Fakulta Elektrotechnická
Katedra mikroelektroniky

Bakalářská práce

Digitální vzorkovací osciloskop na bázi SoC – řídicí a zobrazovací část

Josef Čada

Studijní obor:
Elektronika a komunikace (BP77)

Vedoucí práce:
prof. Ing. Pavel Hazdra, CSc.

Praha, 2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Čada** Jméno: **Josef** Osobní číslo: **503220**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra mikroelektroniky**

Studijní program: **Elektronika a komunikace**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Digitální vzorkovací osciloskop na bázi SoC – řídicí a zobrazovací část

Název bakalářské práce anglicky:

SoC Based Digital Sampling Oscilloscope – Control and Display Part

Pokyny pro vypracování:

Seznamte se s problematikou návrhu systémů na čipu řady ZYNQ-7000 a platformou ZYBO. Na bázi této platformy navrhněte a realizujte blok DSP funkcí, řídicí a zobrazovací část dvoukanálového digitálního osciloskopu se šířkou pásma 500 kHz. Uvažte možnost doplnění systému o modul signálového generátoru s digitálním nastavením frekvence a amplitudy. Ověřte činnost klíčových funkcí.

Seznam doporučené literatury:

[1] L. H. Crockett , R. A. Elliot, M. A. Enderwitz, The ZYNQ Book, Xilinx 2015
[2] J. Ledin, Architecting High-Performance Embedded Systems, Packt Publishing, 2021, ISBN 978-1-78995-596-5
[3] Oscilloscope & Signal generator. In: redpitaya [online]. Dostupné z: https://redpitaya.com/applications-measurement-tool/oscilloscope-signal-generator/

Jméno a pracoviště vedoucí(ho) bakalářské práce:

prof. Ing. Pavel Hazdra, CSc. katedra mikroelektroniky FEL

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

Datum zadání bakalářské práce: **17.02.2023** Termín odevzdání bakalářské práce: **26.05.2023**

Platnost zadání bakalářské práce: **22.09.2024**

prof. Ing. Pavel Hazdra, CSc. podpis vedoucí(ho) práce	prof. Ing. Pavel Hazdra, CSc. 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.

. Datum převzetí zadání

Podpis studenta

ABSTRAKT

Tato bakalářská práce se zaměřuje na návrh a realizaci řídicí a zobrazovací části digitálního osciloskopu a přidruženého signálového generátoru na desce Zybo Z7-20 s využitím čipu Zynq 7000. Práce byla realizována ve spolupráci s Michalem Navrátilem, který se ve své praci naměřuje na signálovou část.
První částí je implementace řídícího kódu pro komunikaci mezi FPGA a procesorem ARM Cortex a řízení komunikace mezi vývojovou deskou a osobním počítačem, přes rozhraní UART. Druhou částí je návrh modulu pro digitální zpracování signálu a signálového generátoru pro FPGA. Poslední částí je realizace aplikace pro zobrazování signálu v prostředí LabVIEW.

Výsledkem je tedy aplikace schopná zobrazovat navzorkovaná data z desky Zybo Z7, konfigurační soubor pro desku ZyboZ7 realizující digitální zpracování signálu a signálový generátor, a v neposlední řadě kód v jazyce C konfigurující komunikaci mezi FPGA a ARM Cortex procesorem a řízení UART komunikace. Po spojení obou prací vzniká dvoukanálový osciloskop o vzorkovací frekvenci 500 MSPS na kanál, šířkou pásma 500 kHz a generátorem signálu se sinovým, trojúhelníkovým, pilovým a PWM výstupem.

KLÍČOVÁ SLOVA

FPGA, SoC, DSO, Osciloskop, Digitální osciloskop, LabView, ZyboZ7, ZYNQ 7000

ABSTRACT

This bachelor thesis focuses on the design and implementation of control and display of a digital oscilloscope and associated signal generator on a board Zybo Z7-20 using Zynq 7000 chip. The work was carried out in collaboration with Michal Navrátil, who focused his work on the signal part.

The first part is the implementation of control code for communication between FPGA and ARM Cortex processor and control of UART communication. The second part is the design of the module for digital signal processing and signal generator for the FPGA. The last part is the implementation of the signal display application in the LabVIEW environment.

Thus, the result is an application capable of displaying the sampled data from the board Zybo Z7, a configuration file for the ZyboZ7 board implementing digital signal processing and a signal generator, finally, a C language code configuring the communication between the FPGA and the ARM Cortex processor and controlling the UART communication. Combining the two works results in a two-channel oscilloscope with a sampling rate of 500 MSPS per channel, a bandwidth of 500 kHz and a signal generator with sine, triangle, sawtooth and PWM outputs.

KEYWORDS

FPGA, SoC, DSO, Oscilloscope, Digital oscilloscope, LabView, ZyboZ7, ZYNQ 7000

PODĚKOVÁNÍ

Rád bych poděkoval profesoru Ing. Pavlu Hazdrovi, CSc. za vedení práce a za podněty k návrhu, realizaci a psaní práce. Dále bych rád poděkoval Michalu Navrátilovi za spolupráci a výpomoc. Také bych rád poděkoval Ing. Pavlu Mlejnkovi, Ph.D. za cenné připomínky k návrhu LabVIEW programu, a docentu Ing. Stanislavu Vítkovi, Ph.D. za zapůjčení vybavení pro testování práce.
Své díky bych rád vyjádřil i přítelkyni a rodině za podporu.

PROHLÁŠENÍ

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 26. května 2023

Josef Čada

Obsah

Seznam použitých obrázků

Seznam tabulek

Seznam použitých zkratek

ADC Analog to Digital Converter, Analogově digitální převodník.
APSoC All Programmable System-on-a-Chip, Plně programovatelný systém na čipu.
ASIC Application-Specific Integrated Circuit.
AXI Advanced eXtensible Interface.
AXI GPIO Advanced eXtensible Interface General Purpose Input/Output.

CLB Configurable Logic Block, Konfigurovatelný logický blok.
CLK Clock, Hodinový signál, Frekvence.

DAC Digital to analog converter, Digitálně analogový převodník.
DPO Digital Phosphor Oscilloscope, Digitální fosforový osciloskop.
DSO Digital storage oscilloscope, Digitální osciloskop.

FPGA Field Programmable Gate Array, Programovatelné hradlové pole.

GPIO General Purpose Input/Output, Všeobecné vstupně/výstupní rozhraní.
GSPS Giga Samples Per Second, Miliarda vzorků za sekundu.

LUT Look-up table, Vyhledávací tabulka.

MDO Mixed Domain Oscilloscope, Osciloskop se smíšenou doménou.
MSO Mixed Signal Oscilloscope, Osciloskop pro smíšené signály.

PL Programmable logic, Programovatelná logika.
PS Processing system.
PWM Pulse Width Modulation, Pulzně šířková modulace.

RAM Random Access Memory, Paměť s přímým přístupem.
RST Reset.

SDK Software Development Kit, Sada pro vývoj softwaru.
SoC System on a Chip, Systém na čipu.

UI User interface, Uživatelské prostředí.

VI Virtual Instrument, Virtuální přístroj.
VLSI Very large-scale integration, Velmi rozsáhlá integrace.

1 Úvod

Osciloskopy jsou základním vybavením každé laboratoře, jedná se o nenahraditelný přístroj při analýze a diagnostice systémů. Jsou to přístroje schopné zobrazovat napětí signálu v závislosti na čase nebo v závislosti na napětí jiného signálu. První osciloskopy byly analogové, ty zobrazovaly signál v reálném čase a byly limitovány přesností v závislosti na maximální frekvenci.

Digitální osciloskopy čtený signál zpracují do digitální podoby a uloží si jej do paměti. Ukládání dat také umožňuje další zpracování signálu a jeho zpětné zobrazení. V dnešní době se používají při tvorbě digitálních osciloskopů programovatelná hradlová pole, která jim umožňují vysokou flexibilitu a rekonfiguraci navržené digitální části. Právě na využití programovatelných polí, pro realizaci osciloskopu, je tato práce soustředěna.

Celkem se bakalářská práce zabývá vytvořením jednoduchého digitálního osciloskopu s použitím systému na čipu s programovatelným hradlovým polem a následným zobrazením pomocí stolního počítače. Vzhledem k náročnosti práce je využito již existující prototypovací desky systému na čipu ZYNQ 7000 a návrh osciloskopu byl rozdělen do dvou bakalářských prací. Autor tohoto textu se zaměřil na zpracování a vizualizaci signálů na uživatelském rozhraní prostřednictvím aplikace LabVIEW a na návrh a implementaci přidruženého generátoru signálů.

Michal Navrátil [1] se na druhou stranu soustředí na návrh analogové části systému, řízení ADC a vytvoření desky plošných spojů s programovatelným zesilovačem.

Účelem obou prací je naučit se implementovat digitální osciloskop na prototypovací desce hradlového pole, využít co nejméně komponent mimo vybranou desku a výsledkem této části práce je systém měření a zpracování signálů do digitální podoby.

2 Teoretický úvod

2.1 Osciloskop

Dle [2] je osciloskop elektronickým zařízením, které slouží k vizualizaci elektrických signálů ve formě grafu a je pravděpodobně nejpoužívanějším laboratorním měřicím přístrojem. Technologie osciloskopů je přímo svázána s vývojem katodové trubice (CRT). Ta byla poprvé vynalezena v roce 1897, Karlem Ferdinandem Braunem. Na toto navázal v roce 1899 Jonathan Zenneck tím, že přidal paprsky tvarující desky a lineární horizontální magnetické vychylovací pole a vytvořil tak první oscilogram. Od této chvíle se začalo s implementací CRT do laboratorních osciloskopů. První zařízení měli různé vývojové problémy, hlavně díky vakuu a problémům s horkou katodou. V roce 1931, Dr. Vladimir K. Zworykin publikoval detaily o první trvale utěsněné CRT s horkou katodou ve vysokém vakuu, která bylo vhodné pro použití v laboratorních přístrojích. Díky tomuto průlomu bylo možné CRT považovat za součástku a ne za proces. A konstruktéři přístrojů ve společnosti General Radio představili první moderní osciloskop.

Mezi lety 1990 a 2000, se začali designéři zaměřovat na LCD displeje, a to z důvodu velkých fyzických rozměrů CRT, křehkosti a obtížnému výrobnímu procesu.

Od roku 2000 šla technologie v oblasti osciloskopů rychle ku předu. V dnešní době je již možno pořídit osciloskop s šířkou pásma 110 GHz a vzorkovací frekvencí 256 GSPS.

2.1.1 Digitální osciloskopy (DSO)

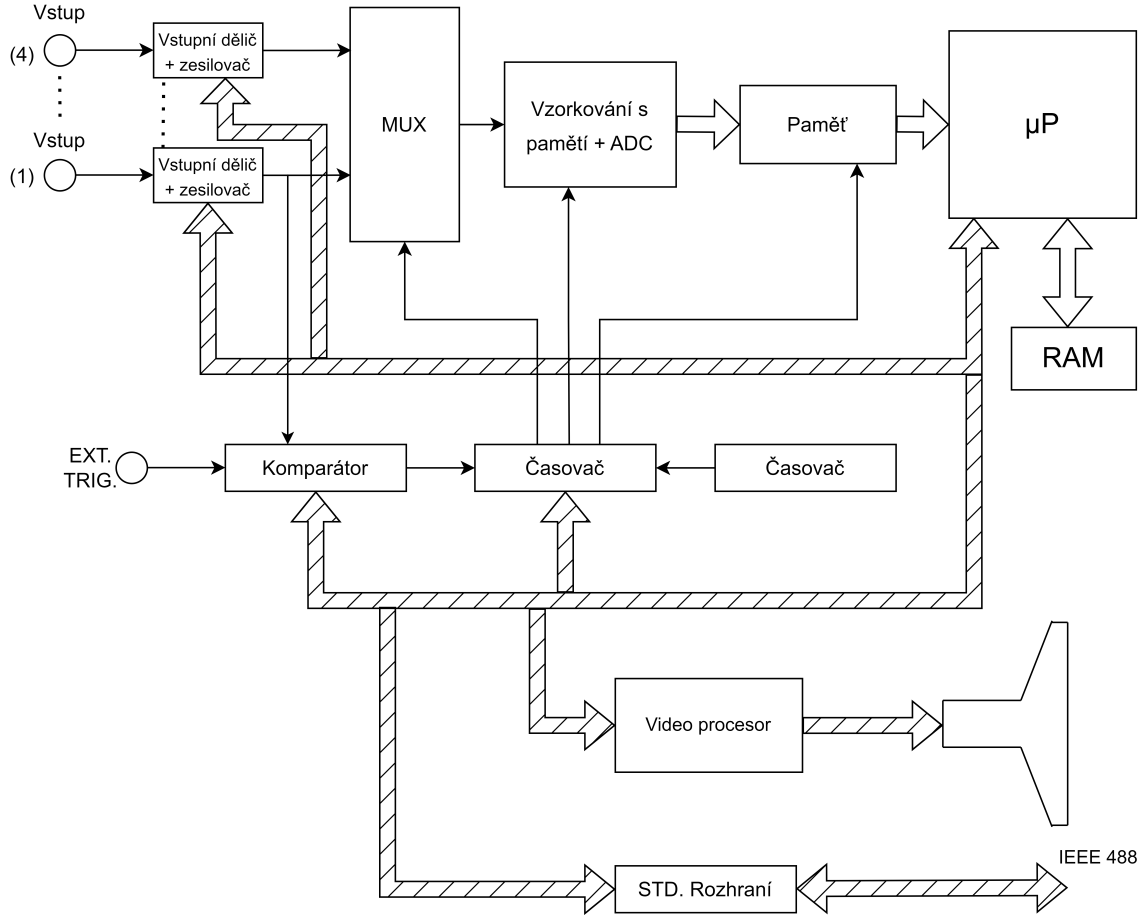
Jak je uvedeno v [3], digitální osciloskopy, neboli DSO, se liší od analogových osciloskopů tím, že zaznamenávají a zobrazují elektrické signály v digitální podobě. Toto umožňuje lepší možnosti pro jejich zpracování a analýzu. Dále umožňují snadnější zpracování a analýzu měřených dat, například pomocí počítače.

Literatura [4], uvádí následující výhody DSO:

- záznam jednorázových přechodných dějů,
- nastavitelná velikost „záporného zpoždění"(možnost sledování signálu před příchodem spouštěcího signálu, anglicky *pretrigger mode*),
- lepší přesnost než u analogových přístrojů,
- rychlé měření buď s využitím kurzorů nebo s automatickým výpočtem a zobrazením řady parametrů měřeného průběhu,
- možnost exportu dat pro následující zpracování,

- programovatelnostt měření a samočinné nastavení přístroje pro měření,

Na počátku vývoje byly digitální osciloskopy „vylepšené analogové"s digitální pamětí a analogovým zobrazováním, viz. [5]. Kvůli limitaci pamětí a vzorkovací frekvence, byly zpočátku digitalní řešení vnímána jako náhrada perových záznamníků průběhu. Dnes již téměř vytlačili analogové osciloskopy. Jelikož je téměř ve všech parametrech překonávají.



Obr. 2.1: Blokové schéma digitálního osciloskopu s multiplexovanými vstupy, převzato a upraveno z [4].

2.1.2 Dnešní trendy u osciloskopů

V současných trendech digitálních osciloskopů lze pozorovat několik klíčových oblastí, které se neustále vyvíjejí a zlepšují. Jedním z těchto trendů je zvýšení šířky pásma a vzorkovací frekvence. Tento trend je reakcí na rychlý rozvoj telekomunikací a radiotechniky, kde se často vyskytují frekvence v řádu GHz.

Dalším trendem je rozšíření konektivity a integrace s jinými zařízeními. Novější osciloskopy často zahrnují možnosti bezdrátového připojení, které umožňují snadné sdílení dat a ovládání přístroje na dálku. Kromě toho jsou některé osciloskopy navrženy

tak, aby se mohly snadno integrovat s jinými měřicími zařízeními, jako jsou spektrální analyzátory nebo generátory signálů.

V oblasti uživatelských rozhraní a softwaru se také objevují inovace. Moderní digitální osciloskopy nabízejí intuitivní dotyková uživatelská rozhraní, která zjednodušují ovládání a zvyšují efektivitu práce s přístrojem. Běžné je exportování dat na flash disk nebo vzdálené připojení přes WiFi a následné stažení dat.

2.1.3 Digitální fosforové osciloskopy

Digitální fosforové osciloskopy (DPO) představují další vývojový krok v oblasti digitálních osciloskopů, který zahrnuje technologii fosforových obrazovek používaných v analogových osciloskopech. DPO kombinují výhody rychlého zpracování digitálních signálů s vlastnostmi intenzitního zobrazení, které poskytují analogové fosforové osciloskopy.

Ale jak udává literatura [6] termín digitální fosforový osciloskop, DPO, se dnes již tolik nepoužívá. Byl to termín, který byl rozšířenější v době, kdy byly digitální osciloskopy poprvé uvedeny na trh. DPO má jinou architekturu než tradiční digitální osciloskop, což mu umožňuje rychlejší zpracování signálů.

Zvýšení rychlosti zpracování u digitálního fosforového osciloskopu, DPO je dosaženo použitím paralelního zpracování namísto tradičnější a přímočařejší architektury sériového zpracování.

Pomocí technik paralelního zpracování a specializovaného procesoru je DPO schopen snadněji zachytit přechodové jevy, které se vyskytují v digitálních systémech. Mezi ně mohou patřit falešné impulsy, glitche a přechodové chyby. Emuluje také zobrazovací vlastnosti analogového osciloskopu a zobrazuje signál ve třech rozměrech: čas, amplituda a rozložení amplitudy v čase, vše v reálném čase.

