BPC-NDI Pomocná Aritmetická Jednotka

Závěrečná zpráva

	Jméno		Datum
Autor:	Petr	Čechura	15.01.2023

1. DOCUMENT CHANGE LOG

Verze	Datum	Autor	Strany	Popis změn
1.0			Všechny	První verze

2. OBSAH

3. SEZNAM OBRÁZKŮ

4. SEZNAM TABULEK

1 ÚVOD

Pomocná aritmetická jednotka, kterou se tato práce zabývá, by v praxi jen těžko našla své uplatnění. Ovšem na principu fungování, architektuře a problémech, které její realizace provází, se dají dobře demonstrovat úskalí návrhu digitálních obvodů, protože obsahuje jak problematiku zpracování asynchronních signálů, tak práci s obvyklou reprezentací čísel, a v neposlední řadě také potřebu komplexnějšího verifikačního prostředí.

Projekt má tak spíše edukativní charakter a dává si za cíl simulovat práci na reálném projektu ve firmě, což se odráží i v požadavku na kompletní a přehlednou dokumentaci.

2 APLIKOVATELNÉ A ODKAZOVANÉ DOKUMENTY

Všechny dokumenty sloužící jako zadávací dokumentace nebo odkazované v tomto textu jsou uvedeny v tabulkách dole. Pokud není známá přesná verze, je použito datum vydání dokumentu (měsíc/rok). V případě že není známé ani datum vydání, je jako verze uvedena hodnota 0.

2.1 Seznam aplikovatelných dokumentů

Ref.	Název	Číslo dokumentu	Verze
AD01		-	-
AD02	-	-	-

Table 2-1 Seznam aplikovatelných dokumentů

2.2 Seznam odkazovaných dokumentů

Ref.	Název	Číslo dokumentu	Verze
RD01	Pomocná aritmetická jednotka Semestrální projekt	1	1
RD02	Requirement Specification	2	1
RD03	Synthesis Report	3	1
RD04	Post-PAR Static Timing Report	4	1

Table 2-2 Seznam odkazovaných dokumentů

3 DEFINICE A SEZNAM ZKRATEK

3.1 Definice

- **Paralelní slovo** – Posloupnost bitů, která byla zpracována (deserializerem) a uložena do paměti, takže je možné pracovat s celou hodnotou, kterou reprezentuje.
- **Rámec** – Data, která v bitové reprezentaci obsahují jedno číslo. Délka je 16 bitů.
- **Paket** – Skládá se ze dvou rámců, tj. ze dvou čísel.

3.2 Psaní čísel

Čísla se vyskytují v binární a decimální soustavě. Zápis binárních čísel je uvozen prefixem „0b“.

3.3 Jednotky

Pouze jednotky ze soustavy SI jsou použity v textu.

3.4 Zkratky

AAU	Auxiliary Arithmetic Unit
FSM	Finite state machine
TBC	To Be Confirmed
TBD	To Be Defined
DUT	Design under test
BFM	Bus Functional Model

4 PŘEDSTAVENÍ PROJEKTU

Výsledkem projektu je architektura samostatné výpočetní jednotky, která komunikuje přes rozhraní SPI a je schopna provádět operace **sčítání** a **násobení**.

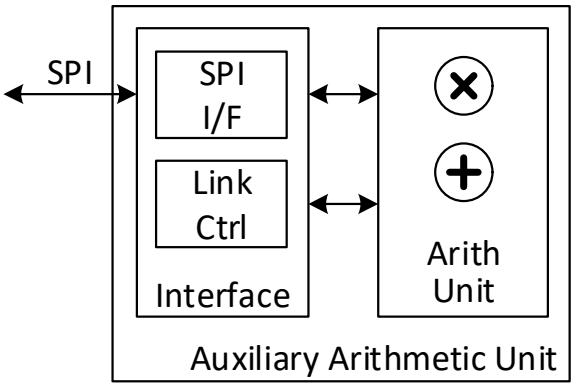
Jednotka obsahuje následující vstupy a výstupy:

- **CS_b** *(vstup)* – Signál, kterým *master* zahajuje komunikaci překlopením do log. 0. Pokud komunikace neprobíhá, signál musí být v log. 1.
- **MOSI** *(vstup)* – Datový signál, obsahující bity pro provedení aritmetických operací.
- **MISO** *(výstup)* – Datový signál, obsahující výsledky aritmetických operací.
- **SCLK** *(vstup)* – Synchronizační signál, určující frekvenci komunikace; při náběžné hraně jednotka posílá bit do MISO a *master* jej přijímá, při sestupné hraně *master vysílá bity do jednotky a ta je přijímá.*
- **clk** *(vstup)* – Hodinový signál, který je pro všechny části architektury stejný.
- **reset** *(vstup)* – Resetovací signál, společný pro všechny části architektury.

Komunikace probíhá ve formě paketů, kde každý paket obsahuje dva rámce. Protože je využita jak náběžná, tak sestupná hrana signálu SCLK, součástí jednoho rámce jsou bity pro provedení operací (jdoucí do jednotky) a výsledek operace s čísly z předchozího paketu (jdoucí do *master*).

Čísla jsou ve formátu desetinných čísel s dvojkovým doplňkem a s pevnou řádovou čárkou na 8. bitu. Protože jeden rámec má velikost 16 bitů, maximální číslo, které je možné vyjádřit pomocí jednoho rámce, je 127.9960938; nejmenší pak -128. Pokud výsledek některé z operací převyšuje maximální rozsah rámce, jednotka jej automaticky zaokrouhlí.

Architektura je navržena pro FPGA Spartan3 firmy Xillinx, konkrétně model XC3S200. Hodinový signál má frekvenci 50 MHz.



