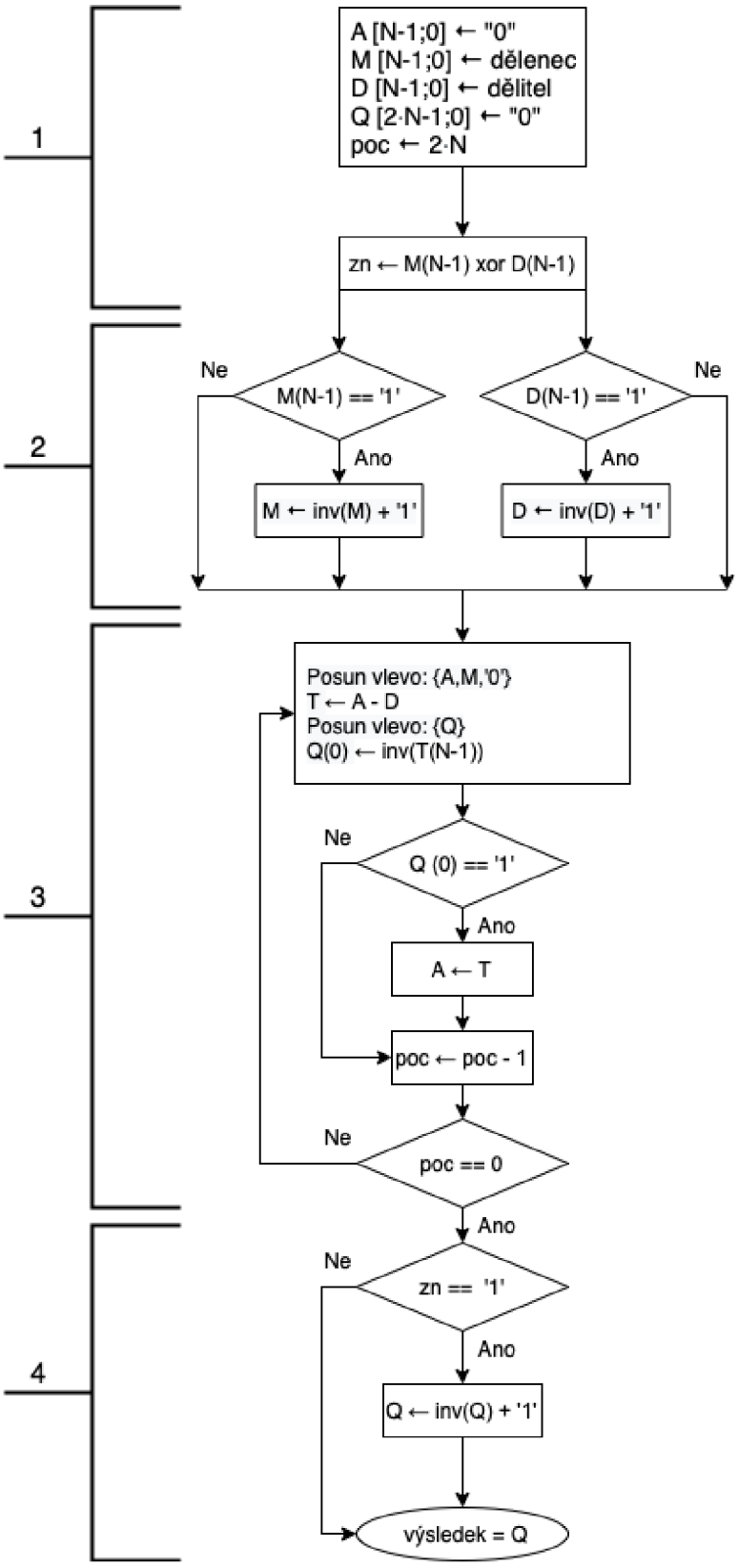
Tyto metody v každé iteraci daného algoritmu vyprodukují jeden bit výsledku, čehož dosáhnou díky opakovanému sčítání nebo odčítání vhodných členů [8].

Dělení s obnovením pomocného registru

Základním algoritmem dělení je algoritmus dělení s obnovením, který jako základní operace využívá bitový posun vlevo, odčítání dělitele od pomocného registru a případné obnovení hodnoty pomocného registru. Princip tohoto algoritmu je zobrazen na obrázku 2.3.

Algoritmus je rozdělen na celkem 4 sekce. V první části jsou přiřazeny vstupní hodnoty registrům M a D o šířce N dané bitovou šířkou vstupních hodnot, kdy registr M obsahuje hodnotu dělence a registr D hodnotu dělitele. Dále je inicializován pomocný registr A o stejné šířce a registr Q pro výsledek s dvojnásobnou šířkou, oba jsou naplněny nulovou hodnotu. Také je nastaveno počítadlo na hodnotu 2·N a přiřazeno znaménko pro výsledek jako XOR mezi hodnotami MSB registrů M a D.

Ve druhé sekci je ošetřeno počítání se zápornými čísly. Vzhledem k tomu, že je již znaménko výsledku nastaveno z operace XOR, dále je vhodné počítat jen s kladnými čísly. V případě záporného čísla se tedy provede převod čísla na kladné pomocí dvojkového doplňku.



Obrázek 2.3 Dělení s obnovením pomocného registru [8]

Ve třetí části začíná samotný výpočetní cyklus. Registry A a M jsou posouvány vlevo společně tak, že se MSB registru A zahodí, na LSB registru A se nasouvá MSB registru M a na LSB registru M se nasouvá nula. Je vytvořen dočasný registr, do kterého se přiřadí výsledek operace A-D. Následuje bitový posun registru Q a na hodnotu jeho LSB se nasune inverzní hodnota MSB dočasného registru. Pokud se takto dosadí hodnota '1', je pak do registru A přiřazena hodnota dočasného registru, v opačném případě zůstává hodnota A nezměněna. Následně se provádí dekrementace hodnoty počítadla a postupuje se k další iteraci, dokud není počítadlo vynulováno.

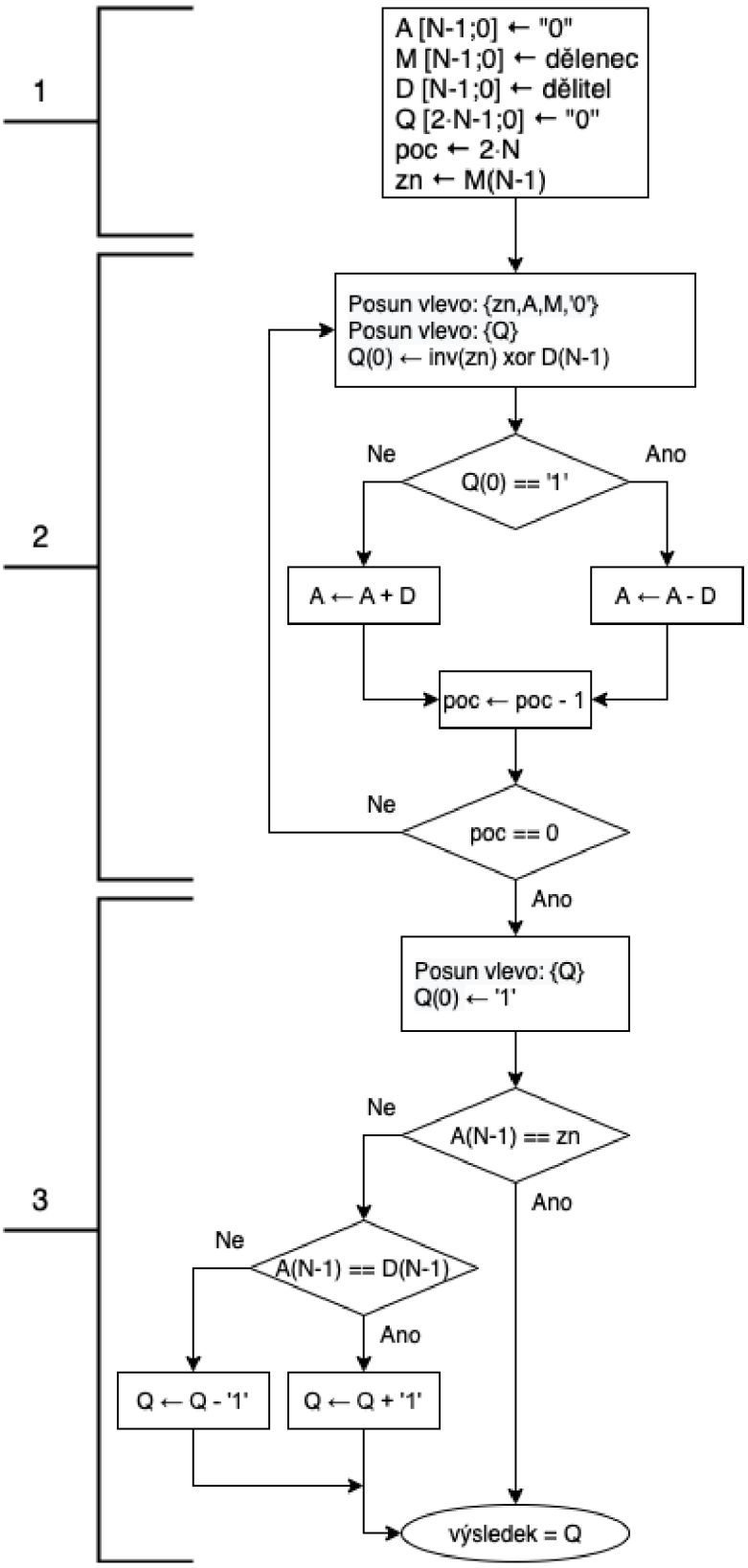
Ve čtvrté části je v registru Q hodnota podílu. V případě, že výsledek má vyjít záporný, je nutné dle indikátoru znaménka převést hodnotu výsledku na zápornou hodnotu pomocí dvojkového doplňku. Vzhledem k tomu, že výsledek je dvojnásobně široky oproti formátu vstupních hodnot, je potřeba jej korigovat zaokrouhlením a případným ošetřením přetečení tak, aby formát výsledku odpovídal požadavku.