Ačkoli název DPO může naznačovat, že je založen na chemickém luminoforu, nemusí tomu tak být, protože se používají modernější displeje. Má však mnoho aspektů fosforového osciloskopu a zobrazuje tím intenzivnější obraz, čím častěji prochází průběh určitým bodem.

2.1.4 Osciloskop pro smíšené signály

Mixed Signal Oscilloscope (MSO) je moderní typ osciloskopu, který kombinuje funkcionalitu digitálního osciloskopu a logického analyzátoru. Tento přístroj umožňuje současně analyzovat jak analogové, tak digitální signály, což je velmi užitečné v dnešním světě, kde se často setkáváme s komplexními smíšenými signály v elektronických systémech.

Podle [7] má zpravidla méně kanálů než samostatný logický analyzátor. Umí také naměřená digitální data interpretovat, například jako BCD pro sedmisegmentový displej a na zobrazit na svém displeji jakému segmentu informace odpovídá.

Firma Keysight udává [8], že MSO jsou vybaveny možnostmi spouštění i dekódování sériových sběrnic. Logické analyzátory však podobnou technologii spouštění a de-

kódování nemají. Bez spouštění protokolu není možné osciloskop nastavit tak, aby se spouštěl na konkrétní pakety. Můžete například nastavit MSO tak, aby se spustil, když dojde ke čtení I2C na určitou adresu s určitou hodnotou dat. Případně můžete spouštět na určitém datovém paketu SPI nebo na začátku výčtu USB.

Firma Keysight i umožňuje vylepšit DSO na MSO.

2.1.5 Osciloskop se smíšenou doménou

Mixed domain osciloskopy (MDO) představují další vývojový krok v oblasti měřicích přístrojů, jelikož kombinují vlastnosti digitálních osciloskopů s funkčností spektrálního analyzátoru. MDO umožňují měření a analýzu signálů jak v časové, tak i frekvenční oblasti, což zvyšuje jejich univerzálnost a použitelnost v různých aplikacích. MDO jsou vhodné pro vysokofrekvenční aplikace
Společnost Tektronix poznamenává[9], že jejich MDO mají všechny možnosti MSO a ještě více:

- 16 digitálních kanálů pro ladění smíšených signálů,
- Automatické dekódování a spouštění sériové sběrnice,
- Všestranné spouštění, dlouhá délka záznamu a snadná navigace a vyhledávání průběhů.
- Vestavěný hardware spektrálního analyzátoru,
- Vestavěný libovolný/funkční generátor,

2.1.6 Signálový generátor

Jak je uvedeno v [10], generátor signálu je elektrický přístroj, který generuje různé předem definované signály pro testování nebo navrhování elektronických obvodů a zařízení.

Generátor signálu může poskytovat „ideální"průběhy nebo může přidat známé, opakovatelné množství a typy zkreslení (nebo chyby) do signálu, který dodává.

Existuje mnoho typů generátorů signálů, včetně generátorů funkcí, generátorů libovolných průběhů a generátorů vektorových signálů. Jednotlivé typy jsou následující:

- **Signálový generátor:** Obecný název kategorie pro analogové a digitální zdroje elektronických signálů.
- **Funkční generátor:** Generátory které se obvykle používají v případě potřeby běžných průběhů, jako je sinusový, vlnový, trojúhelníkový atd.
- **Generátor náhodného průběhu(Arbitrary function generator):** generátory, které jsou schopny sestavovat náhodné průběhy.
- **Arbitrární generátor průběhu(Arbitrary waveform generator):** Arbitrární generátory průběhů se většinou používají, když je potřeba sestavit vlastní průběhy (spíše než přednastavené běžné průběhy).

- **RF signálový generátor:** Generátory RF signálu se používají pro bezdrátové aplikace a obvykle poskytují i běžnou analogovou modulaci, jako je AM, FM a PM.
- **(RF) Vektorový generátor signálu:** Generátory RF vektorových signálů podporují analogovou i vektorovou modulaci na RF nosných pro digitální komunikační aplikace.

2.1.7 Programovatelné hradlové pole

Podle [11] je FPGA (Field-Programmable Gate Array) technologie programovatelného obvodu využívající univerzální konfigurovatelné logické bloky a programovatelné propojení, využívanou pro návrh, ladění a implementaci hardwarových řešení bez potřeby vytvářet vlastní integrované obvody.

Charakteristickým rysem FPGA je, že jsou programovatelné na straně zákazníka, a to pomocí HDL jazyků. Akt programování FPGA se nazývá konfigurace, aby se odlišil od načítání jakýchkoli souvisejících softwarových programů.

Tato skutečnost dává FPGA prostor v oblasti prototypování, kde zákazníci mohou programovat tato zařízení a implementovat svá řešení. Produkty s FPGA lze tak snadno vylepšit, případně zbavit chyb. Na druhou stranu návrh na bázi FPGA může být při implementaci stejné logiky na čipu dražší než u jiných variant ASIC (standardní buňky, hradlová pole), neboť se většinou nevyužije celá plocha čipu. U FPGA mohou náklady vzrůst až 1,5 - 10× oproti ASIC, ale s každou novou generací litografie roste i cena u ASIC. Z těchto důvodů většina středních a menších systémů spoléhá na kombinaci FPGA, díky jejich adaptabilitě, a ASIC nebo ASSP (Application-Specific Standard Parts) a pamětí.

Technologie FPGA ovlivnila vývoj téměř všech produktů v oblasti elektroniky od svého uvedení na konci 80. let.

Zynq Book [12] uvádí, že Architektura FPGA se skládá z následujících komponent:

- **Konfigurovatelný logický blok (CLB) a Plátky(slices):** CLB tvoří malá, pravidelná seskupení logických prvků uspořádaná do dvourozměrného pole v PL. Tato seskupení jsou navzájem propojena pomocí programovatelných propojení. Vedle každého CLB je umístěna přepínací matice a v rámci CLB jsou obsaženy dva logické plátky. Tyto plátky, vybavené prostředky pro realizaci kombinatorických a sekvenčních logických obvodů, se skládají ze čtyř vyhledávacích tabulek, osmi klopných obvodů a dalších logických prvků.
- **Blok univerzální logické funkce generované tabulkou (LUT):** Toto flexibilní zařízení umožňuje implementovat logické funkce až pro šest vstupů, malou paměť typu ROM nebo RAM, nebo posuvný registr. LUT lze dle potřeby kombinovat pro vytvoření větších logických funkcí, pamětí nebo posuvných registrů.

- **Flip-flop (FF):** FF je sekvenční obvodový prvek, který implementuje jednobitový klopný obvod s funkcí resetování. Jedním z FF lze volitelně realizovat latch.
- **Přepínací matice:** Přepínací matice poskytuje možnost vytváření propojení mezi prvky uvnitř CLB a mezi jednotlivými CLB a dalšími zdroji v PL.
- **Přenosová logika:** Přenosová logika slouží k přenosu mezisignálů mezi sousedními plátky(slicy). Skládá se z řetězce tras a multiplexorů, které propojují plátky ve vertikálním sloupci.
- **Vstupní/výstupní bloky (IOB):** IOB umožňují propojení mezi logickými prvky PL a fyzickými „podložkami"zařízení pro připojení k externím obvodům. Každý IOB je schopen zpracovat jednobitový vstupní nebo výstupní signál a obvykle jsou umístěny po obvodu zařízení.

Návrhové nástroje Xilinx automaticky syntetizují návrh do architektury daného hradlového pole (LUT, FF, IOB atd.) a následně optimálně rozmístí, propojí a namapují do konkrétní cílové architektury (čipu).

2.1.8 Systém na čipu

Dle literatury [13] je SoC komplexní integrovaný obvod, který integruje hlavní funkční prvky kompletního koncového výrobku do jediného čipu nebo čipové sady. Obecně návrh SoC zahrnuje programovatelný procesor, paměť na čipu a akcelerační funkční jednotky implementované v hardwaru. Má také rozhraní s periferiemi. Návrhy SoC zahrnují jak hardwarové, tak softwarové komponenty. Protože návrhy SoC mohou mít rozhraní s reálným světem, často obsahují analogové komponenty a v budoucnu mohou zahrnovat také komponenty opto/mikroelektronického mechanického systému (O/MEMS).

Louise H. Crockett [12] uvádí, že smyslem SoC je implementace funkcí celého systému za použití jednoho křemíkového čipu, místo toho aby se využilo několika různých čipů. SoC může kombinovat všechny aspekty digitálního systému: zpracování, vysokorychlostní logiku, rozhraní, paměť atd. Všechny tyto prvky by jinak mohly být realizovány pomocí fyzicky oddělených zařízení, a kombinací těchto funkcí by bylo možné dosáhnout dohromady do systému na úrovni desky plošných spojů (PCB). Avšak SoC je levnější umožňuje rychlejší a bezpečnější přenos dat mezi jednotlivými prvky systému, má vyšší celkovou rychlost systému, nižší spotřebu energie, menší fyzické rozměry, a vyšší spolehlivost.

2.2 Průzkum trhu

V dnešní době je možné zprovoznit nějakou variantu osciloskopu na velkém množství prototypovacích desek či levných zařízeních, ale i na přístrojích za stovky tisíc korun. V následující sekci bych se rád podíval na nějaká z nich a diskutoval o jejich parametrech, výhodách a nevýhodách a provedení.

- **DSO138:** Levný a jednoduchý osciloskop s 50MHz šířkou pásma, 1 GSPS vzorkovací frekvencí a 8bitovým rozlišením. Oblíbený mezi amatérskými uživateli a školními laboratořemi [14]. Existuje také projekt DLO-138 nabízí vylepšený firmware, který rozšiřuje jeho funkcionalitu [15]. DSO138 je vhodný pro jednoduché úkoly a úpravy firmware poskytují další možnosti pro pokročilé uživatele.
- **Rigol DS1052E:** Tento osciloskop nabízí skvělý poměr cena výkon, s 50 MHz šířkou pásma, 1 GSPS vzorkovací frekvencí a 12bitovým rozlišením. Oblíbený mezi amatérskými uživateli a školními laboratořemi. Navíc nabízí možnost dekódovat sériovou komunikaci a upgradovat firmware pro rozšíření funkcí. Jeho dostupnost a cenovka jej činí ideálním pro začínající uživatele [16].
- **Tektronix TBS1000C:** Osciloskop střední třídy s 50 - 200 MHz šířkou pásma, 1 GSPS vzorkovací frekvencí a 8bitovým rozlišením. Vhodný pro výuku a průmyslové aplikace. Tento osciloskop nabízí výkonnější řešení pro uživatele, kteří potřebují širší škálu funkcí a šířku pásma [17].
- **Tektronix TBS2000B:** Vyšší třída osciloskopu s 70 - 200 MHz šířkou pásma, 1-2 GSPS vzorkovací frekvencí a 8bitovým rozlišením. Nabízí rozšířené možnosti triggeru a kompatibilitu s TekVPI sondami. TBS2000B je ideální pro ty, kteří potřebují rozšířené funkce a vyšší výkon pro pokročilé aplikace [18].
- **Rohde & Schwarz RTB2000:** Osciloskop s vysokým rozlišením a širokým spektrem funkcí. Nabízí 70 - 300 MHz šířku pásma, 2,5 GSPS vzorkovací frekvenci a 10bitové rozlišení ADC. RTB2000 je vhodný pro profesionální uživatele, kteří potřebují vysokou úroveň přesnosti a širokou škálu možností pro analýzu signálů [19]. Navíc nabízí pokročilé možnosti měření a analýzy, což usnadňuje práci s náročnými úkoly.
- **Tektronix MSO22:** Vysoký výkon s 500 MHz šířkou pásma, 2,5 GSPS vzorkovací frekvencí a 8-16bitovým rozlišením. Kombinuje analogové a digitální kanály pro flexibilní analýzu signálů. MSO22 je vhodný pro uživatele, kteří potřebují kombinaci analogových a digitálních kanálů pro komplexní analýzu signálů [20].
- **Rohde & Schwarz RTM3004:** Osciloskop vyšší třídy s 10bitovým ADC, desetinásobnou pamětí a 10,1"dotykovým displejem. Nabízí 70 - 300 MHz šířku pásma, 2,5 GSPS vzorkovací frekvencí a širokou škálu funkcí pro analýzu signálů. RTM3004 je vhodný pro profesionální uživatele, kteří potřebují vysokou úroveň přesnosti a pokročilé možnosti analýzy signálů. Tento osciloskop byl také použit pro měření v této práci [21].

V následujících tabulkách je pak celkové srovnání.

Tab. 2.1: Srovnání klíčových parametrů osciloskopů (část 1)

Parametr	**DSO138**	**Tektronix TBS1000C**
Šířka pásma	50 MHz	50 - 200 MHz
Počet kanálů	1	2
Vzorkovací frekvence	1 GSPS	1 GSPS
Rozlišení ADC	8 bitů	8 bitů

Tab. 2.2: Srovnání klíčových parametrů osciloskopů (část 2)

Parametr	**Tektronix TBS2000B**	**Tektronix MSO22**
Šířka pásma	70 - 200 MHz	500 MHz
Počet kanálů	2-4	2-4 (analog) + 16 (digitální)
Vzorkovací frekvence	1 GSPS	2,5 GSPS
Rozlišení ADC	8 bitů	8 bitů

Tab. 2.3: Srovnání klíčových parametrů osciloskopů (část 3)

Parametr	**Rohde & Schwarz RTM3000**	**Rohde & Schwarz RTB2000**
Šířka pásma	100 - 500 MHz	70 - 300 MHz
Počet kanálů	2-4	2-4
Vzorkovací frekvence	5 GSPS	2,5 GSPS
Rozlišení ADC	10 bitů	10 bitů

2.3 Řešení nabízená firmou Digilent

Firma Digilent, má několik svých kitů které mohou plnit funkci osciloskopu.

2.3.1 Systém Analog Discovery 2

Digilent Analog Discovery 2, jak uvádí [22], je USB osciloskop, logický analyzátor a multifunkční nástroj, který umožňuje uživatelům měření, vizualizaci, generování, záznam a ovládání smíšených signálových obvodů různých druhů. Tento testovací a měřicí přístroj, vyvinutý ve spolupráci se společností Analog Devices a podporovaný univerzitním programem Xilinx, je kompaktního rozměru, aby se vešel do kapsy, ale zároveň poskytuje dostatečný výkon, aby mohl nahradit řadu laboratorního vybavení. Poskytuje tak inženýrům, studentům, hobbistům a elektronickým nadšencům volnost při práci s analogovými a digitálními obvody v prakticky jakémkoli prostředí, ať už v laboratoři nebo mimo ni. Analogové a digitální vstupy a výstupy lze k obvodu připojit prostřednictvím jednoduchých drátových sond, nebo lze pro připojení a využití vstupů a výstupů použít adaptér Analog Discovery BNC a sondy BNC.

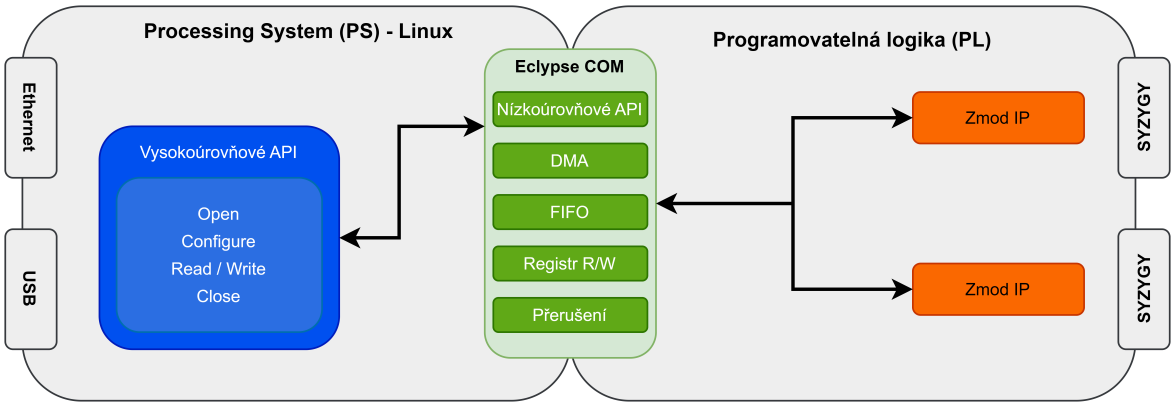
Analog Discovery 2 je ovládán pomocí softwaru WaveForms, který je dostupný pro počítače, a může být dále konfigurován prostřednictvím javascriptu nebo LabVIEW.

Obr. 2.2: Analog Discovery 2, převzato z [22].

2.3.2 Deska Eclypse Z7

Podle informací od firmy Digilent [23] je Eclypse Z7 speciální prototypovací platforma, která využívá Zynq-7000 APSoC (All Programmable System-on-Chip) . Zynq-7000 kombinuje dvoujádrový procesor ARM Cortex-A9 s programovatelnou logikou Xilinx řady 7. Navíc obsahuje dva vysokorychlostní rozšiřující porty SYZYGY pro připojení specifického hardwaru. Do těchto portů se zapojují Zmod rozšiřující karty, které rozšiřují desku o AD či DA převodníky a zároveň umožňují různé finální použití, například generátor signálu. Tyto karty lze kombinovat, což umožňuje přizpůsobení projektů pro různé aplikace, jako jsou lékařská zařízení nebo osciloskopy. Zmod desky jsou plně integrované do vývojového systému Vivado [24]. Pro lepší kompatibilitu jsou na desce také dostupné dva Pmod konektory. Mezi další výbavu patří ethernet a slot pro SD kartu.