Obrázek 4-1 Blokové schéma AAU

Obsahem této zprávy je podrobný rozbor architektury a požadavků, které musí splňovat, a následně i verifikace, pomocí které jsou prověřovány. Poslední kapitola se pak věnuje implementaci obvodu do FPGA, výpisu použitých zdrojů a realizačních omezení.

5 PLÁN VÝVOJE

Vývoj výpočetní jednotky začal definicí všech požadavků a shromážděním doporučených postupů pro co největší eliminaci hrubých chyb, které by vedly k významným opravám v kódu. Návrh architektury se do velké míry řídil instrukcemi z dokumentace RD01.

Nejprve byl navržen funkční blok pro zpracování příchozího signálu dle standardu SPI (**SPI_Interface)**. Ten obsahuje obvody pro zpracování asynchronního signálu a ošetření možných chyb, které při přenosu mohou vzniknout. Následovala samostatná verifikace SPI rozhraní, kterou bylo ověřeno, že jednotka je schopna zpracovat příchozí signál a z něj vytvořit paralelní slovo.

Druhým funkčním blokem je **PKT_Control**, jehož součástí je stavový automat o 4 stavech, který má řídit průběh komunikace a adekvátně předávat data z SPI rozhraní do výpočetní jednotky (a naopak) podle současného stavu. Po návrhu znovu následovala verifikace, součástí které byla simulace všech možných situací, které v jednotce mohou nastat.

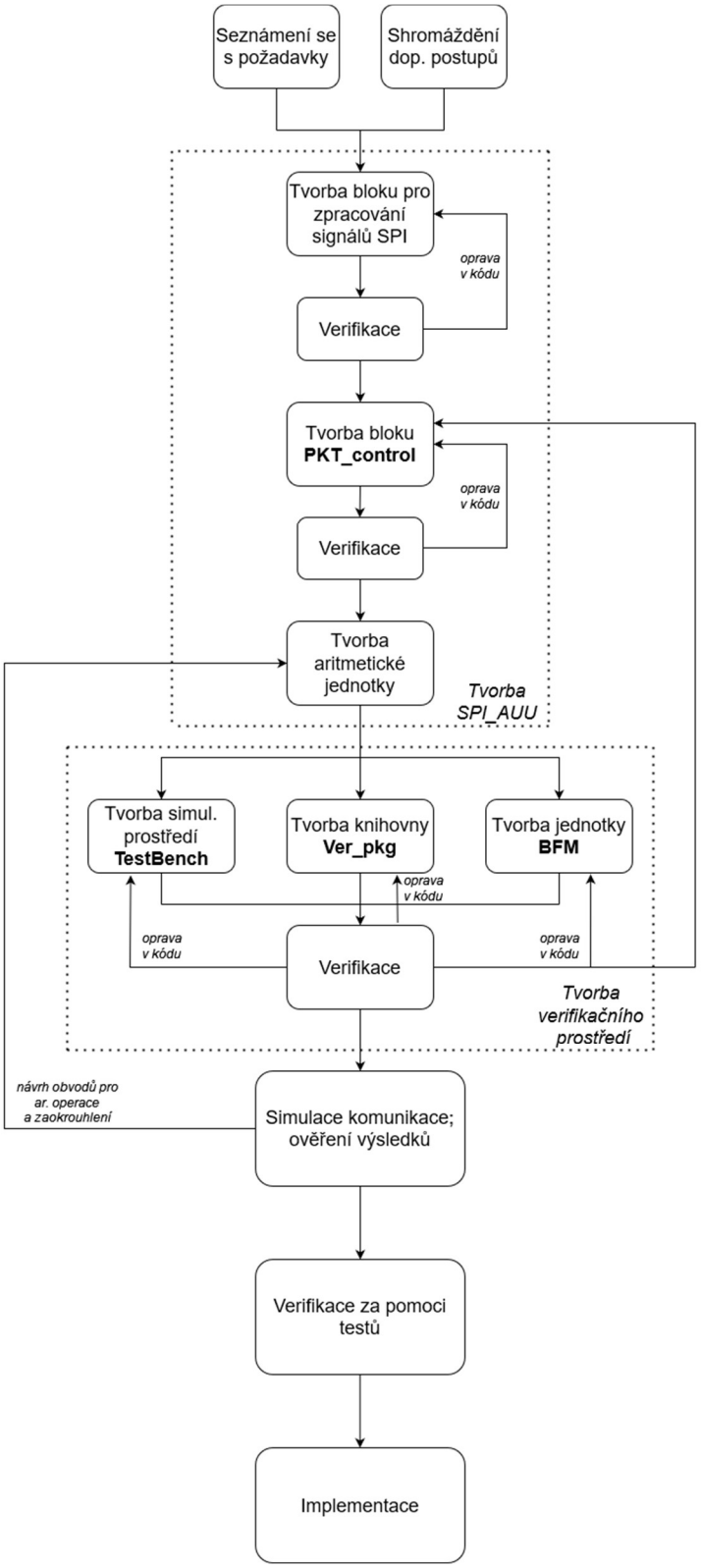
Třetí funkční blok je samotná aritmetická jednotka (**Aritmetical_unit**), ve které jsou prováděny aritmetické operace a která řídí i proces zaokrouhlování. Výsledek je pak poslán zpět do SPI rozhraní, aby byl v dalším paketu vysílán do řídící jednotky. Pro usnadnění verifikace byla aritmetická jednotka v raných fázích zjednodušena tak, aby předávala stále stejný rámec – obvody pro výpočet a zaokrouhlování výsledku byly vytvořeny až ke konci vývoje (*viz dále*).

