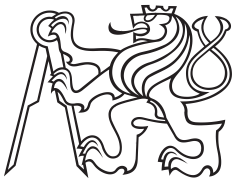
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra telekomunikační techniky

Realizace jednoduchého navigačního systému pomocí FPGA a jazyka VHDL s výstupem na monitor

Jiří Čala

Vedoucí: Ing. Pavel Lafata, Ph.D.
Studijní program: Elektrotechnika, elektronika a komunikační technika
Květen 2022

Poděkování

Chtěl bych velmi poděkovat svému vedoucímu práce, panu Ing. Pavlu Lafatovi, Ph.D., za vedení práce. Rovněž bych chtěl poděkovat své přítelkyni Zdeňce za podporu.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškerou použitou literaturu.

V Praze, 20. května 2022

Abstrakt

Cílem práce je navrhnout a realizovat jednoduchý navigační systém na určování polohy, směru a rychlosti pohybu postavený na některém z dostupných kitů s FPGA, např. Zybo či Nexys a navigačním modulu Pmod NAV: 9-axis IMU Plus Barometer s využitím jazyka VHDL. Výstup navigačního systému bude zobrazen pomocí standardního rozhraní VGA na připojeném monitoru. Výstupem budou VHDL knihovny a VHDL moduly pro obsluhu navigačního modulu a komunikaci s jeho jednotlivými čidly, VHDL moduly pro zpracování a vyhodnocení údajů a VHDL moduly pro zobrazení výsledku pomocí VGA rozhraní na připojeném monitoru.

Klíčová slova: FPGA, VHDL, Pmod NAV, SPI, VGA

Vedoucí: Ing. Pavel Lafata, Ph.D.

Abstract

The aim of the bachelor thesis is creation and implementation of a simple navigation system using FPGA and VHDL. Digilent NAV module containing various sensors will be used in order to create a navigation system to obtain position and movement information. The result will be displayed via standard VGA output using VGA display.

Keywords: FPGA, VHDL, Pmod NAV, SPI, VGA

Title translation: A Simple Navigation System Based on FPGA and VHDL Using VGA Output

Obsah

Obrázky

Tabulky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Čala** Jméno: **Jiří** Osobní číslo: **457616**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra telekomunikační techniky**

Studijní program: **Elektrotechnika, elektronika a komunikační technika**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Realizace jednoduchého navigačního systému pomocí FPGA a jazyka VHDL s výstupem na monitor

Název bakalářské práce anglicky:

A Simple Navigation System Based on FPGA and VHDL Using VGA Output

Pokyny pro vypracování:

Navrhněte a realizujte jednoduchý navigační systém na určování polohy postavený na některém z dostupných kitů s FPGA, např. Zybo či Nexys a navigačním modulu Pmod NAV: 9-axis IMU Plus Barometer s využitím jazyka VHDL. Výstup navigačního systému zobrazte pomocí standardního rozhraní VGA na připojeném monitoru. Navrhněte a implementujte navigační systém, ověřte jeho přesnost a využití v různých podmínkách. Výstupem budou VHDL knihovny a VHDL moduly pro obsluhu navigačního modulu a komunikaci s jeho jednotlivými čidly, VHDL moduly pro zpracování a vyhodnocení údajů a VHDL moduly pro zobrazení výsledku pomocí VGA rozhraní na připojeném monitoru.

Seznam doporučené literatury:

[1] Zybo Zynq, Nexys4 Digilent manuály k deskám
[2] Pmod NAV Digilent manuál a aplikační poznámky
[3] PINKER, J. a POUPA, M. Číslicové systémy a jazyk VHDL. Praha : BEN - technická literatura, 2006. 349 s. ISBN 80-7300-198-5.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Pavel Lafata, Ph.D., katedra telekomunikační techniky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **16.09.2021** Termín odevzdání bakalářské práce: _____

Platnost zadání bakalářské práce: **19.02.2023**

Ing. Pavel Lafata, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

20.10.2021 _____

Datum převzetí zadání Podpis studenta

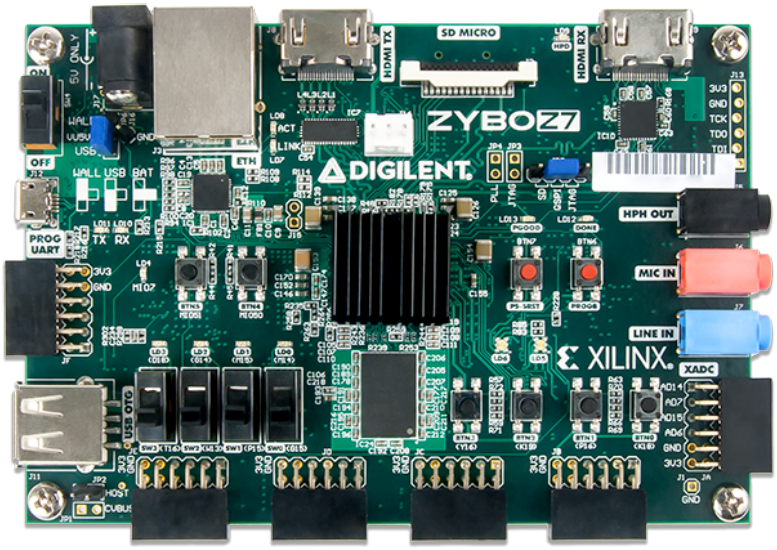
Kapitola 1

Úvod

Cílem práce bylo vytvořit návrh, který pomocí SPI rozhraní získá data o pohybu a poloze z modulu Pmod NAV (akcelerometr, gyroskop, magnetomer), tato data zpracuje a výsledné hodnoty využije ke grafickému zobrazení pomocí VGA rozhraní na monitoru. Návrh je vytvořen ve VHDL a realizován na FPGA na desce Digilent Zybo Z7-20.

Práci může být rozdělena do těchto částí:

- Na úvod je popsána technologie FPGA, deska a moduly, které budou v práci použity.
- V druhé a třetí kapitole jsou řešena rozhraní pro SPI komunikaci s modulem Pmod NAV, a rozhraní pro VGA výstup. Navazuje čtvrtá kapitola, která řeší zpracování dat z modulu Pmod NAV. Každá z těchto tří kapitol začíná teoretickým úvodem do problematiky, a poté následuje popis výsledného návrhu a jeho testování.
- Čtvrtá kapitola práce se pak věnuje spojením předešlých návrhů do jednoho celku. Jsou zde řešeny podpůrné menší entity a popsán způsob vizualizace získaných dat.
- Poslední část práce popisuje testování návrhu a jeho vyhodnocení.



Obrázek 1.1: Vývojová deska Digilent Zybo Z7-20 [1]

Programovatelné logické buňky	85 tis.
LUT tabulky	53 200
Klopné obvody (Flip-Flop)	106 400
Paměť RAM (počet 36 Kb bloků)	4.9 Mb (140)
DSP bloky (18x25 MACC)	220

Tabulka 1.1: Parametry FPGA čipu Xilinx Zynq 7000. [20]

1.1 Vývojová deska Digilent Zybo Z7-20

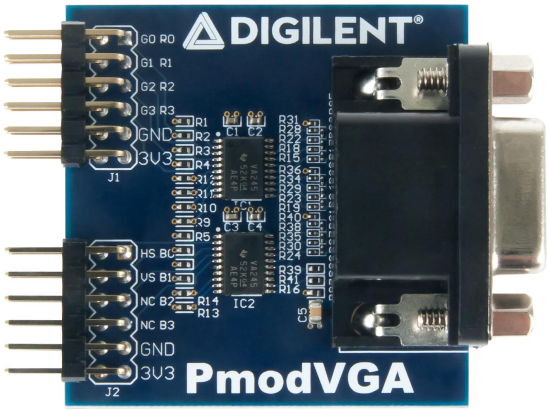
Digilent Zybo Z7-20 je vývojová deska osazená čipem Xilinx Zynq 7000, který v sobě integruje procesor ARM Cortex-A9 se 7. generací Xilinx FPGA architektury, ekvivalentní s řadou Xilinx Artix-7. Pro tuto práci bude využita pouze programovatelná logika FPGA. Základní parametry jsou zobrazeny v tabulce 1.1.

Na vývojové desce budou dále využity Pmod porty (GPIO rozhraní) pro komunikaci s periferiemi – především senzory a přídavnými moduly. Tyto porty obsahují 2 piny pro napájení 3.3V, 2 piny pro uzemnění a 8 pinů pro připojení logických signálů. Celkem je k dispozici 6 těchto portů. [1]

Pro programování desky je použito vývojové prostředí Vivado Design Suite 2018.3 od společnosti Xilinx. Návrh je vytvořen v jazyce VHDL 2008, a poté je syntetizován a implementován do výsledného zapojení hradel. Posledním krokem je vytvoření bitstreamu – konfiguračního souboru, který je nahrán do zařízení. Vivado také přímo obsahuje nástroje simulace – behaviorální, post-syntetizační nebo post-implementační. Pro tyto simulace lze v prostředí vytvářet testovací soubory (test bench).

1.2 Modul Digilent Pmod VGA

Modul Pmod VGA umožňuje připojení zobrazovacích zařízení k vývojovým deskám pomocí standardu VGA a jeho rozšíření. Dokáže přenášet obrazový signál s 12bitovou barevnou hloubku (až 4096 barev) a základní frekvencí až 150 MHz. Je vybaven R-2R DAC převodníky s žebříčkovou rezistorovou sítí, které na základě 4bitového digitálního vstupu vytváří analogový signál s výstupním napětím o 16 úrovních v rozmezí 0 V až 0.7 V při impedanci 75 Ω na vstupu zobrazovacího zařízení. Modul disponuje dvěma vstupními 12pinovými Pmod porty J1 a J2 (samec) a jedním výstupním VGA portem (samice). Celkem 14 pinů je určeno pro vstupní obrazový signál – 3 × 4 piny pro RGB kanály (R0-R3, G0-G3, B0-B3), 1 pin pro H-sync a 1 pin pro V-sync.[12]



Obrázek 1.2: Modul Digilent Pmod VGA.[12]

Pin	Signal	Description
1	R0	Red 0
2	R1	Red 1
3	R2	Red 2
4	R3	Red 3
5	GND	Power Supply Ground
6	VCC3V3	Positive Power Supply
7	B0	Blue 0
8	B1	Blue 1
9	B2	Blue 2
10	B3	Blue 3
11	GND	Power Supply Ground
12	VCC3V3	Positive Power Supply

Tabulka 1.2: Vstupní port J1 modulu Pmod VGA.[12]

Pin	Signal	Description
1	G0	Green 0
2	G1	Green 1
3	G2	Green 2
4	G3	Green 3
5	GND	Power Supply Ground
6	VCC3V3	Positive Supply Ground
7	HS	Horizontal Sync
8	VS	Vertical Sync
9	NC	Not Connected
10	NC	Not Connected
11	GND	Power Supply Ground
12	VCC3V3	Positive Power Supply

Tabulka 1.3: Vstupní port J2 modulu Pmod VGA. [12]

1.3 Modul Digilent Pmod NAV

Modul Digilent Pmod NAV je přídavným modulem pro vývojové desky. Disponuje senzory pro určení orientace a polohy. Modul je osazen dvěma integrovanými obvody se zabudovanými senzory:

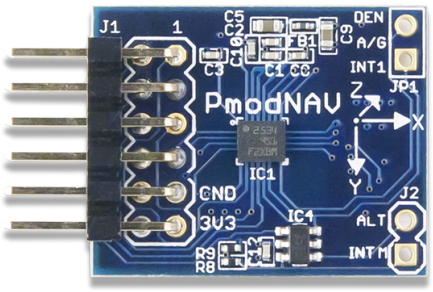
STMicroelectronics LSM9DS1

- 3osý akcelerometr s volitelnými rozsahy ±2/±4/±8/±16 g
- 3osý gyroskop s volitelnými rozsahy ±245/±500/±2000 dps
- 3osý magnetometr s volitelnými rozsahy ±4/±8/±12/±16 gauss
- vestavěný senzor teploty pro teplotní kompenzaci s operačním rozsahem -40 až +85°C [2]

STMicroelectronics LPS25HB

- piezorezistivní MEMS senzor tlaku s rozsahem 260-1260 hPa
- vestavěný senzor teploty pro teplotní kompenzaci s operačním rozsahem -30 až +105°C

Modul disponuje třemi vstupními porty – jedním 12pinovým Pmod portem J1, a dvěma doplňkovými 2pinovými porty J2 a JP1.



Obrázek 1.3: Modul Digilent Pmod NAV.

Pin	Signal	Description
1	CS_A/G	Chip select for Accel/Gyro
2	SDI	Master Out Slave In (MOSI)
3	SDO	Master In Slave Out (MISO)
4	SPC	Serial Clock
5	GND	Power Supply Ground
6	VCC	Power Supply (3.3V/5V)
7	INT	Interrupt pin for all components
8	DRDY_M	Data ready for the Magnetometer
9	CS_M	Chip Select for the Magnetometer
10	CS_ALT	Chip Select for the Altimeter
11	GND	Power Supply Ground
12	VCC	Power Supply (3.3V/5V)

Tabulka 1.4: Vstupní port J1 modulu Pmod NAV. [11]

Port J1 obsahuje 6 pinů pro připojení kanálů komunikačního SPI rozhraní:

- SPC – kanál pro hodinový signál SCLK
- SDI – MISO kanál
- SDO – MOSI kanál
- CS_A/G – SS kanál pro akcelerometr/gyroskop
- CS_M – SS kanál pro magnetometr
- CS_ALT – SS kanál pro barometr

Ostatní piny jsou pro připojení napájení, uzemnění a přerušení (INT). [11]

Kapitola 2

Ovladač VGA rozhraní pro zobrazení na monitoru

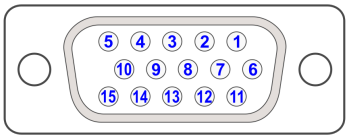
2.1 Historie a princip fungování VGA rozhraní

VGA (video graphic array) je standard počítačového rozhraní, které bylo představeno společností IBM v roce 1987. Bylo vytvořeno k propojení počítače s analogovým monitorem. Na základě tohoto rozhraní postupně vznikaly další, rozšířené standardy, jako např. SVGA, XGA nebo SXGA, které umožňují použití vyšších rozlišení a obnovovací frekvencí. Označujeme je jako zobrazovací režimy. VGA rozhraní je postupně nahrazováno digitálními rozhraními (např. HDMI nebo DisplayPort), které vznikly v souvislosti s příchodem digitálních zobrazovacích panelů (např. LCD), a které nabízejí vyšší kvalitu obrazu. Výhodou VGA oproti novým standardům je jeho jednoduchost. Základní zobrazovací režim VGA rozhraní je definován rozlišením 640×480 a frekvencí 60Hz.

2.1.1 VGA konektor

VGA rozhraní využívá 15pinový konektor DE-15, dnes běžně označovaný jako VGA konektor. Na obrázku 2.1 a v tabulce 2.1 jsou popsány jednotlivé piny konektoru. Piny 1 až 3 slouží pro připojení analogových RGB kanálů (Red, Green, Blue), které slouží k řízení intenzity 3 základních barevných složek

– červené, zelené a modré. Piny 5-8 a 10 slouží jako uzemnění jednotlivých kanálů. Pin 9 je nezapojen, případně je na něj připojeno napětí +5V. Piny 13 a 14 pak zajišťují připojení synchronizačních kanálů H-sync a V-sync (pro horizontální a vertikální synchronizaci). Kanály na pinech 4, 11, 12 a 15 původně sloužily pouze pro identifikaci monitoru (standard non-DDC). Způsob použití těchto kanálů se postupně měnil (standard DDC1) a v posledních standardech (DDC2B a E-DDC) VGA rozhraní umožňuje také konfiguraci monitoru, kdy kanály na pinech 12 a 15 fungují podle komunikačního standardu I²C. Ten je tvořen SCL kanálem pro hodinový signál a SDA kanálem pro datový signál, který je obousměrný a umožňuje čtení a zápis nastavení z a do paměti monitoru. [6]



Obrázek 2.1: VGA konektor, uspořádání pinů.

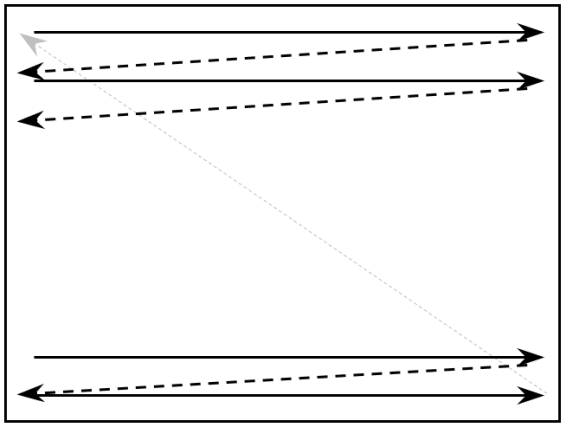
Pin	Signál
1	R (červená) *(analogový)*
2	G (zelená) *(analogový)*
3	B (modrá) *(analogový)*
5	GND
6	GND pro R
7	GND pro G
8	GND pro B
9	Nezapojen nebo +5V
10	GND pro H-sync a V-sync
13	H-sync
14	V-sync
4	Identifikace monitoru
11	Identifikace monitoru
12	SDA*)
15	SCL*)

*) jen u novějších standardů DDC2B a E-DDC, u starších pouze pro identifikaci monitoru

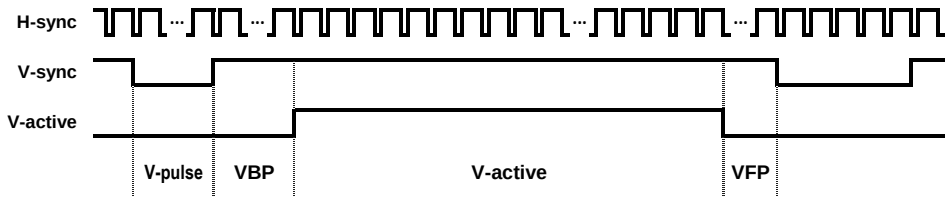
Tabulka 2.1: Popis pinů VGA konektoru. [6]

2.1.2 Zobrazování obrazu

Obrazovka monitoru je obnovována postupně po jednotlivých pixelech na každém řádku zleva doprava, a jednotlivé řádky pak odshora dolů. To znamená, že obnovovací cyklus začíná u pixelu nejvíce vlevo a nahoře a končí pixelem nejvíce vpravo a dole (obrázek 2.2). U základní frekvence 60 Hz je obrazovka, a tedy i každý její pixel obnoven 60krát za sekundu.
Obrazovky obsahují svítící elementy o třech barvách - červené, zelené a modré. Ty mohou být rozsvíceny s různou intenzitou a jejich kombinace vytvoří zobrazený barevný odstín. U CRT obrazovek jsou tyto tři barevné elementy tvořeny fluorescenčním malými tečkami nanesenými na stínítku, které jsou aktivovány elektronovým paprskem. Rozlišení u CRT tedy záleží na velikosti barevných teček v nátěru a šířce paprsku. LCD obrazovky jsou složeny z matice pixelů, které jsou tvořeny třemi segmenty, každý svítící jednou základní barvou. Rozlišení u LCD je dáno počtem pixelů v matici.
V počítači je obraz rozdělený na jednotlivé pixely. Sytost každé ze 3 barevných složek pixelu je kódována jako číslo, běžně 8bitové (0 až 255), dohromady tedy hovoříme o 24bitové hloubce, která umožnuje kódování přibližně 16.5 miliónů barev. Můžeme se ale setkat s nižší nebo vyšší bitovou hloubkou, např. až 48bitovou. DAC převodníkem jsou binární hodnoty základních 3 složek převedeny na analogové signály s výstupním napětím v rozmezí 0 V až 0.7 V (vyšší napětí odpovídá vyšší sytosti barevné složky). Pouze jeden pixel na obrazovce může být řízen v daný moment. Tehdy jeho složky dostávají patřičný pulz určující intenzitu rozsvícení. Poté je řízen následující pixel a předchozí začne zhasínat. Na základě obnovovací frekvence dostává pixel pravidelné krátké pulzy, které pixel opět rozsvítí, což při správné kalibraci vytvoří vjem požadovaného odstínu. Zároveň tak mohou být na jednotlivých pixelech obrazovky nastaveny různé barvy současně, a tím vytvořit zobrazený obraz. Ten navíc není statický, ale může se v čase velmi rychle měnit.