Dělení bez obnovení pomocného registru

Algoritmus dělení bez obnovení je postaven na podobném principu jako algoritmus s obnovením, ovšem není využíváno obnovování registru A dočasným registrem. Princip algoritmu dělení bez obnovení je zobrazen na obrázku 2.4.

V první části jsou opět inicializovány vstupní registry, které mají totožný význam jako v případě dělení s obnovením. Rozdíl je pouze u indikátoru znaménka, kdy jemu je pouze přiřazen znaménkový bit dělence. V tomto případě není nutné počítat pouze s kladnými hodnotami, a proto se ve druhé části již provádí výpočet. Posuv registrů funguje podobně, s tím rozdílem, že se MSB registru A nezahazuje, ale zapisuje do indikátoru znaménka. Shodně se posouvá vlevo i s registrem Q, ale na jeho pozici LSB se nasune výsledek XOR mezi negací znaménkového indikátoru a MSB registru D. Pokud se takto dosadí hodnota '1', pak se od registru A odečte hodnota D a výsledek se ponechá v A, v opačném případě je k registru A přičtena hodnota D. Následuje stejný proces dekrementace počítadla a další iterace.

V poslední části je však nutné více korigovat výsledek než v případě algoritmu s obnovením. Po skončení všech iterací je ještě potřeba naposled posunout registrem Q a na pozici LSB dosadit '1'. Následuje porovnání indikátoru znaménka s MSB registru A, pokud se hodnoty liší, je nutné k výsledku přičíst nebo odečíst hodnotu '1' dle toho, zda se hodnoty MSB registru A a D liší, jak je vidět v třetí sekci na obrázku 2.4.



Obrázek 2.4 Dělení bez obnovení pomocného registru [8]

2.4.2 Multiplikativní metody dělení

Tyto metody mají základ v násobení a mají kvadratickou konvergenci, tedy v každém kroku produkují dvojnásobek bitů výsledku než v kroku předchozím [8]. Tyto metody musí vždy pracovat pouze s kladnými čísly.

Newton-Raphsonova metoda

Tato metoda spočívá v převedení operace podílu na operaci součinu ve formě násobení dělence převrácenou hodnotou dělitele. Aby bylo možno přistoupit k násobení, je nutné aproximovat přibližnou hodnotu převrácené hodnoty dělitele [8].

Pro volbu počáteční hodnoty aproximace je vhodné, ale byl dělitel z intervalu $\langle 0{,}5;\ 1\rangle$, čehož je dosaženo bitovým posuvem dělitele i dělence. Pak je možné pomocí vhodné konstanty určit počáteční aproximaci a od této hodnoty je následně formou Newtonovy metody určena převrácená hodnota dělitele. Počet opakování aproximace je dán s dostatečnou přesností jako logaritmus od základu dva z dvojnásobku bitové šířky vstupních hodnot, nicméně není přesně určeno, kolikrát je nutné provádět bitový posuv při určování počáteční aproximace.

Pro získání výsledků je nakonec nutné vynásobit dělence aproximovanou převrácenou hodnotou dělitele a výsledek korigovat podle znamének vstupních hodnot.

Goldschmidtova metoda

Tento algoritmus je založen na principu násobení hodnoty dělence i dělitele vhodnou shodnou konstantou tak, aby dělitel postupně konvergoval k hodnotě 1 a tím pádem dělenec konvergoval k hodnotě podílu.

Pro nalezení náležité násobící konstanty je vhodné předpokládat hodnotu dělitele v intervalu $\langle 0;\ 1\rangle$, čehož je opět dosaženo bitovým posuvem obou vstupních hodnot. Pak je možné tuto konstantu určit odečtením hodnoty dělitele od hodnoty 2. Dále je nutné touto konstantou násobit dělitele i dělence a pokračovat v dalších iteracích, jejichž počet je dán rovněž logaritmem od základu dva z dvojnásobku bitové šířky vstupních hodnot.

V tomto případě je výsledek rovnou v registru dělence a je pouze nutné upravit znaménko.

2.4.3 Porovnání jednotlivých metod

Algoritmy dělení s nebo bez obnovení patří mezi pomalé metody, což je dáno velkým počtem opakovacích cyklů. Jejich velkou výhodou je však velmi malá využitá plocha na čipu [8]. Pro aplikace, které nejsou velmi citlivé na zpoždění je tedy výhodné použít rekurentní metody dělení.

Oproti tomu jsou multiplikativní metody vhodné použít pro aplikace vyžadující menší zpoždění výpočtu. Jejich implementace je však náročnější na hardware, jednak z pohledu velikosti plochy na čipu, ale i z pohledu nejasného počtu cyklů při určování hodnoty počáteční iterace. Tento problém se však dá v rámci implementace řešit například pro Glodschmidtovu metodu pomocí náhledové tabulky.

2.5 Metody zaokrouhlování

Zaokrouhlování je operace, při které dochází k zmenšování počtu platných číslic, čímž se zvětšuje nepřesnost výsledku, ale zachovává se požadovaný rozsah pro jednodušší práci s daným výsledkem.

Metody zaokrouhlování lze dělit dle dvou kritérií. Prvním kritériem je, zda daná metoda bere ohled na střed intervalu a jak případnou střední hodnotu mezi dvěma čísly řeší. Druhým kritériem je pak směr zaokrouhlování.

Nejčastěji je využíváno metod klasického aritmetického zaokrouhlování, kdy je zaokrouhlováno směrem k nejbližší hodnotě a případná střední hodnota se zaokrouhlí směrem ke kladnému nekonečnu. Dále jsou hojně využívány zaokrouhlování nahoru nebo dolů, které neberou ohled na střed intervalu a zaokrouhlují vždy ke kladnému nebo k zápornému nekonečnu. Existuje však mnoho dalších metod zaokrouhlování, jejichž přehled je zobrazen v tabulce 2.1 [9].

Tabulka 2.1 Metody zaokrouhlování

Metoda zaokrouhlování	Směr zaokrouhlování	Řešení střední hodnoty
Aritmeticky	Nejbližší hodnota	Směrem ke kladnému nekonečnu
Aritmeticky dolů	Nejbližší hodnota	Směrem k zápornému nekonečnu
Aritmeticky k nule	Nejbližší hodnota	Směrem k nule
Aritmeticky od nuly	Nejbližší hodnota	Směrem od nuly
Aritmeticky k lichému číslu	Nejbližší hodnota	Směrem k lichému číslu
Aritmeticky k sudému číslu	Nejbližší hodnota	Směrem k sudému číslu
Absolutně dolů	Záporné nekonečno	Neřeší
Absolutně nahoru	Kladné nekonečno	Neřeší
Absolutně k nule	Směrem k nule	Neřeší
Absolutně od nuly	Směrem od nuly	Neřeší

2.6 Zjednodušená modulární redukce

Modulární redukce je matematická operace určující zbytek po celočíselném dělení. Aby nebylo nutné využívat komplikovaného dělení, lze pro určité případy modulární redukci zjednodušit na pouhé porovnávání hodnot a případné sčítání, respektive odečítání.

Pro možnost zjednodušení této operace je nutné uvažovat s tím, že vstupní hodnota musí být menší, než je hodnota dvojnásobku modulu. V případě, že je vstupní hodnotou výsledek jiné matematické operace, jejíž vstupní hodnoty byly menší, než je modulo, pak touto matematickou operací může být pouze sčítání, respektive odčítání. V případě součinu nebo podílu by mohlo dojít k tomu, že výsledek je větší než dvojnásobná hodnota modulu, a tedy by zjednodušená modulární redukce poskytla výsledek větší než modulo.

Výpočet zjednodušené modulární redukce spočívá v porovnání velikosti vstupní hodnoty s hodnotou modulu a případnému přičtení nebo odečtení modulu k této vstupní hodnotě. Tuto operaci lze provádět buď asymetricky, kdy je výsledkem hodnota z intervalu $\langle 0; mod)$, nebo také symetricky, kdy je výsledek z intervalu $\left\langle -\frac{1}{2}mod; \frac{1}{2}mod \right)$.

2.7 Goniometrické funkce

Určení goniometrických funkcí ve formátu FX je možné pomocí několika způsobů. Mezi základní z těchto způsobů patří metoda náhledové tabulky a využití algoritmu CORDIC.

Metoda náhledové tabulky spočívá ve vygenerování a uložení určitého počtu hodnot do paměti. Každá hodnota uložená v LUT odpovídá hodnotě dané goniometrické funkce pro konkrétní úhel. Při výpočtu hodnoty goniometrické funkce daného úhlu je pak tato hodnota pouze vyčtena z LUT [10].

Algoritmus CORDIC vychází z rotace vektoru po jednotkové kružnici, výpočet je prováděn iterační metodou a využívá operace bitového posunu, sčítání/odčítání a porovnávání hodnot [11].