Tři funkční bloky dohromady tvoří kompletní výrobek, který bylo potřeba otestovat. K tomu bylo vytvořeno verifikační prostředí, jehož součástí je **TestBench**, ve kterém jsou v samostatném procesu prováděny testy. Testy využívají procedury z vlastní knihovny **Ver_pkg**, která byla vytvořena pro přehlednost. Procedury jsou celkem tři (***SendRightPacket***, ***SendWrongPacket***, ***SendFrame***) a simulaci komunikace s testovanou jednotkou (**SPI_AUU**) provádějí za pomoci bloku **BFM**, který podle pokynů z bloku **TestBench** vysílá do jednotky datové signály podle standardu SPI.

Po vytvoření funkčního verifikačního prostředí (podoba se několikrát změnila), schopného odeslat a přijmout paket, byl proveden první test, a sice odeslání platného paketu a příjem odpovědi. Pro správnou funkci byla nutná drobná úprava v architektuře jednotky (problém byl v datovém toku výsledků aritmetických operací uvnitř jednotky).

Jakmile přenos probíhal bez problémů, pozornost byla zaměřena na aritmetickou jednotku, tedy správnost výsledků a ošetření zaokrouhlování. Pro snadnější simulaci byl vytvořen v jazyce Python skript (**Converter.py)**, který převádí binární číslo do dekadické podoby s pevnou řádovou tečkou a s dvojkovým doplňkem.

Po ověření, že jednotka je schopna reagovat na platné rámce a odeslat výpočetně správné výsledky, byly za pomoci vytvořených procedur provedeny další testy, kterými byla verifikace dokončena.

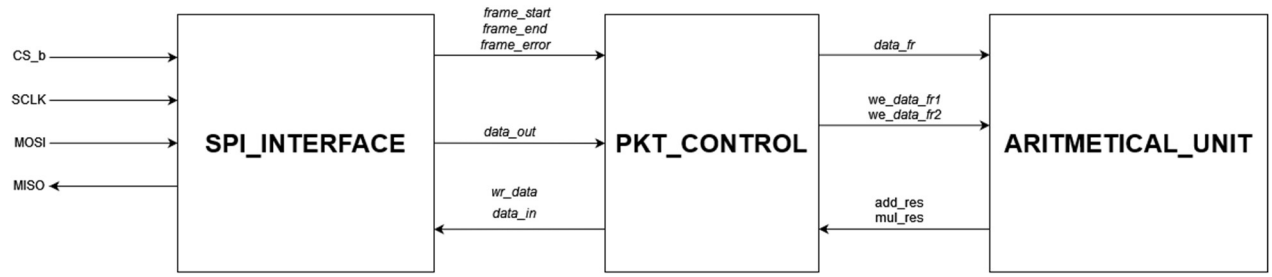


Obrázek 5-1 Vývojový diagram

6 POPIS NÁVRHU

Jednotka se skládá ze tří hlavních funkčních bloků:

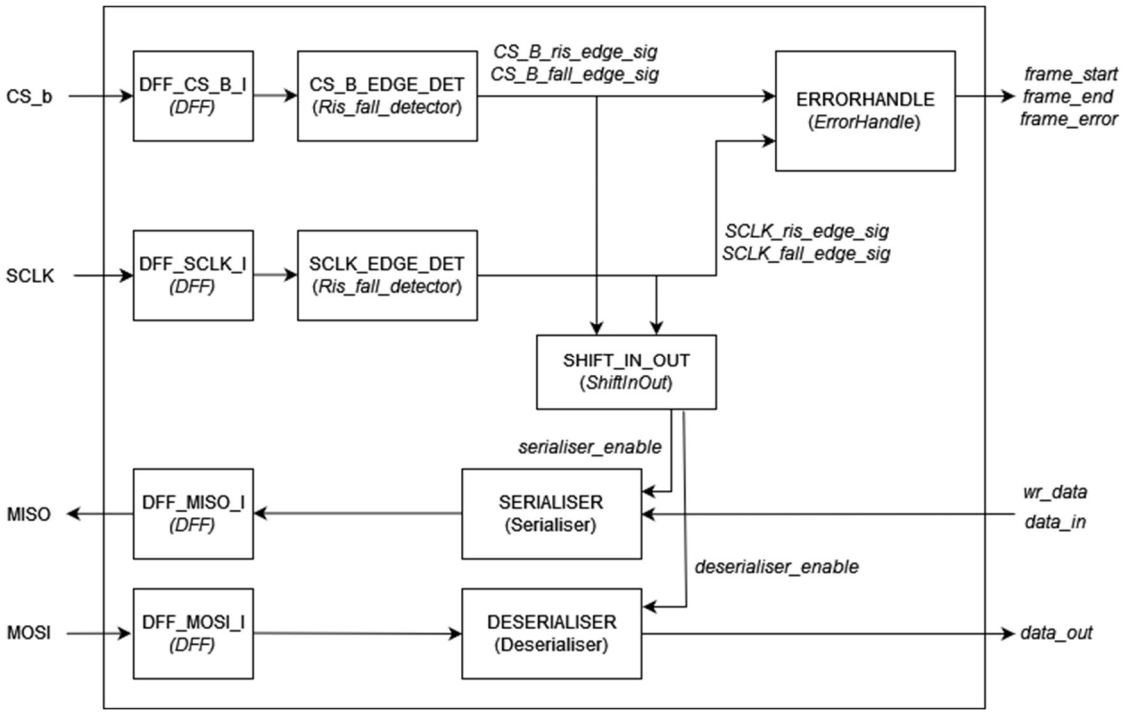
- **SPI_Interface (**rozhraní SPI),
- **PKT_Control** (řadič paketů)
- **Aritmetical unit** (aritmetická jednotka)



Obrázek 6-1 Architektura AAU

6.1 SPI_Interface

Blok, představující nejnižší vrstvu komunikace, který je navržen tak, aby zpracovával vstupní signály (tj. převáděla je na paralelní slovo) a vysílal výstupní signály do řídící jednotky podle standardu SPI. Součástí jsou i obvody pro detekci chyb přenosu; na případnou chybu reaguje blok **PKT_Control** (*viz dále*).