Obrázek 2.2: Směr obnovování pixelů obrazovky.



Obrázek 2.3: H-sync cyklus

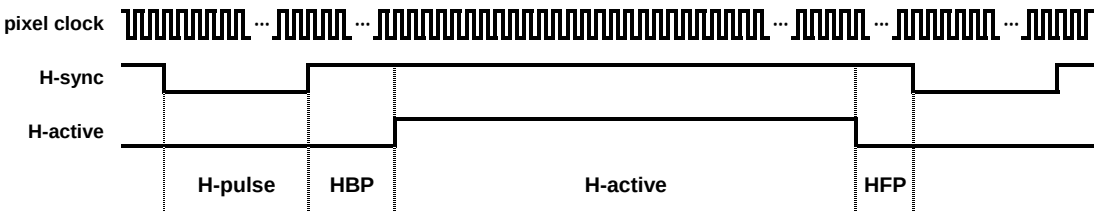
Zobrazovací režim			Pixel clock (MHz)	Polarita	Horizontální časování (počet pixel clock pulzů)				
Název	Rozlišení	Frekvence (Hz)			H-pulse	HBP	H-active	HFP	Total
VGA	640 × 480	60	25.175	–	96	48	640	16	800
VGA	640 × 480	75	31.5	–	96	48	640	16	800
SVGA	800 × 600	60	40	+	128	88	800	40	1056
SVGA	800 × 600	75	49.5	+	80	160	800	16	1056
SVGA	800 × 600	85	56.25	+	64	152	800	32	1048
XGA	1024 × 768	60	65	–	136	160	1024	24	1344
XGA	1024 × 768	85	94.5	+	96	208	1024	48	1376
SXGA	1280 × 1024	60	108	+	112	248	1280	48	1688

Tabulka 2.2: Horizontální (H-sync) časování pro vybrané zobrazovací režimy. [13]

2.1.3 Princip fungování VGA rozhraní

Výchozí hodinový signál pro VGA rozhraní je označován jako pixel clock, jehož frekvence je u základního zobrazovacího režimu 25.175 MHz. Na základě této frekvence jsou generovány synchronizační signály V-sync a H-sync.
Pro synchronizaci jednoho řádku (horizontální) je generován signál H-sync. Zde rozlišujeme 4 fáze: H-pulse, HBP (Horizontal Back Porch), H-active, HFP (Horizontal Front Porch), viz obrázek 2.3. Délky jednotlivých fází jsou určeny přesně daným počtem pixel clock pulzů (tabulka 2.2). Během H-pulse fáze je změněna logická úroveň signálu, což vytváří synchronizační pulz. Důležitá je polarita tohoto pulzu. Pokud je signál ve výchozím stavu na logické '1' a při synchronizačním pulzu je nastaven na logickou '0', pak se jedná o negativní polaritu, v opačném případě jde o pozitivní polaritu.

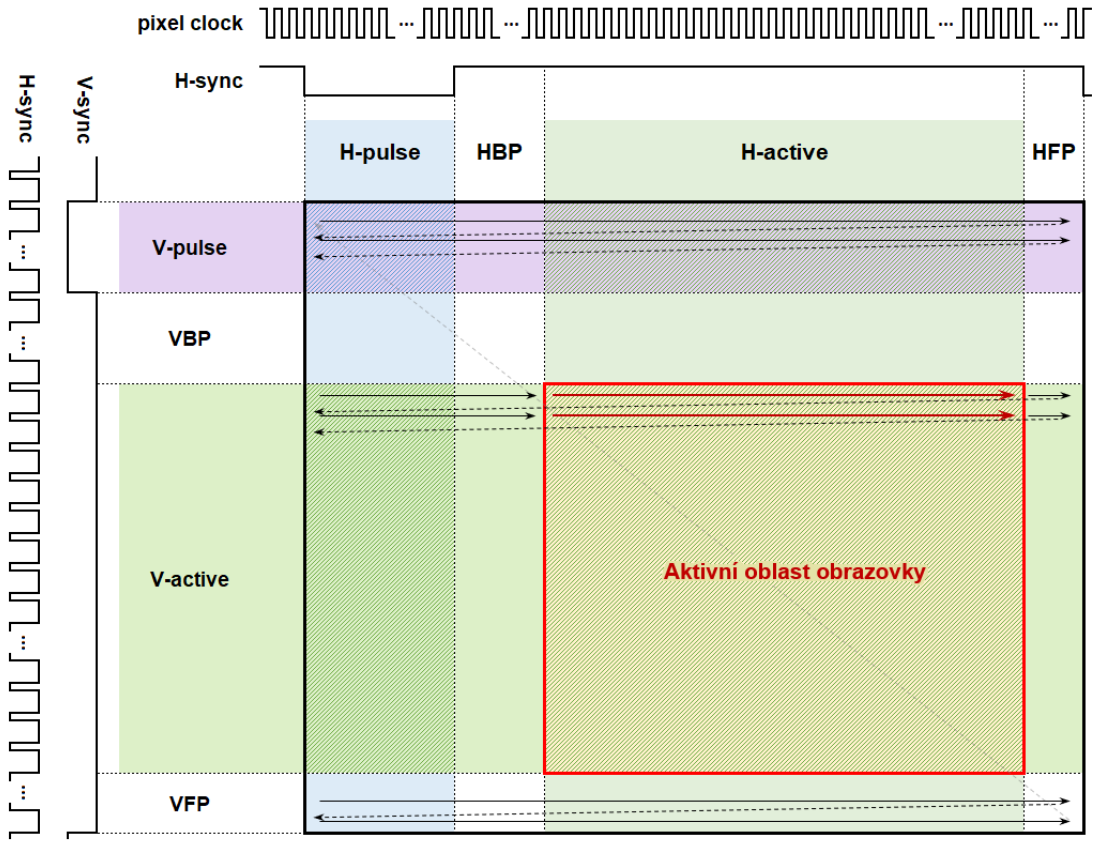
Pro vertikální synchronizaci je generován signál V-sync, kde podobně jako u H-sync rozlišujeme 4 fáze: V-pulse, VBP (Vertical Back Porch), V-active, VFP (Vertical Front Porch), viz obrázek 2.4. Délky těchto fází jsou určeny přesně daným počtem H-sync cyklů (tabulka 2.3). Během V-pulse fáze je generován synchronizační pulz. Logická úroveň během fází je opět nastavena podle toho, zda se jedná o pozitivní nebo negativní polaritu. U většiny zobrazovacích režimů je polarita V-sync a H-sync stejná, existují ale výjimky.



Obrázek 2.4: V-sync cyklus

Zobrazovací režim			Pixel clock (MHz)	Polarita	Vertikální časování (počet H-sync cyklů / řádků)				
Název	Rozlišení	Frekvence (Hz)			V-pulse	VBP	V-active	VFP	Total
VGA	640 × 480	60	25.175	–	2	33	480	10	525
VGA	640 × 480	75	31.5	–	2	33	480	10	525
SVGA	800 × 600	60	40	+	4	23	600	1	628
SVGA	800 × 600	75	49.5	+	3	21	600	1	625
SVGA	800 × 600	85	56.25	+	3	27	600	1	631
XGA	1024 × 768	60	65	–	6	29	768	3	806
XGA	1024 × 768	85	94.5	+	3	36	768	1	808
SXGA	1280 × 1024	60	108	+	3	38	1024	1	1066

Tabulka 2.3: Vertikální (V-sync) časování pro vybrané zobrazovací režimy. [**?**]



Obrázek 2.5: Celý cyklus obnovení obrazovky řízený signály H-sync a V-sync.

Monitor podle pulzů H-pulse a V-pulse nastaví na své straně frekvenci pro přepínání pixelů a načasuje začátek a konec aktivních fází – V-active a H-active. V momentu překryvu těchto aktivní fází pak přepíná mezi pixely a nastavuje barevný odstín podle napěťových úrovní RGB signálů. Jejich výstup proto musí být přesně načasován se synchronizačními pulzy.
Základní zobrazovací režim pracuje na principu negativní polarity, tedy ve výchozím stavu je logická '1' a sychronizační pulzy mají logickou '0'. Celková časová délka jednoho H-sync cyklu, neboli jednoho řádku, je u základního rozlišení $96 + 48 + 640 + 16 = 800$ pixel clock impulzů. Jeden V-sync cyklus obsahuje $2 + 33 + 480 + 10 = 525$ H-sync cyklů (řádků). Celkově se jedná o $800 \times 525 \times 60 = 25.2 \times 10^6$ clock pulzů za sekundu, což odpovídá základní pixel clock frekvenci 25.175 MHz.

2.2 Návrh VGA rozhraní ve VHDL

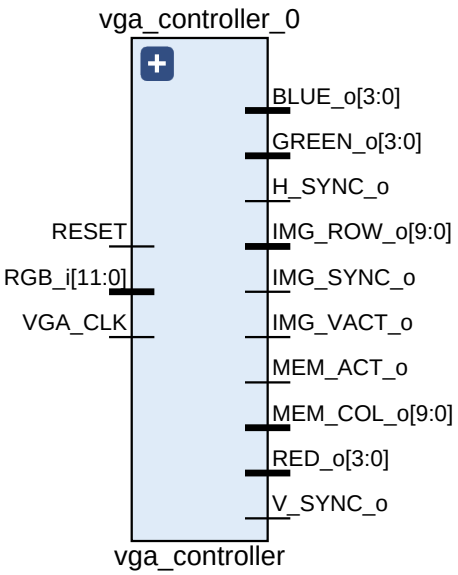
Celý návrh VGA rozhraní je vytvořen jako jedna entita ***vga_controller*** v souboru vga_controller.vhd.

Vstupní porty entity:

- **VGA_CLK** – VGA hodinový signál
- **RESET** – reset signál

Výstupní porty pro VGA modul:

- **RED_o** – červená barva
- **GREEN_o** – zelená barva
- **BLUE_o** – modrá barva
- **H_SYNC_o** – horizontální synchronizace
- **V_SYNC_o** – vertikální synchronizace



Obrázek 2.6: Schéma komponenty *vga_controller*

Výstupní porty pro generátor obrazu:

- **IMG_ROW_o** – aktivní řádek obrazovky
- **IMG_SYNC_o** – synchronizační signál začátku řádku
- **IMG_VACT_o** – signál aktivní vertikální oblasti

Porty pro čtení z paměti obrazu:

- **MEM_COL_o** – adresa pro čtení z paměti
- **MEM_ACT_o** – příznak povolení čtení
- **RGB_i** – 12bitová sběrnice pro barevný odstín čtený z paměti

Dále jsou deklarovány Generic konstanty pro délky jednotlivých fází vertikální a horizontální synchronizace a pro polarity signálů. Hodnoty konstant jsou nastaveny v entitě ***top*** a pomocí Generic jsou přiřazeny do ostatních entit. To umožňuje kdykoliv později upravit projekt na jiné rozlišení změnou konstant na jednom místě.

2.2.1 Architektura entity vga_controller

Hodinový signál

Základním hodinovým signálem pro řízení VGA rozhraní je ***vga_clk***, který je generován IP komponentou *ip_vga_clk_gen*. Ta byla vytvořena pomocí konfigurátoru Clocking Wizard. Komponenta je připojena k systémovému hodinovému signálu *SCLK* o frekvenci 125 MHz a frekvence *vga_clk* je nastavena na 25.125 MHz.

Řízení vertikální synchronizace - proces H_signals_gen

V procesu **H_signals_gen** jsou na základě počitadla ***H_cntr***, které je inkrementováno s každým vga_clk pulzem, určeny čtyři fáze horizontální synchronizace. V těchto fázích jsou nastavovány logické úrovně signálů ***H_sync*** a ***H_active***. Počitadlo ***H_cntr*** je resetováno při dosažení celkové délky horizontálního cyklu (součet délek fází cyklu).

Řízení vertikální synchronizace - proces V_signals_gen

V procesu **V_signals_gen** jsou na základě počitadla ***V_cntr***, které je inkrementováno po dokončení horizontálního cyklu (při resetu *H_cntr*), určeny čtyři fáze vertikální synchronizace. V těchto fázích jsou nastavovány logické úrovně signálů ***V_sync*** a ***V_active***. Počitadlo *V_cntr* je resetováno při dosažení celkové délky vertikálního cyklu.

Generování obrazu

Obraz je generován entitou *image_generator*. Mezi generátor a VGA rozhraní je vložena paměť, která má funkci řádkového bufferu. Do ní jsou generátorem ukládány hodnoty barev jednotlivých pixelů na aktivním řádku. Paměť je vygenerovaná pomocí Block Memory Generator 8.4. Jedná se o Simple Dual

Port RAM s datovou šířkou 12bitů a hloubkou 640. Je k ní vytvořen entita *image_row_mem_wrp* ve které je vytvořena instance paměti.

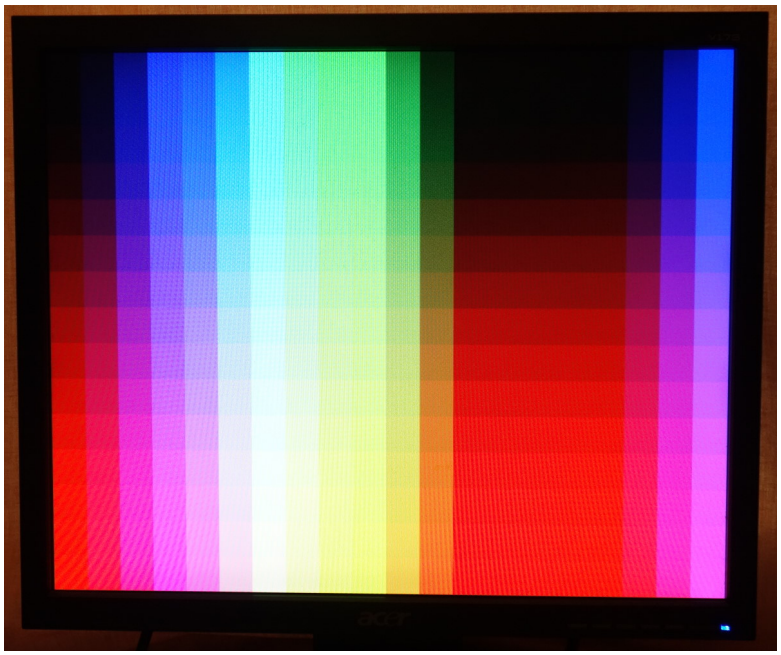
VGA entita na základě aktivního sloupce nastaví adresu pro čtení z paměti. Jelikož vybavení dat z dané adresy má zpoždění dva hodinové cykly, tak jsou do VGA entity vloženy zpožďující registry.

RGB barvu aktivního pixelu z paměti přenáší 12bitová sběrnice ***RGB_i***. Signály jsou převedeny do 4bitových signálů ***red***, ***green*** a ***blue***. Ty jsou poté spolu se zpožděnými synchronizačními signály připojeny na výstup.

2.2.2 Testování návrhu

Pro ověření funkce entity, především správného časování signálů, je vytvořen test bench v souboru tb_vga_controller.vhd a provedena behaviorální simulace ve Vivadu. Pro první praktický test zobrazení na monitoru byla vytvořena matice barev (obrázek 2.7), která ověřila, že synchronizace signálů a nastavování barev pixelů funguje správně.

Obrázek 2.7: Matice barev pro test VGA rozhraní.



Obrázek 2.8: Zobrazení matice barev na obrazovce monitoru při textování VGA rozhraní.

Kapitola 3

Ovladač SPI rozhraní pro komunikaci se senzorem

3.1 Historie, využití a princip SPI rozhraní

SPI (Serial Peripheral Interface) je synchronní sériové rozhraní pro komunikaci zařízení na krátkou vzdálenost. Bylo vyvinuto společností Motorola v 80. letech 20. století a stalo se velmi rozšířeným standardem. Využívá se především pro komunikaci mezi vestavěnými systémy a senzory. Rozhraní funguje v režimu Master/Slave. Master jen jeden a nemění se. Slave zařízení může být více. SPI rozhraní definuje 4 typy přenosových kanálů:

- **SCLK** – kanál pro řídící hodinový signál generovaný Master zařízením. Udává frekvenci přenosu dat.
- **MOSI** (Master Out, Slave In) – datový kanál, na kterém jsou data vysílány z Master směrem ke Slave.
- **MISO** (Master In, Slave Out) – datový kanál, na kterém Master přijímá data ze Slave.
- **SS** (Slave Select) – kanál mezi Master a Slave. Pro každý Slave je samostatný, tzn. že narozdíl od předchozích kanálů je SS kanálů více – jejich počet je roven počtu Slave zařízení připojených k Master. Těmito kanály Master určuje, se kterým Slave bude komunikovat.