Čip Zynq-7000 může provozovat Linux, a existují speciální distribuce Linuxu pro tuto platformu, jako například PetaLinux či Xillinux. Díky své architektuře (viz obr. 2.3) je Eclypse Z7 zvláště přizpůsoben pro použití vysokoúrovňových programovacích jazyků, což umožňuje vývojářům využít FPGA bez nutnosti přímé interakce prostřednictvím nižších jazyků. Tato vlastnost zvyšuje flexibilitu a rychlost prototypování na této desce.

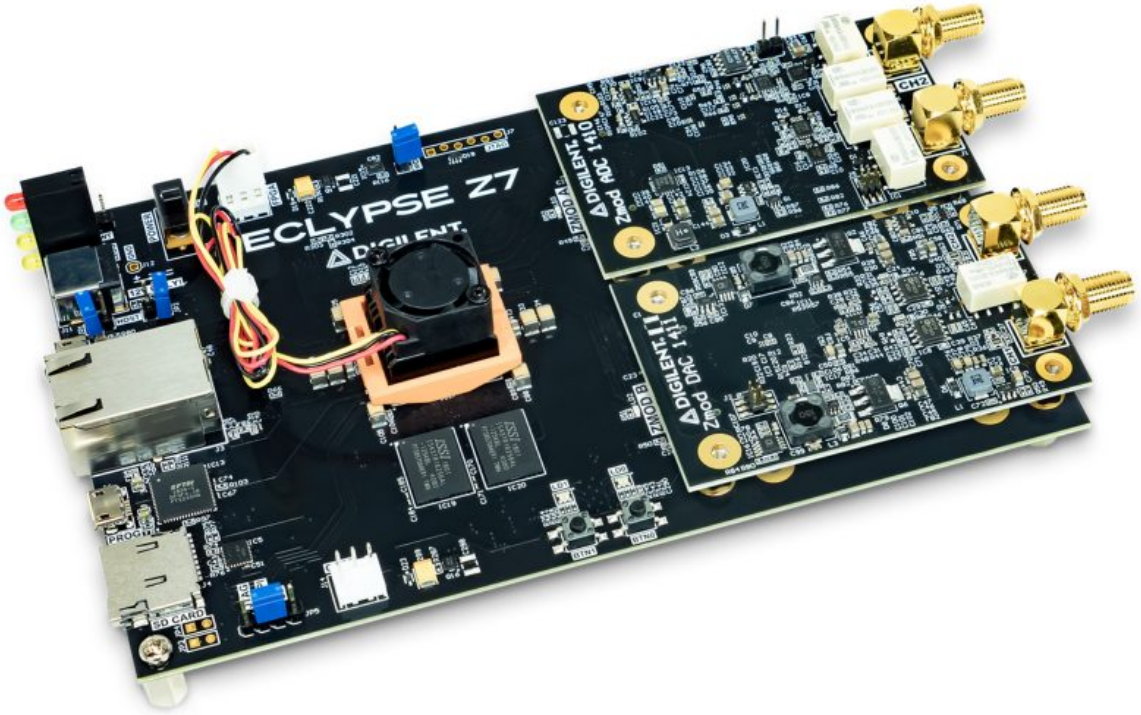


Obr. 2.3: Softwarová architektura Eclypse Z7, převzato a upraveno z [25].

Díky uvedeným vlastnostem je Eclypse Z7 ideální pro vysokorychlostní sběr a analýzu analogových dat. Navíc umožňuje rychlé prototypování testovacích a měřicích aplikací, potenciálně včetně softwarově definovaného rádia, ultrazvuku, lékařských zařízení a dalších. Platforma je dostupná i v balíčku pro vývoj osciloskopu s sondami, rozšiřujícími deskami pro generátor funkcí a dvěma nebo čtyřmi kanály, a to za cenu 635 dolarů. Díky integraci Linuxu je zde také možnost využít WaveForms.

V důsledku těchto vlastností je tento kit velmi vhodný pro prototypování a designování ve větších firmách, umožňuje rychlé a efektivní testování funkčnosti budoucích

modelů.



Obr. 2.4: Eclypse Z7, napravo jsou vidět Zmod rozšiřující desky, převzato z [23].

2.3.3 Vývojová deska Zybo Z7

Zybo Z7 [26] (na obrázku 2.5) je vývojová deská vytvořena kooperací firem Digilent a Xilinx. Tato deska je určena pro návrhy a vývoj SoC založených na SoC Zynq-7000, podobně jako Eclypse Z7.

Zynq-7000 nabízí vývojářům velkou flexibilitu díky těsnému propojení mezi FPGA a mikroprocesorem, což umožňuje vytváření systémů s vysokou výkonností a spolehlivostí [12].

Zybo Z7 obsahuje různé paměťové subsystémy pro rychlé ukládání programů a dat. Mezi tyto subsystémy patří 16GB DDR3 RAM jako hlavní systémová paměť a rychlá SRAM pro dočasné ukládání dat. Kromě mikroprocesoru a paměťových subsystémů je na desce také řada periferních zařízení, jako je ethernet, USB, HDMI a další.

V tabulce 2.4 je srovnání dvou variant Zybo Z7. Jak lze vidět, výbava obou variant se výrazně liší, zejména co se týče počtu vyhledávacích tabulek (LUT), flip-flopů a bloků RAM. Pro naše účely je vhodnější varianta Z7-20, zejména díky většímu počtu bloků RAM, což usnadňuje práci s pamětí. Na desce jsou také Pmod porty, které

výrazně rozšiřují možnosti připojení rozšiřujících modulů, jako jsou rotační enkodéry, GPS modul, bluetooth modul a mnoho senzorů. Deska také obsahuje vestavěný dvoukanálový ADC s vzorkovací frekvencí 1 MSPS.

Tab. 2.4: Porovnání modelů Zybo Z7

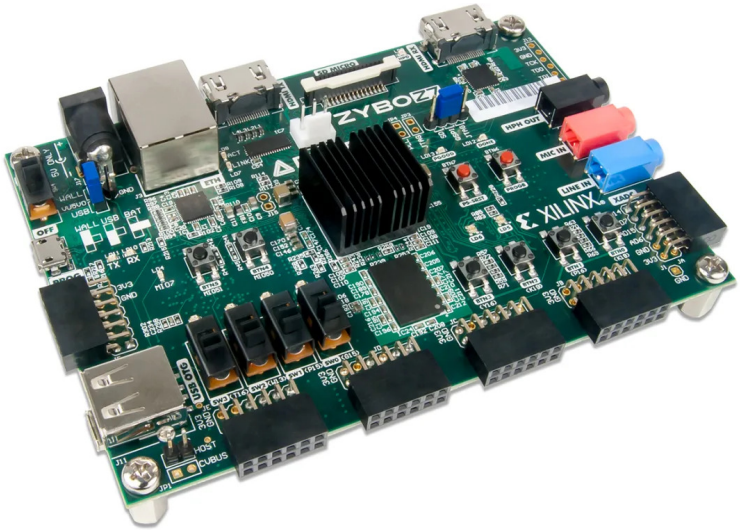
Produktová varianta	**Zybo Z7-10**	**Zybo Z7-20**
Zynq část	XC7Z010-1CLG400C	XC7Z020-1CLG400C
Look-up tabulky (LUT)	17,600	53,200
Flip flop	35,200	106,400
Blok RAM	270 KB	630 KB
Počet Pmod portů	5	6
Konektor na ventilátor	Ne	Ano
Heatsink	Ne	Ano
Podpora HDMI CEC	Pouze TX port	TX i RX port
RGB LED	1	2

Díky přítomnosti dvou HDMI portů je Zybo Z7 často preferováno pro zpracování videosignálů, zejména model Z7-20, který je pro tento účel Digilentem doporučen.

Podobně jako u Eclypse Z7, existuje i zde možnost spuštění Linuxu. Avšak na rozdíl od Eclypse Z7, Zybo Z7 nenabízí API pro pohodlnější práci s vyššími programovacími jazyky. Pro řešení komplexních úloh je potřeba využít kombinaci následujících vývojových nástrojů od společnosti Xilinx:

- Vivado: Tento hlavní nástroj pro návrh hardwarové platformy umožňuje tvorbu vlastních IP (ve VHDL/Verilogu) a nabízí nástroje a průvodce pro připojení IP a vytvoření systému na čipu.
- Xilinx SDK: Tento nástroj je určený pro tvorbu softwaru pro hardwarovou platformu, která byla vytvořena v prostředí Vivado. Kromě využití ovladačů Xilinx pro standardní IP, může uživatel pomocí SDK vytvářet firmware potřebný ke správě vlastních IP a aplikace pro běh nad platformou, buď bez operačního systému nebo s podporou operačního systému (FreeRTOS nebo Linux), což například umožňuje větší kontrolu nad časováním.
- Vivado HLS: Tento nástroj umožňuje vytváření vlastních IP v jazyce C/C++, které lze následně syntetizovat pomocí Vivada a začlenit do hardwarové platformy.

- SDSoC: Pomocí tohoto nástroje lze vytvořit kompletní platformu SoC od její softwarové definice v jazyce C/C++. SDSoC, s využitím HLS, vytvoří IP z jazyka C/C++, ale také veškerou softwarově definovanou síť konektivity, což umožňuje snadný přenos softwarových funkcí z PS do PL. Obsahuje také nástroj pro profilování, který umožňuje efektivně identifikovat úzká místa a najít kandidáty na hardwarovou akceleraci [27].
- Petalinux: Tato embedded Linux distribuce od společnosti Xilinx je zaměřena na velmi komplexní aplikace, jako je například strojové učení.



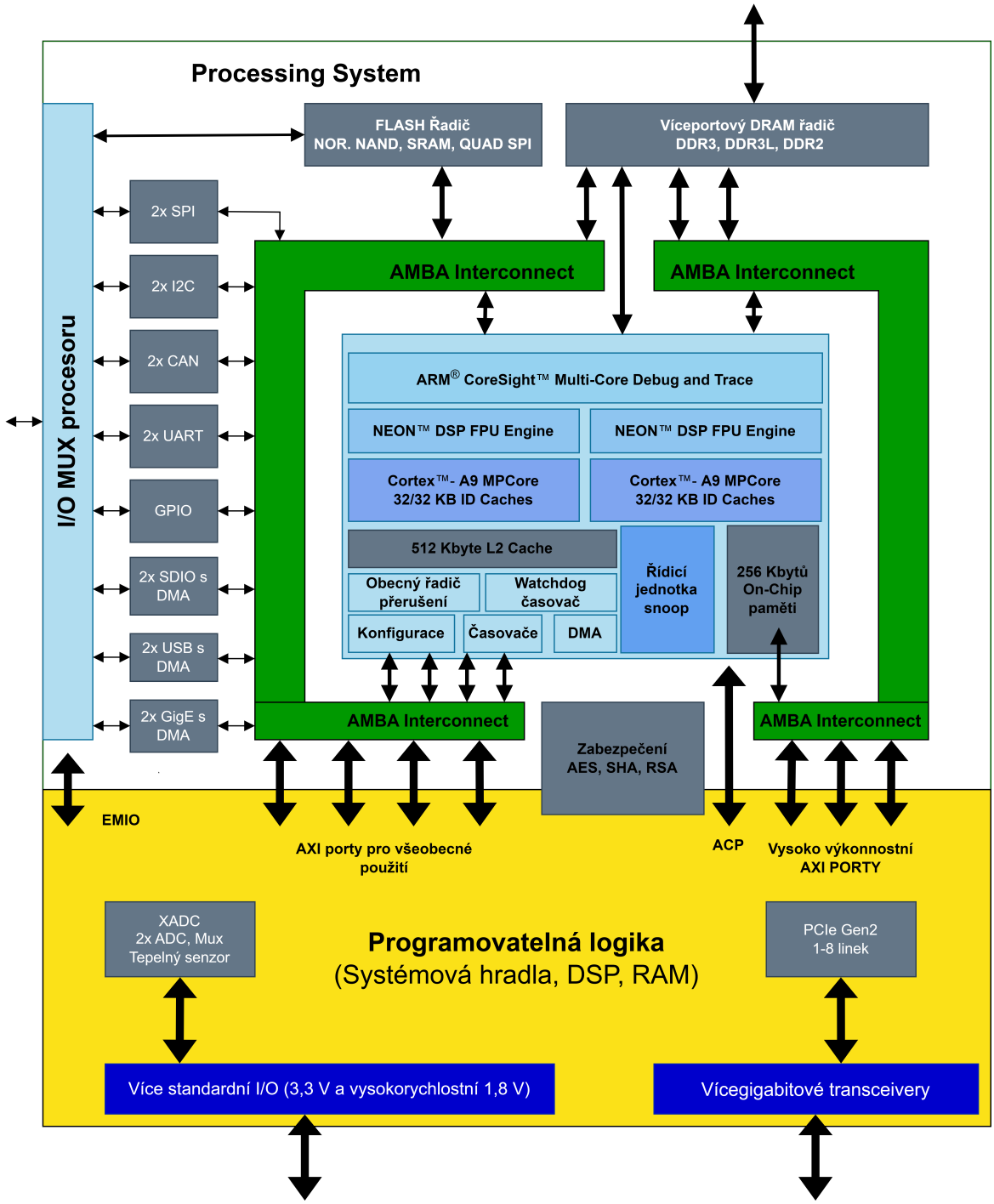
Obr. 2.5: Zybo Z7 - 20, převzato z [26].

2.3.4 Architektura systému na čipu ZYNQ Z7

Konkrétní model čipu XC7Z020-1CLG400C, který na desce nalezneme, je rozdělen na dva subsystémy: Processing System - PS a a Programovatelnou Logiku - PL (viz. obr2.7)

To znamená, že desku lze použít v jakékoli z následujících konfigurací:

- Pouze PS: Použití dvou procesorů ARM Cortex-A9 s využitím všech integrovaných periferních řadičů (paměť a I/O) pro připojení ke zdrojům desky. V tomto případě lze trochu obecněji tvrdit, že desku provozujeme v principu jako mikrokontrolér.
- Pouze PL: Využití pouze FPGA a připojení desky pouze přes I/O piny a integrovaných periferií PS.
- PS + PL: Využití kombinace prostředků dostupných v obou systémech.



Obr. 2.6: Architektura čipu XC7Z020-1CLG400C, převzato a upraveno z [26].

Programovatelná logika je takřka identická s běžným FPGA Xilinx 7. řady, má pouze několik portů a sběrnic pro lepší komunikaci s Processing Systémem. Také není přesně identický konfigurována ale musí být konfigurována procesorem či přes JTAG port. Processing System se skládá z více částí. Jde o:

- Jednotka APU (Application Processing Unit), která obsahuje 2 procesory Cortex-

A9.

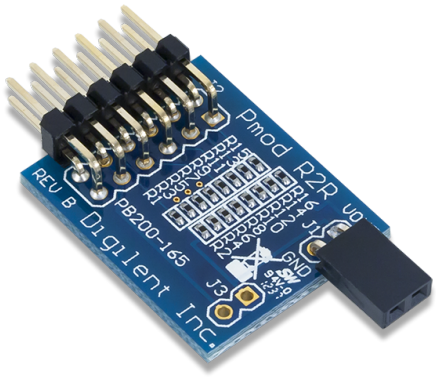
- Propojení sběrnice AMBA (Advanced Microcontroller Bus Architecture)
- Paměťový řadič DDR3
- a různé periferní řadiče, jejichž vstupy a výstupy jsou multiplexovány na 54 vyhrazených pinů (tzv. multiplexované I/O nebo MIO piny).

Periferní řadiče jsou připojeny k procesorům jako podřízené jednotky prostřednictvím AMBA a obsahují řídicí registry s možností čtení/zápisu, které jsou adresovatelné v paměťovém prostoru procesorů.

2.3.5 Pmod digitálně analogový převodník R2R

Pmod (Peripheral Module) je rozhraní vytvořené společností Digilent pro připojení periferních modulů s nízkou frekvencí a malým počtem I/O pinů k řídícím desce. Toto rozhraní se vyznačuje šesti nebo dvanácti piny, které podporují různé protokoly, jako jsou SPI, I²C, UART, I2S, H-bridge a GPIO [28]. Moduly Pmod mohou být připojeny přímo k řídící desce nebo přes kabely a jsou napájeny hostitelem prostřednictvím napájecích a zemních pinů rozhraní.

Pmod R2R je konkrétní typ modulu, který funguje na principu digitálně-analogového převodníku s konfigurací rezistorového žebříčku R-2R. Uživatelé mohou do modulu poskytnout 8 bitů paralelních dat prostřednictvím GPIO signálů. Tento modul umožňuje 8-bitovou digitálně-analogovou konverzi až do frekvence 25 MHz [29].



Obr. 2.7: Pmod R2R, převzato z [29].

2.3.6 Prostředí LabVIEW a NI VISA

Dle [30] je LabVIEW (Laboratory Virtual Instrument Engineering Workbench). Tento software, který vyvinula společnost National Instruments, slouží pro systémový design a je významně využíván v aplikacích potřebujících testování, měření a kontrolu. Jeho unikátní vlastností je především rychlý přístup k hardwaru a datovým informacím.

LabVIEW je založen na grafickém programovacím jazyku nazývaném G. Tato technologie umožňuje výkonné a flexibilní vytváření systémů pro sběr dat, měření, monitorování a množství dalších inženýrských aplikací.

Jednou z klíčových výhod LabVIEW je jeho možnost integrace s širokou škálou hardwarových zařízení a jeho rozsáhlá podpora hardwarových rozhraní. Díky tomu je LabVIEW schopný snadno implementovat distribuované a paralelní systémy.