Výhodou řešení goniometrických funkcí pomocí LUT je jednoduchá implementace a rychlost, která je omezena pouze výčtem hodnoty z paměti. Ovšem nevýhodou je velká plocha na čipu při implementaci do hardwaru v případě požadavku na velkou přesnost výsledků. Oproti tomu je implementace algoritmu CORDIC náročnější, ale je vhodná pro aplikace vyžadující menší plochu na čipu při větší přesnosti výsledků.

3 MODEL ARITMETICKÉ JEDNOTKY

Tato kapitola se věnuje popisu modelu aritmetické jednotky pro práci s čísly v pevné řádové čárce v Matlabu. V kapitole jsou popsány funkce reprezentující jednotlivé operace a následně výsledky testování jednotlivých operací vzhledem k referenčnímu výpočtu v plovoucí řádové čárce. V závěru kapitoly je představena knihovna funkcí, která usnadňuje modelování v prostředí Simulink.

3.1 Popis funkcí implementovaných v Matlabu

Společným rysem všech funkcí implementovaných v prostředí Matlab je možnost nastavit jejich chování podle požadavků uživatele. Konfigurace je sdílena pro všechny operace pomocí společné proměnné, která uchovává nastavení jednotlivých požadavků a je předávána jednotlivým funkcím při jejich volání. Toto nastavení obsahuje

- celočíselnou hodnotu bitové šířky vstupních hodnot,
- hodnotu určující pozici řádové čárky,
- logickou hodnotu určující volbu ošetření přetečení,
- volbu typu zaokrouhlování,
- hodnotu modulu společně s volbou symetrického nebo asymetrického výpočtu modulární redukce a
- náhledovou tabulku pro operaci sinus, která je automaticky generována v závislosti na bitové šířce zlomkové části a požadovaném počtu hodnot.

Vzhledem k tomu, že Matlab nativně počítá s čísly ve formátu plovoucí řádové čárky, jsou implementovány dvě funkce pro převod mezi formáty FP a FX. Funkce pro **převod čísel z formátu FP na formát FX** využívá konfigurační proměnnou pro nastavení pozice řádové čárky a typu zaokrouhlení. Samotné zaokrouhlení je při převodu realizováno pomocí standardních funkcí dostupných v Matlabu, konkrétně pro aritmetické zaokrouhlení (*round*), zaokrouhlení absolutně dolů (*floor*), absolutně nahoru (*ceil*) a absolutně k nule (*fix*). Zaokrouhlení je doplněno o vlastní funkci zaokrouhlení absolutně od nuly. Funkce převodu FP na FX také ošetřuje případ, kdy číslo v plovoucí řádové čárce přesahuje rozsah čísla v pevné řádové čárce dle nastavení volby přetečení spolu s vypsáním varování. Funkce pro **převod čísel z formátu FX do FP** pouze matematickou operací převede dané číslo při znalosti pozice řádové čárky.

Dále jsou implementovány samostatně funkce pro jednotlivé aritmetické operace. Funkce **operace součtu** provede aritmetický součet dvou čísel a ověří, zda se výsledek vejde do požadovaného bitového rozsahu. V případě přetečení nebo podtečení výsledku, dle konfigurační volby, tento výsledek ošetří požadovaným způsobem. Funkce implementující **operaci rozdílu** nejprve přiřadí druhému členu opačné znaménko a následně volá funkci operace součtu.

Funkce **násobení** provádí aritmetické násobení dvou čísel, následně provede zaokrouhlení, a nakonec ošetření přetečení. V kroku zaokrouhlení funkce nejprve rozdělí vstupní hodnoty na znaménko a absolutní číselnou hodnotu, kterou následně převede na bitový vektor. Tento bitový vektor se rozdělí dle požadovaného rozsahu zaokrouhlení a převede na zlomkové číslo ve formátu FP, kterému se následně přiřadí znaménko. Toto číslo je následně zaokrouhleno stejně, jako v případě zaokrouhlování při převodu z formátu FP na FX. Pro dokončení operace násobení je stejně jako v případě součtu ověřen rozsah výsledků a případné přetečení nebo podtečení je dle konfiguračního požadavku ošetřeno.

Dělení je implementováno pomocí funkce obsahující algoritmus dělení s obnovením pomocného registru tak, jak je popsán v kapitole 2.4.1. Vstupní hodnoty jsou převedeny na bitové vektory, je inicializován pomocný vektor a vektor pro výsledek. Dále je nastaveno počítadlo cyklu a uchováno znaménko pro výsledek. Pomocný vektor je spojen s vektorem dělence a pomocí smyčky *for* postupně vypočten výsledek tak, jak je vidět na třetí části obrázku 2.3. Výsledek je dále zaokrouhlen pomocí stejné funkce jako v případě násobení a obdobným způsobem je ošetřeno i případné přetečení nebo podtečení.

Implementace **zjednodušené modulární redukce** je řešena jednoduchým porovnáváním, zda vstupní hodnota leží v požadovaném intervalu hodnot, který je dán konfigurací této operace a hodnotou modulu. V případě, že je hodnota menší, než je spodní hranice intervalu, k této hodnotě se přičte hodnota modulu. Pokud je větší než horní hranice, pak se od této hodnoty modul odečte. Pokud vstupní hodnota intervalu vyhovuje, je tato hodnota ponechána beze změny.

Operace sinus je proveden pomocí vyhledávání v náhledové tabulce uložené v konfigurační proměnné. Kvůli tomu, aby bylo možné využívat pouze bity zlomkové části čísla a znaménkový bit, není možné získat jako výsledek hodnotu 1. V takovém případě je výsledkem maximální možná hodnota nižší než 1, tedy všechny bity zlomkové části nastaveny v '1' a znaménkový bit v '0'. **Operace kosinus** je řešena pouze voláním operace sinus s posunem fáze vstupní hodnoty odpovídajícím posunu o $\frac{\pi}{2}$.

3.2 Testování modelu aritmetické jednotky

Testování modelu aritmetické jednotky v prostředí Matlab bylo provedeno na velkém množství náhodně generovaných čísel vždy z maximálního možného rozsahu pro aktuální konfiguraci aritmetické jednotky. Testování bylo prováděno pro různé nastavení bitové šířky vstupních hodnot, pozice řádové čárky, různé typy zaokrouhlování i ošetření přetečení.

Jednotlivé náhodně generované vstupní hodnoty byly převedeny do formátu FX a zaokrouhleny tak, aby se jejich celá i zlomková část vešla do požadovaného bitového rozsahu. S každou dvojicí hodnot byly provedeny všechny 4 základní aritmetické operace (sčítání, odčítání, násobení a dělení). Zaokrouhlené vstupní hodnoty byly převedeny zpět

do formátu plovoucí řádové čárky a pomocí standardních aritmetických funkcí v Matlabu byly mezi nimi provedené stejné operace. Výsledky operací ve formátu FX byly následně převedeny na formát FP a porovnány s výsledky operací v tomto formátu. Při porovnání výsledku bylo ověřeno, že absolutní odchylky mezi výsledky jsou menší než váha LSB v daném formátu, případně že došlo k saturaci nebo přetečení dle požadovaného nastavení.

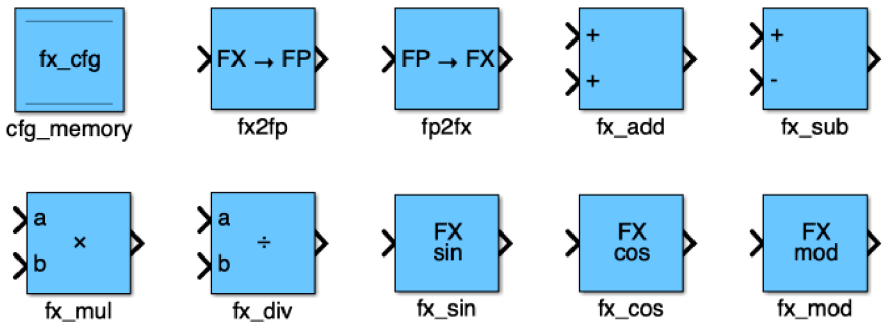
Testování goniometrických funkcí bylo provedeno samostatně, opět pro různě nastavené konfigurace aritmetické jednotky. Při testu bylo rozlišení goniometrických funkcí krokováno v rozmezí 2^8 až 2^{10} a vždy byly pro vstupní hodnoty vypočteny hodnoty sinu i kosinu ve formátu FX i FP. Tyto výsledky byly konvertovány do stejného formátu a bylo ověřeno, že absolutní odchylka mezi nimi je menší, než je váha LSB.

Zjednodušená modulární redukce byla ověřena rovněž samostatným testem s různým nastavením konfigurace a různými velikostmi modulu. Testem byla ověřena správná funkce symetrické i asymetrické konfigurace této operace při vstupních hodnotách odpovídajícím maximálně velikosti modulu pro symetrickou, respektive velikosti dvojnásobku modulu pro asymetrickou konfiguraci.

3.3 Model aritmetické jednotky v prostředí Simulink

Pro usnadnění modelování složitých systémů byla vytvořena vlastní knihovna funkcí pro prostředí Simulink využívající funkce popsány v předchozí kapitole. Tato knihovna také obsahuje funkční bloky pro simulaci konkrétních funkcí v prostředí Simulink, které jsou vidět na obrázku 3.1.