Rozhraní funguje na frekvencích až v řádech jednotek MHz. Pro senzor

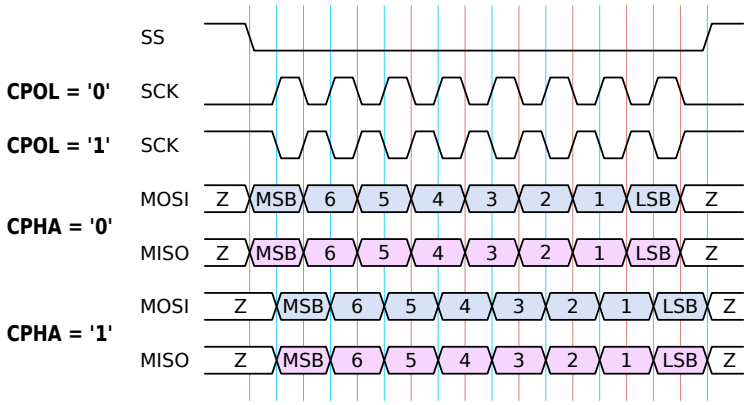
polohy a pohybu (modul Digilent Pmod NAV), použitý v tomto projektu, je maximální podporovaná frekvence 10 MHz. Některá zařízení mohou podporovat i frekvence vyšší než 10 MHz.

3.1.1 Průběh SPI komunikace

Master řídí kanály SCLK, MOSI a SS, Slave řídí kanál MISO. Ve stavu nečinnosti je na kanálech SS nastavena logická ’1’. Kanály MOSI a MISO jsou ve stavu vysoké impedance ’Z’. Na kanále SCLK záleží na zvolené polaritě CPOL (viz dále).

Pro zahájení komunikace je na kanále SS k požadovanému Slave zařízení nastavena logická ’0’ na ostatních ponechána logická ’1’. Následně je zahájeno vysílání hodinového signálu na kanále SCLK a vysílání dat na kanále MOSI. Časování těchto dvou kanálů záleží na zvolené polaritě a fázi hodinového signálu – CPOL a CPHA (clock polarity, clock phase). Jedná se o jednobitové parametry, tzn. mohou mít hodnotu logická ’1’ nebo ’0’. CPOL určuje polaritu SCLK v klidovém stavu. CPHA určuje, která hrana SCLK signálu je hlavní a která vedlejší. Pro CPHA = ’0’ je hlavní hranou ta, kde se mění SCLK signál z aktivní do klidové úrovně a hrana opačným směrem je vedlejší. Pro CPHA = ’1’ je to přesně naopak, tzn. hlavní hranou je přechod SCLK z klidové do aktivní úrovně. Kombinace parametrů CPOL a CPHA dávají 4 možné režimy časování SPI komunikace (znázorněny na obrázku 3.1).

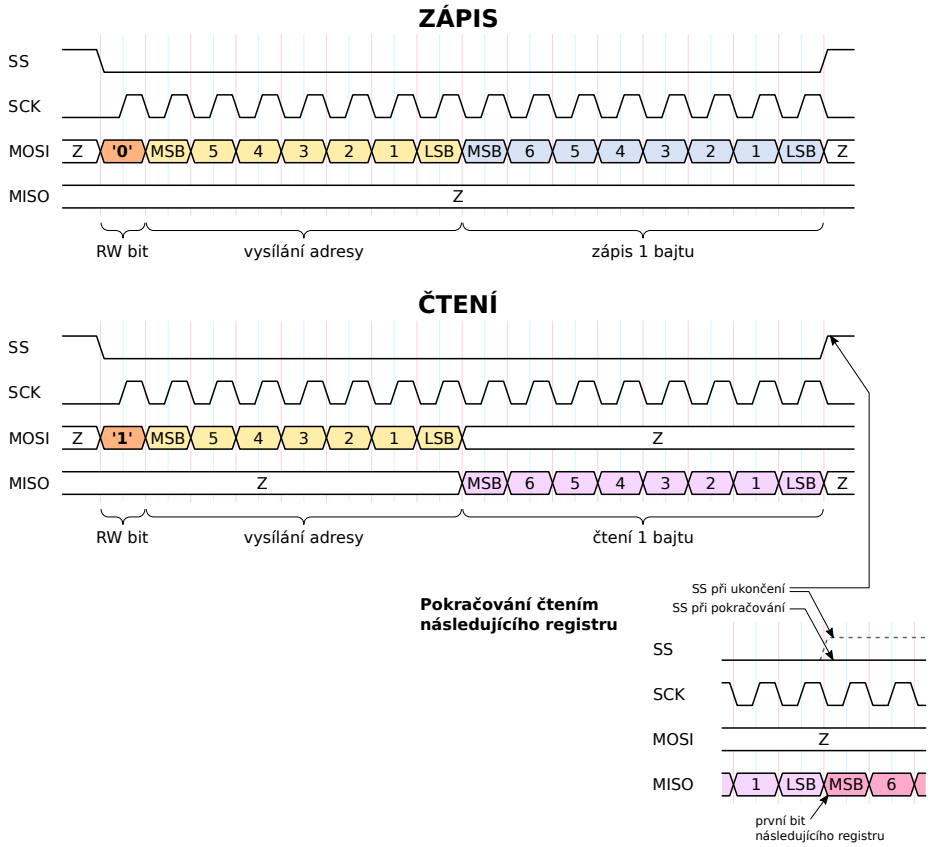
Prvním bitem vysílaným na MOSI kanále je RW bit (Read/Write), který určuje, zda bude Master číst data z registru Slave zařízení nebo do něj zapisovat. Logická ’0’ signalizuje zápis (Write), logická ’1’ čtení (Read). Poté



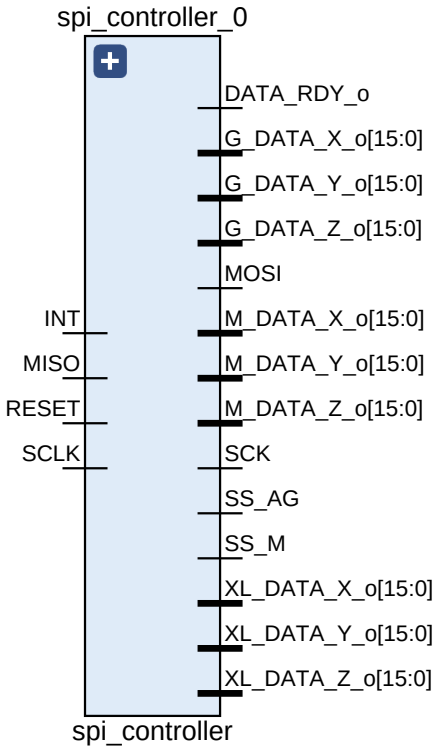
Obrázek 3.1: Režimy časování SPI komunikace určené parametry CPOL a CPHA

je na MOSI postupně odvysíláno 7 bitů kódující adresu cílového registru. Spolu s RW bitem je tímto zkompletován první bajt. Dále následuje přenos dalších 8 bitů (další celý bajt). Pokud byl RW bit nastaven na zápis (RW bit = '0'), tak jsou bity vysílány Master zařízením na MOSI kanále a Slave data zapíše do registru. V opačném případě (čtení, tzn. RW bit = '1') Slave zařízení na MISO kanále vysílá data uložená v registru. Nevyužitý kanál (MISO nebo MOSI) je nastaven do stavu vysoké impedance. Všechny přenášené bity (včetně adresy) jsou vysílány v pořadí od nejvýznamnějšího (MSB – most significant bit) k nejméně významnému (LSB – least significant bit). Logická úroveň nového bitu je nastavena vždy na hlavní hranu SCLK. Čtení bitů Master nebo Slave zařízením je naopak prováděno vždy na vedlejší hranu.

Po odvysílání bajtu může Master komunikaci ukončit nastavením aktivního SS na logickou '1'. Pokud SS zůstane nezměněn (aktivní, tzn. logická '0'), pak Slave buď opakuje vysílání bajtu nebo automaticky nastaví adresu následujícího registru a probíhá přenos 8 bitů pro zápis/čtení tohoto registru, stejným způsobem jako u předchozího.



Obrázek 3.2: Časování SPI komunikace pro zápis a čtení (CPOL = '0', CPHA = '0').



Obrázek 3.3: Schéma komponenty *spi_controller*

3.2 Návrh SPI rozhraní ve VHDL

Celý návrh SPI rozhraní je vytvořen jako jedna komponenta ***spi_controller*** v souboru *spi_controller.vhd*. Tato komponenta komunikuje s porty, ke kterým jsou připojeny senzory pomocí SPI kanálů, a další komponenty, které dále zpracovávají přijímaná data.

Vstupní porty:

- **SCLK** – systémový hodinový signál
- **RESET** – reset
- **MISO**
- **INT** – interrupt pro data gyroskopu

Výstupní porty pro SPI:

- **MOSI**
- **SCK**
- **SS_AG** – SS kanál pro akcelerometr a gyroskop
- **SS_M** – SS kanál pro magnetometr

Výstupní porty pro komponenty, dále zpracovávají přijímaná data:

- Devět 16bitových sběrnic, které přenášejí souřadnice ze tří senzorů (gyroskopu, akcelerometru, magnetometru)
- **DATA_RDY_o** – pro signalizaci, že jsou k dispozici nová data

3.2.1 Architektura komponenty spi_controller

Pomocný hodinový signál – proces aux_clk_gen

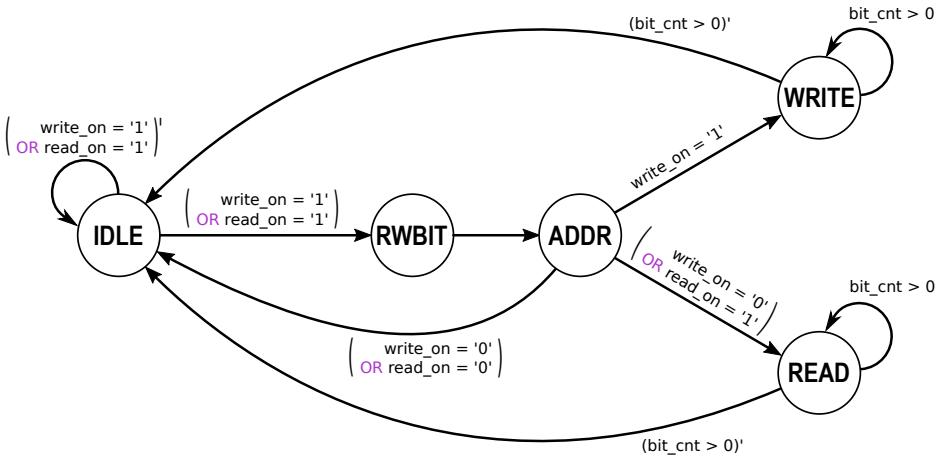
Pomocný hodinový signál ***aux_clk*** o frekvenci 100 kHz je generován procesem **aux_clk_gen**, který funguje jako dělička systémového hodinového signálu.

Konečný stavový automat pro řízení SPI cyklu

Na základě popisu průběhu SPI komunikace, je v procesu ***state_comb_logic*** vytvořen konečný stavový automat, který zajišťuje správné sekvence vysílaných signálů na jednotlivých kanálech. Stavy automatu nastavují logické úrovně na výstupních portech ***SCK***, ***SS_AG***, ***SS_M***, ***SS_ALT***, ***MOSI*** a dále nastavují pomocné signály ***nx_state*** (kóduje následující stav) a ***bit_timer*** (kóduje kolikrát bude následující stav zopakován, což je použito pro stavy, kde je vysíláno nebo čteno více bitů za sebou – např. vysílání adresy registru nebo zápis/čtení jednoho bajtu registru). Časování signálů generovaných automatem je v režimu CPOL = '0' a CPHA = '0', tzn. polarita hodinového signálu SCK je v klidovém stavu nastavena na logickou '0'. Hlavní hranou, na kterou je vysílán nový bit, je sestupná hrana. Vedlejší hranou, na kterou jsou čteny bity, je náběžná hrana.

Použité stavy automatu (schéma je zobrazeno na obrázku 3.4):

- **idle** – stav nečinnosti – na všech kanálech jsou nastaveny klidové signály.
- **rwbit** – stav pro nastavení RW bitu, který určí jestli bude probíhat čtení z registru nebo zápis do registru.
- **addr** – stav pro vysílání bitů adresy registru na MOSI, stav je opakován 7krát, což je nastaveno pomocí *bit_timer* v předchozím stavu rwbit a poté odečítáno počitadlem *bit_cnt*.
- **write** – stav pro vysílání bitů k zápisu na MOSI. Stav je opakován 8krát (pro zápis do jednoho 1bajtového registru). Může být opakován i vícekrát (v násobcích osmi – 16krát, 24krát apod.) pro zápis do více po sobě jdoucích 1bajtových registrů, ale tato možnost není v tomto projektu využita.
- **read** – stav pro čtení bitů z MISO. Stav je opakován v násobcích osmi, většinou 8krát, 16krát nebo 48krát pro čtení jednoho nebo více po sobě jdoucích 1bajtových registrů. Narozdíl od write zde již je tato možnost plně využita. Počet opakování je určen opět pomocí *bit_timer* (jako v případě 7 bitů u addr), který je nastaven ve stavu addr jako 8násobek čísla *bytes_to_rw* a pak odečítáno počitadlem *bit_cnt*.



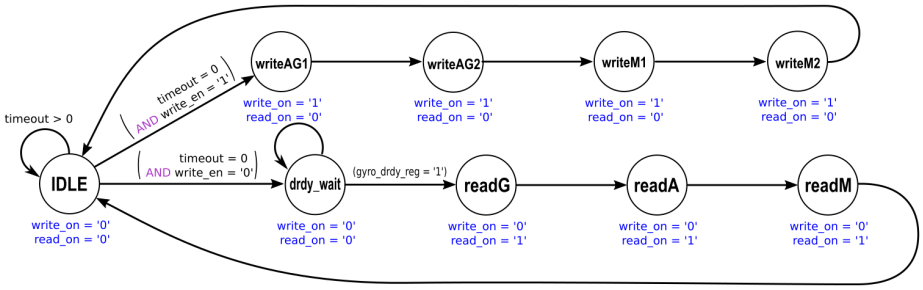
Obrázek 3.4: Schéma konečného stavového automatu pro řízení SPI cyklu.

Signály pro výstupní SPI porty jsou automatem nastavovány takto:

- ***MOSI*** – ve stavech *idle* a *read* je ve stavu vysoké impedance 'Z', v ostatních stavech (*rwbit*, *addr*, *write*) jsou vysílány bity pro Slave (RW bit, adresa, data k zápisu).
- ***SCK*** – ve stavu *idle* je na logické '0', v ostatních stavech je vysílán invertovaný pomocný hodinový signál *aux_clk*.
- ***S_AG***, ***SS_M*** nebo ***SS_ALT*** – ve stavu *idle* jsou všechny nastaveny na logickou '1' (klidová), v ostatních stavech je vždy jeden na základě *S_select* nastaven na logickou '0' (aktivní).

Stav je určen vstupním signálem ***pr_state***. Kromě něj jsou výstupní signály pro SPI porty určeny také dalšími vstupními signály:

- ***S_select*** – 2bitový, pro určení zařízení (senzoru) se kterým bude komunikováno
- ***write_on*** – pro zahájení cyklu SPI komunikace, pro nastavení RW bitu na logickou '0' (zápis), a pro nastavení stavu write po odvysílání adresy.
- ***read_on*** – pro zahájení cyklu SPI komunikace, pro nastavení RW bitu na logickou '1' (čtení), a pro nastavení stavu read po odvysílání adresy. V případě, že *write_on* je aktivní, tak hodnota *read_on* není relevantní.
- ***bit_cnt*** – číslo 0 až 47, počitadlo bitů zbývajících k odvysílání. Stav je opakován tolikrát, dokud je hodnota vyšší než 0. S každým opakováním je řídícím procesem hodnota snížena.
- ***bytes_to_rw*** – číslo 1 až 6, kóduje počet bajtů k zápisu/čtení v konkrétním jednom SPI cyklu.
- ***reg_addr*** – adresa registru, do kterého bude zapisováno nebo ze kterého bude čteno
- ***data_tx*** – data (1 bajt), která budou zapsána. Pokud je cyklus určen ke čtení, tak jsou všechny bity nastaveny na logickou '0'.



Obrázek 3.5: Schéma konečného stavového automatu pro řízení SPI fází.

Konečný stavový automat pro řízení fází

SPI cyklus je několikrát opakován. Při jednotlivých cyklech se liší jejich řídící vstupní signály. Např. protože je potřeba měnit Slave zařízení, se kterým je komunikace vedena, nebo jsou některé cykly určeny pro zápis a jiné pro čtení. K nastavení a spuštění těchto SPI cyklů je vytvořen další konečný stavový automat. Pro odlišení od předchozího automatu nejsou jednotlivé kroky nazývány jako stavy, ale jako fáze, a celek jako fázový automat. V každé fázi je spuštěn jeden SPI cyklus. Takto jsou ve stanoveném pořadí spuštěny SPI cykly pro zapsání konfiguračních dat do registrů, a poté cykly pro získání všech potřebných dat z registrů senzorů. Fáze je určena na základě signálu ***pr_phase*** a jsou během ní nastaveny všechny již dříve zmíněné vstupní signály pro probíhající SPI cyklus (*S_select*, *write_on*, *read_on*, *bytes_to_rw*, *reg_addr*, *data_tx*).

Přehled použitých fází (schéma je zobrazeno na obrázku 3.5):

- **idle** – fáze nečinnosti – žádný SPI cyklus neprobíhá. Pokud je nastaven timeout (přestávka mezi fázovými cykly), tak je odpočítáván.
- **writeAG1** a **writeAG2** – zápisy do konfiguračních registrů gyroskopu
- **writeM1** a **writeM2** – zápisy do konfiguračního registrů magnetometru
- **drdy_wait** – čekání na interrupt signál
- **readG** – čtení 6 bajtů pro data naměřená gyroskopem (2 bajty pro každou osu – x, y, z)
- **readA** – čtení 6 bajtů pro data naměřená akcelerometrem (2 bajty pro každou osu)
- **readM** – čtení 6 bajtů pro data naměřená magnetometrem (2 bajty pro každou osu)

Fáze určené pro zápis (konfiguraci) jsou přepnuty pouze jednou při prvním běhu systému nebo po resetování, a v dalších cyklech jsou přeskakovány. Během konfigurace jsou nastaveny tyto parametry pro rozsahy a rozlišení výstupních dat:

- Akcelerometru je nastaven na rozsah ±2g, rozlišení 1 bitu je 0,061mg.
- Magnetometru je nastaven na rozsah ±4gauss, rozlišení 1 bitu je 0,14mGs.
- Gyroskop je nastaven na rozsah ±500dps a frekvence vzorkování je 119Hz, rozlišení 1 bitu je 17,5mdps.