Rychlý vývoj aplikací je v LabVIEW umožněn díky přímému zobrazení výstupů a dat, což umožňuje okamžitou zpětnou vazbu a usnadňuje ladění. Aplikace vytvořené v LabVIEW se nazývají virtuální nástroje (VI), jelikož jejich vzhled a ovládání napodobují fyzické přístroje, jako jsou osciloskopy nebo multimetry.

Programy v LabVIEW, označované jako VI (virtuální nástroje), mají tři hlavní komponenty:

- Přední panel - slouží jako uživatelské rozhraní.
- Blokové schéma - obsahuje grafický zdrojový kód, který definuje funkcionalitu VI.
- Ikona a panel konektorů - identifikují rozhraní VI, tak aby bylo možné použít VI v jiném VI jako podprogram.

Výhodou tohoto přístupu je, že je možné vytvářet složitější systémy skládáním jednodušších VI dohromady, což vede k modularitě a snadnému znovu použití kódu.

National Instruments Virtual Instrument Software Architecture (NI VISA) je softwarové rozhraní, které poskytuje schopnost komunikovat s širokým spektrem zařízení přes rozličné sběrnice, včetně GPIB, VXI, PXI, Serial, Ethernet a USB [31]. Jako standardizovaný nástroj pro automatizaci testů a měření nabízí konzistentní prostředí pro komunikaci s přístroji bez ohledu na typ rozhraní, což zjednodušuje vývoj a údržbu softwaru pro testování a měření. VISA je multiplatformní a podporuje vývoj aplikací pro různé operační systémy a programovací jazyky.

Komponenta VISA Serial je součástí NI VISA, která umožňuje komunikaci s zařízeními přes sériové rozhraní. VISA Serial umožňuje otevřít, konfigurovat a zavřít sériové porty, číst a zapisovat data a také ovládat sériové linky a monitorovat jejich stav [32]. Toto je obzvlášť užitečné pro zařízení, která využívají sériové komunikační standardy, jako je RS-232 nebo RS-485. VISA Serial poskytuje efektivní a univerzální nástroj pro správu a řízení sériové komunikace v rámci širšího ekosystému NI VISA.

3 Řešení

3.1 Naše koncepce

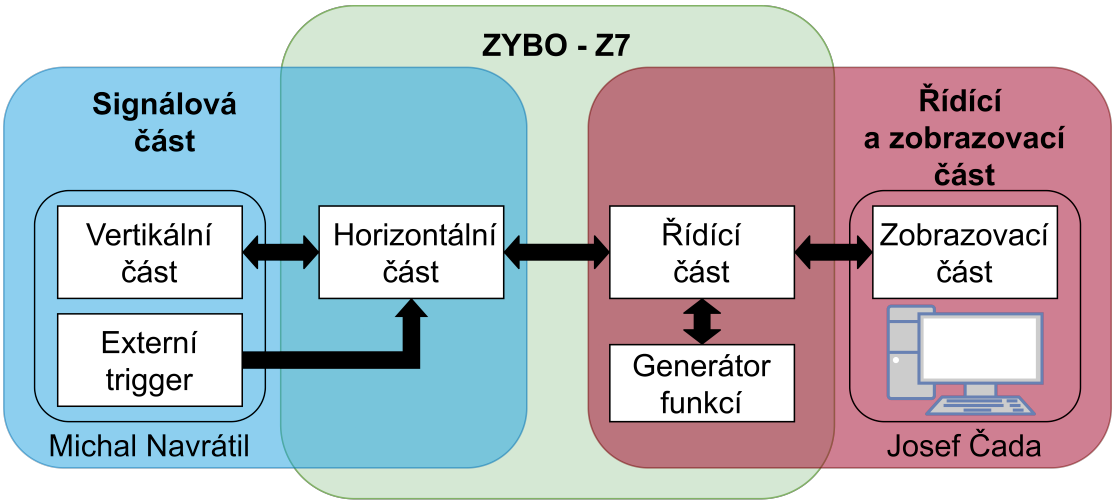
Základním kamenem naší koncepce je deska Zybo-Z7-20, jejíž detaily jsou popsány v kapitole 2.3.3. Vzhledem k rozhodnutí využít co nejvíce funkcí na desce, byl zvolen jako hlavní převodník vnitřní ADC čipu XC7Z020-1CLG400C této desky. Využití tohoto převodníku postačí pro realizaci osciloskopu s pásmem 500 kHz , pomocí prokládání vzorků jednotlivých kanálu.

Zybo-Z7-20 dovoluje návrh pouze digitálních částí osciloskopu, bude tedy rozšířena o plošný spoj obsahující vertikální část a externí trigger osciloskopu a program vytvořený pomocí LabVIEW na počítači pro zobrazování a uživatelský vstup. Celkové schéma s rozdělením pracovních částí je ilustrováno na obrázku 3.1.

Signál je zesilován a upravován a zesilován analogovými obvody na PCB (detailněji toto téma rozvíjí Michal Navrátil ve své práci [1]). Poté vnitřní ADC desky zde vzorkuje vzokrovací frekvencí 1MSPS singál na diskrétní hodnoty. Ty jsou pak digitálně zpracovány FPGA na desce, tam se řídí trigger level a ovládá PGA.

Stanovili jsme si následující hlavní cíle:

1. Navrhnout analogovou část systému, která zahrnuje řízení a konverzi analogových signálů pomocí ADC, a vytvořit desku plošných spojů s programovatelným zesilovačem.
2. Navrhnout a implementovat zpracování signálů, což zahrnuje akvizici, analýzu a vizualizaci signálů na uživatelském rozhraní prostřednictvím softwarové aplikace LabVIEW.
3. Vyvinout přidružený generátor signálů, který bude moci generovat různé typy signálů pro účely testování a kalibrace.
4. Při návrhu využít co nejvíce komponent a schopností desky Zybo Z7 a minimalizovat počet externích komponent.



Obr. 3.1: Celkové schéma práce

3.2 Souhrn mého řešení

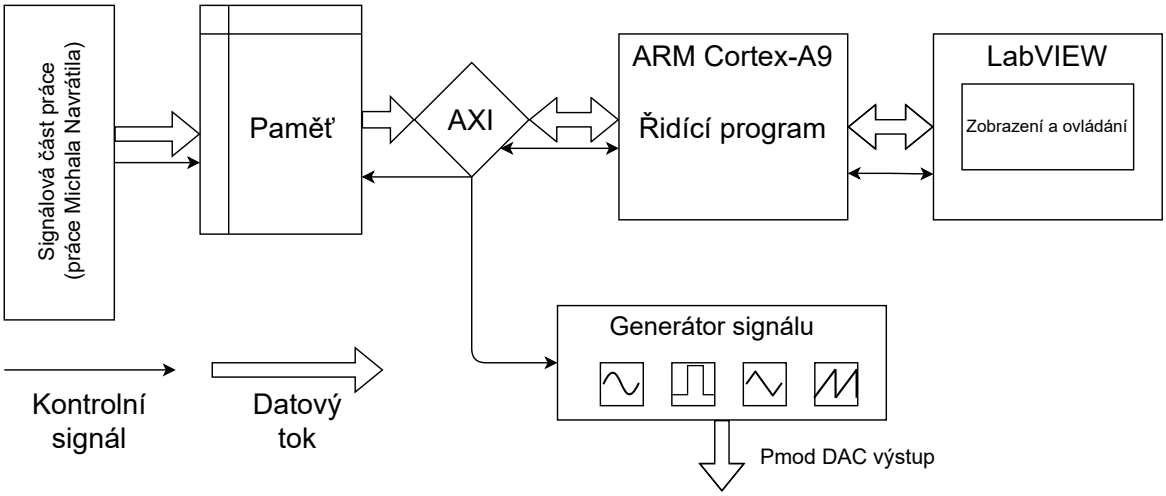
Cílem této práce je navrhnout a realizovat řídící a zobrazovací část v programu LabVIEW. Pro řídící část bylo zvoleno prostředí Xilinx Vitis, které slouží k programování, ladění, nahrávání a správě kódu pro ARM Cortex procesor.

V rámci tohoto kódu se zabýváme UART komunikací v obou směrech přes UART rozhraní. Data jsou pak formátována a předávána do počítače, kde jsou zachycena a dále zpracovávána programem LabVIEW.

Prostřednictvím front panelu virtuálního instrumentu v LabVIEW je možné také nastavit frekvenci, amplitudu, střídu a průběh signálu pro signálový generátor. Pro PWM je také možné nastavit střídu signálu.

Všechny parametry signálu je možné ovládat pomocí front panelu, což zahrnuje nastavení počtu vzorků, úrovně triggeru pro jednotlivé kanály, výběr externího triggeru nebo vynucení triggeru. LabVIEW přes UART posílá instrukce na procesor SoC, který data zapisuje a upravuje podle toho chování digitálního designu na FPGA.

Pro zobrazení, bylo cílem mít zobrazitelnou část signálu dle zadaného času, kontinuální zobrazování signálu, zobrazení měřených veličin.



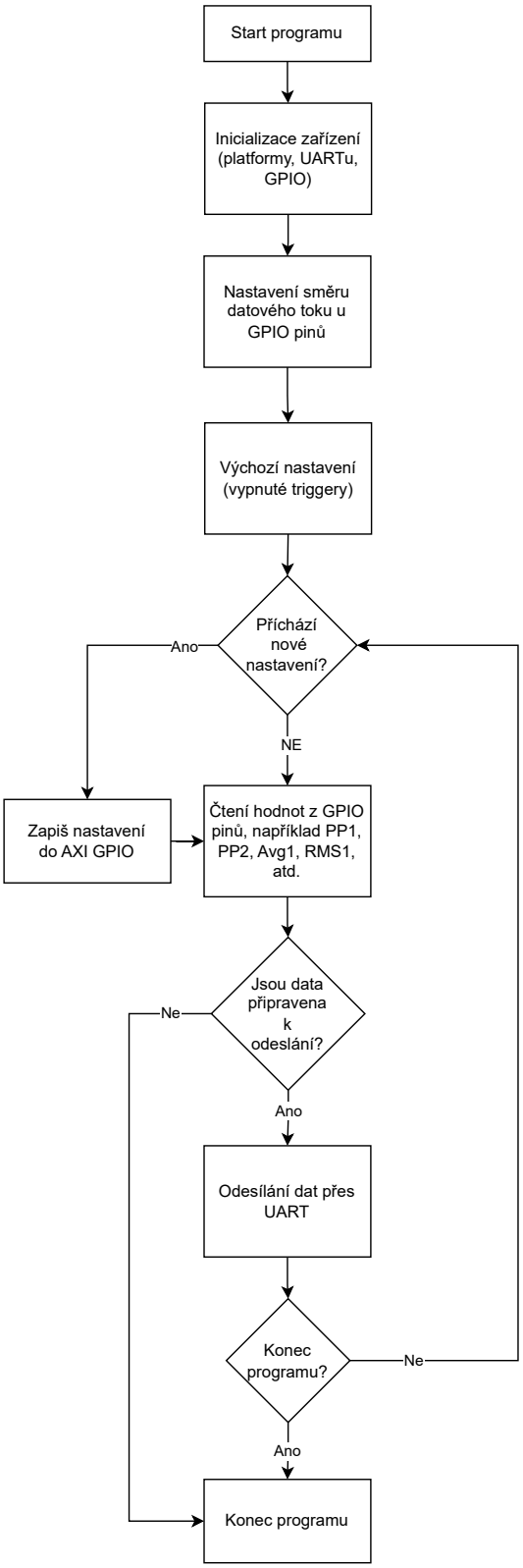
Obr. 3.2: Má část projektu

3.3 Souhrn kódu pro řízení komunikace mezi FPGA a ARM procesorem

Následuje krátký souhrn práce kódu, celý kód je dostupný v digitální příloze jako *řídící_kód.c*. Tento kód slouží k čtení a zápisu dat z/do FPGA a řízení UART komunikace mezi deskou a stolním počítačem.

1. Inicializace: Hlavní funkce main začíná inicializací několika komponent. To zahrnuje UART komunikaci, nastavení a inicializaci různých GPIO pinů pro různé periferní jednotky a jejich konfiguraci pro vstupní nebo výstupní provoz.
2. Nastavení hodnot: Definuje několik konstant a proměnných, které se používají pro konfiguraci a ovládání systému. Tyto hodnoty jsou pak použity v rámci různých funkcí a operací v rámci hlavní smyčky.
3. Definice funkcí: Definuje několik pomocných funkcí (getAddr, getData, getBit, Header, ReadVals), které se používají pro manipulaci s daty, generování hlaviček pro komunikaci a čtení hodnot přes UART komunikaci.
4. Hlavní smyčka: Vstupuje do nekonečné smyčky, kde se provádí hlavní funkčnost programu. Tato smyčka zahrnuje čtení hodnot přes UART, aktualizaci systémových nastavení podle přijatých dat, ovládání různých pinů GPIO pro ovládání různých periferních jednotek a provádění dalších operací na základě přijatých dat.

Na obrázku 3.3 je vývojový diagram řidícího programu.



Obr. 3.3: Vývojový diagram řídicího kódu na ARM procesoru

3.4 Vybrané části kódu

Vzhledem k rozsahu kódu jsou zde prezentovány pouze vybrané části, zejména ty, které interagují s AXI GPIO.

```
    short status;// Proměnná pro kontrolu stavu
    XGpio Trig_data, Trig_conf, Data_read, Bord_cont;
    // Definice GPIO proměnných
    init_platform();
    // Inicializace platformy
    XUartPs_Config *Config;
    // Vytvoření ukazatele na konfiguraci UART
    Config = XUartPs_LookupConfig(UART_DEVICE_ID);
    // Hledání konfigurace UART zařízení
    status = XUartPs_CfgInitialize(&Uart_Ps, Config,
       Config->BaseAddress); // Inicializace UART zařízení
     if (status != XST_SUCCESS) {
         // Kontrola úspěšnosti inicializace
         return XST_FAILURE;
     }
```

Kód. 3.1: Inicializace platformy a UART

Kód uvedený výše (3.1) představuje inicializaci platformy a UART zařízení. Nejdříve je definována proměnná pro kontrolu stavu a proměnné pro různé GPIO funkce. Následuje inicializace platformy. Poté se inicializuje UART zařízení a kontroluje se úspěšnost této operace. Pokud inicializace selže, funkce vrátí hodnotu ‘XST_FAILURE‘.

```
        XGpio_SetDataDirection(&Trig_data ,1 , 0xFFFFFFFF);
        // Nastavení prvního portu Trig_data na vstup
        XGpio_SetDataDirection(&Trig_data ,2 , 0xFFFFFFFF);
        // Nastavení druhého portu Trig_data na vstup
        print("\n\rTrigger data gpio set\n\r");
        // Výpis informace o nastavení

        XGpio_SetDataDirection(&Trig_conf ,1 , 0x0);
        // Nastavení prvního portu Trig_conf na výstup
        XGpio_SetDataDirection(&Trig_conf ,2 , 0x0);
        // Nastavení druhého portu Trig_conf na výstup
        XGpio_DiscreteWrite(&Trig_conf, 1, Header(0));
        // Nastavení triggeru 1
        XGpio_DiscreteWrite(&Trig_conf, 2, Header(0));
        // Nastavení triggeru 2
        print("Trigger config gpio set\n\r");
        // Výpis informace o~nastavení
```

Kód. 3.2: Konfigurace GPIO pro Trig data a Trig conf

Kód uvedený výše (kód 3.2) představuje konfiguraci GPIO pro Trig_data a Trig_conf. Pro Trig_data jsou nastaveny dva porty jako vstupní, což je signalizováno výpisem do konzole. Poté jsou nastaveny dva porty Trig_conf jako výstupní a jsou konfigurovány dva triggery. Po dokončení těchto operací je do konzole vypsána informace o úspěšném nastavení. Takto se inicializují všechny potřebné proměnné co využívají AXI GPIO.

```
1            // Nastavení portů Trig_conf-1 a Trig_conf-2 na
                 výstup a konfigurace triggerů
2            XGpio_SetDataDirection(&Trig_conf ,1 , 0x0);
3            XGpio_SetDataDirection(&Trig_conf ,2 , 0x0);
4            XGpio_DiscreteWrite(&Trig_conf, 1, Header(0));
5            XGpio_DiscreteWrite(&Trig_conf, 2, Header(0));
6            print("Trigger config gpio set\n\r");
7
8            // Konfigurace portů Data_read
9            XGpio_SetDataDirection(&Data_read ,1 , 0x0);
10           XGpio_SetDataDirection(&Data_read ,2 , 0xFFFFFFFF);
11           XGpio_DiscreteWrite(&Data_read, 1, 0x0);
12           print("Data read gpio set\n\r");
13
14           // Konfigurace portů Bord_cont
15           XGpio_SetDataDirection(&Bord_cont ,1 , 0x0);
16           XGpio_SetDataDirection(&Bord_cont ,2 , 0x0);
17           XGpio_DiscreteWrite(&Bord_cont, 1, 0x0);
18           XGpio_DiscreteWrite(&Bord_cont, 2, 0x0);
19           print("Board control gpio set\n\r");
```

Kód. 3.3: Konfigurace GPIO pro Trig_data, Trig_conf, Data_read, a Bord_cont

Tento výňatek kódu v jazyce C demonstruje konfiguraci různých GPIO portů na systému s využitím funkcí XGpio z knihovny Xilinx. V první části se porty Trig_-conf nastavují jako výstupní a dále se konfigurují triggery. Funkce XGpio_SetDataDirection() nastavuje směr dat pro daný port a funkce XGpio_DiscreteWrite() zapisuje hodnotu do portu. Toto se opakuje pro druhý port.