V první řadě model aritmetické jednotky vyžaduje základní konfiguraci aritmetické jednotky, což je implementováno v bloku konfigurační paměti. V tomto bloku je možné nastavení bitové šířky jednotlivých částí vstupního čísla, volba nastavení zaokrouhlování a ošetření přetečení a další parametry, jako je velikost modulu nebo rozlišení goniometrických funkcí. V modelu jsou dále implementovány bloky pro konverzi čísel mezi pevnou a plovoucí řádovou čárkou. Nakonec jsou v modelu zavedeny jednotlivé aritmetické operace pomocí samostatných funkcí.



Obrázek 3.1 Bloky aritmetické jednotky pro použití v prostředí Simulink

4 REALIZACE ARITMETICKÉ JEDNOTKY VE VHDL

Při návrhu aritmetické jednotky v jazyce VHDL byla tato jednotka rozdělena na samostatné komponenty, které implementují vždy jedinou aritmetickou operací. Jednotlivé komponenty byly navrženy jako čistě kombinační logika s možností použití vstupních a výstupních datových registrů v nadřazené komponentě dle požadavků konkrétní aplikace. Výjimkou je operace podílu, kterou je nutné provádět sekvenčně a obsahuje tedy jak kombinační, tak i sekvenční část.

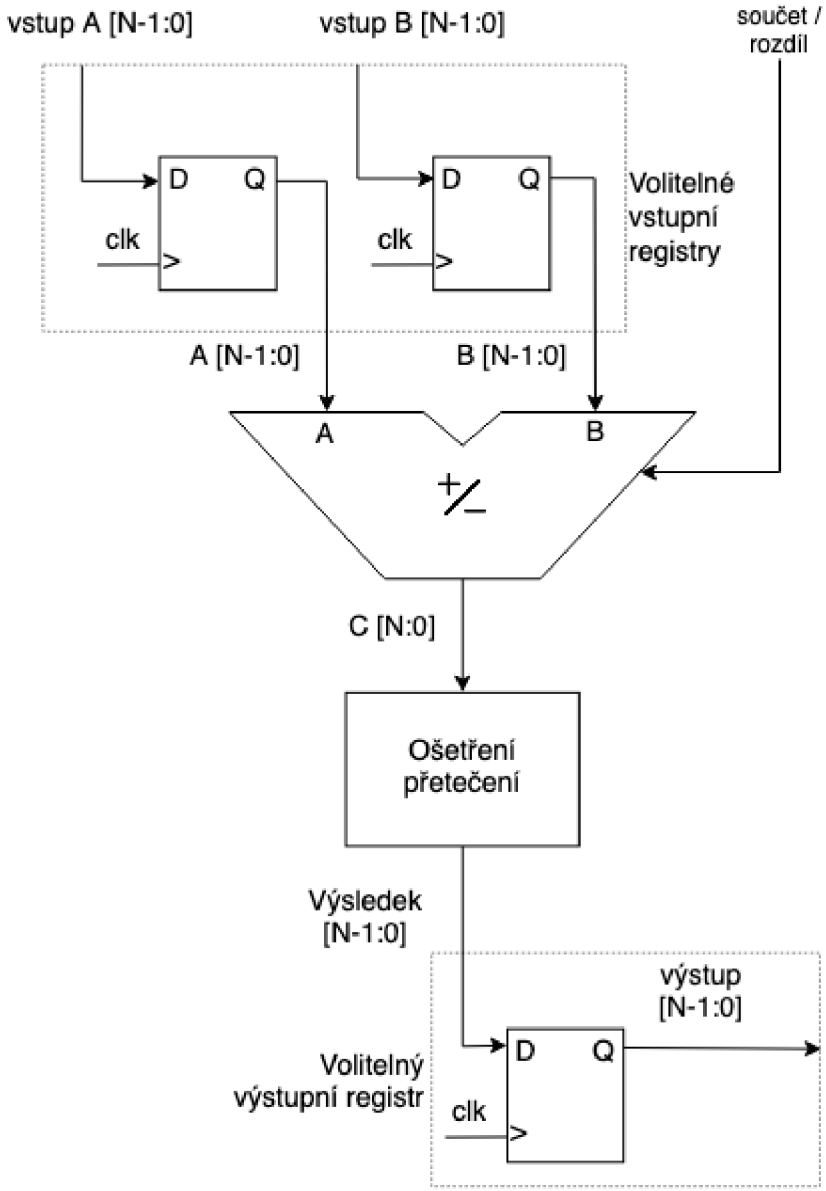
Pro možnost konfigurace je vytvořena konfigurační sloha, která je připojena v každé komponentě. V konfigurační sloze je možné nastavit konstanty určující bitovou šířku vstupních a výstupních čísel, pozici řádové čárky v těchto číslech, volbu ošetření přetečení a volbu typu zaokrouhlení. V samostatném souboru je pak uložena náhledová tabulka pro výpočet goniometrických funkcí.

4.1 Součet a rozdíl

Součet a rozdíl jsou prováděny pomocí jediné komponenty, jejíž blokové schéma je znázorněno na obrázku 4.1. Přepínání mezi operacemi součtu a rozdílu je realizováno pomocí dedikovaného řídicího signálu. Samotné operace jsou provedeny standardní funkcí z knihovny IEEE NUMERIC_STD a mezivýsledek je uložen do pomocné proměnné, která je o jeden bit širší, než je vstupní bitová šířka jednotlivých čísel.

V případě, že je povoleno přetečení výsledků, je na výstup této operace přiřazena hodnota pomocné proměnné tak, aby bylo zachováno znaménko, ale zároveň zahozeny nejvyšší bity daného čísla.

Při konfiguraci požadující saturaci výstupní hodnoty je v případě, kdy by mělo dojít k přetečení výstupu, nastavena na výstup saturační hodnota odpovídající maximální nebo minimální hodnotě při daném bitovém rozsahu čísel. V opačném případě je hodnota uložená v pomocné proměnné upravena na požadovanou bitovou šířku a zapsána na výstup.



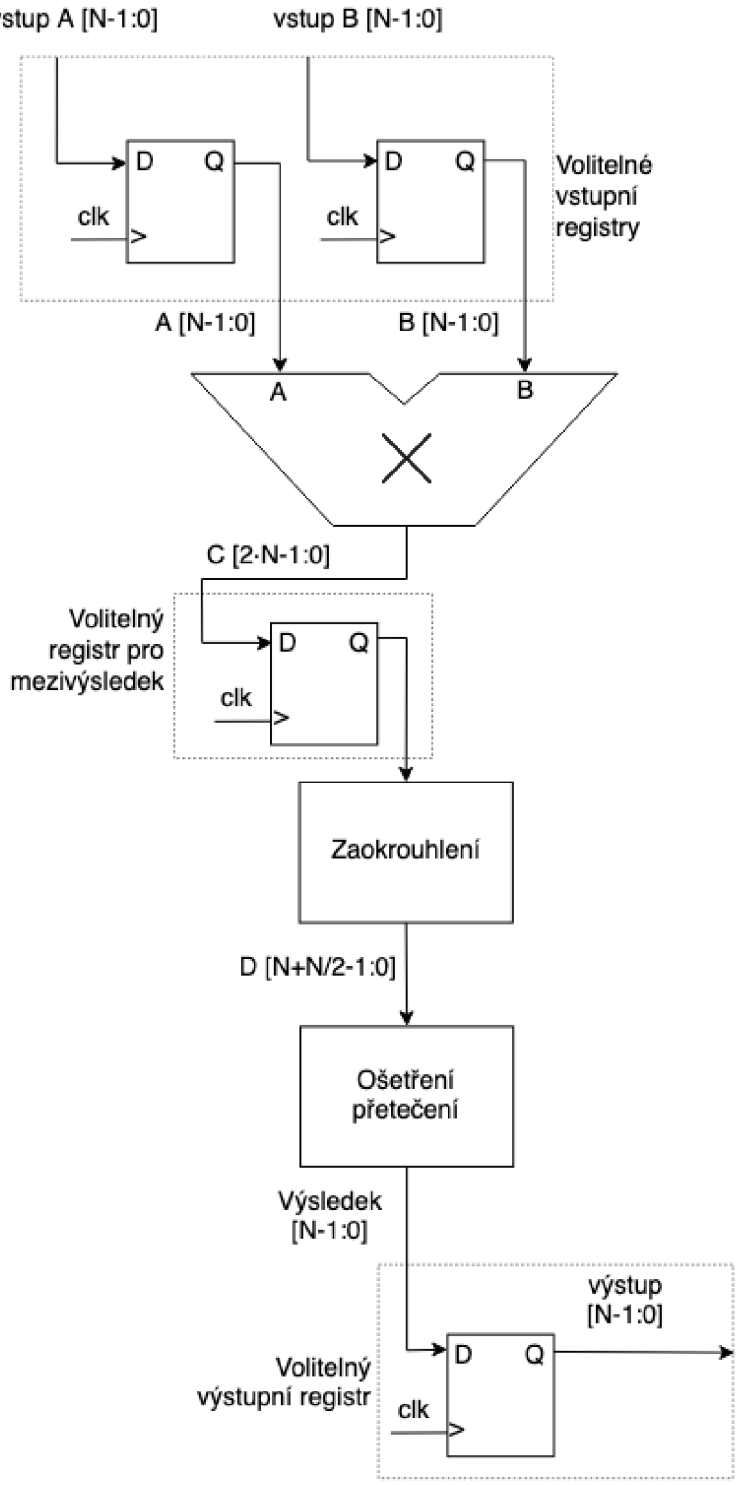
Obrázek 4.1 Blokové schéma komponenty součtu

4.2 Součin

Blokové schéma komponenty provádějící součin je zobrazeno na obrázku 4.2. Operace součinu je opět provedena standardní funkcí z knihovny IEEE NUMERIC_STD a mezivýsledek je uložen do pomocné proměnné, která má dvojnásobnou bitovou šířku, než je šířka požadovaná.