Řízení stavového a fázového automatu

Fáze a stavy jsou přepínány procesem **state_phase_ctrl**. Zahájení první fáze je spuštěno interrupt signálem *INT* v logické '1', který je nastaven gyroskopem v momentu, kdy jsou připravena nová data ke čtení. Přepnutí nového stavu (*pr_state* podle *nx_state*) je vždy na náběžnou hranu hodinového signálu *aux_clk*. Následující fáze je přepnuta (podle *nx_phase*) po ukončení probíhajího SPI cyklu, tzn. s přechodem stavového automatu do *idle* stavu. Proces dále nastavuje a řídí počitadlo bitů *bit_cnt* u stavů, kde je vysíláno nebo přijímáno více bitů (*addr*, *write*, *read*). Proces umožňuje celkový reset obou cyklů a je jím nastavován a odpočítáván timeout v případě, že mezi fázemi má být časová prodleva, např. po zapsání konfiguračních registrů.

Čtení dat a řízení výstupu

Čtení dat je zajištěno procesem **read_ctrl**. Jednotlivé bity přijímané na kanále MISO jsou zaznamenány na sestupnou hranu hodinového signálu *aux_clk* a zapisovány do ***data_rx***. Po přečtení celého bajtu jsou data na základě zdrojového senzoru a pořadí právě přečteného bajtu uloženy do příslušného bufferu.

3.2.2 Testování návrhu

Stejně jako u komponenty pro VGA rozhraní, je i u komponenty pro SPI rozhraní otestována její funkce. Pro ověření správného časování signálů a nastavo-

vaných logických úrovní, je vytvořen test bench v souboru tb_spi_controller.vhd a provedena behaviorální simulace ve Vivadu. Zároveň je zachycena komunikace pomocí logického analyzátoru SL-USB8CH (8 kanálový s šířkou pásma 24MHz) a záznamového prostředí Saleae Logic Software. Snímky obrazovky zachycující simulované průběhy signálů a záznamy z logického analyzátoru jsou v příloze B-1 až B-3.

Kapitola 4

Určení polohy v prostoru pomocí senzorů

4.1 Inerciální měřící jednotka

Inerciální měřící jednotka (IMU = inertial measurement unit) je elektronické zařízení, které umožňuje měřit působení určitých sil a zrychlení, které na toto zařízení působí. IMU obsahuje tři typy senzoru: akcelerometr, gyroskop a magnetometr. Na základě měření těmito senzory je možné určit polohu v prostoru, rychlost a směr pohybu, anebo také změny tohoto pohybu - zrychlení. [10] V této práci budou naměřené hodnoty těchto senzorů využity k určení polohy.

4.1.1 Akcelerometr

Akcelerometr je senzor měřící hodnoty zrychlení v určitém směru. 3osý akcelerometr dokáže měřit současně zryhlení ve směru všech tří os. Při určení polohy je cílem vyžít jeho schopnost měřit statické zrychlení, tzn. směr a velikost gravitačního zryhlení Země, a použít jej pro určení náklonu. Z tohoto principu vyplývá, že akcelerometrem nelze určit úplnou polohu senzoru v prostoru, jelikož při otáčení kolem osy kolmé k povrchu země se poloha senzoru mění, ale data měřená akcelerometrem zůstavají stejná. K tomuto účelu je nezbytné použít kombinaci s magnetometrem.

Při měření akcelerometrem je potřeba vzít v úvahu rušivé vlivy v podobě dynamických zrychlení, která vznikají při pohybu senzorem. Mohou to být odstředivá zrychlení při pohybu po křivce, vynulování gravitačního zrychlení při volném pádu, nebo krátkodobá zrychlení působící v mnoha směrech způsobena chvěním nebo vibracemi. Odchylky jsou rovněž způsobeny nedokonalostí samotného senzoru. Krátkodobá zrychlení lze odstranit aplikací jednoduchého filtru typu dolní propust, který zprůměruje určitý počet posledních naměřených hodnot, a tím vyhladí průběh. Dojde tím ale zpoždění reakce na změnu polohy. Lepším způsobem jak odstranit vliv dynamických zrychlení, je použití v kombinaci s gyroskopem.

4.1.2 Magnetometr

Magnetometr měří intenzitu magnetického pole, a proto může být použit pro měření magnetického pole Země. Směr tohoto pole je konstantní (pokud neměníme geografickou polohu, a pokud zanedbáme změny v čase v řádu měsíců a let). Kombinací se směrem gravitačního zrychlení již můžeme jednoznačně určit polohu senzoru v prostoru. Měření magnetometru jsou ovlivněny třemi hlavními vlivy:

- Poloha senzoru vůči směru gravitačního zrychlení - náklon senzoru. Pro určení polohy senzoru je potřeba provést tzv. tilt compensation pomocí dat z akcelerometru.
- Hard-iron efekt - konstantní magnetické pole, které nezávisí na poloze senzoru (vzniká uvnitř senzoru). Jeho neměnná hodnota je přičtena k velikosti měřeného magnetického pole. Výsledkem je konstantní směr vychýlení naměřených hodnot. Hard-iron efekt lze nejjednodušeji zjistit rotací senzoru kolem všech tří os v co nejširším rozsahu. Ze získaných hodnot určit maximální a minimální hodnotu pro každou osu a tyto dvě hodnoty zprůměrovat. Výsledkem je trojrozměrná hodnota výchylky, kterou lze odečíst od měřených dat.
- Soft-iron efekt - magnetické pole, které závisí na poloze senzoru. Výsledné hodnoty jsou ovlivněny jen v určitém směru. Projevuje se elipsovitým protažením celkového rozložení měřených hodnot v tomto směru. Pro kompenzaci je potřeba nalézt úhel, směr a velikost protažení.

Je důležité zmínit, že pomocí magnetometru určíme polohu senzoru vůči magnetickému severnímu pólu. Pro určení polohy vůči zeměpisnému severnímu pólu, je potřeba ještě vzít v úvahu magnetickou deklinaci, což je rozdíl mezi

těmito dvěma póly. Ten se liší v závislosti na geografické poloze. V mapě deklinací geomagnetického pole lze nalézt hodnotu příslušné odchylky a tu použít pro korekci.

4.1.3 Gyroskop

3osý gyroskop je senzor měřící úhlové rychlosti ve třech na sebe kolmých směrech. Integrací úhlových rychlostí podle času získáme velikosti změn úhlů, o které byl senzor pootočen za určitý časový úsek. Postupným sčítáním těchto přírůstků v čase a jejich přičtením k počáteční poloze lze určit polohu bez vlivu přímočarých zrychlení. Ovšem pouze samotný gyroskop také nelze použít, protože při delší době měření se vzniklé odchylky gyroskopu postupně nasčítají a skutečná poloha se bude lišit. Zároveň je také potřeba mít určenu počáteční polohu.

Měření gyroskopu vykazují určitou odchylku, která se projevuje tím, že i v klidovém stavu u senzoru jsou naměřeny malé přírůstky úhlů. Jelikož jsou většinou tyto přírůstky v čase konstantní, je možné je v rámci klidové kalibrace změřit, a poté od každého měření odečíst.

4.1.4 Komplementární filtr

Nejlepším řešením pro určení polohy je použití kombinace všech tří senzorů. K tomu lze použít různé typy filtrů. V této práci je použit komplementární filtr, který kombinuje data v určitém poměru. Poměr vlivu gyroskopu musí být výrazně vyšší oproti akcelerometru a magnetometru (např. 98:2), aby v krátkodobém časovém úseku polohu určovala především měření gyroskopu, a akcelerometr spolu s magnetometrem pouze korigovali nepřesnosti gyroskopu v delším čase. Přesné určení poměru závisí na požadavcích a konečném použití návrhu.

4.2 Souřadnicové systémy senzoru a země, Eulerovy úhly

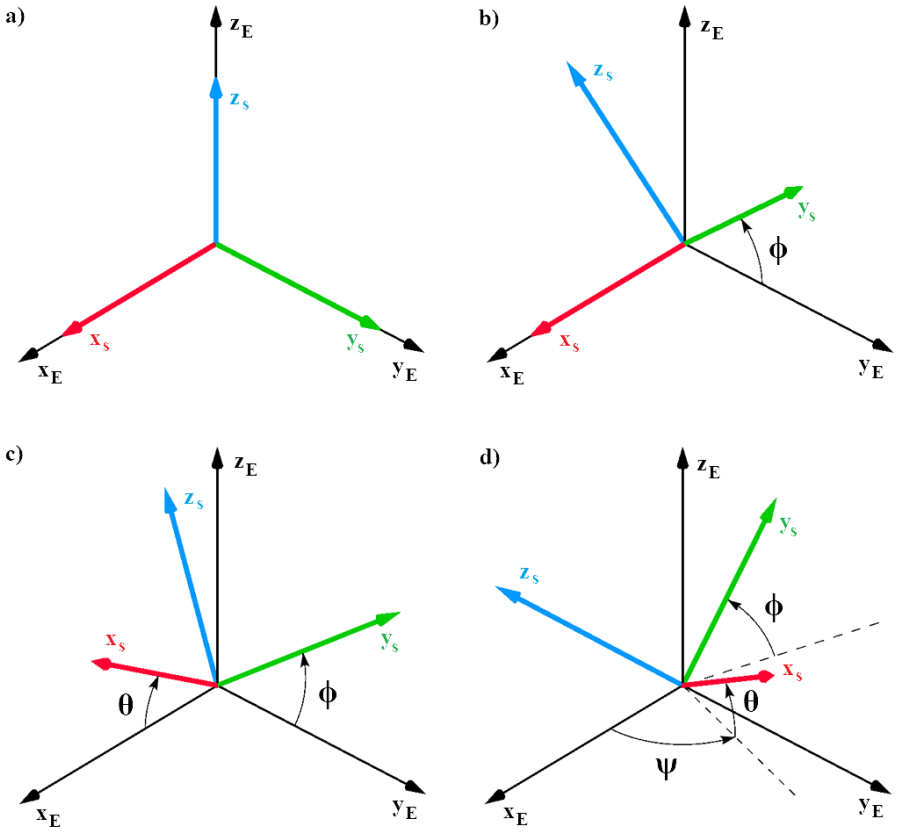
Pro určení polohy senzoru v prostoru je zapotřebí rozlišit dva pravotočivé kartezské souřadnicové systémy.[8] Souřadnicový systém senzoru (x_s, y_s, z_s) a souřadnicový systém země (x_E, y_E, z_E). Osa z_E je orientována kolmo k zemskému povrchu a směřuje od povrchu vzhůru. Osa x_E je kolmá na tuto osu (rovnoběžná s povrchem) a je orientována směrem k magnetickému severu (nikoliv ke geografickému severu - tato odchylka bude popsána v textu později). Poloha senzoru v prostoru je určena jako poloha těchto souřadnicových systémů vůči sobě. K tomu jsou použity tři Eulerovy úhly - yaw, pitch a roll.[9] Pro jednoznačnost určení je důležité pořadí ve kterém prováděny rotace senzorového souřadnicového systému vůči souřadnicovému systému Země. V základní poloze jsou osy senzoru orientovány souhlasně s osami země. Senzor nejprve otáčen pravotočivě o úhel ϕ (roll) kolem osy x_E, poté levotočivě o úhel θ (pitch) kolem osy y_E, a nakonec pravotočivě o úhel ψ (yaw) kolem osy z_E (obr. 4.1). Pro jednoznačné určení jakékoliv polohy jsou potřeba tyto rozsahy úhlů:

$$\begin{aligned}\psi &= \langle -\pi; \pi)\\ \theta &= \left\langle -\frac{\pi}{2}; \frac{\pi}{2} \right\rangle\\ \phi &= \langle -\pi; \pi)\end{aligned}$$

Změny souřadnic senzoru pro jednotlivé rotace lze zapsat pomocí matic R_E, P_E, Y_E (roll, pitch, yaw):

$$\begin{aligned}R_E &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{pmatrix}\\ P_E &= \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix}\\ Y_E &= \begin{pmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix}\end{aligned} \tag{4.1}$$

Konečná poloha, určená všemi třemi úhly, je poté získána vynásobním jednotlivých matic. Výsledná matice je označena M_E:



Obrázek 4.1: Určení polohy souřadnicového systému senzoru pomocí Eulerových úhlů: a) výchozí pozice, b) rotace systému o úhel ϕ (roll), c) rotace systému o úhel θ (pitch), d) rotace systému o úhel ψ (yaw)

$$M_E = R_E \times P_E \times Y_E = \begin{pmatrix} \cos\psi\cos\theta & \sin\psi\cos\theta & \sin\theta \\ -\sin\psi\cos\phi - \cos\psi\sin\theta\sin\phi & \cos\psi\cos\phi - \sin\psi\sin\theta\sin\phi & \cos\theta\sin\phi \\ \sin\psi\sin\phi - \cos\psi\sin\theta\cos\phi & -\cos\psi\sin\phi - \sin\psi\sin\theta\cos\phi & \cos\theta\cos\phi \end{pmatrix} \quad (4.2)$$

Pro další použití jsou označeny členy matice M_E a rozepsány jejich závislost na úhlech:

$$M_E = \begin{pmatrix} F_x & F_y & F_z \\ S_x & S_y & S_z \\ U_x & U_y & U_z \end{pmatrix} \quad (4.3)$$

$$
\begin{aligned}
F_{T,x} &= \cos\Delta\psi\cos\Delta\theta \\
F_{T,y} &= \Delta\psi\cos\Delta\theta \\
F_{T,z} &= \Delta\theta \\
S_{T,x} &= -\Delta\psi\cos\Delta\phi - \cos\Delta\psi\,\Delta\theta\,\Delta\phi \\
S_{T,y} &= \cos\Delta\psi\cos\Delta\phi - \Delta\psi\,\Delta\theta\,\Delta\phi \\
S_{T,z} &= \cos\Delta\theta\,\Delta\phi \\
U_{T,x} &= \Delta\psi\,\Delta\phi - \cos\Delta\psi\,\Delta\theta\cos\Delta\phi \\
U_{T,y} &= -\cos\Delta\psi\,\Delta\phi - \Delta\psi\,\Delta\theta\cos\Delta\phi \\
U_{T,z} &= \cos\Delta\theta\cos\Delta\phi
\end{aligned}
\tag{4.4}
$$

Členy matice určují polohu os senzoru v souřadnicovém systému Země. F_x, F_y, F_z určují polohu osy x_s, členy S_x, S_y, S_z polohu osy y_s a U_x, U_y, U_z polohu osy z_s. Tato matice nám umožní zobrazit polohu senzoru tak, jak je vnímáma pozorovatelem, tzn. z pohledu Země.

Hodnoty, které jsou získány měřením senzoru, popisují polohu Země vůči senzoru, jsou to tedy souřadnice v souřadnicovém systému senzoru. Pro popis polohy Země v tomto systému, za použití stejných úhlů, je zapotřebí provádět rotace okolo os senzoru. Z tohoto důvodu mají rotace obrácený směr (např. pokud je senzor otočen doprava, tak z pohledu senzoru se souřadnicový systém Země otočil doleva) a pořadí provádění rotací je opačné. Obrácená orientace úhlu se projeví na funkci sinus, která je lichá, tzn. že platí $sin(-\alpha) = -sin(\alpha)$. Funkce kosinus je sudá ($cos(-\alpha) = cos(\alpha)$), a proto změna úhlu hodnotu funkce nemění. V maticích R_Z, P_Z, Y_Z jsou proto všechny funkce sinus vynásobeny hodnotou -1. Takto upravené matice jsou označeny R_s, P_s, Y_s:

$$
\begin{aligned}
R_s &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{pmatrix} \\
P_s &= \begin{pmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{pmatrix} \\
Y_s &= \begin{pmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix}
\end{aligned}
\tag{4.5}
$$

Opačné pořadí provádění rotací je zohledněno změnou pořadí nasobení matic, a výsledná matice je označena M_s:

$$M_s = Y_s \times P_s \times R_s = \\ = \begin{pmatrix} \cos\theta\cos\psi & -\cos\phi\sin\psi - \cos\psi\sin\theta\sin\phi & \sin\phi\sin\psi - \cos\phi\cos\psi\sin\theta \\ \cos\theta\sin\psi & \cos\phi\cos\psi - \sin\theta\sin\phi\sin\psi & -\cos\psi\sin\phi - \cos\phi\sin\theta\sin\psi \\ \sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{pmatrix} \quad (4.6)$$

Členy matice M_s jsou označeny (použitím malých písmen k odlišení od členů matice M_E) a jsou rozepsány jejich závislosti na úhlech:

$$M_s = \begin{pmatrix} f_x & f_y & f_z \\ s_x & s_y & s_z \\ u_x & u_y & u_z \end{pmatrix} \quad (4.7)$$

$$\begin{aligned} f_x &= \cos\psi\cos\theta \\ f_y &= -\sin\psi\cos\phi - \cos\psi\sin\theta\sin\phi \\ f_z &= \sin\psi\sin\phi - \cos\psi\sin\theta\cos\phi \\ s_x &= \sin\psi\cos\theta \\ s_y &= \cos\psi\cos\phi - \sin\psi\sin\theta\sin\phi \\ s_z &= -\cos\psi\sin\phi - \sin\psi\sin\theta\cos\phi \\ u_x &= \sin\theta \\ u_y &= \cos\theta\sin\phi \\ u_z &= \cos\theta\cos\phi \end{aligned} \quad (4.8)$$

Porovnáním členů matic M_E a M_s lze získat převodní vztahy ze souřadnicového systému senzoru (naměřené hodnoty, členy napravo) do souřadnicového systému Země (členy nalevo):

$$\begin{aligned} F_x &= f_x \\ F_y &= s_x \\ F_z &= u_x \\ S_x &= f_y \\ S_y &= s_y \\ S_z &= u_y \\ U_x &= f_z \\ U_y &= s_z \\ U_z &= u_z \end{aligned} \tag{4.9}$$

V maticovém tvaru:

$$M_E = \begin{pmatrix} f_x & s_x & u_x \\ f_y & s_y & u_y \\ f_z & s_z & u_z \end{pmatrix} \tag{4.10}$$

Je tedy patrné, že pro převod do M_E matici M_s pouze transponujeme.

Na závěr této části je také potřeba definovat převodní vztahy z matice M_E zpět na úhly ψ, θ, ϕ.

$$\begin{aligned} \psi &= \operatorname{arctg} \frac{F_y}{F_x} && F_x \neq 0 \\ \theta &= \operatorname{arctg} \frac{\sqrt{F_x^2 + F_y^2}}{F_z} && F_z \neq 0 \\ \phi &= \operatorname{arctg} \frac{S_z}{U_z} && U_z \neq 0 \end{aligned} \tag{4.11}$$

Definiční obory těchto vztahů jsou omezeny. K odstranění těchto omezení je použita funkce atan2:

$$\begin{aligned} \psi &= \operatorname{atan2}(F_y, F_x) \\ \theta &= \operatorname{atan2}(\sqrt{F_x^2 + F_y^2}, F_z) \\ \phi &= \operatorname{atan2}(S_z, U_z) \end{aligned} \tag{4.12}$$

4.3 Určení polohy na základě měření akcelerometrem a magnetometrem

Měřením akcelerometrem je získán vektor zrychlení $\vec{a}$.

$$\vec{a} = (a_x, a_y, a_z) \tag{4.13}$$

Pokud je senzor v klidu, tak tento vektor odpovídá směru a velikosti gravitačního zrychlení Země, směřuje kolmo k jejímu povrchu, a pokud je převrácen a normován, tak získá vektor $\vec{u}$, který přímo určuje osu z_Z souřadnicového systému Země.

$$\vec{u} = (u_x, u_y, u_z) = -\frac{\vec{a}}{|\vec{a}|} \tag{4.14}$$

Ovšem nelze podle něj určit směry zbývajících os.