Následně je konfigurace opakována pro porty Data_read, kde je jeden port nastaven jako výstupní a druhý jako vstupní.

V poslední části se provádí stejný postup pro porty Bord_cont, kde jsou oba porty nastaveny jako výstupní.

Po každé konfiguraci je pro kontrolu vypsána odpovídající zpráva do konzole.

```
    u32 vals[10] = {0,0,maxi_trig,0,200,0,0,0,0,0};
       //počáteční hodnoty proměnných z~desky
    u32 length = vals[4];
    u8 chan = 0, ext;
    u32 tr, data, addr;
    print("\n\rInitialization complete, starting reading\n\r");
    trig_lev = maxi_trig;
    while(1){
        u8 num = ReadVals(&Uart_Ps,vals, 10);
        if (num == 10){
        u8 x = 2*vals[0]+vals[1];
        XGpio_DiscreteWrite(&Bord_cont, 2, x);
        trig_lev = vals[2];
        chan = vals[3];
        length = vals[4];
        freq_div = vals[5];
        ext = vals[0];
        }
    }
```

Kód. 3.4: Inicializace proměnných a smyčka pro zápis GPIO dat

Tento výňatek kódu ukazuje inicializaci několika proměnných pro další zpracování dat a hlavní smyčku, která kontinuálně čte hodnoty přijaté z UART komunikace. Po inicializaci proměnných a výpisu inicializační zprávy do konzole se program vstoupí do hlavní smyčky.

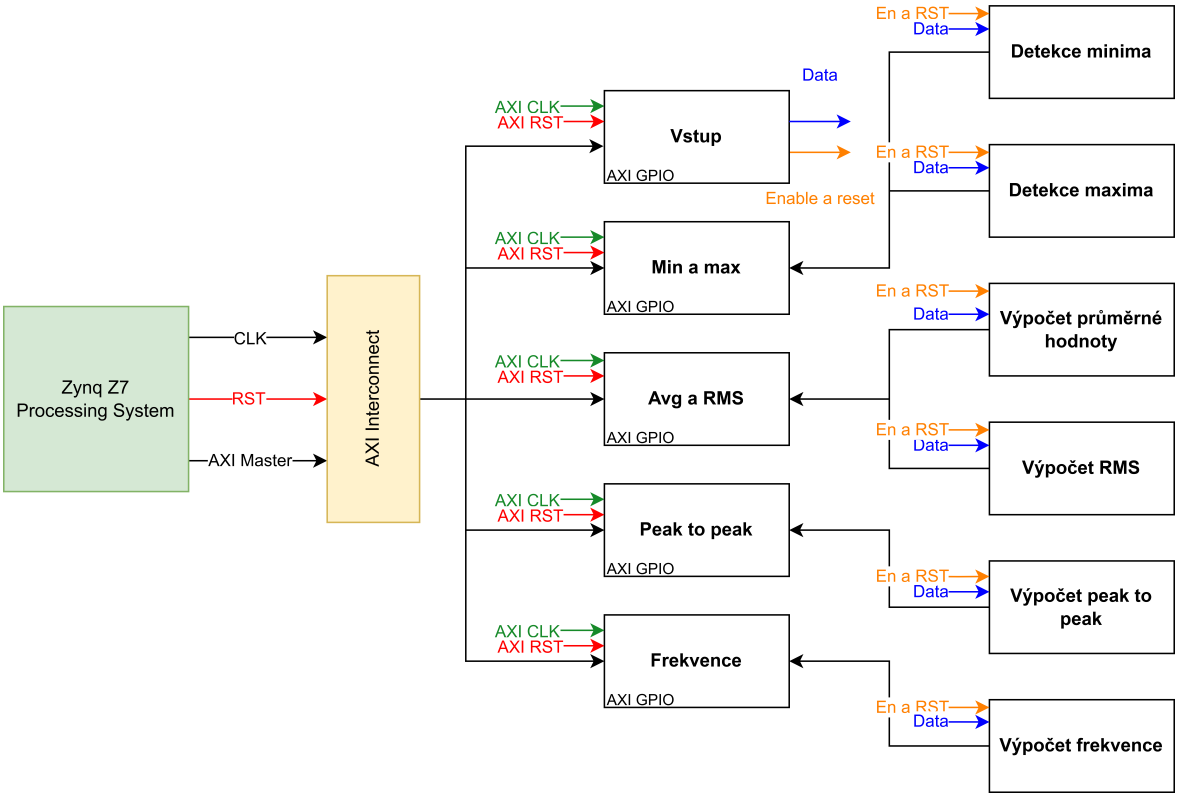
Ve smyčce se z UART přijímají hodnoty do pole vals. Pokud je přijato správné množství hodnot (v tomto případě 10), pak se provádí aktualizace proměnných a zápis do GPIO portu. Pokud není přijato správné množství hodnot, smyčka pokračuje na další iteraci bez aktualizace proměnných.

Hodnoty v poli vals představují různé konfigurační a řídící hodnoty pro desku. Tyto hodnoty se poté používají pro nastavení různých parametrů v systému, jako je úroveň triggeru, délka, frekvenční dělič a další.

3.5 Zpracovaní signálu

Vzorkovaný signál je nejlepší využít rovnou na desce, a poté zpracovaný signál posílat přes UART do počítače. Je to takto přesnější, rychlejší a příjemnější pro pozdější zpracování.

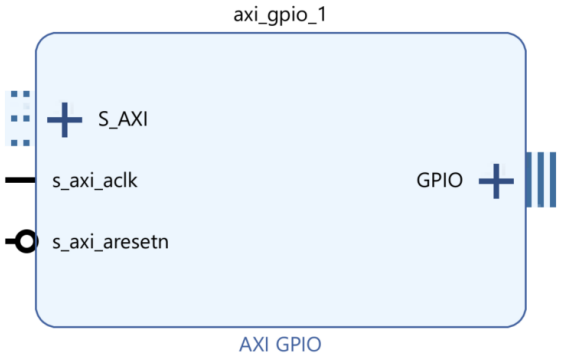
Pro tento účel byl vytvořen blokový design s pracovním názvem Calc. Jeho zjednodušené schéma je na 3.4. Celé schéma z Vivada je pak dostupné v příloze A.3.



Obr. 3.4: Zjednodušené schéma modulu pro zpracování signálu

Blok ZYNQ7 Processing System a blok AXI Interconnect řídí AXI GPIO bloky, umožňují tak jednoduchý přenos dat mezi FPGA a ARM procesorem. Signál, který byl vzorkován a je reprezentován v 12bitovém formátu (vychází to 12bitového ADC), je přijímán prostřednictvím AXI GPIO. Tento signál je následně zpracován pomocí navržených bloků. Výsledná data, po provedení výpočtů a zpracování, jsou předána dalším specifickým AXI GPIO blokům. Z těchto bloků jsou data následně čtena procesorem prostřednictvím programu v jazyce C.

3.5.1 AXI GPIO

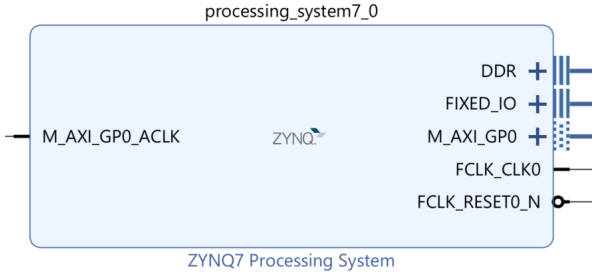


Obr. 3.5: AXI GPIO IP blok ve Vivadu

AXI GPIO je IP blok, který funguje jako spojnice mezi obecným vstupním/výstupním rozhraním a AXI4-Lite rozhraním [33]. Tento blok může být konfigurován jako jednokanálové nebo dvoukanálové zařízení, přičemž šířka každého kanálu lze nastavit nezávisle. Díky možnosti dynamické konfigurace portů pro vstup či výstup prostřednictvím povolení či zakázání 3-stavového bufferu nabízí AXI GPIO velkou flexibilitu. Kanály lze navíc nastavit tak, aby generovaly přerušení při detekci přechodu na jakémkoli ze svých vstupů.

V kontextu tohoto designu je AXI GPIO využíván jako „brána", která umožňuje komunikaci mezi FPGA a procesorem. Tento blok tedy hraje klíčovou roli při přenosu dat a koordinaci mezi těmito dvěma komponentami.

3.5.2 Zynq7 Processing System



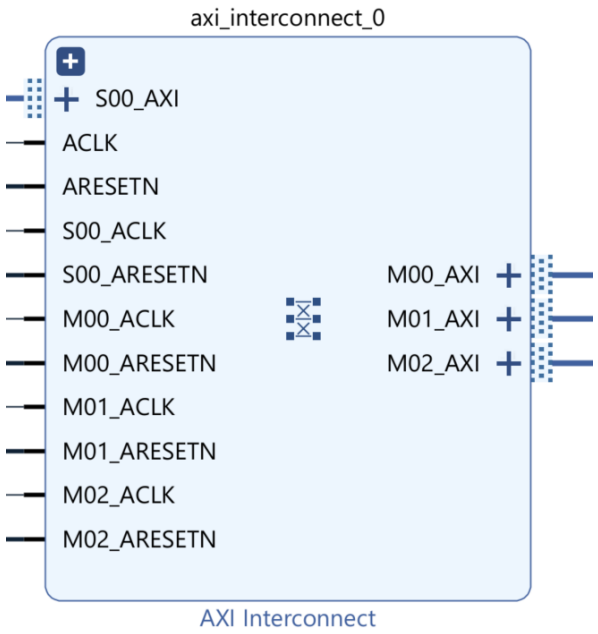
Obr. 3.6: Zynq7 Processing System v konfiguraci pro použití v modulu pro zpracování signálu

IP blok Zynq7 Processing System tvoří část Zynq čipu, která nespadá do kategorie FPGA, ale pod Processing System (PS) [34]. Jeho význam je zásadní pro návrhy vyžadující interakci s procesorem či konfiguraci periferií, hodinových signálů a dalších nastavení.

Processing System 7 wrapper inicializuje systémovou složku Zynq-7000 APSoC pro programovatelnou a externí deskovou logiku. Tento obalový prvek udržuje původní konektivity a u některých signálů zahrnuje také logické funkce [34].

Tento IP blok je tedy nezbytný pro nastavení a komunikaci s AXI, konfiguraci UART komunikace a mnoha dalších funkcí.

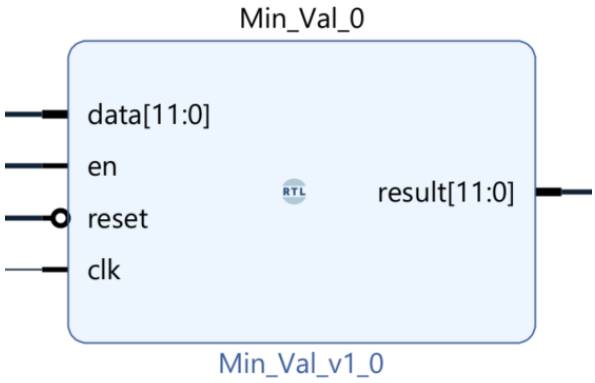
3.5.3 AXI Interconnect



Obr. 3.7: AXI Interconnect v konfiguraci pro použití v modulu pro zpracování signálu

Blok AXI Interconnect slouží jako propojovací prvek mezi jedním nebo více AXI Master zařízeními a jedním nebo více Slave zařízeními. Tyto zařízení jsou paměťově mapované.

3.5.4 Detekce minima



Obr. 3.8: Blok pro detekci minimální hodnoty

Tento modul je navržený tak, aby sledoval série hodnot přicházejících na svůj datový vstup a udržoval aktuální minimální hodnotu této série.

Jeho hlavní funkci lze popsat následujícím způsobem:

1. Na začátku, nebo když je aktivován reset, je výstup result nastaven na maximální možnou hodnotu pro 12-bitové číslo (4096). To znamená, že jakákoli následující hodnota, co příjde na vstup bude menší než result.
2. Při každé náběžné hraně hodinového signálu clk modul kontroluje, zda je povolen. Pokud není povolen (en je neaktivní), tak se žádná další akce se neprovádí.
3. Pokud je modul povolen (en je aktivní), kontroluje se, zda je aktuální hodnota data menší než aktuální minimální hodnota uložená v result. Pokud ano, result se aktualizuje na tuto novou minimální hodnotu.
4. Tento proces se opakuje pro každou náběžnou hranu hodinového signálu, dokud je modul povolen, což umožňuje modulu sledovat a udržovat aktuální minimální hodnotu série vstupních hodnot.

Výstupem tohoto modulu je tedy nejmenší hodnota, kterou kdy přijal na svém vstupu, od posledního resetu nebo od začátku operace. Tato hodnota je k dispozici na výstupu a posílá se na AXI GPIO blok, kde ji zpracuje procesor a pošle přes UART.

3.5.5 Detekce maxima

Obr. 3.9: Blok pro detekci maximalní hodnoty

Tento blok funguje algoritmicky stejně jako blok pro detekci minima, ale pro detekci maxima.

Jeho hlavní funkci lze popsat následujícím způsobem:

1. Na začátku, nebo když je aktivován reset, je výstup result nastaven na minimální možnou hodnotu pro 12-bitové číslo (0). To znamená, že jakákoli následující hodnota co příjde na vstup bude větší než result.
2. Při každé náběžné hraně hodinového signálu clk modul kontroluje, zda je povolen. Pokud není povolen (en je neaktivní), tak se žádná další akce se neprovádí.
3. Pokud je modul povolen (en je aktivní), kontroluje se, zda je aktuální hodnota data větší než aktuální maximální hodnota uložená v result. Pokud ano, result se aktualizuje na tuto novou maximální hodnotu.
4. Tento proces se opakuje pro každou náběžnou hranu hodinového signálu, dokud je modul povolen, což umožňuje modulu sledovat a udržovat aktuální maximální hodnotu série vstupních hodnot.

Výstupem tohoto modulu je tedy největší hodnota, kterou přijal na svém vstupu, od posledního resetu nebo od začátku operace. Tato hodnota je k dispozici na výstupu a také se posílá přes AXI GPIO na UART.

3.5.6 Výpočet průměrné hodnoty

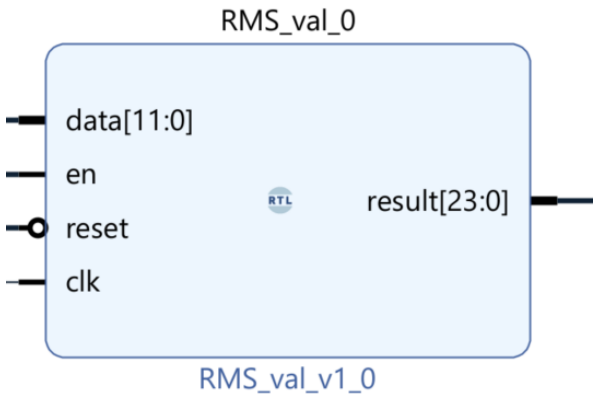
Obr. 3.10: Blok pro detekci průměrné hodnoty

Tento modul je navržen tak, aby vypočítával průměr hodnot, které přijímá na svém datovém vstupu. Jeho hlavní operace lze shrnout následovně:

1. Na počátku, nebo když je reset aktivován, jsou proměnné result, sum a count nastaveny na nulu. Tím se zajistí čistý start pro výpočet průměru.
2. Při každé náběžné hraně hodinového signálu clk, modul zkontroluje, zda je povolen. Pokud není povolen (en je neaktivní), žádné další akce se neprovádějí.
3. Pokud je modul povolen (en je aktivní), dojde k přičtení aktuální hodnoty data k současné hodnotě sum a inkrementaci count. To umožňuje současně udržovat celkový součet a počet vstupních hodnot.
4. Následně je vypočítán průměr tím, že se hodnota sum(celková hodnota) vydělí hodnotou count(počet hodnot) a výsledek se uloží do result.
5. Tento postup se opakuje pro každou náběžnou hranu hodinového signálu, pokud je modul povolen, což umožňuje modulu vypočítávat a udržovat průměr ze série vstupních hodnot.

Výstupem tohoto modulu je tedy průměrná hodnota ze série hodnot, které přišly na jeho vstup od posledního resetu nebo od začátku operace. Tato hodnota je dostupná na výstupu result.

3.5.7 Výpočet RMS



Obr. 3.11: Blok pro výpočet RMS hodnoty

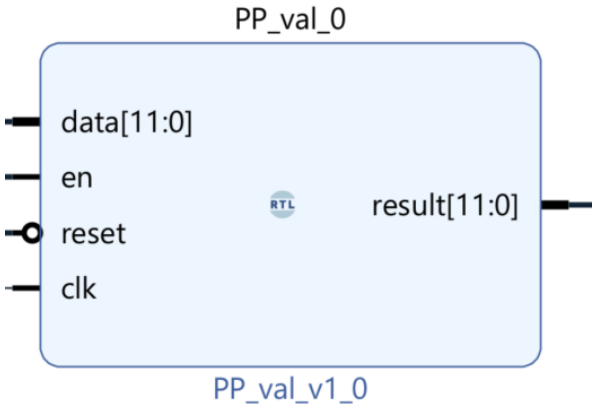
Modul RMS_val je navržen tak, že vypočítává průměr čtverců hodnot, které jsou přijaty na vstupu data. Toto pak dává na výstup result. Tato hodnota se poté používá k výpočtu RMS na procesoru, kde je v jazyce C dostupná odmocnina a je tedy dokončen výpočet RMS. Bylo nutné to takto vyřešit jelikož Verilog nemá vestavěnou funkci pro výpočet druhé odmocniny

Stručný popis funkce modulu:

1. Inicializace nebo reset nastaví sum_squared (součet čtverců) a count(počet hodnot) na nulu.
2. Při každé náběžné hraně hodinového signálu se zkontroluje, zda je modul povolen (en je aktivní). Pokud není povolen, neděje se nic.
3. Pokud je modul povolen, pak se inkrementuje počet hodnot a ke sumě se přičte čtverec aktuální hodnoty data.
4. Tento proces pokračuje s každou náběžnou hranou hodinového signálu, dokud je modul povolen.
5. Na výstupu result je poskytnut průměr čtverců vstupních hodnot, tedy hodnota sum_squared dělená hodnotou count. Tato hodnota je připravena k dalšímu zpracování, konkrétně k výpočtu druhé odmocniny, která vede k výsledné RMS hodnotě.