V dalším kroku je provedeno zaokrouhlení tohoto mezivýsledku dle nastavené konfigurace vybíráním správných bitů z mezivýsledků a případnou inkrementací této hodnoty o 1 dle požadovaného typu zaokrouhlení. Komponenta součinu implementuje aritmetické zaokrouhlení, zaokrouhlení absolutně nahoru a absolutně dolů.

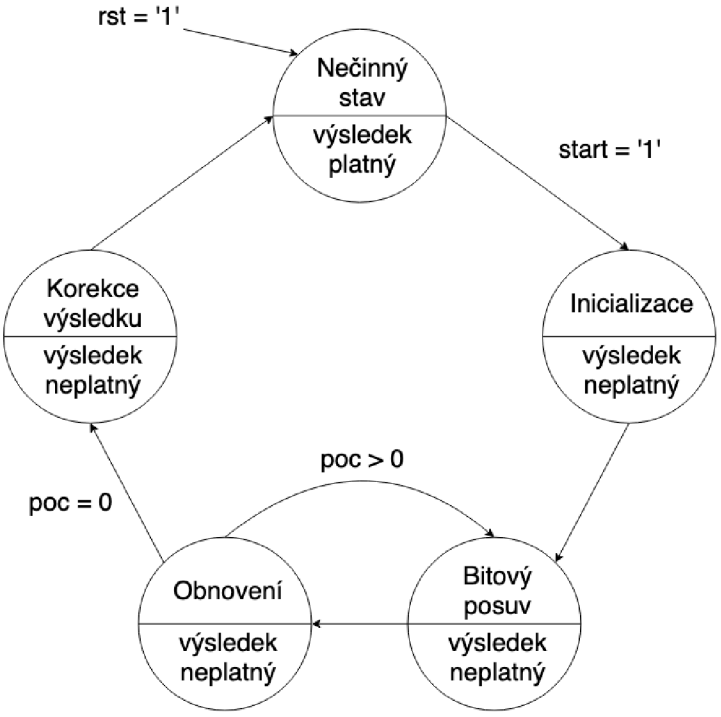
V dalším kroku je na již zaokrouhleném mezivýsledku provedeno ošetření přetečení dle nastavení v konfiguračním souboru. Toto je provedeno podobným způsobem, jako v případě operace součtu popsané v kapitole 4.1, jen v případě přetečení je zahozeno více nejvýznamnějších bitů, jejichž počet odpovídá polovině bitové šířky vstupních čísel.



Obrázek 4.2 Blokové schéma komponenty součinu

4.3 Podíl

Pro výpočet podílu byl použit algoritmus dělení s obnovením pomocného registru, tak jak je popsán v kapitole 2.4.1 a na obrázku 2.3. Jedná se o sekvenční dělení a tato komponenta je tedy implementována jako sekvenční obvod. Základem sekvenční části jsou registry pro všechny pomocné a datové signály a stavový automat zajišťující řízení samotného výpočtu jehož stavový diagram je zobrazen na obrázku 4.3. Vzhledem k tomu, že je výpočet pomalý a trvá řadu hodinových taktů, byla provedena optimalizace tak, aby bylo využito co možná nejméně stavů FSM. Touto optimalizací bylo dosaženo pouze pěti stavů, které jsou nutné pro správnou funkci FSM, z čehož jsou pouze dva stavy využity ke smyčce výpočtu, která se nejvíce podílí na délce trvání výpočtů.

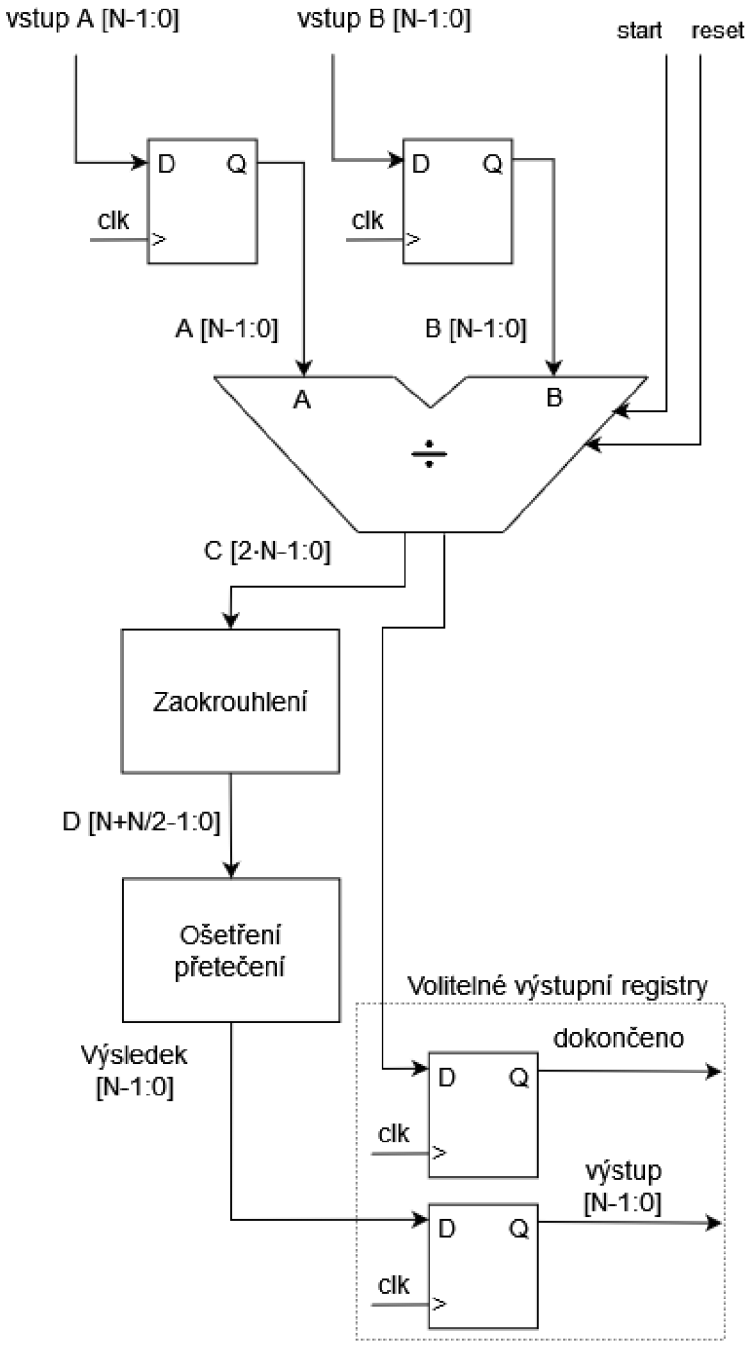


Obrázek 4.3 Konečný stavový automat implementující algoritmus dělení s obnovením pomocného registru

Ovládání komponenty je zajištěno povolovacím vstupním signálem, na základě jehož aktivní hodnoty přejde stavový automat do stavu *Inicializace*, v němž dochází k načtení vstupních hodnot a inicializaci pomocných proměnných, nastavení počítadla a určení znaménka výsledku. Z této fáze přechází FSM do stavu *Bitový posuv*, v rámci kterého je prováděn bitový posuv signálu dělence vlevo, určení hodnoty pomocné proměnné a dekrementace počítadla. Následuje stav *Obnovení*, kdy je vypočtena nová hodnota výsledku a případné obnovení signálu dělence pomocnou proměnnou v závislosti na jejím MSB. V případě, že počítadlo nebylo v předchozím stavu vynulováno, FSM se vrací do stavu *Bitový posuv*, v opačném případě přechází do stavu *Korekce výsledku*. V tomto stavu je výsledek v případě záporného znaménka převeden na jeho dvojkový doplněk.

Po dokončení výpočtu se stavový automat vrací do *Nečinného stavu* a o platnosti výsledku informuje nadřazenou komponentu pomocí dedikovaného signálu.

Blokové schéma celé komponenty dělení ukazuje obrázek 4.4, kde blok děličky představuje FSM vyobrazený na obrázku 4.3. Mezivýsledek dán výpočtem FSM je dále upraven podobně, jako v případě výpočtu součinu popsaném v kapitole 4.2. Nejprve je provedeno zaokrouhlení podle nastavení v konfigurační sloze a následně je ošetřeno přetečení.



Obrázek 4.4 Blokové schéma komponenty podílu

4.4 Zjednodušená modulární redukce

Komponenta zjednodušené modulární redukce přijímá na vstupu kromě vstupní hodnoty také konfigurační hodnotu modulu a požadavek, zda se jedná o symetrickou nebo asymetrickou konfiguraci.

Tato operace je v případě asymetrické modulární redukce implementována pomocí porovnávání vstupní hodnoty s nulou a s hodnotou modulu, v případě symetrické modulární redukce pak s kladnou a zápornou polovinou hodnoty modulu, a následným případným přičtením nebo odečtením hodnoty modulu ke vstupní hodnotě. Součet, respektive rozdíl, je prováděn základní operací z knihovny IEEE NUMERIC_STD.