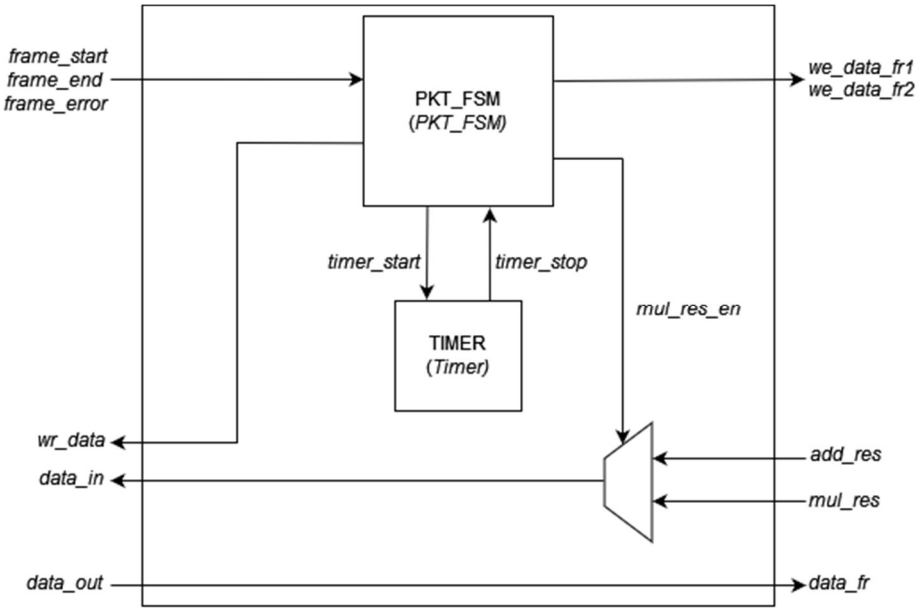
Obrázek 6-2 Blokové schéma funkčního bloku SPI_Interface

Celkem obsahuje 6 funkčních bloků:

- **Ris_fall_detector** – Detektor náběžné a sestupné hrany; použit pro detekci změny signálu SCLK a CS_b.
- **ShiftInOut** – Podle náběžných a sestupných hran SCLK a CS_b řídí funkci serializeru a deserializeru.
- **Deserialiser** – Registr, převádějící vstupní signál na paralelní slovo, které lze dále zpracovat v FPGA.
- **Serialiser** – Registr, obsahující data (výsledky aritmetických operací) k odeslání.
- **ErrorHandle** – Podle počtu náběžných a sestupných hran SCLK zjišťuje, zda přenos probíhá bez chyby. Zároveň detektuje začátek a konec komunikace.
- **DFF** – Dva klopné obvody typu D v řadě za sebou pro odstranění metastability.

6.2 PKT_Control

Zprostředkovává komunikaci mezi rozhraním SPI a aritmetickou jednotkou, představuje řídící blok celé komunikace. Součástí je stavový automat a časovač, který je použit pro měření času 1 ms; pokud není do této doby přijat druhý rámec, celý paket je považován za neplatný (*viz* **požadavky**).

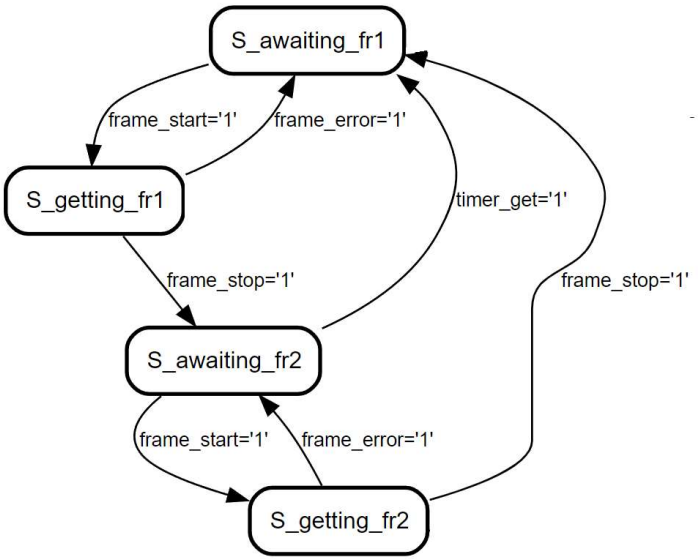


Obrázek 6-3 Blokové schéma funkčního bloku PKT_Control

Stavový automat má celkem 4 stavy:

- **S_awaiting_fr1** – Výchozí stav při neprobíhající komunikaci. Při začátku komunikace posílá výsledek součtu z předchozího paketu do rozhraní SPI a přechází v další stav.
- **S_getting_fr1** – Jednotka přijímá první rámec. Vyskytla-li se chyba, dochází k návratu do výchozího stavu. Bezchybný rámec je poslán do AU, do serializeru je vyslán součin z předchozího paketu a nastává přechod do dalšího stavu.
- **S_awaiting_fr2** – Jednotka čeká na druhý rámec. Časovač měří dobu 1 ms; trvá-li čekání déle, stavový automat přechází do výchozího stavu.