Tedy, výstupem tohoto modulu je průměr čtverců vstupních hodnot, který je poté použit pro výpočet RMS hodnoty v další části systému.

3.5.8 Výpočet Peak to Peak



Obr. 3.12: Blok pro výpočet Peak to Peak hodnoty

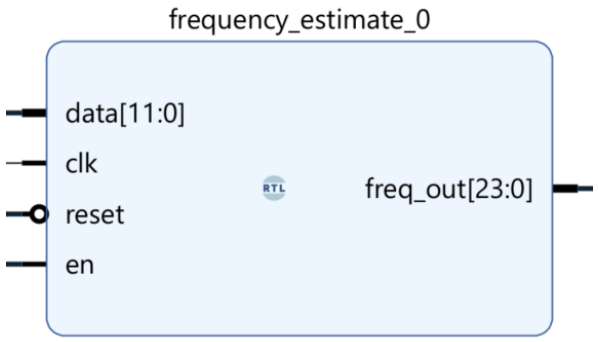
Modul PP_val (Peak to Peak) je navržen pro sledování a udržování maximální a minimální hodnoty vstupních dat. Výstupem tohoto modulu je rozdíl mezi maximální a minimální hodnotou, což představuje hodnotu peak to peak.

Jeho funkce lze popsat následovně:

1. Při inicializaci, nebo když je aktivován reset, jsou max (maximální hodnota) a min (minimální hodnota) nastaveny na své extrémní hodnoty. Max je nastaven na 0 a min na 4096. To znamená, že jakákoli následující hodnota, co příjde na vstup, bude větší než max a menší než min.
2. Při každé náběžné hraně hodinového signálu clk modul zkontroluje, zda je povolen. Pokud není povolen (en je neaktivní), tak se žádná další akce se neprovádí.
3. Pokud je modul povolen (en je aktivní), kontroluje se, zda je aktuální hodnota data větší než aktuální maximální hodnota uložená v max, nebo menší než aktuální minimální hodnota uložená v min. Pokud ano, max nebo min se aktualizují na tuto novou hodnotu.
4. Tento proces se opakuje pro každou náběžnou hranu hodinového signálu, dokud je modul povolen, což umožňuje modulu sledovat a udržovat aktuální maximální a minimální hodnotu série vstupních hodnot.

Výstupem tohoto modulu je tedy rozdíl mezi největší a nejmenší hodnotou, kterou kdy přijal na svém vstupu, od posledního resetu nebo od začátku operace. Tato hodnota je k dispozici na výstupu result a představuje hodnotu peak to peak

3.5.9 Čítač frekvence



Obr. 3.13: Blok pro čítání frekvence

Modul frequency_estimate je navržen pro odhad frekvence signálu na základě vstupních dat z ADC. Tento modul funguje na principu detekce nulových průchodů, kdy počítá počet vzorků mezi dvěma po sobě jdoucími nulovými průchody signálu.

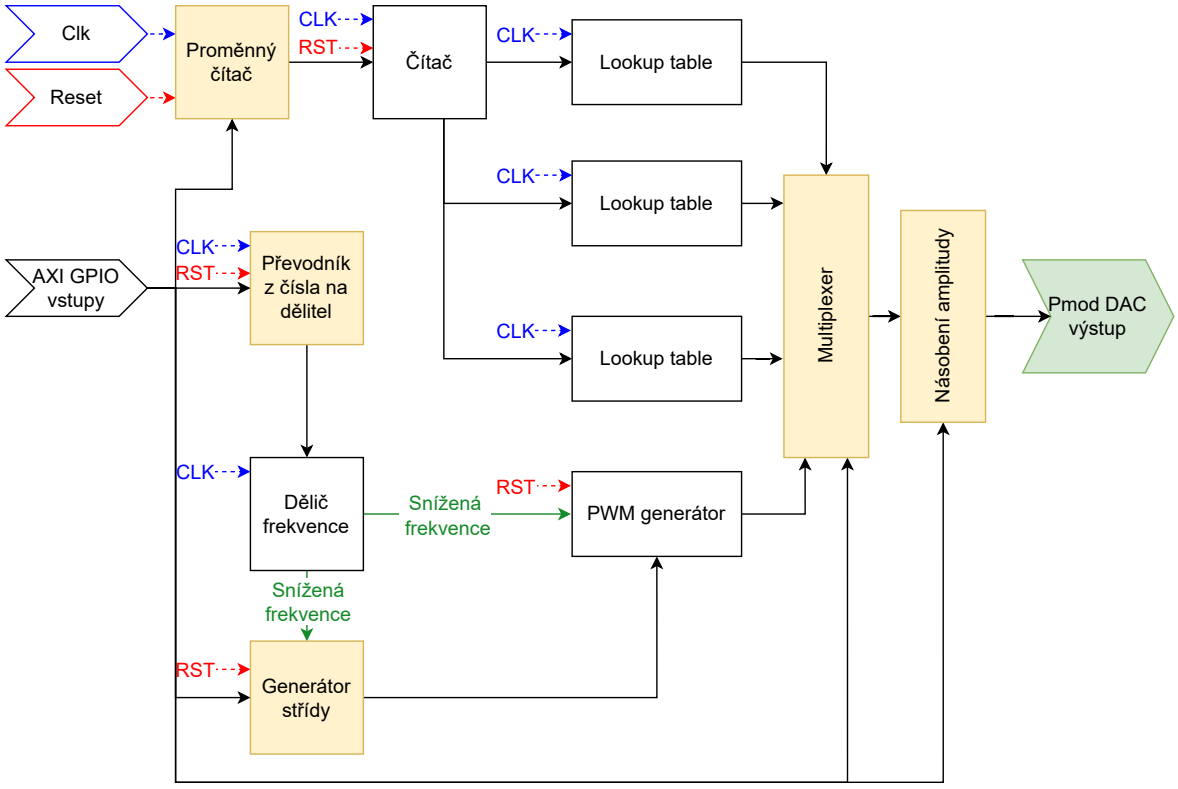
Podrobný popis funkce modulu:

1. Inicializace nastává buď při prvním spuštění, nebo když je aktivováno resetování. Poslední vzorek, čítače nulových průchodů, periody a frekvence jsou všechny nastaveny na nulu.
2. S každou náběžnou hranou hodinového signálu se prověřuje, zda je modul povolen pomocí signálu en. Pokud není povolen, žádná další akce není vykonávána.
3. Když je modul aktivní, kontroluje se, zda aktuální vzorek dat překročil střední hodnotu (2048 pro 12bitový vzorek) odspodu, což by indikovalo vzestupný průchod nulou. Pokud ano, frekvence je vypočtena jako reciproká hodnota počtu vzorků mezi dvěma nulovými průchody, což je frekvence ve vzorcích na periodu. Tato frekvence je pak vynásobena vzorkovací frekvencí (1 MHz v tomto případě), což dává frekvenci v hertzích. Poté je čítač nulových průchodů resetován na nulu.
4. Tento proces pokračuje pro každou náběžnou hranu hodinového signálu, dokud je modul povolen, což umožňuje modulu sledovat a udržovat aktuální frekvenci vstupních dat.

Výstupem tohoto modulu je tedy odhadovaná frekvence vstupního signálu, která je k dispozici na výstupu freq_out.

3.6 Návrh signálového generátoru

Blokové schéma signálového generátoru je zobrazeno na obrázku 3.14. Komponenty, které jsou ovládány prostřednictvím UART z uživatelského rozhraní vizualizačního programu v LabView, jsou zvýrazněny žlutě.



Obr. 3.14: Zjednodušené principiální schéma implementovaného generátoru signálu

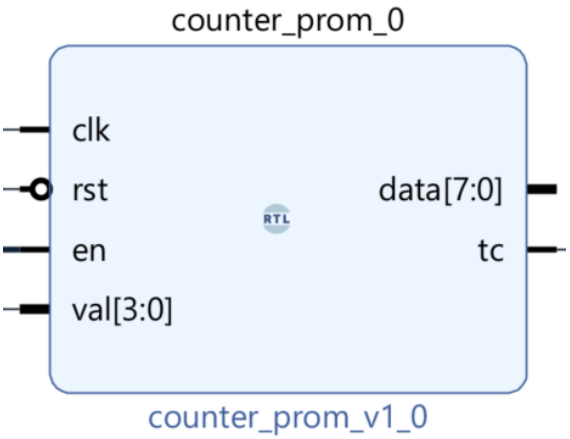
První blok, proměnný čítač, určuje frekvenci výstupního signálu tím, že určuje, jak často se změní index v look-up tabulce. Druhý čítač pak slouží k procházení adres v look-up tabulce, čímž určuje, která hodnota signálu se má v daný čas použít. Tři look-up table bloky načítají 256 hexadecimálních hodnot uložených v RAM paměti desky (sinus, trojúhelník a pila). Výstupní hodnoty z look-up tabulky jsou poté předány do bloku, kde se signál násobí podle zadané amplitudy. Výstup z tohoto bloku je pak odeslán na Pmod DAC modul, který převádí digitální hodnoty na analogový signál.

Z LabView, také nastavujeme požadovanou frekvenci od 1kHz do 500kHz s logaritmickými skoky. Toto nastavení mění časování prvního čítače a tím se mění výsledná frekvence.

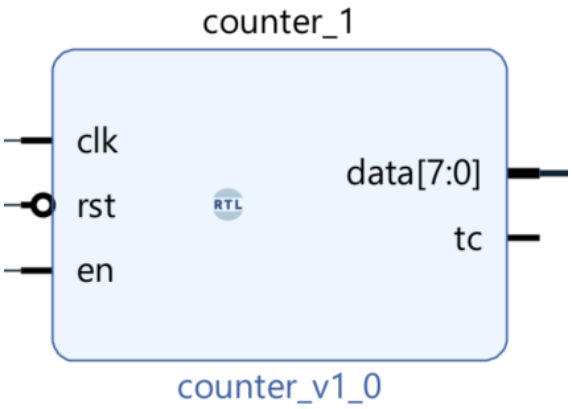
Frekvenci a střídu PWM generátoru lze rovněž měnit prostřednictvím LabView. Pro jednoduchost ovládání bylo zvoleno pět hodnot pro střídu: 0%, 25%, 50%, 75%

a 100%. Přestože by bylo technicky možné přidat více možností pro nastavení střídy, výsledné ovládání by se stalo zbytečně komplexním.

3.6.1 Čítače



Obr. 3.15: Blok proměnného čítače



Obr. 3.16: Blok čítače

Čítače jsou základem pro generování signálu. Každý čítač má vstup CLK a RST. První čítač má kromě toho vstup na ovládací hodnotu a výstup k ovládání druhého čítače. Druhý čítač má mimo CLK a RST vstupů, ještě vstup na povolení inkrementace a datový výstup.

První čítač (proměnný), přijímá čtyřbitovou hodnotu z inkrementálního čítače, která určuje maximální hodnotu do které se proměnný čítač inkrementuje. Tato maximální hodnota určuje, jak rychle se čítač inkrementuje a kdy dosáhne své maximální hodnoty. Jakmile dosáhne proměnný čítač své maximální hodnoty, tak na výstup dá logickou 1.

Tato 1 je přivedena na vstup druhého čítače s velmi podobným kódem. Ten se inkrementuje do 255 a má 8bitový datový výstup. Další inkrementace způsobí přetečení a čítač se vrátí na hodnotu 0. Tento čítač se bude inkrementovat pouze tehdy, když na svém vstupu povolení dostane logickou 1 od prvního čítače. Pokud je signál enable nastaven na logickou 0, čítač zůstane ve svém současném stavu a nebude inkrementovat. Tímto způsobem první čítač ovládá, kdy se druhý čítač bude inkrementovat a tím ovlivňuje celkovou frekvenci výstupního signálu.

Na osmibitovém datovém výstupu budou hodnoty od 0-255. Tyto hodnoty dále jdou do look-up table bloku.

Proměnnost frekvence zajišťuje proměnná maximální hodnota do které se bude inkrementovat první čítač. Toto časování můžeme vyjádřit následujícím vzorcem:

$$T_{promnenny\_citac} = \frac{f_{zakladni}}{f_{cilova} * T_{citac}}.$$

Víme že:

- referenční frekvence Zybo Z7 je 125 MHZ
- perioda čítače je 255 (osmibitový výstup)
- za cílovou frekvenci můžeme dosadit např. 5kHz

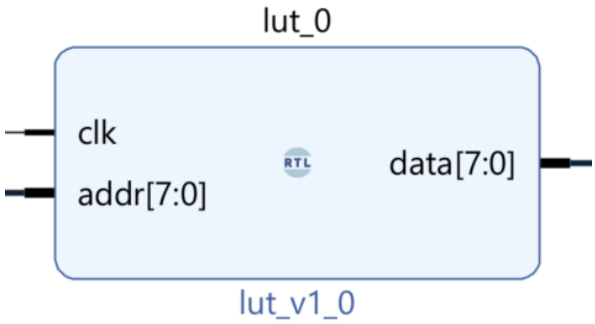
Když poté dosadíme do rovnice, tak nám vyjde časování pro proměnný čítač:

$$T_{promnenny\_citac} = \frac{125MHz}{5kHz * 255} \approx 98.$$

Z tohoto tedy plyne, že pro frekvenci signálu 5kHz je nutné, aby se první čítač inkrementoval do 98.

Takto jde frekvence signálu od 1kHz až do cca. 500kHz. U vyšších frekvencí, ale začíná být maximalní inkrementační hodnota velmi nízká (u 500kHz je to 1), a tento přístup tak naráží na svůj limit.

3.6.2 Look-up table



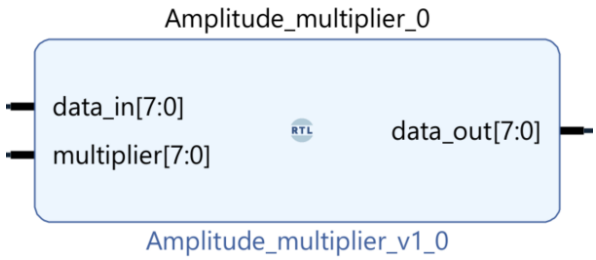
Obr. 3.17: Look-up table blok

Blok look-up table je navržen pro ukládání a načítání dat z paměti. Má CLK vstup, adresní vstup (addr) a datový výstup (data). Nepotřebuje RST vstup, jelikož je přímo vázán na čítače které při stisknutí RST tlačítka přestanou pracovat. Činnost modulu je následující:

1. **Inicializace:** Blok načte data z externího souboru .hex do RAM. Soubor musí mít 256 položek, v hexadecimálním tvaru.
2. **Vstupy:** Blok přijímá dva vstupy - CLK (pro synchronizaci činnosti s jinými bloky) a adresu, která je použita k vyhledání konkrétní uložené hodnoty v souboru.
3. **Vyhledávání hodnoty:** Na základě vstupní adresy modul vyhledá odpovídající hodnotu v souboru.
4. **Výstup:** Nalezená hodnota je poslána na výstup modulu.

Výstup se poté posílá na násobení signálu a osmibitový DAC Pmod, který převede diskrétní hodnoty na spojitý signál.

3.6.3 Násobič amplitudy



Obr. 3.18: Blok pro násobení amplitudy

Blok násobiče amplitudy je navržen k regulaci amplitudy vstupního signálu podle příchozích dat z LabView. Má tři vstupy: data_in přijímá vstupní signál, multiplier definuje koeficient násobení a CLK synchronizuje činnost modulu. Výstupem modulu je upravený signál data_out. Modul nevyžaduje vstup RST, jelikož žádný jeho stav není potřeba resetovat.

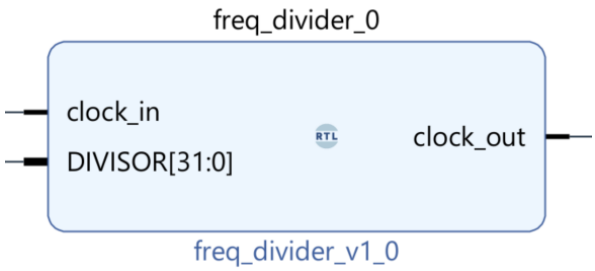
Činnost modulu je následující:

1. **Vstupy:** Blok přijímá tři vstupy - data_in (vstupní signál k upravení), multiplier (koeficient násobení, který definuje míru změny amplitudy signálu) a CLK (pro synchronizaci činnosti s ostatními bloky).
2. **Výpočet násobku:** Blok používá vstup multiplier k určení koeficientu násobení. Předpokládá se, že hodnota multiplier je v rozsahu [0,3]. Podle hodnoty multiplier je vybrán odpovídající koeficient násobení. Pro multiplier = 3 je koeficient násobení 1,0, pro multiplier = 2 je to 0,6 a pro multiplier = 1 je to 0.3.
3. **Násobení signálu:** Blok násobí vstupní signál hodnotou získanou z předchozího kroku. Pokud je výsledek větší než maximální povolená hodnota, je výsledek omezen na tuto maximální hodnotu.
4. **Výstup:** Upravený signál je poslán na výstup modulu.