4.5 Goniometrické funkce

Výpočet goniometrických funkcí je řešen pomocí náhledové tabulky pro operaci sinus, jejíž hodnoty byly vygenerované pomocí skriptu v Matlabu. Tato implementace není konfigurovatelná pomocí konstant v konfigurační sloze, ale je nutné pro každou konfiguraci aritmetické jednotky vytvořit samostatnou náhledovou tabulku odpovídající jak bitové šířce zlomkové části čísel, tak i požadovanému rozlišení goniometrických funkcí.

Samotný výpočet hodnoty sinu dle vstupní hodnoty je realizován jako čtení z náhledové tabulky. Vstupní hodnota musí být v rozmezí daném právě rozlišením LUT, což je možné ošetřit například pomocí zjednodušené modulární redukce (kapitola 4.4) se správně nastavenou hodnotou modulu, případně správně nastaveným bitovým rozsahem vstupní hodnoty.

Implementace výpočtu hodnoty kosinu je provedena s použitím stejné LUT, avšak ke vstupní hodnotě se připočte hodnota odpovídající posunu o $\frac{\pi}{2}$ v závislosti na konfiguraci aritmetické jednotky.

4.6 Testování funkčnosti aritmetické jednotky

Pro testování základních aritmetických operací bylo vybráno několik základních konfigurací aritmetické jednotky a pro každou z nich bylo pomocí skriptu v Matlabu vygenerováno deset tisíc náhodných vstupních hodnot a očekávaných výsledků s využitím funkcí popsaných v kapitole 3.1.

Dále bylo vytvořeno verifikační prostředí v jazyce VHDL a pomocí simulátoru Xilinx Vivado byly vždy pro jednotlivé konfigurace načteny ze souboru vstupní hodnoty a očekávané výsledky a porovnány výsledky jednotlivých operací aritmetické jednotky s očekávanými výsledky. Porovnání bylo zapsáno do výsledkové tabule ve formě ověřovacího signálu. Výsledky testování pro jednotlivé konfigurace jsou uvedeny v tabulce 4.1.

Z tabulky 4.1 je vidět, že v případě konfigurace aritmetické jednotky na bitovou šířku 32 bitů a povolenému přetečení došlo k malému množství chyb v případě operace součinu. Počet chyb byl v řádu nižších jednotek z deseti tisíc. Rozdíl výsledků v těchto případech činil vždy pouze 1 bit na pozici LSB. Tato drobná chyba je způsobena dvojím zaokrouhlováním v rámci operace zaokrouhlení v Matlabu, kdy je číslo nejprve převedeno na formát FP, čímž dojde k určité ztrátě přesnosti a následně je provedeno zaokrouhlení pomocí standardní funkce Matlabu nad tímto číslem.

Tabulka 4.1 Výsledky testování aritmetické jednotky

Konfigurace aritmetické jednotky				**Počet zjištěných chyb z 10 000 testovacích hodnot**		
Bitová šířka	**Zlomková část**	**Ošetření přetečení**	**Typ zaokrouhlení**	**Součet**	**Součin**	**Podíl**
16	8	Přetečení	Aritmetické	0	0	0
16	8	Přetečení	Dolů	0	0	0
16	8	Přetečení	Nahoru	0	0	0
16	8	Saturace	Aritmetické	0	0	0
16	8	Saturace	Dolů	0	0	0
16	8	Saturace	Nahoru	0	0	0
24	16	Přetečení	Aritmetické	0	0	0
24	16	Přetečení	Dolů	0	0	0
24	16	Přetečení	Nahoru	0	0	0
24	16	Saturace	Aritmetické	0	0	0
24	16	Saturace	Dolů	0	0	0
24	16	Saturace	Nahoru	0	0	0
32	16	Přetečení	Aritmetické	0	**3**	0
32	16	Přetečení	Dolů	0	**5**	0
32	16	Přetečení	Nahoru	0	**2**	0
32	16	Saturace	Aritmetické	0	0	0
32	16	Saturace	Dolů	0	0	0
32	16	Saturace	Nahoru	0	0	0

Testování goniometrických funkcí bylo provedeno pro konfiguraci aritmetické jednotky s 16-bitovým rozsahem zlomkové části čísla. Pro testování byla vygenerována LUT obsahující 512 17-bitových hodnot, které odpovídají zlomkové části čísla a znaménkovému bitu. Pro možnost uložení hodnoty pouze ve zlomkových bitech čísla byla zanesena chyba do těchto hodnot v případě, kdy je očekávaná hodnota sinu rovna jedné. V takovém případě je výsledkem největší možné kladné číslo uložené v tomto rozsahu, což odpovídá hodnotě větší než 0,9999. Jako vstupní hodnoty byly použity hodnoty z intervalu $\langle 0; 511 \rangle$ a pro tyto hodnoty byla vygenerována hodnota sinu a kosinu dle LUT.

Zjednodušená modulární redukce byla testována pomocí čítače na vstupu komponenty. Testování bylo provedeno pro symetrickou i asymetrickou redukci při různých hodnotách modulu. Výsledky testování goniometrických funkcí i zjednodušené modulární redukce byly porovnány s výsledky těchto operací v modelu aritmetické jednotky v Matlabu a nebyly zjištěny žádné odchylky.

4.7 Implementace do obvodu FPGA

Syntéza a implementace aritmetické jednotky byla provedena pomocí prostředí Xilinx Vivado do obvodu FPGA Artix-7, kdy byla zvolena standardní strategie implementace jako kompromis mezi množstvím využitých zdrojů v obvodu a celkovou rychlostí.

Syntéza i implementace byly provedeny pro stejné základní konfigurace jako v případě testování funkčnosti jednotky. Pro každou komponentu byla syntéza provedena samostatně a byly vždy upravovány relevantní konfigurace pro danou komponentu. Celkové výsledky počtu využitých zdrojů jsou brány jako výsledky syntézy jednotlivých komponent bez volitelných vstupních a výstupních registrů.

Pro lepší přesnost určení maximální pracovní frekvence jednotlivých komponent byly všechny komponenty doplněny o jednoduché vstupní a výstupní registry tam, kde to bylo relevantní. Takto rozšířené komponenty byly samostatně syntetizovány, návrh byl optimalizován, rozmístěn a propojen tak, aby bylo možné odečíst kritické trasy. Z délky zpoždění na těchto trasách byla následně určena maximální pracovní frekvence.

Výsledky syntézy komponenty součtu a rozdílu jsou vidět v tabulce 4.2. Jelikož tato komponenta nevyužívá konfiguraci zaokrouhlení, nebylo s touto konfigurací uvažováno. Tato komponenta využívá pouze LUT jako základní logické prvky. Z této tabulky je vidět, že v případě ošetření přetečení pomocí saturace je využito o 50 % více zdrojů.

Tabulka 4.2 Výsledky syntézy a implementace komponenty součtu/rozdílu

Konfigurace aritmetické jednotky				**Počet využitých zdrojů**			**Max. prac. frekvence [MHz]**
Bitová šířka	**Zlomková část**	**Ošetření přetečení**	**Typ zaokrouhlení**	**LUT**	**Registry**	**DSP**	
16	8	Přetečení	---	16	0	0	513
16	8	Saturace	---	24	0	0	329
24	16	Přetečení	---	24	0	0	468
24	16	Saturace	---	36	0	0	297
32	16	Přetečení	---	32	0	0	432
32	16	Saturace	---	48	0	0	261

Tabulka 4.3 zobrazuje výsledky syntézy komponenty součinu. Tato komponenta využívá kromě LUT také hardwarové násobičky obsažené ve zdroji zpracování digitálního signálu (DSP). V případě Artix-7 se jedná o blok DSP48E1 obsahující násobičku 25 x 18 bitů a další funkční bloky. Vzhledem k bitové šířce této násobičky je pro konfigurace s větší bitovou šířkou využito více těchto bloků.

Dále je z této tabulky zřetelné, že typ zaokrouhlení má výrazný vliv na využité zdroje. Zaokrouhlení absolutně dolů je na využití zdrojů nejméně náročné, oproti tomu aritmetické zaokrouhlení vyžaduje relativně velký počet LUT ke své funkci. Nejmenší náročnost na zdroje má tedy vždy pro danou bitovou šířku konfigurace s povoleným přetečením a zaokrouhlením absolutně dolů.

Tabulka 4.3 Výsledky syntézy a implementace komponenty součinu

Konfigurace aritmetické jednotky				**Počet využitých zdrojů**			**Max. prac. frekvence [MHz]**
Bitová šířka	**Zlomková část**	**Ošetření přetečení**	**Typ zaokrouhlení**	**LUT**	**Registry**	**DSP**	
16	8	Přetečení	Aritmetické	18	0	1	132
16	8	Přetečení	Dolů	0	0	1	497
16	8	Přetečení	Nahoru	2	0	1	161
16	8	Saturace	Aritmetické	36	0	1	100
16	8	Saturace	Dolů	18	0	1	172
16	8	Saturace	Nahoru	20	0	1	129
24	16	Přetečení	Aritmetické	28	0	2	108
24	16	Přetečení	Dolů	0	0	2	234
24	16	Přetečení	Nahoru	4	0	2	146
24	16	Saturace	Aritmetické	54	0	2	92
24	16	Saturace	Dolů	26	0	2	142
24	16	Saturace	Nahoru	30	0	2	109
32	16	Přetečení	Aritmetické	83	0	4	86
32	16	Přetečení	Dolů	47	0	4	205
32	16	Přetečení	Nahoru	51	0	4	113
32	16	Saturace	Aritmetické	116	0	4	68
32	16	Saturace	Dolů	80	0	4	94
32	16	Saturace	Nahoru	84	0	4	81

Komponenta podílu, jak je vidět z tabulky 4.4, vyžaduje oproti komponentám součtu a součinu velké množství LUT, a navíc velké množství registrů, které jsou využity v rámci implementovaného FSM.