Měřením magnetometrem je získán vektor směru magnetického pole.

$$\vec{m'} = (m'_x, m'_y, m'_z) \tag{4.15}$$

Tento vektor obsahuje složku, která směruje k magnetickému severu, a složku, která směřuje dolů kolmo k zemskému povrchu. Vektor tedy sám o sobě není rovnoběžný se zemským povrchem a nelze jej rovnou použít pro určení polohy osy x_Z. Vektor je rovněž normován.

$$\vec{m} = (m_x, m_y, m_z) = -\frac{\vec{m'}}{|\vec{m'}|} \tag{4.16}$$

Vektorovým součinem obou normovaných vektorů a normováním výsledku je získán vektor $\vec{s}$, který určuje osu y_Z.

$$\vec{s'} = \vec{a} \times \vec{m} \tag{4.17}$$

$$\vec{s} = (s_x, s_y, s_z) = -\frac{\vec{s'}}{|\vec{s'}|} \tag{4.18}$$

Dalším vektorovým součinem je pak získán vektor $\vec{f}$, který určuje osu x_Z.

$$\vec{f} = \vec{s} \times \vec{a} \tag{4.19}$$

Získáné vektory svými členy odpovídají jednotlivým řádkům v matici M_s. Lze je tedy pomocí převodních vztahů převést na matici ve tvaru M_E a určit tak polohu senzoru. Tato matice souřadnic vůči Zemi získaná pomocí akcelerometru a magnetometru je pro další použití označena jako $M_{a/m}$. Je potřeba zdůraznit, že poloha takto zjištěná platí pouze pokud je senzor v klidu. Pokud je senzor v pohybu, tak směr vektoru $\vec{a}$ bude ovlivněn dynamickými zrychleními, což se projeví i na výsledná matici.

4.4 Určení polohy senzoru na základě měření gyroskopem

Gyroskop umožňuje měřit úhlové rychlosti rotace kolem os senzoru - výsledkem tohoto měření je vektor $\vec{\omega_g}$.

$$\vec{\omega_g} = (\omega_x, \omega_y, \omega_z) \tag{4.20}$$

Platí, že integrací úhlové rychlosti v čase od t_1 do t_2 je získána velikost úhlu o který byla rotace za daný čas provedena.

$$\Delta\phi = \int_{t_1}^{t_2} \omega dt \tag{4.21}$$

Digitální senzory neměří spojitě, ale diskrétně ve vzorcích. Pokud t_1 a t_2 představují časy dvou po sobě jdoucích měření a ω_1 je hodnota prvního měření, pak je integrace aproximována násobením:

$$\Delta\phi = (t_2 - t_1)\omega_1 = \Delta t\omega_1 \tag{4.22}$$

Pro určení velikosti změny úhlu gyroskopem tedy nestačí pouze naměřené hodnoty, ale je potřeba znát dobu Δt mezi jednotlivými měřeními. Ta je určena nastavenou frekvencí vzorkování f_s

$$\Delta t = \frac{1}{f_s} \tag{4.23}$$

Pro rotace kolem jednotlivých os gyroskopu jsou zavedeny Eulerovy úhly ψ_g, θ_g, ϕ_g. Orientace a pořadí skládní je definováno tak, že první je prováděna rotace pravotočivě o úhel ϕ_g kolem osy x_s (osa x senzoru - gyroskopu), poté levotočivě o úhel θ_g kolem osy y_s, a nakonec pravotočivě o úhel ψ_g kolem osy z_s. Pro závislosti jejich přírůstků na úhlových rychlostech ω_g platí vztahy:

$$\begin{aligned} \Delta\psi_g &= \Delta t\omega_z = \frac{1}{f_s}\omega_z \\ \Delta\theta_g &= \Delta t\omega_y = \frac{1}{f_s}\omega_y \\ \Delta\phi_g &= \Delta t\omega_x = \frac{1}{f_s}\omega_x \end{aligned} \tag{4.24}$$

Směr a pořadí rotací jsou definovány stejně jako v případě polohových úhlů senzoru vůči Zemi (označení bez indexu g). Rotace o tyto úhly jsou ale prováděny v jiných souřadnicových systémech. Nelze je tedy sčítat. Řešením je zavedení přechodové matice mezi polohami a převodu přírůstků úhlů na tuto matici.

Polohy senzoru vůči Zemi v čase t_1 a t_2 lze označit jako P_1 a P_2. Poloha P_1 je určena úhly ψ_1, θ_1 a ϕ_1. Tyto úhly jsou dosazeny do matice M_E. Výsledkem

je matice M_{P1} se souřadnicemi určujících polohu senzoru vůči Zemi v čase t_1. Stejným způsobem lze získat matici M_{P2} se souřadnicemi určující polohu senzoru vůči Zemi v čase t_2. Pro přechod mezi M_{P1} a M_{P2} je zavedena přechodová matice T

$$M_{P2} = T \times M_{P1} \tag{4.25}$$

Ta představuje matici souřadnic polohy P_2 vůči souřadnicovému systému senzoru v poloze P_1.

$$T = \begin{pmatrix} F_{T,x} & F_{T,y} & F_{T,z} \\ S_{T,x} & S_{T,y} & S_{T,z} \\ U_{T,x} & U_{T,y} & U_{T,z} \end{pmatrix} \tag{4.26}$$

Pokud jsou rotace o úhly $\Delta\psi$, $\Delta\theta$ a $\Delta\phi$ v tomto souřadnicovém systému prováděny se stejnou orientací jako rotace o ψ, θ a ϕ v souřadnicovém systému Země, pak lze k získání souřadnic v matici T použít stejné převodní vztahy jako v matici M_E:

$$\begin{aligned}
F_{T,x} &= cos(\Delta\psi)cos(\Delta\theta) \\
F_{T,y} &= sin(\Delta\psi)cos(\Delta\theta) \\
F_{T,z} &= sin(\Delta\theta) \\
S_{T,x} &= -sin(\Delta\psi)cos(\Delta\phi) - cos(\Delta\psi)sin(\Delta\theta)sin(\Delta\phi) \\
S_{T,y} &= cos(\Delta\psi)cos(\Delta\phi) - sin(\Delta\psi)sin(\Delta\theta)sin(\Delta\phi) \\
S_{T,z} &= cos(\Delta\theta)sin(\Delta\phi) \\
U_{T,x} &= sin(\Delta\psi)sin(\Delta\phi) - cos(\Delta\psi)sin(\Delta\theta)cos(\Delta\phi) \\
U_{T,y} &= -cos(\Delta\psi)sin(\Delta\phi) - sin(\Delta\psi)sin(\Delta\theta)cos(\Delta\phi) \\
U_{T,z} &= cos(\Delta\theta)cos(\Delta\phi)
\end{aligned} \tag{4.27}$$

Přírůstky úhlů jsou tedy převedeny na souřadnice relativní k předchozí poloze, a ty jsou maticově vynásobeny s absolutními souřadnicemi předchozí polohy. Pro další postup je upraveno označení matic. Matice předešlé polohy (M_{P1}) bude dále značena jako M_o a matice nové polohy M_{P2} bude značena M_g. Převodní matice meci polohami bude značna T_{dg}. Pro jednu iteraci měření platí

$$M_g = T_{dg} \times M_o \tag{4.28}$$

Pro další iteraci matici M_g použijeme jako M_o a opět vynásobíme s novou T_{dg}. Ovšem opakování tohoto postupu pouze s naměřenými daty gyroskopu, by po větším počtu iterací vedlo k odchylce získaných souřadnic od reálné polohy. Důvodem jsou nedokonalosti senzoru a nahrazení integrace násobením, které způsobí drift. Tuto odchylku lze korigovat pomocí souřadnic získaných akcelerometrem/magnetometrem, které určuje matice $M_{a/m}$.

4.5 Komplementární filtr

Komplementární filtr je použit ke kombinaci dat z gyroskopu a akcelerometru/magnetometru. Obecně jej lze jej popsat vztahem

$$C = nA + (1 - n)B \qquad (4.29)$$
$$n = (0; 1)$$

kde A jsou data jednoho senzoru, B data druhého, n je poměr s jakým jsou data A a B komplementovány, a C jsou výsledná data. V tomto vztahu platí, že zvyšování n zvyšuje podíl dat A a snižuje podíl B. Do vztahu ovšem nelze za A a B dosadit přímo matice $M_{a/m}$ a M_g, protože souřadnice ve výsledné matici by neurčovaly vždy pravoúhlé vektory. Místo toho jsou matice převedeny na úhly:

$$\begin{aligned} \psi_{a/m} &= \operatorname{atan2}\left(F_{y,a/m}, F_{x,a/m}\right) \\ \theta_{a/m} &= \operatorname{atan2}\left(\sqrt{F_{x,a/m}^2 + F_{y,a/m}^2}, F_{z,a/m}\right) \\ \phi_{a/m} &= \operatorname{atan2}\left(S_{z,a/m}, U_{z,a/m}\right) \end{aligned} \qquad (4.30)$$

$$\begin{aligned} \psi_g &= \operatorname{atan2}\left(F_{y,g}, F_{x,g}\right) \\ \theta_g &= \operatorname{atan2}\left(\sqrt{F_{x,g}^2 + F_{y,g}^2}, F_{z,g}\right) \\ \phi_g &= \operatorname{atan2}\left(S_{z,g}, U_{z,g}\right) \end{aligned} \qquad (4.31)$$

Filtr je poté aplikován na tyto úhly. Výsledkem jsou konečné výstupní úhly:

$$\begin{aligned} \psi_o &= n\psi_{a/m} + (1 - n)\psi_g \\ \theta_o &= n\theta_{a/m} + (1 - n)\theta_g \\ \phi_o &= n\phi_{a/m} + (1 - n)\phi_g \end{aligned} \qquad (4.32)$$

4.6 Shrnutí použitých výpočtů

Na začátku každé iterace určení polohy máme k dispoci nová data ze senzorů v podobě vektorů $\vec{a}$ z akcelerometru, $\vec{m}$ z magnetometru, $\vec{\omega_g}$ z gyroskopu. Dále máme k dispozi matici M_o, která obsahuje výsledné souřadnice polohy

senzoru vůči Zemi z předcházející iterace. Vektory $\vec{a}$ a $\vec{m}$ jsou převedeny na matici $M_{a/m}$. Vektor $\vec{\omega_g}$ je převeden na přírůstky úhlů a poté na přechodovou matici relativních souřadnic T_{dg}. Matice je vynásobena s maticí M_o a je získána matice M_g. Matice $M_{a/m}$ a M_g jsou převedeny na úhly, na které je aplikován komplementární filtr. Výsledkem jsou úhly ψ_o, θ_o, ϕ_o, které opět převedeme na novou matici M_o, která je pak použita v další iteraci. Posledně zmíněné úhly a matice (s indexem "o") slouží jako výstup celého systému. Mohou být využity pouze úhly nebo pouze matice, záleží na typu aplikace, ve které jsou data využity.

4.7 Návrh entity ve VHDL pro zpracování dat na FPGA

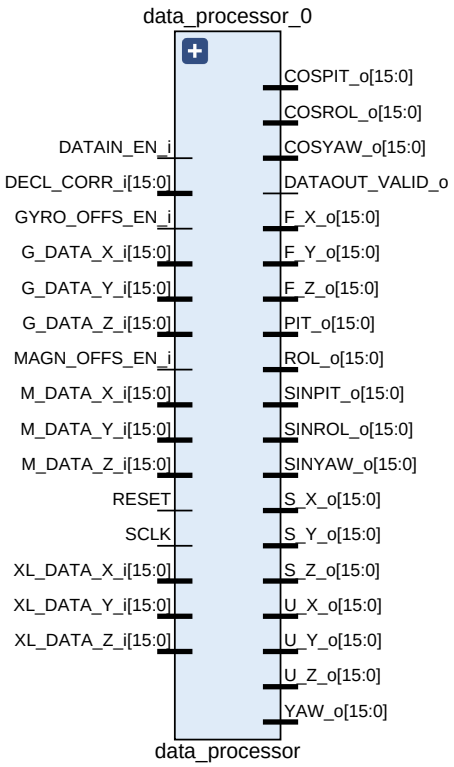
Pro zpracování data získáných se senzoru přes SPI rozhraní (v entitě *spi_controller*) byla navržena entita ***data_processor***.

Na vstupní porty entity jsou připojeny tyto signály a sběrnice:

- hodinový signál 125MHz a reset
- signály tlačítek a přepínačů, které jsou určeny k ovládání některých procesů uživatelem
- 16 bitové datové sběrnice, které jsou určeny pro data získaná SPI rozhraním. Pro každou osu a pro každý senzor je sběrnice zvlášť, tzn. celkem je jich 9. Každá kóduje číslo dvojkovém doplňku.
- signál pro příznak z SPI rozhraní, který signalizuje nová data na vstupních sběrnicích
- 16 bitová datová sběrnice, která přenáší hodnotu pro korekci magnetické deklinace jako číslo Q1.14. Zdrojovou entitou pro tuto sběrnici je ***deklin_corr_unit***, která je umístěna mimo entitu *data_processor* a bude popsána v kapitole Kompletace návrhu.

Signály a sběrnice na výstupních portech jsou:

- signál pro příznak platnosti výstupních dat
- 16 bitové datové sběrnice pro výstup vypočtených úhlů. Jedná se o 16 bitová čísla, která jsou škálovaná aby celým svým rozsahem kódovaly hodnotu v rozsahu $\langle -\pi; \pi)$ nebo $\langle 0; 2\pi)$. To znamená, že jeden bit 16bitového čísla kóduje úhel $2pi/2^{16}$. Oba zmíněné rozsahy jsou zaměnitelné. Výhodou tohoto přístupu je, že pro hodnoty úhlu je využit celý rozsah 16bitového čísla a zároveň není potřeba při sčítání nebo odečítání dvou úhlů řešit přetečení, resp. podtečení. Protože se tím dostaneme do další periody a směrové vektory budou stále odpovídat skutečnosti. A poslední výhodou je možnost číslo použít jako signed nebo unsigned bez potřeby převodu. Nevýhodou je, že hodnoty většiny celých stupňů nelze takovým číslem přesně kódovat. Vyjmenované výhody ale výrazně zjednodušují celý návrh, a proto byla škálovaná čísla pro hodnoty úhlů použita.



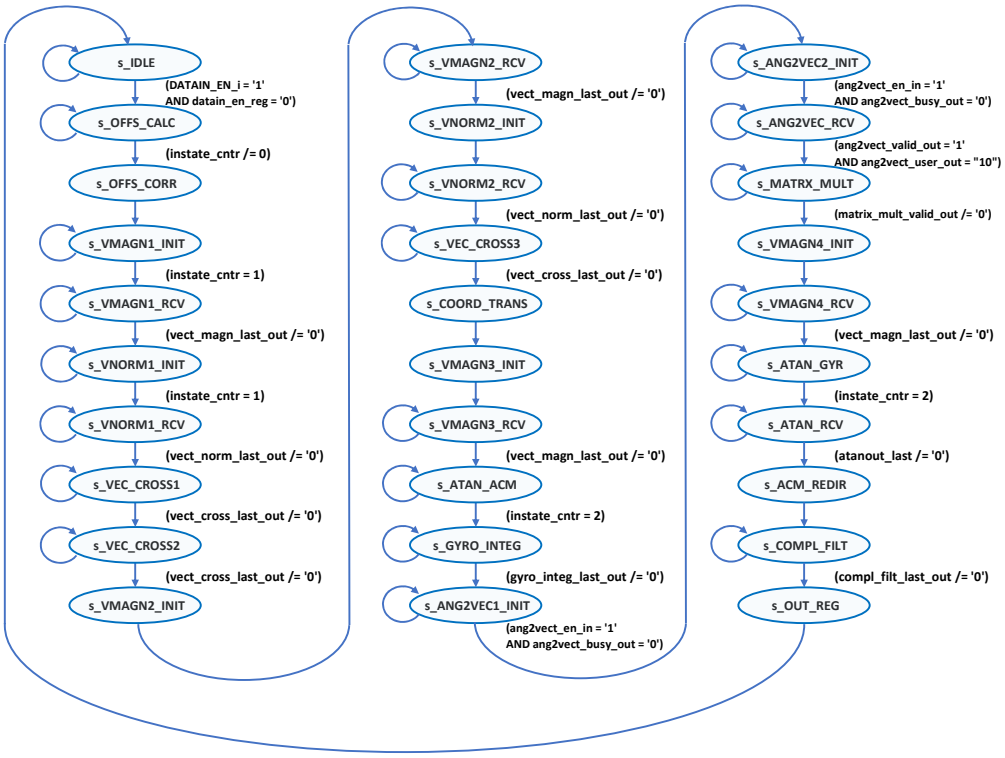
Obrázek 4.2: Schéma komponenty *data_processor*

- 16 bitové datové sběrnice pro výstup vypočtených sinů a kosinů úhlů. Hodnoty jsou přenášeny jako čísla Q1.14, které kódují hodnotu v rozsahu $\langle -1.0; 1.0 \rangle$ (Číslo Q1.14 umožňuje větší rozsah, ale ten není plně využit).
- 16 bitové datové sběrnice pro výstup souřadnic polohy senzoru. Sběrnice přenášejí čísla Q1.14, které kódují hodnoty souřadnic normované na rozsah $\langle -1.0; 1.0 \rangle$.