Důležitou poznámkou je, že kvůli omezením Verilogu v práci s desetinnými čísly je v tomto modulu použita normalizace hodnot. To znamená, že koeficienty násobení jsou převedeny na celá čísla, přičemž zachovávají stejné relativní poměry mezi sebou. Díky tomu může modul efektivn

Blok je je díky AXI sběrnici ovládán přes UART, a tím pádem i přes LabVIEW.

3.6.4 Dělič frekvence



Obr. 3.19: Blok pro dělení frekvence

Jelikož PWM generátor nepracuje na principu look-up table, tak bylo nutné vymyslet jak pomocí inkrementačního čítače měnit frekvenci u PWM signálu. Nejvhodnějším řešením je implementace děliče frekvence. Vstupy do bloku jsou CLK a dělitel frekvence. Výstupem je nižší frekvence.

V kódu se neprovádí explicitní dělení vstupního hodinového signálu. Místo toho se frekvence vstupního hodinového signálu efektivně dělí pomocí čítače a porovnávání s hodnotou dělitele.

Algoritmus pracuje následovně:

1. Čítač se zvyšuje o jedna, při každém náběžném hraně vstupního hodinového signálu.
2. Jakmile hodnota čítače dosáhne hodnoty (dělitel - 1), je čítač resetován na nulu
3. Výstupní hodinový signál je nastaven na logickou 1, pokud je hodnota čítače menší než polovina hodnoty dělitele. Pokud je hodnota čítače větší nebo rovna polovině hodnoty dělitele, je nastaven na logickou 0.

Pokud máme vstupní hodinový signál s frekvencí 125 MHz a chceme generovat výstupní hodinový signál s frekvencí 1 kHz, budeme potřebovat nastavit hodnotu dělitele tak, aby odpovídala poměru mezi těmito frekvencemi.

V tomto případě nastavíme hodnotu dělitele na 125 000, protože 125 MHz / 1 kHz = 125 000. Tímto způsobem dělíme frekvenci vstupního hodinového signálu 125 000krát.

1. Čítač začíná na hodnotě 0.
2. Při každé náběžné hraně vstupního hodinového signálu se hodnota čítače zvýší o 1. V tomto případě to znamená, že čítač bude zvýšen 125 milionkrát za sekundu.
3. Jakmile čítač dosáhne hodnoty (dělitel - 1), což je 124 999, čítač se resetuje na 0.

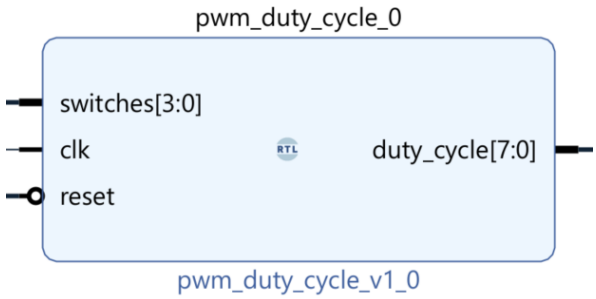
4. Výstupní hodinový signál je nastaven na logickou 1, pokud je hodnota čítače menší, než polovina hodnoty dělitele (62 500). Pokud je hodnota čítače větší nebo rovna 62 500, je nastaven na logickou 0.

Čítač projde hodnotami 0 až 124 999 a pak se vrátí na 0. To se stane 1 000krát za sekundu, což dělí frekvenci vstupního hodinového signálu (125 MHz) 125 000krát. Výsledkem je výstupní hodinový signál s frekvencí 1 kHz.

Před děličem frekvence je blok převodníku, který přiřadí pro každému výstupu inkrementačního čítače dělitel, který slouží jako vstup do deliče frekvence.

LabVIEW program automaticky přiřazuje dělitele na základě výběru frekvence. Tato hodnota je odeslána přes UART do procesoru, který následně přenese hodnotu přes AXI sběrnici do FPGA. Díky tomu je dělič správně nastaven podle požadované frekvence.

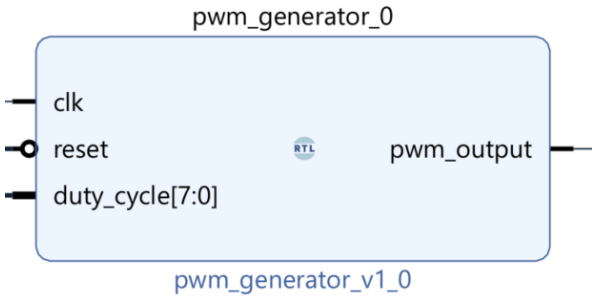
3.6.5 Generátor střídy



Obr. 3.20: Blok pro výpočet Peak to Peak hodnoty

Tento blok je zodpovědný za převod čtyřbitového čísla, představujícího polohu přepínačů, na osmibitové číslo, které reprezentuje různé úrovně střídy PWM signálu. Vstupní čtyřbitové číslo je získáno z konfigurace čtyř přepínačů. Výstupní osmibitové číslo, které reprezentuje střídu PWM signálu, se poté použije v následujícím řízeném bloku pro generování PWM signálu s odpovídající střídou.

3.6.6 PWM Generátor



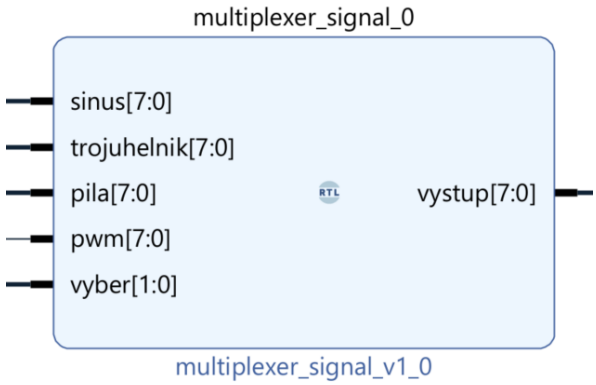
Obr. 3.21: Blok pro generování PWM signálu

PWM generátor pracuje na principu srovnávání hodnoty osmibitového čítače s hodnotou střídy, která je také osmibitová. Čítač se inkrementuje s každou hranou hodinového signálu. Proces je následující:

1. Čítač je inicializován na nulu a inkrementuje s každou hranou hodinového signálu.
2. Hodnota čítače je porovnávána s hodnotou střídy.
3. Pokud je hodnota čítače menší než střída, výstup PWM signálu je nastaven na logickou 1.
4. Pokud je hodnota čítače větší nebo rovna střídě, výstup PWM signálu je nastaven na logickou 0.

Tento proces se opakuje v každém cyklu hodinového signálu.

3.6.7 Signálový multiplexer



Obr. 3.22: Blok signálového multiplexeru

Modul multiplexer_signal je 4-to-1 multiplexer, který přijímá čtyři 8bitové vstupní signály: sinus, trojuhelnik, pila a PWM. Dalším vstupem je výběr, to je 2bitový signál sloužící k výběru požadovaného vstupního signálu. Výstupní signál vystup je 8bitový a reprezentuje vybraný vstupní signál.

Výběr signálu je řízen procesorem pomocí AXI sběrnice. Uživatel si nejprve vybere typ signálu na virtuálním přístroji v LabVIEW. Typ signálu je přiřazen určité hodnotě, která je následně odeslána přes UART. Poté procesor přenese tuto hodnotu přes AXI sběrnici do FPGA.

V bloku je definován kombinační proces, který se vykonává vždy, když dojde ke změně některého ze vstupních signálů. V bloku case je použit case statement pro výběr vstupního signálu na základě hodnoty signálu vyber.

1. Při hodnotě 2'b00 je vybrán signál sinus.
2. Při hodnotě 2'b01 je vybrán signál trojuhelnik.
3. Při hodnotě 2'b10 je vybrán signál pila.
4. Při hodnotě 2'b11 je vybrán signál PWM.

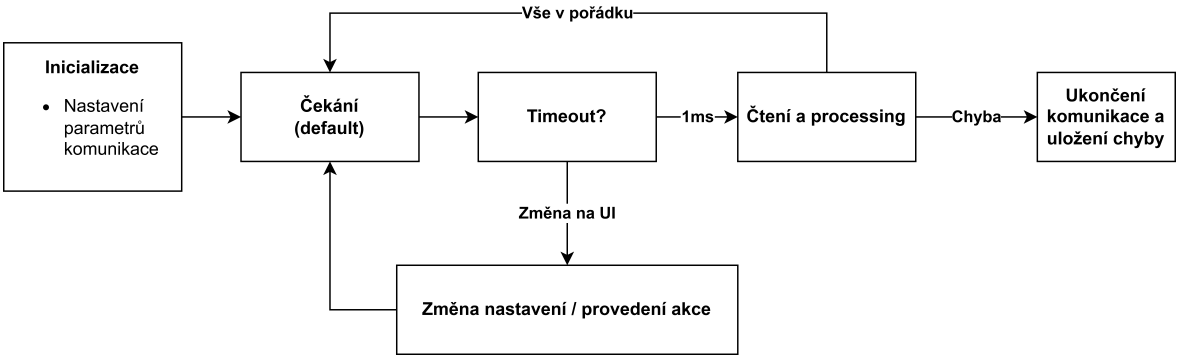
V případě, že hodnota signálu vyber neodpovídá žádnému ze specifikovaných případů, je vybrán signál sinus jako výchozí nastavení.

4 Zobrazovací část

Pro zobrazování byl vybrán program LabVIEW, díky jeho modularitě, zabudovanému zobrazování a dostupnému ovládání. Výhodou je také prostředí NI VISA (2.3.6), které usnadňuje UART komunikaci.

4.1 Program pro zobrazování v LabView

Program v LabView funguje podle principu stavového automatu, jehož zjednodušené schéma je k vidění na obrázku 4.1. Po aktivaci se program přesune do inicializačního stavu, kde se definují parametry UART komunikace. Následně program přechází do základního stavu, kde po dobu 1ms očekává nějakou aktivitu. Pokud v daném časovém intervalu nedojde k žádné interakci uživatele s uživatelským rozhraním, program se periodicky pohybuje mezi čtením a zpracováním dat a čekáním 1ms. Jakmile uživatel s uživatelským rozhraním interaguje, program vykoná požadovanou akci a následně se vrátí do stavu čekání.



Obr. 4.1: Zjednodušený vývojový diagram LabView programu

V současné chvíli má program následující možnosti:

- FFT analýzu.
- Výběr autoscale/manuální nastavení.
- Export grafu do PNG.
- Nastavení vizualních vlastností grafu (šířka, barva stopy, barva pozadí).
- Možnost přednastavených rozsahů (0 - 1V, ±2 V, ±20 V)
- Export dat do excelu.

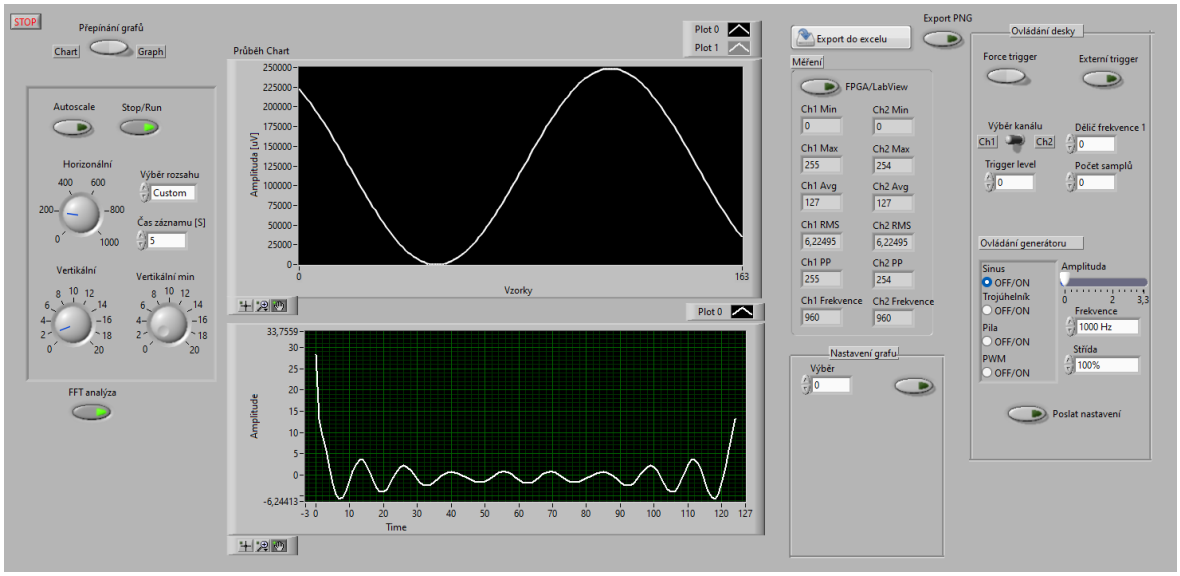
Vzhledem k rozsáhlému charakteru programu, budou v následujících podkapitolách popsány především klíčové segmenty programu.

4.2 Virtuální insturment

Frontpanel LabView programu je zachycen na obrázku 4.2. slouží k zobrazení na vzorkovaných signálů, data příchází z programu na procesoru. Dále také frontpanel slouží k nastavování parametrů jako jsou nastavení triggeru, počet vzorků, dělení frekvence a parametry generátoru signálů.

Ovládání je následující:

- Funkce Autoscale analyzuje data zobrazená na grafu a automaticky přizpůsobí měřítkové rozsahy tak, aby všechna data byla viditelná na grafu.
- Tlačíto "Stop/Run"zastavuje čtení průběhu.
- Výběr horizontální nastavuje délku záznamu.
- Výběr vertikální min a max nastavuje vertikální rozsah.
- Výběr rozsahu jde vybrat i ze seznamu, kde je na výběr 0 - 1V, ±2 V, ±20 V
- Pokud tlačítko "Run/Stop"není stisknuto a je vybrána délka záznamu, spustí se sběr dat po určitou dobu.
- Stisknutím tlačítka "FFT analýza"se spustí FFT analýza.
- Pod grafy jsou ovládací prvky na zoom, posouvání v historii a další manipulaci.



Obr. 4.2: UI (front panel) programu pro zobrazování

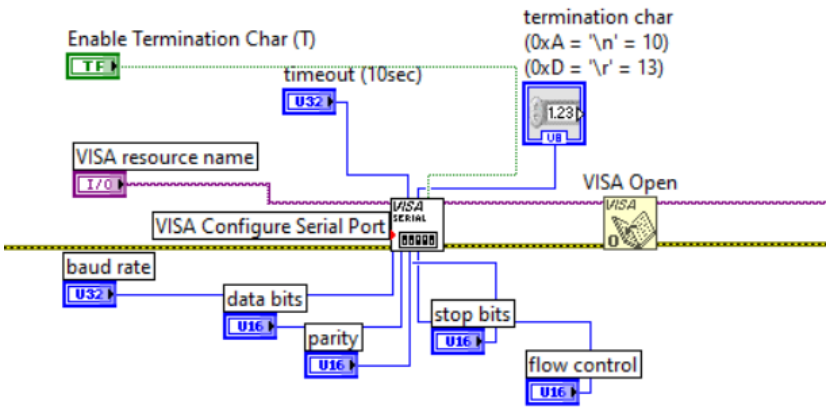
Ovládat/nastavit se dá:

1. vynucení triggeru,
2. hodnota trigger levelu,
3. externí trigger,
4. dělení frekvence,
5. počet vzorků,
6. výber kanálu,
7. prokládání kanálů,
8. výběr typu signálu na výstupu generátoru,
9. amplituda výstupního signálu,
10. frekvence výstupního signálu,
11. u PWM signálu jeho střída,

- V panelu napravo jsou měření parametrů signálu z DSP modulu na FPGA.
- Nad panelem jsou tlačítka pro export grafu do formátu png a Excelu.
- Úplně vpravo je panel pro ovládání parametrů generátoru a vzorkovaní.

4.3 Inicializace a příjem UART komunikace

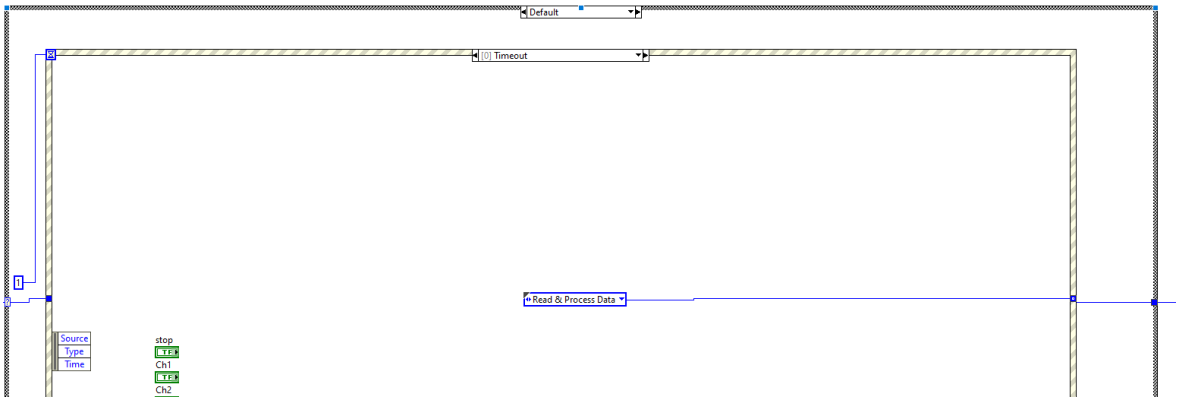
Iniciační stav, ilustrovaný na obrázku 4.3, predstavuje nastavení komunikace UART. V tomto bodě jsou definovány klíčové parametry, jako je COM port zařízení, baud rate a další, které je nutné synchronizovat s blokem Zynq7 Processing System na FPGA pro správné čtení a synchronizaci dat. Blok VISA Open otevírá COM port pro komunikaci. Významnou roli zde hraje také žlutý spoj reprezentující chybový výstup, který je nezbytný pro efektivní zaznamenávání chyb.