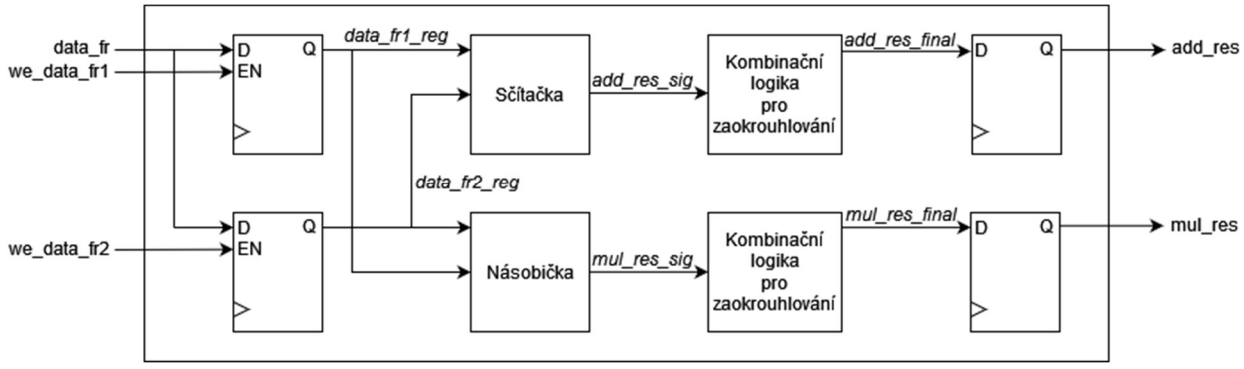
- **S_getting_fr2** – Jednotka přijímá druhý rámec. Je-li přijat bez chyby, předává se dál do aritmetické jednotky a automat přechází do výchozího stavu. Při výskytu chyby se automat vrací do předchozího stavu a očekává opravu.



Obrázek 6-4 Stavový automat ve funkčním bloku PKT_Control

6.3 Aritmetical_unit

Poslední funkční blok přijímá vstupní data pro výpočet součtu a součinu, výsledky pak dává na výstup. Součástí jsou rovněž kombinační obvody, které automaticky zaokrouhlí výsledek v případě přetečení nebo podtečení. Blok neobsahuje samostatné komponenty, je tedy napsán kompletně v jednom souboru.



Obrázek 6-5 Blokové schéma funkčního bloku Aritmetical_unit

7 VERIFIKAČNÍ PLÁN

V rámci verifikačního plánu je představena verifikační matice a následně popsáno verifikační prostředí, pomocí kterého jsou realizovány testy. Následuje výpis jednotlivých testů.

7.1 Verifikační matice

Jednotlivé požadavky jsou sepsány a je naznačenou, jakou metodou jsou splněny. Dominující verifikační metodou je simulace, což zahrnuje testy, které jsou představeny v kapitole 7.3. Zbytek je testován kontrolou dokumentace, přičemž jsou uvedeny odkazy na jednotlivé kapitoly, které se danému požadavku věnují.

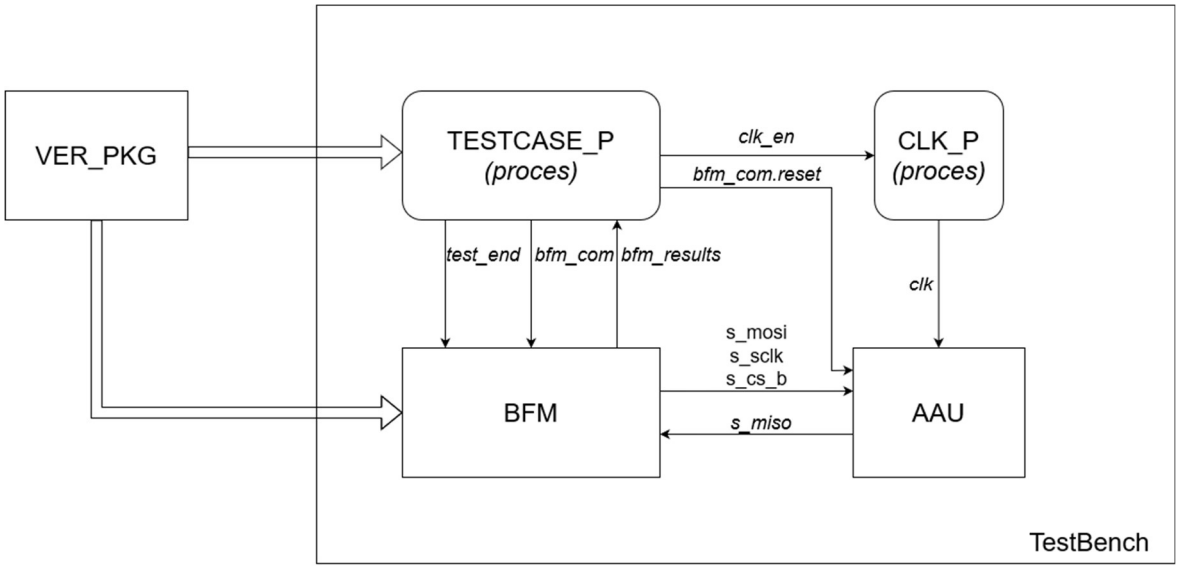
REQ ID	Název	Ver. metoda			Verifikace požadavku	Splněno
		R	S	A		
REQ_AAU_G_001	Cílová technologie	A	-	-	Kapitola 8	A
REQ_AAU_G_002	Synchronní návrh	A	-	-	Kapitola 8	A
REQ_AAU_G_003	Výstupní signály	A	-	-	Kapitola 6.1	A
REQ_AAU_G_004	Vstupní signály	A	-	-	Kapitola 6.1	A
REQ_AAU_G_005	Bezpečná implementace FSM	A	-	-	Kapitola 8	A
REQ_AAU_G_006	Dokumentace	A	-	-	Obsah	A
REQ_AAU_F_010	Výpočetní jednotka	A	-	-	Kapitola 6.3	A
REQ_AAU_F_011	Formát čísel	-	A	-	Test *tc_spi_001*	A
REQ_AAU_F_012	Zaokrouhlování čísel	-	A	-	Test *tc_spi_005*	A
REQ_AAU_F_013	Přetečení operací	-	A	-	Test *tc_spi_005*	A
REQ_ AAU_I_020	Frekvence SPI	-	A	-	Test *tc_spi_001*	A
REQ_ AAU_I_021	Pořadí bitů	-	A	-	Test *tc_spi_006*	A
REQ_ AAU_I_022	Nekompletní rámec	-	A	-	Testy tc_spi_003 a tc_spi_004	A
REQ_ AAU_I_023	Reset komunikace	-	A	-	Test *tc_spi_002*	A
REQ_ AAU_I_024	Formát paketů	-	A	-	Test *tc_śpi_001*	A