Jádro entity je navrženo jako konečný stavový automat. Na základě stavů automatu jsou spouštěny výpočetní jednotky a jejich výstupy jsou přiřazovány do registrů. Cyklus automatu je spuštěn příznakem z SPI rozhraní. Přechody do dalšího stavu jsou buď na základě počitadla, které přepne stav při dosažení určité hodnoty, nebo na základě přijetí příznaku o dokončení výpočetní operace. Schéma stavového automatu je zobrazenona obrázku 4.3. Popis stavů automatu:

- s_OFFS_CALC – výpočet offsetu (gyroskopu a akcelerometru)

- s_OFFS_CORR – odečtení offsetu od naměřených dat (gyroskopu a akcelerometru)
- s_VMAGN1_INIT – zahájení výpočtu velikosti vektorů $\vec{a}$ a $\vec{m}_{corr}$
- s_VMAGN1_RCV – přijetí vypočtených velikostí vektorů $\vec{a}$ a $\vec{m}_{corr}$
- s_VNORM1_INIT – zahájení normování vektorů $\vec{a}$ a $\vec{m}_{corr}$
- s_VNORM1_RCV – přijetí normovaných vektorů $\vec{a}_n$ a $\vec{m}_n$
- s_VEC_CROSS1 – vektorový součin $\vec{a}_n \times \vec{m}_n$
- s_VEC_CROSS2 – vektorový součin $\vec{s} \times \vec{a}_n$
- s_VMAGN2_INIT – zahájení výpočtu velikosti vektoru $\vec{f}$
- s_VMAGN2_RCV – přijetí vypočtených velikostí vektoru $\vec{f}$
- s_VNORM2_INIT – zahájení normování vektoru $\vec{f}$
- s_VNORM2_RCV – přijetí normovaného vektoru $\vec{f}_n$
- s_VEC_CROSS3 – vektorový součit $\vec{a}_n \times \vec{f}_n$
- s_COORD_TRANS – převod ze souřadnicového systém senzoru do systému Země
- s_VMAGN3_INIT – výpočet velikosti $\sqrt{F_x^2 + F_y^2}$
- s_VMAGN3_RCV – přijetí vypočtené velikosti
- s_ATAN_ACM – převod vektorů na úhly
- s_GYRO_INTEG – integrace data gyroskopu
- s_ANG2VEC1_INIT – převod úhlů na vektory pro výstupní úhly minulého cyklu
- s_ANG2VEC2_INIT – převod úhlů na vektory pro přírustky gyroskopu
- s_ANG2VEC_RCV – receive převedených souřadnic vektorů
- s_MATRX_MULT – maticový součin
- s_VMAGN4_INIT – výpočet velikosti $\sqrt{(F_x^2 + F_y^2)}$
- s_VMAGN4_RCV – přijetí vypočtené velikosti
- s_ATAN_GYR – převod vektorů na úhly pro souřadnice určené gyroskopem
- s_ATAN_RCV – přjetí vypočtených úhlů



Obrázek 4.3: Schéma konečného stavového automatu entity *data_processor*

- s_ACM_REDIR – sjednocení orientace úhlů mezi úhly gyroskopu a úhly akcelerometru/magnetometru
- s_COMPL_FILT – komplementární filtr
- s_OUT_REG – výstup dat

4.7.1 Kalibračních procesy

Ihned po přijetí nových dat jsou na začátku cyklu stavového automatu zařazeny stavy určené pro hledání ofsetů a kalibraci dat. To je prováděno pro data magnetometru a gyroskopu.

Kalibrační proces pro magnetometr

Kalibrace magnetometru má za úkol odstranit hard-iron offset, který nejvýrazněji ovlivňuje měření senzoru. Korekce soft-iron ofsetu a jiných menších vlivů nebyla v této práci řešena. Kalibrace je rozložena do dvou procesů. První ***magn_offset_calc*** porovná nová data s maximy a minimy, které jsou pro jednotlivé osy uloženy v registrech. Pokud je nalezeno nové maximum nebo minimum, tak nahradí v registru stávající hodnotu. Pro každou osu je poté nalezen průměr mezi minimem a maximem, který určuje velikost ofsetu. Pro řízení tohoto procesu je na desce přiřazen první přepínač. Proces je spuštěn v každém cyklu jen tehdy, pokud je přepínač v logické '1'. V druhém procesu ***magn_offset_corr*** je nalezený ofset v každém cyklu odečten od nových dat magnetomeru (vektor $\vec{m}$). Výsledkem je upravený vektor $\vec{m}_{corr}$. Tento proces běží v každém cyklu bez závislosti na přepínači.

Kalibrační proces pro gyroskop

Kalibrace je opět rozdělena do dvou procesů. První proces ***gyro_offset_calc*** je spuštěn tlačítkem na desce. To stiskne uživatel v momentu, kdy je deska se senzorem v klidu. Přibližně 1 sekundu probíhá čekání, aby do kalibrace nebyly zahrnuty případně malé rotace způsobnené stisknutím tlačítka. Poté je sbíráno 128 vzorků dat pro každou osu gyroskopu (přibližně 1s), které jsou průběžně sčítány a na konci zprůměrovány. Výsledné hodnoty jsou nastaveny jako klidový ofset gyroskopu. V následujícím procesu ***gyro_offset_corr*** jsou odečteny od naměřených dat gyroskopu. Výsledkem je vektor $\vec{g_{corr}}$.

4.7.2 Návrh výpočetních jednotek

Hlavní částí cyklu stavového automatu tvoří řízení výpočetních operací. Tyto operace byly rozděleny do funkčních celků a pro každý z nich byla navržena samostatná entita. Jak bylo zmíněno v kapitole o SPI rozhraní, tak frekvence vzorkování dat gyroskopu je nastavena na 119Hz. To znamená, že při systémové frekvenci 125MHz je mezi jednotlivými měřeními dostatek času na provedení výpočtů. Proto byly některé výpočty serializovány, čím se sice prodloužila doba zpracování jednoho cyklu, ale byly ušetřeny zdroje na FPGA. Byla tedy dána přednost optimalizaci zdrojů. Návrh je ale vytvořen tak, že by v případě požadavku na rychlejší zpracování, bylo možné návrh celkem snadno upravit na paralelnější architekturu. V další části jsou popsány jednotlivé entity a metody použité pro výpočty.

Aritmetické výpočty na FPGA

Z rozboru výpočtů v teoretické části kapitoly vyplývá, že v rámci entity pro zpracování dat musí být implementovány obvody pro výpočty násobení, sčítání/odčítání, dělení, goniometrických funkcí a odmocňování. K tomuto účelu jsou využity IP (Intelectual Property) komponenty které výrobce Xilinx dává k dispozici ve vývojovém prostředí Vivado. Pro výpočty goniometrických funkcí a odmocňování je použita IP Xilinx CORDIC LogiCore 6.0, která implementuje algoritmus CORDIC. Jedná se o iterativní metodu, která umožňuje efektivně provádět výpočty tohoto typu v digitálních obvodech. Jsou generovány tři instance této IP komponenty:

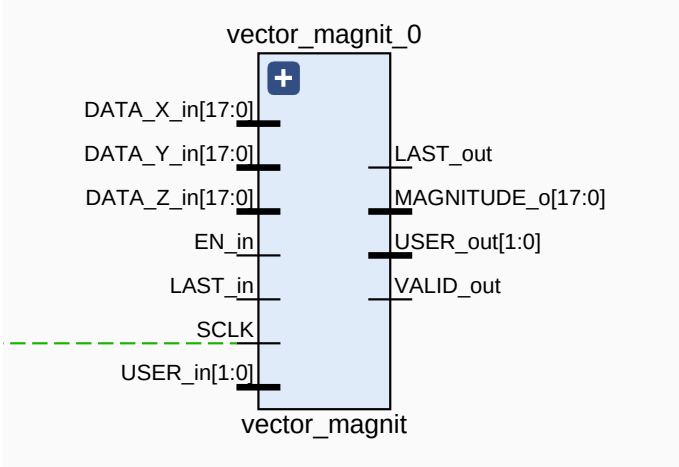
- **ip_codic_sqrt** – pro výpočty odmocniny při normování vektoru nebo při převodu matice souřadnic na úhly, kde je potřeba určit hodnotu $\sqrt{F_x^2 + F_y^2}$. Latence komponenty je 18 hodinových cyklů.
- **ip_codic_sincos** – pro výpočty funkcí sinus a kosinus při převodu z úhlů na matici souřadnic. Latence komponenty je 20 hodinových cyklů.
- **ip_codic_arctan** – pro výpočty funkce atan2 při převodu matice souřadnic na úhly. Latence komponenty je 22 hodinových cyklů.

Dále je využit IP Divider Generator 5.1, kterým je generována komponenta:

- **ip_divider** – pro výpočet dělení při normování vektoru. Latence komponenty je 36 hodinových cyklů.

Výpočty násobení větších čísel jsou Vivadem implementovány s využitím rychlých binárních násobiček, které jsou součástí jednotek DSP48E1 Slice. Tyto násobičky umožňují násobení dvou čísel o bitové šířce až 25 x 18 bitů. Celý návrh je vytvořen tak, že počítá s využitím těchto násobiček, a je s ohledem na ně optimalizován. Šířky signálů jsou nastaveny tak, aby nepřesahovaly maximální bitové šířky vstupů. Některé výpočty jsou serializovány, aby byly implementované DSP48E1 Slice jednotky využity efektivně. Tyto jednotky dokáží provádět i další aritmetické operace, např. sčítání a odečítání čísel o bitové šířce až 48 bitů. Toho je využito při implementaci sčítání a odečítání, které je spojené s násobením (např. u vektorového nebo maticového součinu).

Všechny entity vytvořené pro jednotlivé typy výpočtů nebo převodu obsahují:



Obrázek 4.4: Schéma komponenty *vector_magnit*

- vstupní hodinový signál
- vstupní **enable** signál - ukazatel, že na vstup byla nastavena platná data pro výpočet
- vstupní a výstupní **datové signály**
- výstupní signál **valid** - ukazatel, že na výstupu jsou k dispozici platná data (výsledky výpočtu)
- vstupní a výstupní signál **last** - pro identifikaci posledního datového vstupu anebo výstupu v sérii
- vstupní a výstupní signál **user** - pro identifikaci jednotlivých dat na vstupu anebo výstupu v sérii

Entita vector_magnit

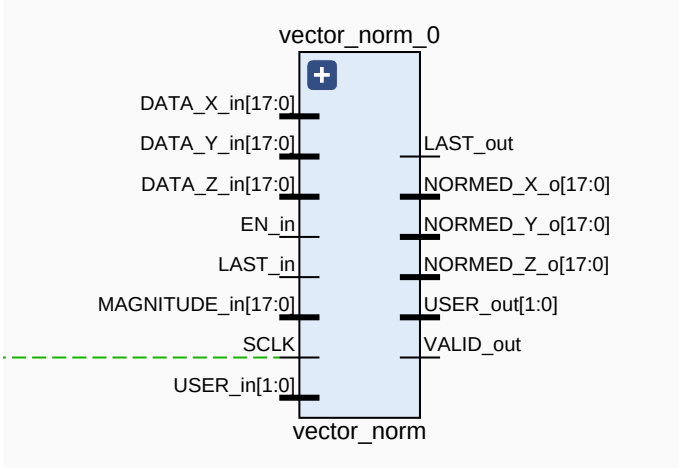
- Datové vstupy: hodnoty souřadnic vektoru $\vec{v} = (x, y, z)$
- Datové výstupy: velikost vektoru $|\vec{v}|$

Entita provede výpočet součtu druhých mocnin souřadnic vektoru a poté jeho odmocnění pomocí komponenty ***ip_cordic_sqrt***. Časové průběhy signálů entity jsou znázorněny na schématu v příloze B-4. Výpočet velikosti vektoru je použit jako předřazený krok pro normování vektorů $\vec{a}$, $\vec{m}_{corr}$, které představují naměřená data z akcelerometru, a naměřená data z magnetometru,

u kterých byla provedena korekce ofsetu. Rovněž je použit před normováním vektoru $\vec{s}$. Dalším využitím je výpočet $\sqrt{F_x^2 + F_y^2}$, který je proveden s nastavením nulové hodnoty pro souřadnici z.

Entita vector_norm

- Datové vstupy: hodnoty souřadnic vektoru $\vec{v} = (x, y, z)$ a velikost vektoru $|\vec{v}|$
- Datové výstupy: normované hodnoty souřadnic vektoru $\vec{v_n} = (x_n, y_n, z_n)$



Obrázek 4.5: Schéma komponenty *vector_norm*

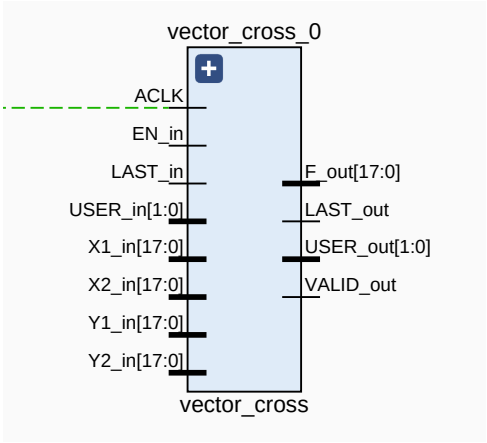
Entita nastaví hodnotu 1 a velikost vektoru $|\vec{v}|$ na vstup děličky. Je provedeno dělení $1/|\vec{v}|$. Jako dělička je použita IP komponenta ***ip_divider***. Na výstupu z děličky je registrován výsledek dělení – koeficient k, 25 bitové číslo Q8.16. Ten je spolu s hodnotami souřadnic vektoru $\vec{v}$ nastaven na vstup entity ***norm_multiplier***, která provádí násobení souřadnic vektoru koeficientem $\vec{v_n} = k\vec{v}$. Výstupem je normovaný vektor, který je nastaven na výstup celé entity *vector_norm*.
Entita je při zpracování dat použita pro normování vektorů $\vec{a}$, $\vec{m}_{corr}$, a také vektoru $\vec{s}$. Časové průběhy signálů entity jsou zobrazeny na schématu v příloze B-5.

Entita vector_cross

Entita je určena pro výpočet vektorového součinu. Počítá jeden člen výsledného vektoru, vyjádřený vztahem:

$$f = x_1 y_2 - x_2 y_1 \tag{4.33}$$

Výpočet tohoto členu je proveden ve 2 iteracích. V první iteraci je proveden druhý součin a ve druhé iteraci je současně proveden první součin a odečtení druhého. Pro výpočet celého vektoru jsou sériově ve 3 hodinových cyklech nastavována vstupní data – souřadnice příslušného vstupního vektoru, a po té jsou ve 3 cyklech registrovány členy výsledného vektoru. Časové průběhy signálů tohoto výpočtu zobrazuje schéma v příloze B-6.



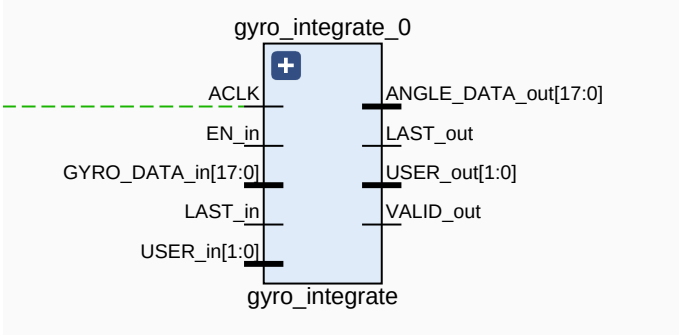
Obrázek 4.6: Schéma komponenty *vector_cross*

Entita gyro_integrate

- Datové vstupy: hodnota úhlové rychlosti naměřená gyrokopem
- Datové výstupy: hodnota přírustku úhlu po integraci, je kódována číslem Q1.14 které je škálováno na rozsah $\langle -\pi; \pi)$.

Základem entity je násobení úhlové rychlosti dobou mezi dvěma vzorkováními, které nahrazuje integraci úhlové rychlosti podle času. Doba je odvozena z vzorkovací frekvence f_s, která je 119Hz. Vztah pro obecný výstupní úhel je:

$$\Delta\alpha_g = \Delta t \omega_z = \frac{1}{f_s}\omega = \frac{1}{119}\omega$$



Obrázek 4.7: Schéma komponenty *gyro_integrate*

Vstupní 16bitová hodnota je v jednotkách dps (degrees per second) a má rozlišení 0,0175 dps na 1 bit. Cílem je aby výstup byl škálovaný na π, a proto je do vztahu přidána převodní konstanta:

$$k_1 = \frac{0,0175 \cdot 2^{16}}{180}$$

$$\Delta\alpha'_g = \frac{1}{119}\,\omega\,k_1 = \frac{1}{119}\,\omega\,\frac{0,0175 \cdot 2^{16}}{180}$$

Všechny konstanty jsou spojeny do jedné převodní konstanty:

$$k = \frac{1}{119}\,\frac{0,0175 \cdot 2^{16}}{180} = 0,053542484$$

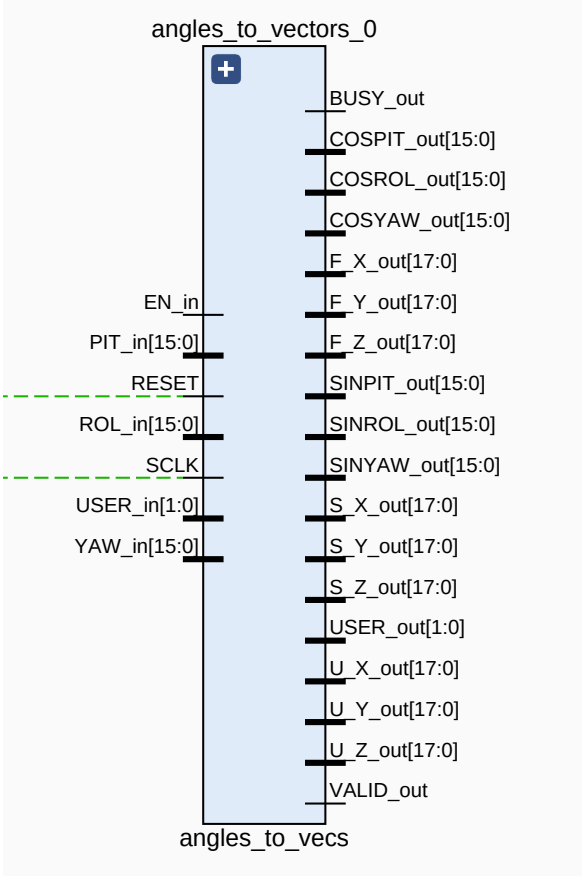
Ta je do návrhu převedena na 24bitovou konstantu ve formátu Q1.22 (hodnota X"06DA7B"). Entita je využita tak, že na vstup jsou seriově ve 3 hodinových cyklech za sebou připojována data gyroskopu - úhlové rychlosti po korekci ofsetu: $\omega_{z,corr}$, $\omega_{y,corr}$, $\omega_{x,corr}$. Na výstupu jsou poté čteny výsledné přírůstky úhlů $\Delta\psi_g$, $\Delta\theta_g$, $\Delta\phi_g$. Časové průběhy signálů entity jsou znázorněny na schématu v příloze B-7.