Pro různé druhy konfigurací platí stejné vzorce v počtech využitých zdrojů, jako tomu bylo u komponenty součinu, tedy nejmenší počet využitých zdrojů je vždy při povoleném přetečení a zaokrouhlení absolutně dolů, naopak největší počet zdrojů vyžaduje ošetření přetečení pomocí saturace v kombinaci s aritmetickým zaokrouhlováním.

Tabulka 4.4 Výsledky syntézy a implementace komponenty podílu

Konfigurace aritmetické jednotky				Počet využitých zdrojů			Max. prac. frekvence [MHz]
Bitová šířka	Zlomková část	Ošetření přetečení	Typ zaokrouhlení	LUT	Registry	DSP	
16	8	Přetečení	Aritmetické	176	139	0	252
16	8	Přetečení	Dolů	157	123	0	333
16	8	Přetečení	Nahoru	159	139	0	260
16	8	Saturace	Aritmetické	194	139	0	170
16	8	Saturace	Dolů	175	131	0	323
16	8	Saturace	Nahoru	177	139	0	226
24	16	Přetečení	Aritmetické	256	203	0	231
24	16	Přetečení	Dolů	229	179	0	286
24	16	Přetečení	Nahoru	231	203	0	293
24	16	Saturace	Aritmetické	285	203	0	140
24	16	Saturace	Dolů	258	195	0	269
24	16	Saturace	Nahoru	260	203	0	178
32	16	Přetečení	Aritmetické	341	267	0	200
32	16	Přetečení	Dolů	305	235	0	212
32	16	Přetečení	Nahoru	309	267	0	201
32	16	Saturace	Aritmetické	374	267	0	125
32	16	Saturace	Dolů	338	251	0	203
32	16	Saturace	Nahoru	342	267	0	147

Tabulka 4.5 Výsledky syntézy a implementace komponenty zjednodušené modulární redukce

Konfigurace aritmetické jednotky				Počet využitých zdrojů			Max. prac. frekvence [MHz]
Bitová šířka	Zlomková část	Ošetření přetečení	Typ zaokrouhlení	LUT	Registry	DSP	
16	8	---	---	88	0	0	211
24	16	---	---	132	0	0	185
32	16	---	---	177	0	0	177

Tabulka 4.6 Výsledky syntézy a implementace komponenty goniometrických funkcí

Konfigurace aritmetické jednotky			Počet využitých zdrojů			Max. prac. frekvence [MHz]
Bitová šířka	Zlomková část	Počet hodnot uložených v náhledové tabulce	LUT	Registry	MUX	
16	8	512	111	0	46	245
16	8	1024	177	0	70	196
24	16	512	151	0	58	243
24	16	1024	201	0	0	197
32	16	512	151	0	58	243
32	16	1024	201	0	0	197

Tabulka 4.5 ukazuje počet využitých zdrojů při zjednodušené modulární redukci, který je závislý pouze na bitové šířce čísel. Pro tuto operaci jsou využity pouze LUT, kterých je zapotřebí relativně velké množství, což může být způsobeno převážně bitovým posuvem hodnoty modulu.

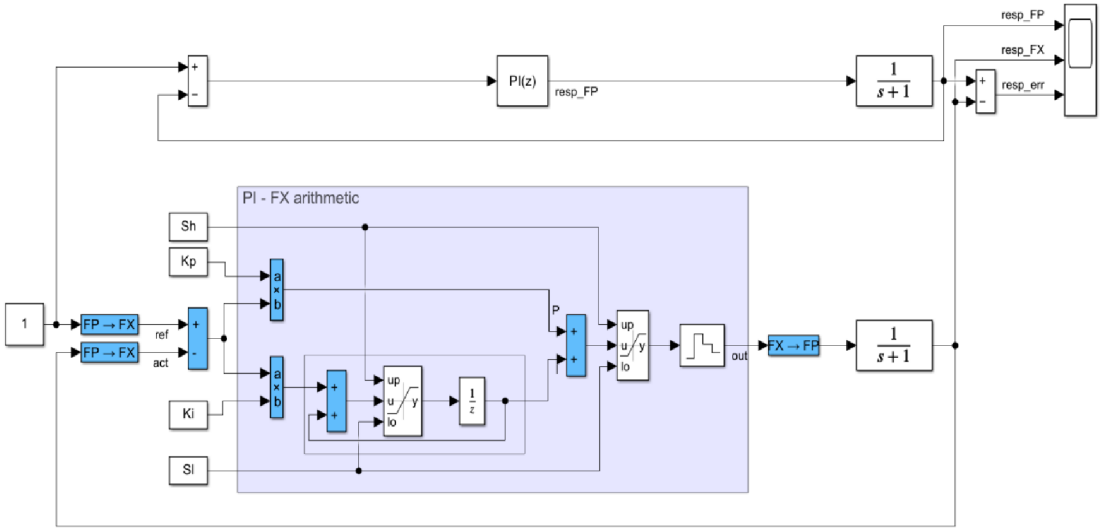
Počet využitých zdrojů pro komponentu goniometrických funkcí závisí na šířce zlomkové části čísel a na rozlišení funkce, tedy na počtu hodnot uložených v náhledové tabulce. Počet využitých zdrojů pro tuto komponentu ukazuje tabulka 4.6.

Z výsledků implementace vyplývá, že největší vliv na maximální pracovní frekvence má komponenta součinu. Pracovní kmitočty této komponenty jsou výrazně ovlivněny typem zaokrouhlování i způsobem ošetření přetečení. Při povoleném přetečení a zaokrouhlování dolů jsou pracovní kmitočty nad 200 MHz pro všechny testované bitové šířky. Ovšem například v případě aritmetického zaokrouhlování může pracovní frekvence klesnout výrazně pod 100 MHz. Oproti tomu komponenta součtu a rozdílu má vliv jen minimální, i pro největší testované konfigurace zůstává vždy pracovní kmitočet vysoce nad hranicí 200 MHz. Vliv komponenty dělení je také relativně malý, což je dáno především tím, že se jedná o sekvenční obvod, ve kterém se nevyskytují dlouhé kombinační cesty. U komponenty zjednodušené modulární redukce je pracovní frekvence závislá na bitové šířce čísel a zůstává poměrně vysoká i pro větší bitové šířky. Kmitočet komponenty goniometrických funkcí klesá s rostoucím rozlišením těchto funkcí. V případě rozlišení 512 nebo 1024 hodnot se kmitočet stále drží na hodnotách okolo 200 MHz.

5 PRAKTICKÁ UKÁZKA VYUŽITÍ NAVRŽENÉHO MODELU

S jednotlivými funkčními bloky navrženého modelu aritmetické jednotky v kombinaci se základními bloky Simulinku je možné v tomto prostředí pracovat a navrhovat různé modely. Jako příklad takového systému je proporcionálně-integrační (PI) regulátor pro systémy řízení, jehož struktura je na obrázku 5.1.

PI regulátor nastavuje výstupní veličinu v závislosti na regulační odchylce, kterou přijímá na vstupu. Tato odchylka je určena rozdílem aktuální vstupní veličiny (signál *act*) od referenční hodnoty (signál *ref*). Do modelu PI regulátoru vstupují parametry *Kp* a *Ki*, kde *Kp* je činitel zesílení proporcionální složky a *Ki* odpovídá zesílení integrační složky. Další možnost konfigurace regulátoru je pomocí horního a spodního saturačního limitu, které jsou nastaveny parametry *Sh* a *Sl*. Hodnoty všech parametrů jsou načítány z pracovního prostředí Matlabu, ve kterém jsou už před spuštěním simulace převedeny od reprezentace ve formátu FX dle konfigurace aritmetické jednotky.