Table 7-1 Verifikační matice

7.2 Popis verifikačního prostředí

Verifikační prostředí se skládá ze tří samostatných modulů:

- **TestBench** – Hlavní platforma, ve které jsou prováděny testy pomocí procedur z **Ver_pkg**. Obsahuje dva samostatně běžící procesy:
- **BFM** – Ve své podstatě simulovaný řadič sběrnice, který je na požádání schopen odeslat a přijmout rámec podle standardu SPI. Komunikuje s modulem **TestBench**, od kterého přijímá data, které má odeslat, a kterému přijaté rámce také předává.
- **Ver_pkg** – Knihovna, obsahující jak procedury pro provádění testů, tak vlastní datové typy pro snadnější manipulaci s daty.



Obrázek 7-1 Blokové schéma verifikačního prostředí

7.2.1 TestBench

Řídící komponenta celého verifikačního prostředí; jsou v ní obsaženy všechny ostatní komponenty a pro realizaci testů jsou použity procedury z Ver_pkg. Skládá se ze dvou samostatných procesů:

- **TESTCASE_P** – Na začátku jsou definovány proměnné pro požadované testy (rámce, velikost rámců a zpoždění). Samotný proces začíná zapnutím hodinového signálu (clk_en), pak následují jednotlivé testy. Na konci je hodinový signál vypnut a signalizován konec testování (test_end) pro **SPI_BFM**.
- **clk_P** – Proces pro generování hodinového signálu, který je distribuován do testované jednotky.

Kromě procesů obsahuje TestBench pomocné signály:

- pro propojení BFM a testované jednotky SPI_AUU
  - *s_mosi*
  - *s_miso*
  - *s_sclk*
  - *s_cs_b*
- pro spojení s BFM v rámci odeslání a přijmu rámců
  - bfm_com
  - bfm_results

7.2.2 BFM

Komponenta, která na základě příkazů, které obdrží ve formě signálu *bfm_com*, posílá do výpočetní jednotky rámce s definovanou hodnotou a délkou.

Komunikace začíná, jakmile komponenta obdrží náběžnou hranu signálu *bfm_com.start*; po odeslání rámce jsou signály, vedoucí do jednotky, vráceny do výchozích hodnot, a proces znovu čeká na náběžnou hranu *bfm_com.start*. Proces končí, jakmile je signál *test_end* v log. 1.

Odpověď z jednotky posílá komponenta formou signálu *bfm_rep*.

Průběh odesílání a příjmu rámců probíhá podle standardu SPI, tj. s náběžnou hranou SCLK jsou data přijata a se sestupnou jsou odeslána. Pokud je délka rámce (*bfm_com.fr_size*) větší než 16, řadič posílá do jednotky požadovaný rámec (který má fixní délku 16) a po jeho odeslání posílá log. 0.

7.2.3 Ver_pkg

Knihovna, kterou používá primárně TestBench pro realizaci testů, obsahuje ale také definice datových typů *record* pro snazší komunikaci mezi moduly.

Datové typy:

- **Bfm_com** – Sdružuje signály, které TestBench posílá do BFM.
  - **Frame** – Obsahuje paralelní slovo rámce, který je potřeba odeslat. Má pevně danou velikost 16 bitů.
  - **Fr_size** – Definuje velikost odeslaného rámce. Může obsahovat jakoukoliv hodnotu.
  - Start – Signalizuje start přenosu.
  - **Reset** – Jediný signál, který není připojen do BFM, ale do AAU. Je použit pro resetování jednotky.
  - **Sclk_period** – Definuje periodu signálu SCLK pro požadovaný přenos.
  - **Bit_order** – Je-li v log. 1, znamená to signál jednotce BFM, aby průběžně vypsala každý bit, který byl přijat nebo odeslán.
- **Bfm_rep** – Sdružuje signály, které BFM posílá do TestBench.
  - **Result** – Výsledný rámec, který BFM získal z výpočetní jednotky.
  - **Done** – Signalizace, že byl obdržen kompletní rámec a je možné jej zpracovat.

Procedury:

- **SendRightPacket** – Procedura k odeslání platného packetu, tj. bez zpoždění a s validními délkami rámců. Začíná resetováním testované jednotky následuje za použití procedury SendFrame odesílání jednotlivých rámců, což je doplněno komentáři pro snadné sledování průběhu a výsledků. Odeslány jsou celkem 4 rámce: 2 obsahují data k odeslání, další dva jsou prázdné a jsou odeslány jen pro obdržení výsledků.
- **SendWrongPacket** – Na rozdíl od předchozí procedury umožňuje odeslat neplatný packet, tj. definovat délku a zpoždění. Protože procedura počítá s tím, že jeden z rámců nebude přijat (protože je neplatný), okamžitě posílá opravený platný rámec. Proto je možné definovat hodnoty třech rámců, aby bylo u porovnání výsledků možné poznat, které rámce jednotka přijala a které ne. Dohromady je tedy odesláno 5 rámců: 3 s daty, 2 pro příjem.
- **SendFrame** – Obsahuje odeslání jednoho rámce a ověření zda byl odeslán.