4.7.3 Entita angles_to_vecs

- Datové vstupy: hodnoty úhlů ψ, θ, ϕ
- Datové výstupy: souřadnice vektorů polohy a hodnoty sinů a kosinů úhlů

Entita provádí výpočet sinů a kosinů. K tomu je použita IP komponenta ***ip_cordic_sincos***. Na tento krok navazuje výpočet souřadnic vektorů polohy. Z převodních vztahů souřadnic 4.4 je patrné, že jsou dva obecné tvary:

$$y_1 = \pm a \cdot b$$
$$y_2 = \pm a \cdot b \pm c \cdot d \cdot e$$

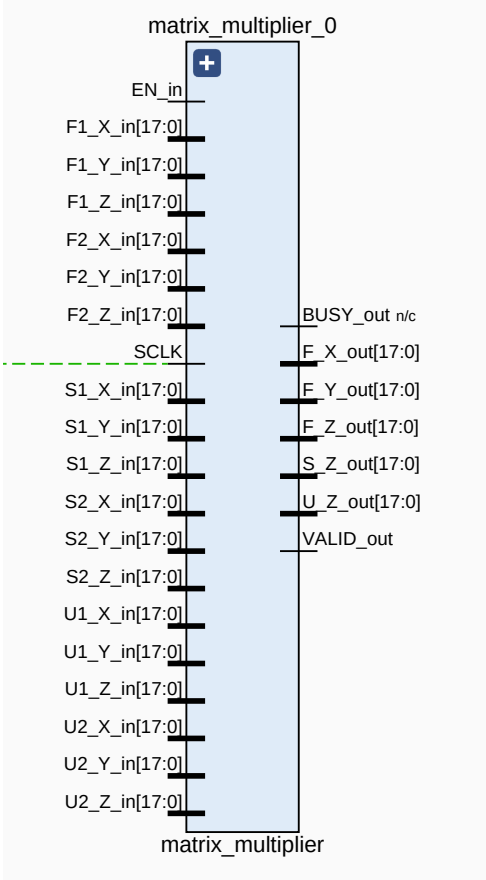


Obrázek 4.8: Schéma komponenty *angles_to_vecs*

K tomuto účelu byly vytvořeny dvě samostané entity – ***angles2vecs_u1*** pro první typ výpočtu a ***angles2vecs_u2*** pro druhý typ výpočtu. Hodnoty sinus a cosinus jsou uloženy v registrecha poté postupně nastavovány na vstupy těchto entit. Výstupní hodnoty jsou registrovány a pak jsou všechny najednou připojeny na výstup. Entita je použita pro převod úhlů ψ_o, θ_o, ϕ_o na matici M_o a poté úhlů $\Delta\psi_g$, $\Delta\theta_g$, $\Delta\phi_g$ na matici T_{dg}. Časové průběhy této a vnořených entit zachycují schémata v přílohách B-8 až B-10. Na schématech je znázorněno jak probíhá převod dvou sad úhlů ihned po sobě. Pro lepší názornost jsou signály patřící k jednotlivým sadám odděleny červenou čarou.

4.7.4 Entita matrix_multiplier

- Datové vstupy: souřadnice matice 1 a matice 2 (celkem 18 datových sběrnic)
- Datové výstupy: 5 souřadnic výsledné matice



Obrázek 4.9: Schéma komponenty *matrix_multiplier*

Entita provádí maticový součin dvou matic. Jelikož souřadnice z výsledné matice jsou dále použity jen pro převod na úhly, a ze vztahů 4.12 je patrné, že je potřeba pouze 5 z celkových 9 souřadnic, tak některé členy výsledné matice nejsou počítány. Úkolem entity je pro každý nový člen (souřadnici) provést výpočet ve tvaru:

$$C_{ij} = A_{1j} \cdot B_{i1} + A_{2j} \cdot B_{i2} + A_{2j} \cdot B_{i2}$$

Návrh je vytvořen tak, že výpočetní jednotka provádí v každém hodinovém cyklu výpočet:

$$Y_n = A \cdot B + Y_{n-1}$$
$$nebo$$
$$Y_n = A \cdot B + 0$$

Přičemž je zaveden signál *add_prod_en*, který indikuje, zda je proveden první nebo druhý výpočet. Tím je zajištěna schopnost sčítat až tři vynásobené členy za sebou, ale zároveň je na celou entitu použita jen jedna násobička. Dále je vytvořen 17bitový posuvný registr, kterým je posouván příznak '1', a podle bitu, ve kterém se právě příznak nachází, jsou nastavovány vstupy pro

výpočetní jednotku a registrovány její výstupy. Výsledné souřadnice jsou pak na výstup nastaveny v posledním kroku všechny najedou. Entita je použita pro výpočet matice $M_g = T_{dg} \times M_o$. Časové schéma entity je v příloze B-11.

4.7.5 Převod souřadnic vektorů na úhly

Převod na úhly se skládá ze dvou kroků. Prvním krokem je výpočet $\sqrt{F_x^2 + F_y^2}$ pomocí entity ***vector_magnit***. Druhým krokem je provedení výpočtu atan2 pomocí IP komponenty ***ip_cordic_arctan***. Převod na úhly je prováděn pro matici $M_{a/m}$ na úhly $\psi_{a/m}$, $\theta_{a/m}$, $\phi_{a/m}$ a pro matici M_g na úhly ψ_g, θ_g, ϕ_g. Při registrování převedeného úhlu $\psi_{a/m}$ (yaw) je od něj odečtena hodnota magnetické deklinace. Tím je zajištěna korekce této odchylky.

4.7.6 Sjednocení orientace yaw úhlu

Proces ***redirect_acm_angles*** předchází komplementární filtr a jeho úkolem je sjednotit směr úhlu yaw získaného akcelerometrem/magnetometrem ($\psi_{a/m}$) s úhlem yaw získaným gyroskopem (ψ_g). Jedním z problémům použití Eulerových úhlů je, že úhel pitch (θ) je definovaný v rozsahu $\langle -\frac{\pi}{2}; \frac{\pi}{2} \rangle$. Pokud je hodnota blízko +90° nebo -90°. Může se pak stát, že dvě polohy, které se od sebe liší jen o malé pootočení mají odlišné úhly yaw a roll. To lze demonstrovat na příkladu, kdy jsou úhly:

$$\begin{aligned} yaw &: \psi = 30^\circ \\ pitch &: \theta = 89^\circ \\ roll &: \phi = 0^\circ \end{aligned}$$

Pokud je provedena další rotace ve směru úhlu yaw o 2° (což by vedlo na úhel o velikosti 91°, tedy mimo definovaný rozsah), tak hodnoty úhlů nové polohy jsou:

$$\begin{aligned} yaw &: \psi = -150^\circ \\ pitch &: \theta = 89^\circ \\ roll &: \phi = -180^\circ \end{aligned}$$

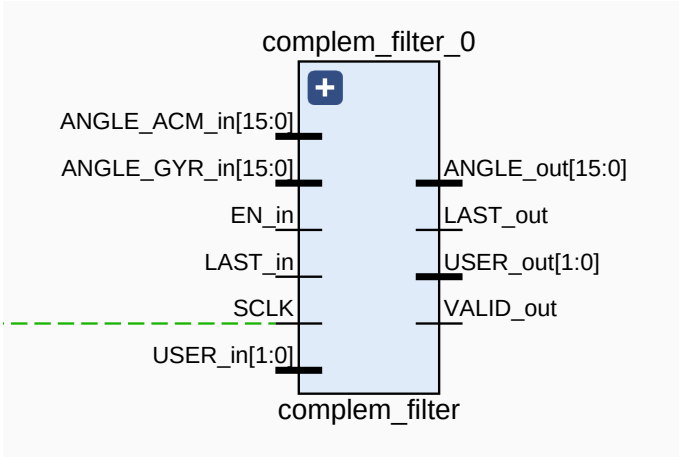
Z toho je zřejmé, že úhly yaw a roll byly otočeny o 180°. Pokud by byly tyto dvě polohy komplementárně sečteny, tak by výsledkem byly neodpovídající úhly. Z tohoto důvodu je povolený rozsah úhlu pitch pro účely komplementárního úhlu rozšířen $\langle -\pi; \pi)$ a je provedena kontrola, jestli rozdíl mezi yaw úhly ψ_g a $\psi_{a/m}$

není větší než 90°. Pokud ano, pak jsou úhly akcelerometru/magnetometru převedeny pomocí vztahů:

$$\psi'_{a/m} = \psi_{a/m} + \pi$$
$$\theta'_{a/m} = \pi - \theta_{a/m}$$
$$\phi'_{a/m} = \phi_{a/m} + \pi$$

4.7.7 Entita complem_filter

- Datové vstupy: úhel získaný z gyroskopu, úhel získaný z data akcelerometru/magnetometru
- Datové výstupy: úhel získaný komplementárním součtem



Obrázek 4.10: Schéma komponenty *complem_filter*

Komlementární filtr provádí poměrový součet. Podíl mezi daty gyroskopu a akcelerometru/magnetometru byl na základě testů nastaven na 63:1. Kromě tohoto byly testovány poměry 127:1, 31:1 a 15:1 – vybírány byly takové, jejichž součet je 2^n kvůli možnosti děleni bitovým posunem. V rámci tohoto filtru je potřeba také provádět jednu korekci, a to v případě, že jsou úhly od sebe vzdálené více jak 180°. Lze to dobře ukázat na příkladu, kdy jeden úhel je -179° a druhý +177°. I když jsou tyto úhly na opačných stranách rozsahu (průměr by nám vyšel -2°), tak ve skutečnosti jsou blízko u sebe (průměr by měl být 179°). Aby bylo možné takové úhly komplementárně sčítat, je nutné v takových případech rozsah rozšířit na $\langle -2\pi; 2\pi)$ a menší z úhlů posunout o 360°:

$$\alpha' = \alpha + 2\pi$$

V uvedeném příkladu by byl posunutím získán úhel 181°, a zde by již průměr odpovídal 179°.

4.7.8 Nastavení výstupních úhlů

Na závěr celého cyklu entity *data_processor* je zařazen proces ***angle_data_output***. Ten nastavuje výstupní hodnoty, ale ještě před tím provádí zpětnou tranformaci, tak aby byl pitch, vrácen do původního rozsahu $\langle -\frac{\pi}{2}; \frac{\pi}{2} \rangle$, pokud tento rozsah přesahuje. To znamená, že provádí stejnou transformaci jako proces ***redirect_acm_angles***, ale s opačným cílem.

Kapitola 5

Kompletace návrhu

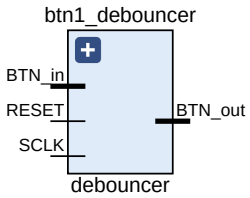
V této kapitole jsou popsány zbývající entity, které byly vytvořeny v rámci celého návrhu a spojení přes top entitu.

5.1 Vstupy pro ovládání uživatelem

V návrhu jsou na hlavní vstupy připojeny některá tlačítka a přepínače, které slouží k ovládání uživatelem:

- resetovací tlačítko
- tlačítko pro spuštění kalibrace gyroskopu
- přepínač aktivující v logické '1' hledání minim a maxim pro data magnetometru, ze kterých je vypočtena korekce ofsetu
- dvě tlačítka, která slouží pro zvyšování a snižování hodnoty pro korekci magnetické deklinace
- přepínač módu zobrazení

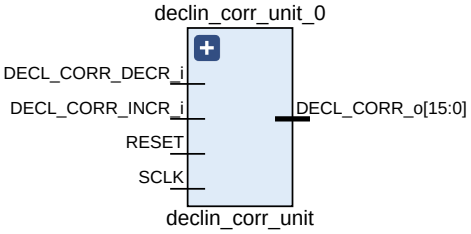
Signál každého tlačítka nebo přepínače je nejprve přiveden na debouncer (entita ***debouncer***) a teprve výstup této komponenty, kde jsou odfiltrovány případné zákmity, je použit pro zapojení v systému.



Obrázek 5.1: Schéma komponenty *debouncer*

5.1.1 Magnetická deklinace

V návrhu byla vytvořena entita ***declin_corr_unit***, která umožňuje nastavit hodnoty úhlu pro korekci magnetické deklinace, způsobené odchylkou magnetického a geografického severního pólu. Jelikož se tato odchylka mění s geografickou polohou, tak není možné bez GPS souřadnic hodnotu nastavit automaticky. Proto bylo vytvořeno rozhraní pro nastavení uživatelem. Nastavování korekčního úhlu je řízeno dvěma tlačítky, jedním se zvyšuje, druhým snižuje. Při delším přidržení stisknutého tlačítka se zvyšování nebo snižování zrychluje.



Obrázek 5.2: Schéma komponenty *declin_corr_unit*

5.2 BCD převodník

BCD (binary coded decimal) způsob kódování čísel, kdy každá číslice je kódovaná zvlášť v podobě 4bitové binárního čísla. Čísla v tomto tvaru jsou používány pro zobrazení na monitoru. V tomto návrhu je cílem rovněž zobrazit některé hodnoty na monitoru, proto byl BCD převodník implementován. Je použit pro zobrazení číselných hodnot tří úhlů - yaw, pitch a roll.

Návrh je rozdělen do dvou entit:

- ***bcd_converter*** – převádí jedno 16bitové číslo na sedm 4bitových číslic
- ***data_bcd_conv*** – postupně nastavuje jednotlivá čísla pro převod a vyčítá převedené BCD čísla

5.2.1 Entita bcd_converter

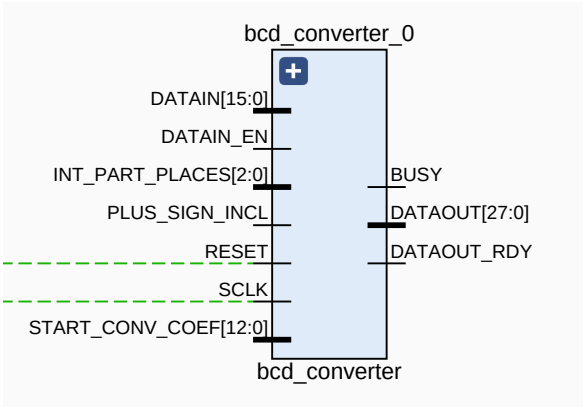
Převodník převádí číslo na jendotlivé číslice 0 až 9. Zároveň kóduje i znaménka plus/mínus, desetinnou tečku a mezeru (prázdný znak). Tabulka kódování výstupního 4bitového čísla:

x"0"	0
x"1"	1
x"2"	2
x"3"	3
x"4"	4
x"5"	5
x"6"	6
x"7"	7
x"8"	8
x"9"	9
x"A"	desetinná tečka
x"B"	plus
x"C"	mínus
x"D"	nepřiřazeno (mezera)
x"E"	nepřiřazeno (mezera)
x"F"	mezera

Převodník je navržen na iterativním principu převodu, kdy v každé iteraci je získána jedna číslice. Vstupního porty:

- sběrnice pro převáděnou hodnotu
- port pro příznak, zda v čísle má být zobrazeno znaménko plus
- datová sběrnice určující kolik číslic má celočíselná část čísla
- 13bitový převodní koeficient

Převodní koeficient slouží k tomu, aby byla v první iteraci správně nastavena první číslice – konstanta musí být zvolena tak, aby převedla vstupní hodnotu



Obrázek 5.3: Schéma komponenty *bcd_converter*

na číslo s jedním celočíselným místem. To zajistí dosazení parametrů čísla do vztahu:

$$k = \frac{2r \cdot (2^9)}{10^{i-1}}$$

r je hraniční kladná hodnota rozsahu 16bitového signed čísla (první číslo, které se již nevejde do rozsahu)
i je počet číslic celočíselné části

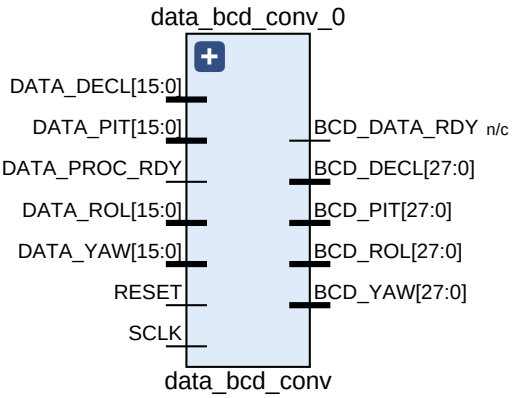
Např. pro převod úhlu kódovaném v tomto 16bitovém čísle je maximální kladná hodnota $r = 180$ a $i = 3$. Převodní koeficient je $k = 1843$, v 13bitovém čísle kódovaný jako x"0733".

V první iteraci je převedeno znaménko, je provedeno násobení koeficientem, a je převedena první číslice. V dalších iteracích je pak číslo vynásobeno 10, tak aby se do celočíselné části dostala další číslice v pořadí, a takto je postupně získáno všech 7 číslic. Násobení 10 je realizované součtem dvou čísel, které vniknou posunutím násobeného čísla o 1bit, resp. o 3 bity doleva.

Proces je ještě doplněn logikou, která zajišťuje, aby nuly na začátku byly nahrazeny mezerou, a aby znaménko bylo umístěno před první nenulovou číslici.

5.2.2 Entita data_bcd_conv

Tato entita je umístěna v úrovni nad *bcd_converter*. Jejími vstupy jsou datové sběrnice, které mají být převedeny na BCD čísla, které jsou poté vysílány sběrnicemi na výstupních portech. Tyto sběrnice využívá generátor obrazu k zobrazování hodnot na obrazovce.



Obrázek 5.4: Schéma komponenty *data_bcd_conv*

5.3 Generování obrazu

Obraz je generován entitou ***image_generator***.

Vstupní datové sběrnice:

- úhly yaw, pitch a roll
- siny a kosiny úhlů yaw, pitch a roll
- souřadnice vektorů F,S,U
- úhly kódované v BCD

Pro zobrazení jsou vytvořené 4 hlavní sekce obrazu:

- Sekce se zobrazením číselných hodnot úhlů yaw, pitch a roll

- Sekce se zobrazením ve stylu technické izometrie, které tvoří barevné úsečky zobrazující osy (x,y,z) a vektory polohy
- Sekce se zobrazením ve stylu leteckého umělého horizontu – ukazatel bočního a předo-zadního náklonu (úhly pitch a roll)
- Sekce se zobrazením kompasu (úhel yaw)

První sekce je zobrazena vždy. Zbylé jsou řízeny přepínačem, který pokud je v logické '1', tak je zobrazena druhá sekce s osami a vektory. Pokud je v logické '0', tak je zobrazen umělý horizont spolu s kompasem.

Jelikož je mezi VGA rozhraní a generátor obrazu zařazena paměť (řádkový buffer), tak s VGA rozhraní je přivedena pouze sběrnice, na které je kódováno číslo právě zobrazovaného řádku, a synchronizační signál indikující začátek řádku. Číslo sloupce si generátor obrazu určuje sám počitadlem *col_int*, které se spustí po přijetí synchronizačního signálu. Barvy pro zobrazení jsou přes výstupní sběrnice ukládány do paměti.