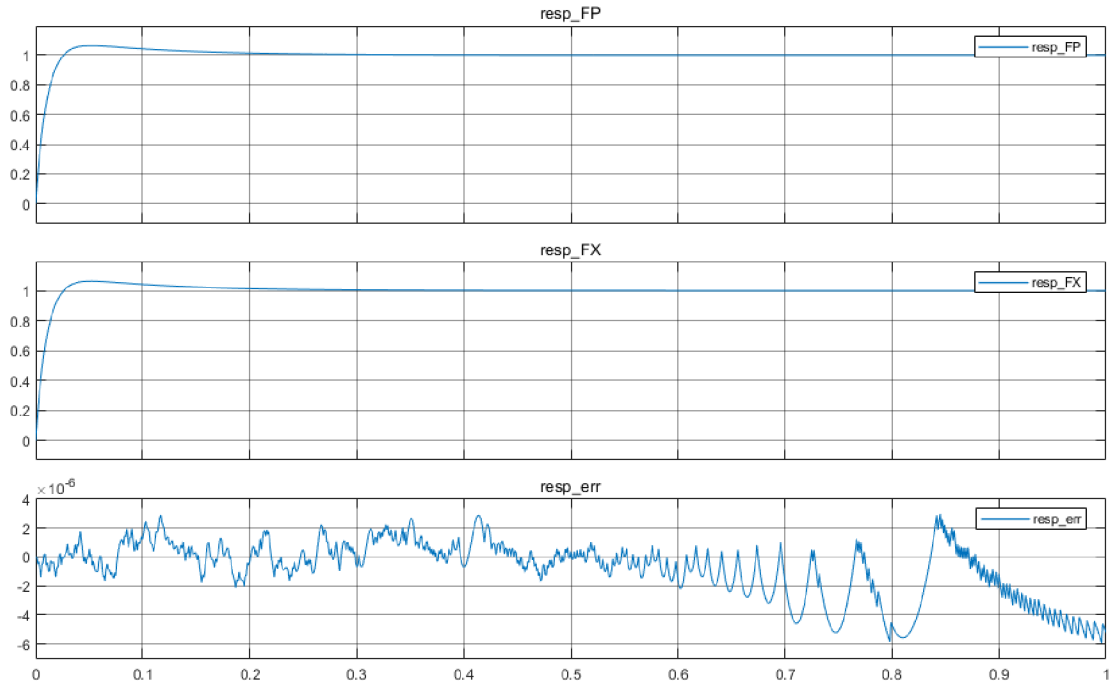


Obrázek 5.1 Simulační model PI regulace v prostředí Simulink

Pro ověření funkčnosti takto navrženého modelu regulátoru PI byla jeho funkce porovnána s blokem regulátoru dostupným v knihovně Simulinku. Konfigurace modelu aritmetické jednotky pro tuto simulaci byla nastavena na 32 bitů s 16 bity zlomkové části, povoleným přetečením a zaokrouhlováním absolutně dolů. Saturační limity byly nastaveny na maximální, respektive minimální hodnotu pro daný bitový rozsah. Koeficienty proporcionálního a integračního zesílení byly nastaveny na vzájemně si odpovídající hodnoty. Vzorkovací perioda byla nastavena na 1 ms.

Jako referenční hodnota byla nastavena konstantní hodnota 1 a na výstup obou bloků byla vložena přenosová funkce, jejíž výstup byl brán jako aktuální hodnota zpět na vstup regulátoru. Průběhy odezvy na jednotkový skok obou regulátorů si navzájem odpovídají,

jak je vidět z obrázku 5.2, kde je také vynesena odchylka, mezi jednotlivými průběhy. Odchylka je v řádu 10^{-6} a je způsobena nepřesností formátu pevné řádové čárky.



Obrázek 5.2 Výsledky porovnání PI regulace ve formátu FP a FX

6 ZÁVĚR

Práce analyzuje možnosti reprezentace číselných hodnot v digitálních obvodech a zaměřuje se na aritmetické operace s čísly reprezentovanými ve formátu pevné řádové čárky. Práce rozebírá základní aritmetické operace sčítání, odčítání, násobení a dělení a také doplňující operace výpočtu goniometrických funkcí a zjednodušené modulární redukce. V případě operace dělení jsou analyzovány možnosti výpočtů pomocí různých typů dělících algoritmů.

V další části práce je představen návrh modelu aritmetické jednotky v Matlabu pro práci s čísly v pevné řádové čárce. Model obsahuje možnost konfigurace aritmetické jednotky a provádí jak základní, tak i doplňující aritmetické operace, kdy operaci podílu vykonává pomocí dělícího algoritmu s obnovením pomocného registru. Pro zjednodušení modelování složitých systémů byla vytvořena knihovna funkčních bloků modelu aritmetické jednotky pro prostředí Simulink.

Funkčnost modelu byla ověřena na sadě náhodně generovaných hodnot a výsledky výpočtů byly porovnány s výsledky operací prováděných ve formátu plovoucí řádové čárky. Výsledky si mezi sebou odpovídaly s odchylkou menší, než je váha nejméně významného bitu při nastavené bitové šířce, respektive v případě přetečení ve formátu pevné řádové čárky došlo správně k jeho ošetření dle konfigurace jednotky.

Dále je v práci představen návrh aritmetické jednotky pro práci s čísly ve formátu pevné řádové čárky v jazyce VHDL, která vykonává všechny výše uvedené aritmetické operace. Funkčnost aritmetické jednotky byla porovnána v simulaci s navrhnutým modelem jednotky v Matlabu. Bylo zjištěno malé množství odchylek pro konfiguraci 32 bitů šířky čísel a 16 bitů zlomkové části s povoleným přetečením, jak je vidět v tabulce 4.1. Tyto odchylky měly vždy váhu nejméně významného bitu a byly způsobeny zaokrouhlováním v modelu v Matlabu pro velké bitové šířky čísel.

Syntéza a implementace navrhnuté aritmetické jednotky do obvodu FPGA byly provedeny pro všechny testované konfigurace aritmetické jednotky a pro každou komponentu samostatně. Počet využitých zdrojů obvodu pro jednotlivé komponenty je kromě bitové šířky čísel velice závislý na typu ošetření přetečení a zaokrouhlení. Stejným způsobem jsou také závislé maximální pracovní frekvence jednotlivých komponent.

Pro ukázku možnosti praktického využití navrhnutého modelu v prostředí Simulink je v poslední části práce jako příklad ukázán model PI regulátoru pro systémy řízení. Tento model je konfigurovatelný kromě základní konfigurace aritmetické jednotky také pomocí koeficientů proporcionálního a integračního zesílení a pomocí saturačních limitů. Jeho funkčnost byla porovnána s blokem regulátoru dostupným v knihovně Simulinku. Na obrázku 5.2 je vidět, že odezva navrhnutého modelu regulátoru na jednotkový skok odpovídá odezvě standardního bloku regulátoru a odchylka mezi jednotlivými průběhy je v řádu 10^{-6}, což je způsobeno nepřesností formátu pevné řádové čárky při konfiguraci 16ti bitů zlomkové části čísel.

LITERATURA

[1] LANDERS, John B. What is an Arithmetic Unit? Easy Tech Junkie [online]. Sparks (NV): Conjecture Corporation, 2013 [cit. 2021-11-19]. Dostupné z: https://www.easytechjunkie.com/what-is-an-arithmetic-unit.htm

[2] TIŠNOVSKÝ, Pavel. Fixed point arithmetic. Root.cz [online]. 2006 [cit. 2021-10-28]. Dostupné z: https://www.root.cz/clanky/fixed-point-arithmetic/

[3] FINLEY, Thomas. Two's Complement [online]. 2000 [cit. 2021-10-28]. Dostupné z: https://www.cs.cornell.edu/~tomf/notes/cps104/twoscomp.html

[4] Pohyblivá řádová čárka. Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2021 [cit. 2021-11-21]. Dostupné z: https://cs.wikipedia.org/wiki/Pohyblivá_řádová_čárka

[5] IEEE 754. Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2021 [cit. 2021-11-21]. Dostupné z: https://cs.wikipedia.org/wiki/IEEE_754

[6] TIŠNOVSKÝ, Pavel. Základní aritmetické operace prováděné ve formátu FX. Root.cz [online]. 2006 [cit. 2021-10-28]. Dostupné z: https://www.root.cz/clanky/zakladni-aritmeticke-operace-provadene-ve-formatu-fx/

[7] ARAR, Steve. Multiplication Examples Using the Fixed-Point Representation. All about circuits[online]. 2017 [cit. 2021-10-28]. Dostupné z: https://www.allaboutcircuits.com/technical-articles/multiplication-examples-using-the-fixed-point-representation/

[8] KUMAR, Deepak, Prabir SAHA a Anup DANDAPAT. Hardware implementation of methodologies of fixed point division algorithms. International journal on smart sensing and intelligent systems [online]. 2017, 10(3), 630-645 [cit. 2021-10-28]. Dostupné z: https://www.exeley.com/exeley/journals/in_jour_smart_sensing_and_intelligent_systems/10/3/pdf/10.21307_ijssis-2017-227.pdf

[9] PIERCE, Rod. Rounding Methods. Math Is Fun [online]. 2020 [cit. 2021-11-19]. Dostupné z: https://www.mathsisfun.com/numbers/rounding-methods.html

[10] RENARDY, Antonius P., Nur AHMADI, Ashbir A. FADILA, Naufal SHIDQI a Trio ADIONO. FPGA implementation of CORDIC algorithms for sine and cosine generator. 2015 International Conference on Electrical Engineering and Informatics (ICEEI) [online]. IEEE, 2015, 2015, 1-6 [cit. 2022-03-29]. ISBN 978-1-4673-6778-3. Dostupné z: https://ieeexplore.ieee.org/document/7352460 doi:10.1109/ICEEI.2015.7352460

[11] TIŠNOVSKÝ, Pavel. Výpočet goniometrických funkcí algoritmem CORDIC. Root.cz [online]. 2006 [cit. 2022-03-29]. Dostupné z: https://www.root.cz/clanky/vypocet-goniometrickych-funkci-algoritmem-cordic/

SEZNAM SYMBOLŮ A ZKRATEK

Zkratky:

FP	Floating-point – Plovoucí řádová čárka
FPGA	Programovatelná hradlová pole
FSM	Finite State Machine – Konečný stavový automat
FX	Fixed-point – Pevná řádová čárka
LSB	Least significant bit – Nejméně významný bit
LUT	Look-up table – Náhledová tabulka
MSB	Most significant bit – Nejvýznamnější bit
XOR	Logická operace exkluzivní disjunkce