7.3 Verifikační testy

Název testu	Ověření funkčnosti při odeslání platného paketu pro různé frekvence SCLK.	**Číslo testu.:**	tc_spi_001
Popis testu	Účelem je ověřit, že při odeslání paketu se správnou délkou rámců a s hodnotami, u kterých nehrozí přetečení, jednotka paket adekvátně vyhodnotí a odešle zpět správné výsledky. Test je proveden pro frekvence 1 MHz, 100 kHz a 10 kHz.		
Reflektované požadavky	REQ_AAU_F_011, REQ_AAU_I_020, REQ_AAU_I_024		

Postup testu

1. Nastavení frekvence SCLK na 1 MHz.
2. Reset DUT
3. BFM: Odeslán platný rámec s hodnotou 0b0001110000111110.
4. Zpoždění 200 µs.
5. BFM: Odeslán platný rámec s hodnotou 0b1111111111000011.
6. BFM: Přijat rámec s hodnotou 0b0001110000000001.
7. BFM: Přijat rámec s hodnotou 0b1111100101000101.
8. Nastavení frekvence SCLK na 100 kHz.
9. Opakování bodů 2. – 7.
10. Nastavení frekvence SCLK na 10 kHz.
11. Opakování bodů 2. – 7.
12. Ověření výsledků:

    Sčítání

    - 0b0001110000111110 + 0b1111111111000011 = 0b0001110000000001
    - 28.2421875 - 0.23828125 = 28.00390625

    Násobení

    - 0b0001110000111110 · 0b1111111111000011 = 0b1111100101000101
    - 28.2421875 · (-0.23828125) = -6.73046875

Název testu	Detekce zpožděného druhého rámce s následnou opravou	**Číslo testu.:**	tc_spi_002
Popis testu	Účelem je ověřit, že při zpoždění druhého rámce jednotka celý paket ignoruje a čeká na správný paket. Frekvence SCLK je 1 MHz.		
Reflektované požadavky	REQ_AAU_I_023		

Postup testu

1. Reset DUT
2. BFM: Odeslán platný rámec s hodnotou 0b0001110000111110.
3. Zpoždění 1100 µs.
4. BFM: Odeslán platný rámec s hodnotou 0b1111111111000011.
5. Zpoždění 100 µs.
6. BFM: Odeslán platný rámec s hodnotou 0b0000001100011100.
7. BFM: Přijat rámec s hodnotou 0b0000001011011111.
8. BFM: Přijat rámec s hodnotou 0b1111111101000010.
9. Ověření výsledků:

   Sčítání

   - 0b1111111111000011 + 0b0000001100011100 = 0b0000001011011111
   - -0.23828125 + 3.109375 = 2.87109375

   Násobení

   - 0b1111111111000011 · 0b0000001100011100 = 0b1111111101000010
   - -0.23828125 · 3.109375 = -0.7421875

Název testu	Detekce chybného prvního rámce	**Číslo testu.:**	tc_spi_003
Popis testu	Účelem je ověřit, zda jednotka ignoruje neplatný první rámec a přijme teprve až ten správný. Nejprve je ověřeno, že jednotka ignoruje příliš krátký rámec, poté je stejná procedura zopakována pro příliš dlouhý rámec. Frekvence SCLK je 1 MHz.		
Reflektované požadavky	REQ_AAU_I_022		

Postup testu

1. Reset DUT
2. BFM: Odeslán neplatný rámec s hodnotou 0b0000001110000111110 (délka 19 bitů).
3. Zpoždění 500 µs.
4. BFM: Odeslán platný rámec s hodnotou 0b1111111111000011.
5. Zpoždění 100 µs.
6. BFM: Odeslán platný rámec s hodnotou 0b0000001100011100.
7. BFM: Přijat rámec s hodnotou 0b0000001011011111.
8. BFM: Přijat rámec s hodnotou 0b1111111101000010.
9. Reset DUT
10. BFM: Odeslán neplatný rámec s hodnotou 0b1110000111110 (délka 12 bitů)
11. Zpoždění 500 µs.
12. BFM: Odeslán platný rámec s hodnotou 0b1111111111000011.
13. Zpoždění 100 µs.
14. BFM: Odeslán platný rámec s hodnotou 0b0000001100011100.
15. BFM: Přijat rámec s hodnotou 0b0000001011011111.
16. BFM: Přijat rámec s hodnotou 0b1111111101000010.
17. Ověření výsledků:

    Sčítání

    0b1111111111000011 + 0b0000001100011100 =

    0b0000001011011111

    -0.23828125 + 3.109375 = 2.87109375

    Násobení

    0b1111111111000011 · 0b0000001100011100 =

    1111111101000010

    -0.23828125 · 3.109375 = -0.7421875

Název testu	Detekce chybného druhého rámce	**Číslo testu.:**	tc_spi_004
Popis testu	Účelem je ověřit, zda jednotka ignoruje neplatný druhý rámec a přijme teprve až ten správný. Nejprve je ověřeno, že jednotka ignoruje příliš krátký rámec, poté je stejná procedura zopakována pro příliš dlouhý rámec. Frekvence SCLK je 1 MHz.		
Reflektované požadavky	REQ_AAU_I_022		