5.3.1 Zobrazení číselných hodnot

Pro toto zobrazení jsou využity úhly v BCD. V entitě definovány konstanty s mapami pro zobrazení číslic a použitých znaků. Velikost číslice je 8x10 pixelů. Pokud je detekována oblast pro zobrazení čísel, tak je načtena hodnota a po té konkrétní číslice, která má být zobrazena. Na základě mapy a souřadnic aktivního pixelu je určeno, zda se nachází na číslici nebo na pozadí. Podobným principem jsou zobrazeny textové popisky před čísly (celkem čtyři), ke kterým jsou vytvořeny mapy o velikosti 42x10 pixelů. V základu jsou zobrazeny tři základní úhly. Pokud dojde ke stisku tlačítek, kterými je nastavována korekce magnetické deklinace, tak je pod základními hodnotami zobrazena ještě hodnota úhlu této korekce. Stiskem je současně nastaveno počitadlo, který se postupně odečítá, a jakmile dosáhne nuly, tak je zobrazení zrušeno a hodnota z obrazovky zmizí.

5.3.2 Zobrazení vektorů v technické izometrii

Technické izometrie je způsob zobrazení 3D prostoru ve 2D. Osy tohoto zobrazení svírají úhel 120°. Vektor ve 3D (reálný), který má souřadnice x,y,z,

je převeden do souřadnic x', y' na základě vztahů (x' je orientováno vodorovně směrem doprava, y' svisle směrem nahoru):

$$x' = (y - x)sin(60°)$$
$$y' = (2z - y - x)cos(60°) = \frac{2z - y - x}{2}$$

Pro toto zobrazení jsou proto využity vstupní porty se souřadnicemi vektorů. Po přepočtu na 2D souřadnice jsou tyto souřadnice porovnány se souřadnicemi aktivního pixelu. Ještě předtím jsou souřadnice aktivního pixelu přepočeny na souřadnice (x_p, y_p) tak, aby začínaly ve stejném nulovém bodě jako 2D souřadnice vektorů (x', y'). Poté jsou porovnány podle vztahů:

$$\frac{y_p}{x_p} = \frac{y'}{x'}$$
$$0 \leq x_p \leq |x'|$$
$$0 \leq y_p \leq |y'|$$

První vztah je upraven tak, aby nebylo potřeba dělení, ale pouze násobení:

$$y_p \cdot x' = y' \cdot x_p \tag{5.1}$$

Na základě těchto vztahů je implementováno ověření, že aktivní pixel leží na vektoru (má stejnou směrnici a neleží dál než dostahuje délka vektoru).

5.3.3 Zobrazení leteckého umělého horizontu

Toto zobrazení využívá hodnoty úhlů a jejich sinů a kosinů. Porovnává souřadnice pixelu se souřadnicemi vektoru podobně jako u předešlého zobrazení, rozdíl je v tom, tento vektor získá přepočtem právě ze sinů a kosinů. Principem přepočtu je vektorová rotace o určitý úhel. Zobrazení má statickou část, kterou tvoří horní stupnice náklonu s číselníkem a středový žlutý ukazatel, a pak pohyblivou část, která se naklání podle úhlu roll (ϕ) a zároveň posouvá podle úhlu pitch (θ). Pro zobrazení nakloněných struktur je použita rotace souřadnic pixelu opačným směrem, než je směr úhlu (jakoby zpět do základí polohy), poté je aplikován posun přičtením přepočtené hodnoty úhlu pitch (θ_{scaled}) od y-souřadnice, a až po té jsou nové souřadnice porovnávány a určeny zobrazení čar a další grafiky. Použité převodní vztahy pro rotaci souřadnic jsou:

$$x_T = x_p \cos\phi + y_p \sin\phi$$
$$y_T = -x_p \sin\phi + y_p \cos\phi + \theta_{scaled}$$

5.3.4 Zobrazení kompasu

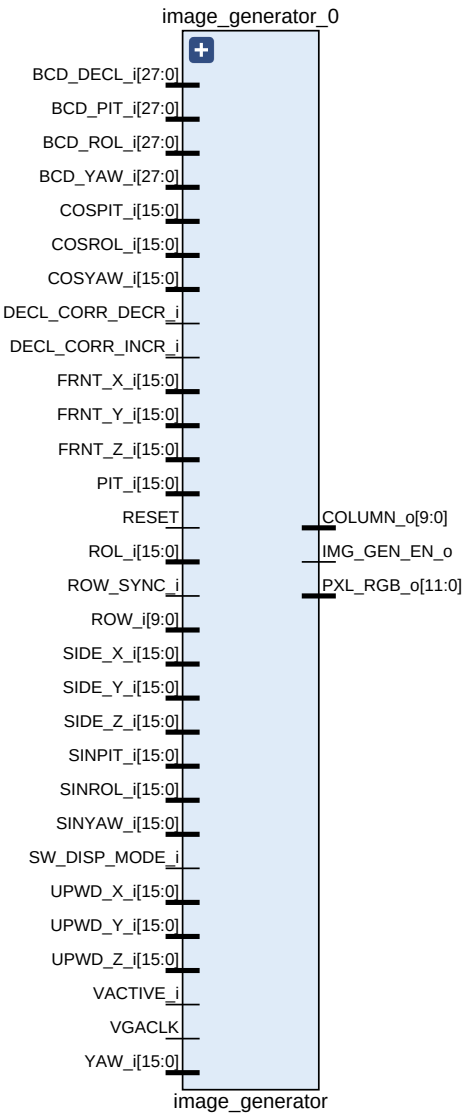
Princip zobrazení je stejný jako u umělého horizontu. Je tady použit jiný úhel pro rotaci – yaw (ψ). Také je zde statická část, tvořená kruhem se značkami a písmeny označujícími světové strany, a pohyblivá část tvořená šipkou. Souřadnice pixelu jsou přepočteny na nulový bod ve středu kruhu. Pokud se nachází uprostřed kruhy, tak jsou rotovány zpětně o úhel yaw, a poté je určováno zobrazení šipky. Použité převodní vztahy pro rotaci souřadnic jsou:

$$x_T = x_p \cos\psi + y_p \sin\psi$$
$$y_T = -x_p \sin\psi + y_p \cos\psi$$

5.4 Návrh top entity

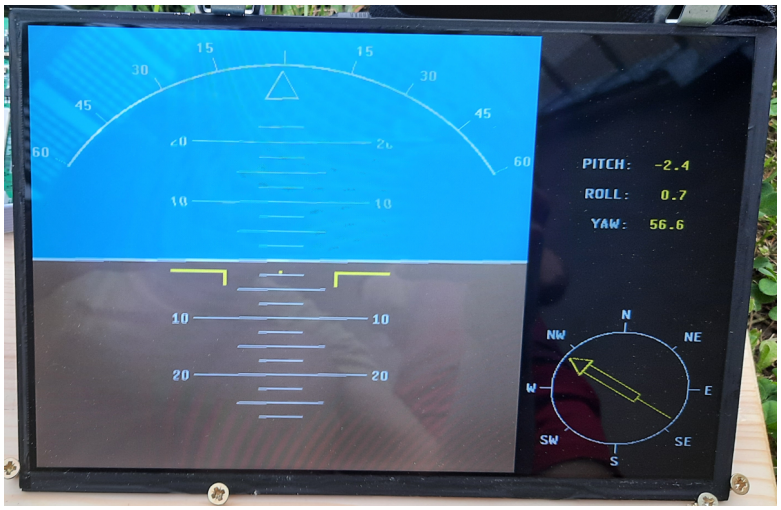
Všechny zmíněné entity jsou v podobě komponent vloženy do hlavní entity ***top*** vzájemně propojeny. Schéma zapojení je v příloze B-12

Dále je v xdc souboru ***Zybo-Z7-Master.xdc*** provedeno přiřazení portů entity k pinům na desce Zybo.



Obrázek 5.5: Schéma komponenty *image_generator*

Obrázek 5.6: Zobrazení vektorů v technické izometrii a číselných hodnot



Obrázek 5.7: Zobrazení umělého horizontu a kompasu

Kapitola 6

Testování návrhu a závěr

6.1 Metody simulace a testování v průběhu vytváření návrhu

V průběhu návrhu byly testovány nejprve menší dílčí a základní celky, např. komunikační SPI rozhraní nebo zobrazení VGA výstupu na monitoru. Po té se přešlo k testování získaných dat.

Jelikož je použito velké množství různých aritmetických operací, které s sebou nesou riziko hazardů. Prvotní otestování bylo provedeno pomocí simulací (behaviorální a post-implementační). Dále bylo potřeba ověřit výsledky získané na reálných datech přímo ze senzorů a vypočtené na hradlovém poli. Pro tyto účely byly využity dvě metody:

- IP komponenta ILA (Integrated Logic Analyzer v6.2 LogiCORE).
- Nástroj FPGA Data Capture v Matlabu.

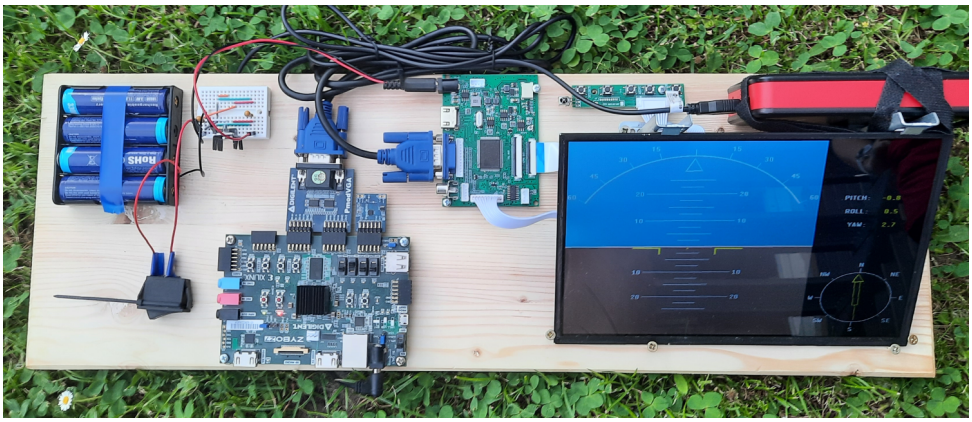
První metoda ILA umožnuje vytvořit sondy pro konkrétní signály, které jsou v určitém časovém úseku zaznamenány, poté přeneseny USB kabelem a následně zobrazeny v prostředí Vivada. Využití je stejné jako v případě externího logického analyzátoru, tzn. je vhodná především pro časovou analýzu a pro odhalování případných hazardů

Pomocí druhé metody je funkcí v Matlabu vygenerována komponenta (VHDL kód a další zdrojové soubory). Tu pak je potřeba přidat přímo do testovaného návrhu a na její vstupy ve VHDL připojit analyzované signály. Po implementaci a spuštění takto upraveného návrhu, můžeme opět pomocí USB kabelu přenášet zaznamenaná data. Výhodou této metody je zejména možnost zaznamenání velkého množství hodnot, které jsou uloženy do matic, a mohout tak být snadno použity pro kontrolní výpočty. Např. pro ověření výsledků aritmetických operací, které byly v FPGA provedeny. Zároveň je možné využít grafických nástrojů Matlabu pro vizualizaci dat. Postupným testováním a úpravami v návrhu bylo dosaženo správného chování všech výpočetních operací. Zároveň byly v průběhu testování a ladění procházeny informace o chybách generované Vivadem během syntézy a implementace. Souvisejicí části návrhu byly na základě těchto výpisů upraveny.

6.2 Závěrečné testování

Testování samostatných rozhraní neodhalilo žádné chyby. Z naměřených průběhů SPI komunikace je patrné, že rozhraní generuje signály se správnými polaritami a správným časováním. Rovněž zobrazený obraz na monitoru je správně napolohován, neobsahuje blikající elementy nebo artefakty a barvy jednotlivých částí se shodují s nastavením v generátoru obrazu.

Celý projekt byl otestován v různých podmínkách. Testovaná sestava je zobrazena na obrázku 6.1.



Obrázek 6.1: Sestava k testování návrhu

Největší vliv na zkreslení zobrazovaných výsldků mají zdroje magnetického pole. To bylo množné pozorovat při testování uvnitř, kdy některá domácí elektronika výrazně měnila směr úhlu yaw, který určen především magneto-

metrem. Naopak při testování venku systém ukazoval polohu velice přesně. Zobrazení reaguje rychle a plynule na skutečnou změnu polohy. Zobrazení číselných hodnot je přesné, ale při změnách polohy čísla rychle přeskakují, což je trochu rušivé a může být složitější hodnoty odečítat. Fotografie z testování jsou připojeny v příloze B-13 až B-15

6.3 Závěr

V bakalářské práci byl navržen navigační systém s využitím desky Zybo Z7-20 s FPGA Xilinx Zynq 7000. Celý návrh byl vytvořen v jazyce FPGA. Byla implementována rozhraní nutná pro komunikaci s moduly Pmod NAV a Pmod VGA. Rovněž byla implementována jednotka pro komplexní zpracování získaných dat, včetně kalibrací, výpočtů a převodů. V této jednotce byla kombinována data ze tří senzorů - akcelerometru, magnetometru a gyroskopu. Pro zobrazení byl implementován generátor obrazu, který na monitoru vykresluje grafické vizualizace zpracovaných dat, a také vypisuje hodnoty polohových úhlů. Zobrazení je plynulé, a zdá se, že negativní vlivy, které na senzory působí, se podařilo odstranit s velmi dobrým výsledkem. Přesto by se daly některé věci vylepšit. Některé entity by mohly být optimalizovány pro snížení množství využitých logických elementů, např. komponenta pro generování obrazu. Také by se dalo vylepšit číselné zobrazení předřazením filtru, aby zobrazované hodnoty nepřeskakovaly tak rychle a plynuleji na sebe navazovaly.

Poznatkem z práce je, že implementace výpočtů na hradlovém poli je poměrně náročná a pro sekvenční zpracování dat může být lepší výpočty provádět na procesoru. Využití by ale návrh nebo některé jeho části mohl mít při zpracování většího množství dat najednou, protože většina navržených procesů může být využita i pro paralelnější architektury, aniž by vzorstlo množství využitých logických elementů na čipu. Naopak největší výhodou použití FPGA pro takovou aplikaci je naprostá kontrola nad průběhem jednotlivých procesů a dat. Z výsledků jsem přesvědčen, že požadavky zadání práce byly všechny splněny.

Příloha A

Literatura

[1] Zybo Z7 Reference Manual
https://digilent.com/reference/programmable-logic/zybo-z7/reference-manual

[2] LSM9DS1 - iNEMO inertial module: 3D accelerometer, 3D gyroscope, 3D magnetometer - Datasheet - production data http://www.st.com/web/en/resource/technical/document/datasheet/DM00103319.pdf

[3] 7 Series DSP48E1 Slice User Guide, UG479 (v1.10) March 27, 2018
https://docs.xilinx.com/v/u/en-US/ug479_7Series_DSP48E1

[4] CORDIC v6.0 LogiCORE IP Product Guide, PG105 August 6, 2021
https://docs.xilinx.com/v/u/en-US/pg105-cordic

[5] ASHENDEN Peter J., The Designer's Guide to VHDL, Third Edition, Morgan Kaufmann, 2008. ISBN: 978-0120887859

[6] HWANG Enoch O.: Digital Logic and Microprocessor Design with VHDL, Thomson 2006, ISBN: 0-534-46593-5

[7] PEDRONI Volnei A.: Digital Electronics and Design with VHDL, Morgan Kaufmann, 2008, ISBN: 0123742706

[8] https://en.wikipedia.org/wiki/Coordinate_system

[9] Euler angles https://en.wikipedia.org/wiki/Euler_angles

[10] Inertial measurement unit https://en.wikipedia.org/wiki/Inertial_measurement_unit

[11] Pmod NAV Reference Manual https://digilent.com/reference/pmod/pmodnav/reference-manual

[12] Pmod VGA Reference Manual https://digilent.com/reference/pmod/pmodvga/reference-manual

[13] Acer V173 User Manual https://www.manualslib.com/download/1078/Acer-V173.html

[14] VGA Signal Timing http://tinyvga.com/vga-timing

[15] Divider Generator v5.1 LogiCORE IP Product Guide, PG151 February 4, 2021 https://docs.xilinx.com/v/u/en-US/pg151-div-gen

[16] Clocking Wizard v6.0 LogiCORE IP Product Guide https://docs.xilinx.com/r/en-US/pg065-clk-wiz

[17] Vivado Design Suite User Guide - Synthesis, UG901 (v2021.2) November 16, 2021 https://www.xilinx.com/content/dam/xilinx/support/documents/sw_manuals/xilinx2022_1/ug901-vivado-synthesis.pdf

[18] Vivado Design Suite User Guide - Logic Simulation, UG900 (v2018.3) December 14, 2018 https://docs.xilinx.com/v/u/2018.3-English/ug900-vivado-logic-simulation

[19] Vivado Design Suite User Guide - Implementation, UG904 (v2019.1) June 25, 2019 https://docs.xilinx.com/v/u/2019.1-English/ug904-vivado-implementation

[20] Zynq-7000 SoC Technical Reference Manual(UG585) https://docs.xilinx.com/v/u/en-US/ug585-Zynq-7000-TRM

Příloha B

Seznam přiložených souborů

B.1 Seznam přiložených zdrojových kódů

angles_to_vecs.vhd
angles2vecs_u1.vhd
angles2vecs_u2.vhd
bcd_converter.vhd
complem_filter.vhd
data_bcd_conv.vhd
data_processor.vhd
debouncer.vhd
declin_corr_unit.vhd
gyro_integrate.vhd
image_generator.vhd
img_row_bram.xci
img_row_mem_wrp.vhd
ip_cordic_arctan.xci
ip_cordic_sincos.xci
ip_cordic_sqrt.xci
ip_divider.xci
ip_vga_clk_gen.xci
matrix_multiplier.vhd
norm_multiplier.vhd
spi_controller.vhd
top.vhd
vector_cross.vhd

vector_magnit.vhd
vector_norm.vhd
vga_controller.vhd
Zybo-Z7-Master.xdc

B.2 Seznam přiložených obrázků

B-1 - Logicky Analyzator - Zaznam SPI komunikace - Zapis.png
B-2 - Logicky Analyzator - Zaznam SPI komunikace - Cteni.png
B-3 - Logicky Analyzator - Zaznam SPI komunikace - Cteni - detail.png
B-4 Časový průběh signálů entity vector_magnit.pdf
B-5 Časový průběh signálů entity vector_norm.pdf
B-6 Časový průběh signálů entity vector_cross.pdf
B-7 Časový průběh signálů entity gyro_integrate.pdf
B-8 Časový průběh signálů entity angles_to_vecs.pdf
B-9 Časový průběh signálů entity angles2vecs_u1.pdf
B-10 Časový průběh signálů entity angles2vecs_u2.pdf
B-11 Časový průběh signálů entity matrix_multiplier.pdf
B-12 Schéma zapojení top entity.pdf
B-13 Testovani Foto1.jpg
B-14 Testovani Foto2.jpg
B-15 Testovani Foto3.jpg