Postup testu

1. Reset DUT
2. BFM: Odeslán platný rámec s hodnotou 0b0001110000111110.
3. Zpoždění 500 µs.
4. BFM: Odeslán neplatný rámec s hodnotou 0b0001111111111000011 (délka 19 bitů).
5. Zpoždění 100 µs.
6. BFM: Odeslán platný rámec s hodnotou 0b0000001100011100.
7. BFM: Přijat rámec s hodnotou 0b0001111101011010.
8. BFM: Přijat rámec s hodnotou 0b0101011111010000.
9. Reset DUT
10. BFM: Odeslán platný rámec s hodnotou 0b0001110000111110.
11. Zpoždění 500 µs.
12. BFM: Odeslán neplatný rámec s hodnotou 0b1111111000011 (délka 12 bitů).
13. Zpoždění 100 µs.
14. BFM: Odeslán platný rámec s hodnotou 0b0000001100011100.
15. BFM: Přijat rámec s hodnotou 0b0001111101011010.
16. BFM: Přijat rámec s hodnotou 0b0101011111010000.
17. Ověření výsledků:

    Sčítání

    0b0001110000111110 + 0b0000001100011100 =

    0b0001111101011010

    28.2421875 + 3.109375 = 31.3515625

    Násobení

    0b0001110000111110 ·0b0000001100011100 =

    0b0101011111010000

    28.2421875 · 3.109375 = 87.8125

Název testu	Schopnost zaokrouhlit výsledky při přetečení nebo podtečení	**Číslo testu.:**	tc_spi_005
Popis testu	Účelem je ověřit, zda je jednotka schopna zaokrouhlit výsledky sčítání a násobení, a to směrem nahoru i dolů. Nejprve jsou odeslány pakety s hodnotami 14.9453125 a 127.97265625, což vyústí v zaokrouhlení na hodnotu 127.99609375, následně pak pakety s hodnotami (-125.5234375, -112.015625) a (14.9453125, -24.02734375), které povedou pro zaokrouhlení dolů. Frekvence SCLK je 1 MHz.		
Reflektované požadavky	REQ_AAU_F_012, REQ_AAU_F_013		

Postup testu

1. Reset DUT
2. BFM: Odeslán platný rámec s hodnotou 0b0000111011110010.
3. Zpoždění 200 µs.
4. BFM: Odeslán platný rámec s hodnotou 0b0111111111111001.
5. BFM: Přijat rámec s hodnotou 0b0111111111111111.
6. BFM: Přijat rámec s hodnotou 0b0111111111111111.
7. Reset DUT
8. BFM: Odeslán platný rámec s hodnotou 0b1000001001111010.
9. Zpoždění 200 µs.
10. BFM: Odeslán platný rámec s hodnotou 0b1000111111111100.
11. BFM: Přijat rámec s hodnotou 0b1000000000000000.
12. BFM: Přijat rámec s hodnotou 0b0111111111111111.
13. Reset DUT
14. BFM: Odeslán platný rámec s hodnotou 0b0000111011110010.
15. Zpoždění 200 µs.
16. BFM: Odeslán platný rámec s hodnotou 0b1110011111111001.
17. BFM: Přijat rámec s hodnotou 0b1111011011101011.
18. BFM: Přijat rámec s hodnotou 0b1000000000000000.

Název testu	Ověření správného pořadí bitů.	**Číslo testu.:**	tc_spi_006
Popis testu	Účelem je ověřit, zda jednotka splňuje požadavek ve specifikaci a přijímá a odesílá LSB bit jako první. Frekvence SCLK je 1 MHz. Test má implementován funkci, že při spuštění zobrazuje každý bit, který byl odeslán nebo přijat, takže lze sledovat pořadí.		
Reflektované požadavky	REQ_AAU_I_021		
Postup testu	1. Reset DUT 2. BFM: Odeslán platný rámec s hodnotou 0b0000000011111111. 3. Zpoždění 200 µs. 4. BFM: Odeslán platný rámec s hodnotou 0b1111111100000000. 5. BFM: Přijat rámec s hodnotou 0b1111111111111111. 6. BFM: Přijat rámec s hodnotou 0b1111111100000001.		

8 VÝSLEDKY IMPLEMENTACE

Výpočetní jednotka byla implementována do obvodu FPGA, konkrétně Spartan3 model XC3S200. Pro implementaci byly nastaveny následující parametry:

Cíl optimalizace *(Optimization goal)*	Rychlost
Náročnost optimalizace *(Optimization effort)*	Normální
Druh enkódování FSM *(FSM encoding algorithm)*	Automaticky
Bezpečná implementace *(safe implementation)*	Ano

Table 8-1 Vstupní parametry pro syntézu (výběr)

Syntéza byla provedena pomocí syntetizéru XST, který je součástí prostřední ISE Design suite 14.7 od firmy Xillinx. Výsledky jsou shrnuty v následující tabulce. Kompletní hlášení ze syntézy je uvedeno v dokumentu RD03. Protože stavový automat byl použit pouze jeden, používá binární enkódování a obsahuje 4 stavy, lze považovat jeho implementaci za bezpečnou. V celém návrhu byl použit pouze jeden hodinový signál (což potvrdila i implementace), takže je synchronní.

Počet klopných obvodů	149
Počet stavových automatů	1
Počet LUT (o 4 vstupech)	151

Table 8-1 Výsledky syntézy (výběr)

Pro implementaci byly použity omezení *(contraints)*, které zahrnovaly nejistotu oscilátoru t_{JITTER}, maximální možné zpoždění na datové cestě $t_{MAX,PATH}$ a čas potřebný ke zpracování vstupních i výstupních signálů. Kompletní souhrn omezení je v dokumentu RD04.

t_{JITTER} [ns]	1
$t_{MAX,PATH}$ [ns]	11.818
Max. frekvence FPGA [MHz]	84.617

Table 8-3 Souhrn důležitých parametrů a výstupů statické časové analýzy

Maximální frekvence, kterou je možné použít, je 84.617 MHz.