Obr. 4.3: Inicializace a příjem UART komunikace v LabView programu pro zobrazování

4.4 Čekání na operaci

Jakmile je inicializace dokončena, program přechází do výchozího stavu, který je charakterizován čekáním. Po dobu 1ms je program v režimu standby, pokud v tomto časovém intervalu nedojde k žádné interakci ze strany uživatele s uživatelským rozhraním (UI). Pokud k takové interakci nedojde, program pokračuje další fází, která spočívá v čtení a zobrazování dat, jak je zřejmé na obrázku 4.4.



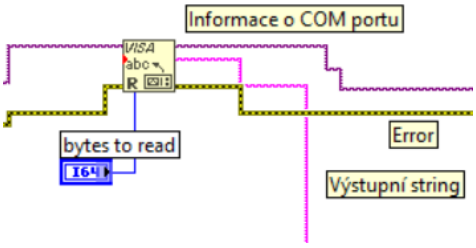
Obr. 4.4: Inicializace a příjem UART komunikace v LabView programu pro zobrazování

4.5 Čtení a filtrace dat

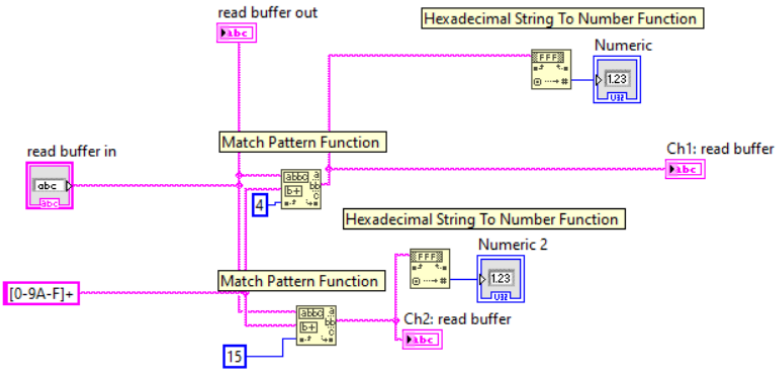
Jakmile je COM port připraven pro čtení, začíná se pomocí NI VISA rozhraní s procesem čtení dat. Tento proces realizuje blok VISA read, který je zobrazen na obrázku 4.5. Tento blok má jako parametr určený počet bytů, které mají být přečteny.

Výstupní string je následně předán do Sub VI bloku pro filtraci. V rámci tohoto bloku je pomocí funkce Match Pattern Function z původního stringu vyfiltrována pouze relevantní data, jako jsou amplituda, frekvence, minimum, maximum a další informace. Podrobnější informace o tomto procesu jsou k dispozici v kapitole 3.5.

Protože data jsou přijímána v hexadecimálním formátu, je nezbytné je převést na formát celých čísel. K tomuto účelu je použita funkce Hexadecimal String To Number Function. Proces filtrace dat z jednotlivých kanálů je znázorněn na obrázku 4.6.



Obr. 4.5: Příjem dat přes VISA read



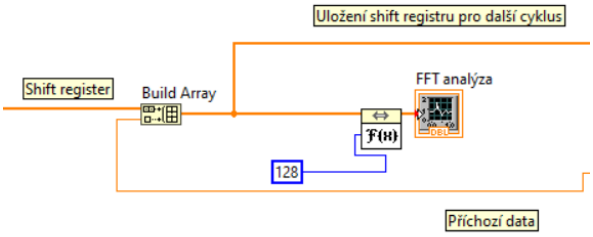
Obr. 4.6: Filtrace příchozích dat

4.6 Další využité vnitřní funkce LabVIEW

V návrhu bylo použito mnoho vnitřních funkcí LabVIEW, následuje výčet těch nejpodstatnějších.

4.6.1 Rychlá Fourierova transformace v LabVIEW

LabVIEW má několik bloků pro výpočet FFT. Pro výpočet a zobrazení FFT z navzorkovanéhosignálu byl zvolen FFT z knihovny signal processing. Zapojení je na obrázku 4.7. Aby nebyl program zbytečně zatěžován, tak byla zvolená délka FFT 128 vzorků.



Obr. 4.7: Použité zapojení pro zobrazení FFT v LabVIEW

4.6.2 Délka záznamu v LabVIEW

Pro ovládání délky záznamu byla využita funkce Elapsed Time na obrázku 4.8. Vstupem do ní je hodnota v sekundách, výstupem je pak hodnota 1 nebo 0. Podmínka je zda uběhl zadaný čas, pokud ano tak funkce vrací logickou 1.

Obr. 4.8: Funkce Elapsed Time v LabVIEW

Dále jsou použity hlavně ovládací prvky, pole, smyčky a další. Celý program je v k dispozici v digitální příloze.

4.7 Výběr zobrazení

V prostředí LabVIEW se pro zobrazování dat často používají dva typy grafů: Waveform Graph a Waveform Chart. Tyto dva typy grafů jsou na první pohled velmi podobné, ale mají několik klíčových rozdílů, které ovlivňují, jak jsou používány pro zobrazování dat.

Waveform Graph

Waveform Graph je ideální pro zobrazování dat, která jsou již kompletně k dispozici. Tento typ grafu přijímá pole dat jako vstup a zobrazí je všechny najednou. To je

ideální pro situace, kdy je důležité vidět všechna data současně, například pro analýzu dat z osciloskopu.

Waveform Chart

Na druhou stranu, Waveform Chart je navržen pro zobrazování dat v reálném čase. Tento typ grafu přijímá jednotlivé hodnoty dat a postupně je přidává na konec grafu. Waveform Chart automaticky upravuje osu x (časovou osu) tak, aby zobrazovala pouze nejnovější údaje. Tato funkce je velmi užitečná pro aplikace, kde je důležité sledovat poslední příchozí data v reálném čase, jako je například monitorování výstupu osciloskopu v daný okamžik.

Pro aktuální informace je vhodnější Waveform Chart a pro dlouhý průběh kde je třeba sledovat historii je vhodný Waveform Graph. Jejich vizuální porovnání je v příloze A.1

Pro optimální řešení bylo zvoleno přepínání mezi jednotlivými typy grafů, aby se mohl zvolit vhodný graf pro specifické použití.

Primárně ale je využíván Waveform Chart

5 Ověření funkčnosti

Ověření funkčnosti jednotlivých segmentů probíhalo nejčastěji pomocí osciloskopu či ještě v simulátoru zabudovaném ve Vivadu. Vybavení používané k měření průběhů a generování signálů je v následující tabulce 5.1.

Tab. 5.1: Seznam použitého vybavení

Typ vybavení	Model
Deska	Zybo Z7-20
Osciloskop	Rhode & Schwarz 3004
Generátor signálu	Rigol DG4162

5.1 Využití čipu Zynq 7000

V tabulce 5.2 je znázorněno využití čipu XC7Z020-1CLG400C naším návrhem. Blokové schéma výsledného návrhu je uvedeno v příloze A.4.

Tab. 5.2: Využité prostředky čipu XC7Z020-1CLG400C celým projektem.

Zdroj	Využité	Celkové	Procentuálně
LUT	4794	53200	9.011278
LUTRAM	63	17400	0.36206898
FF	4787	106400	4.49906
BRAM	6	140	4.2857146
DSP	2	220	0.9090909
IO	17	125	13.6
BUFG	2	32	6.25

- **Look-Up Tables (LUT):** Tyto základní stavební bloky FPGA jsou schopny realizovat libovolné kombinační funkce. V případě Zynq 7000 je využito 4794 z 53200 LUT, což představuje přibližně 9.01% z celkového počtu dostupných LUT. Tato relativně nízká míra využití naznačuje, že je zde dostatek prostoru pro rozšíření návrhu či optimalizaci.
- **LUTRAM (Look-Up Table RAM):** Jde o speciální typ LUT, který je konfigurován jako paměť. V našem případě je využito pouze 63 z 17400, což je 0.36%. Tato velmi nízká míra využití ukazuje velký prostor pro další využití.

- **Flip-Flopy (FF):** Tyto jednoduché jednobitové registry jsou v našem designu využity v počtu 4787 z 106400, což je přibližně 4.5%. Také zde je značný prostor pro další vývoj či optimalizaci.
- **Block RAM (BRAM):** Tato vícebitová paměť slouží pro ukládání dat. Je zde využito 6 z 140 bloků RAM, což je 4.28%. Toto je rovněž poměrně nízká míra využití.
- **Bloky pro digitální zpracování signálů (DSP):** Tyto speciální bloky jsou určeny pro zpracování digitálních signálů. V našem designu je využito pouze 2 z 220 dostupných bloků DSP, což je 0.9%. Zde je tedy velký prostor pro další DSP aplikace.
- **Vstupně/Výstupní bloky (IO):** Tyto bloky slouží pro komunikaci s vnějším světem. Je zde využito 17 z 125 dostupných IO bloků, což je 13.6%. Toto je v porovnání s ostatními zdroji relativně vysoká míra využití.
- **Globální bufery (BUFG):** Tyto speciální bloky se využívají pro distribuci hodinových signálů a resetovacích signálů po celém čipu. Je zde využito 2 z 32 dostupných BUFG, což je 6.25%.

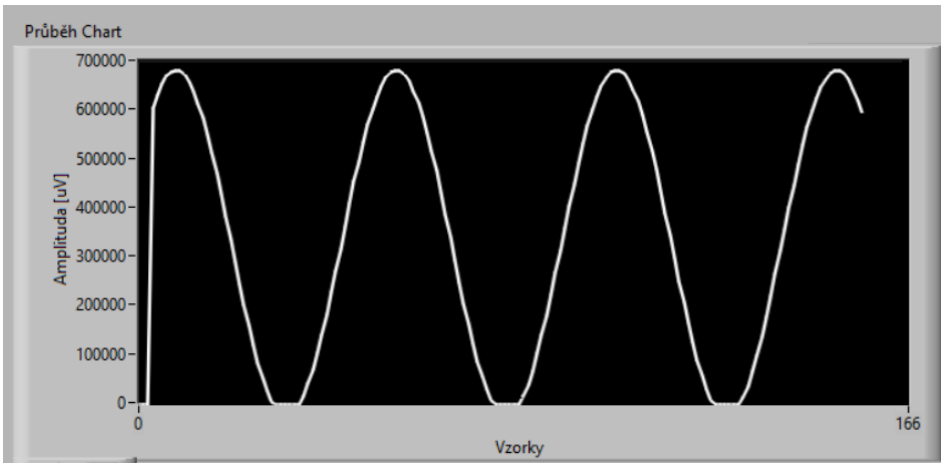
V příloze A.8 je dostupné implementační schéma celého projektu, kde je graficky znázorněno využití čipu XC7Z020-1CLG400C.

5.2 Testování aplikace pro zobrazování

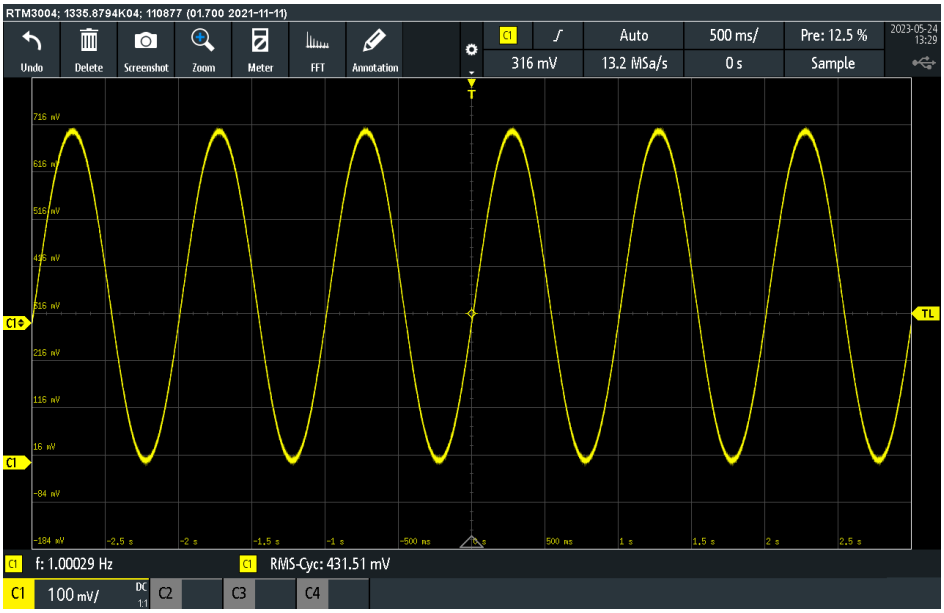
Prvně bylo nutné ověřit správnost zobrazování zachyceného signálu. V průběhu zobrazeném na 5.1 si můžeme všimnout že amplituda signálu je správná. Avšak počet vzorků je velmi nízký. Toto není chyba hardwaru, ale LabVIEW, které nestíhá zpracovávat tolik dat. Program není schopný přijímat a zobrazit data v dostatečné rychlosti, a tak je vykreslení záznamu značně opožděno o proti reálnému snímání.

Na obrázku 5.1 je zobrazen sinový průběh o frekvenci 1 Hz a amplitudě 700 mV. Na obrázku 5.2 je týž signál na osciloskopu Rohde & Schwarz RTM3004.

Více zobrazených signálů v LabVIEW aplikaci je v příloze A.6.

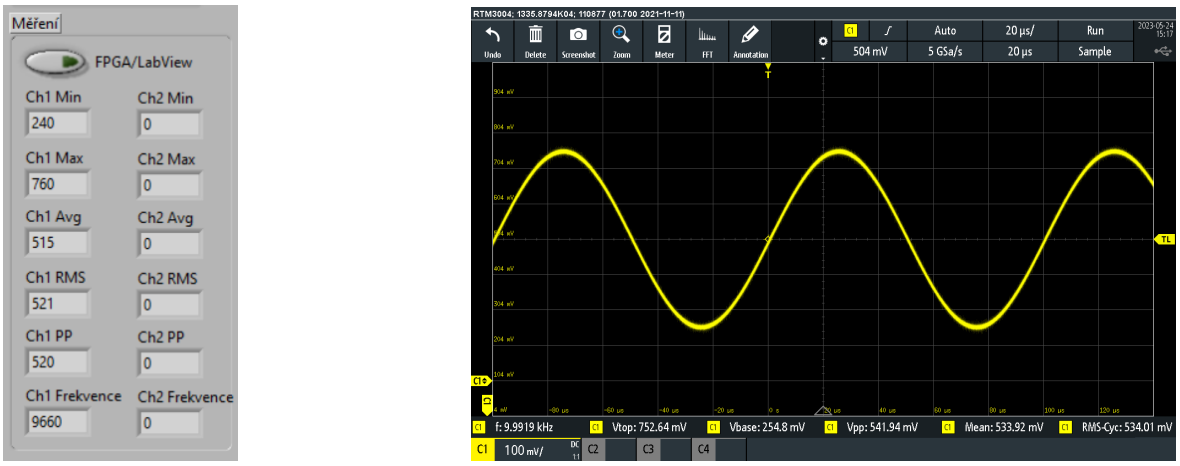


Obr. 5.1: Sinový průběh o frekvenci 1 Hz na frontpanelu v LabVIEW



Obr. 5.2: Sinový průběh o frekvenci 1 Hz na osciloskopu Rhode & Schwarz RTM3004

5.3 Porovnání naměřených hodnot



Obr. 5.3: (a) Naměřené hodnoty sinusového průběhu o amplituě 700 mV a frekvenci 10 kHz z DSP modulu na FPGA, zobrazená na čelním panelu v LabVIEW, (b) Naměřené hodnoty sinusového průběhu o amplituě 700 mV a frekvenci 10 kHz na osciloskopu Rhode & Schwarz RTM3004.

Na tabulce 5.3 je vidět přehledné porovnání naměřených hodnot, vytvořený DSP modul tedy funguje dobře a v těchto parametrech je hodně blízký profesionálnímu řešení. Nevýhodou však je, že s proměnnými těmito proměnnými LabVIEW pracuje v celočíselné tj. zaokrohlené formě. Bylo implementováno i odečítání stejných parametrů přímo v LabVIEW, ale díky tomu jak LabVIEW pracuje s daty, nebylo možné přes LabVIEW odečet vůbec realizovat nebo byl velmi nepřesný.

Tab. 5.3: Porovnání naměřených hodnot sinového signálu o amplitudě 700 mV a frekvenci 10 kHz z generátoru Rigol DG4162

Parametr	DSP modul na FPGA	Osciloskop Rhode & Schwarz RTM30004
Minimum	240 mV	254 mV
Maximum	760 mV	752,43 mV
Průměrná hodnota	515 mV	533,92 mV
RMS	521 mV	534,01 mV
Peak to Peak	520 mV	541,94 mV
Frekvence	9660 Hz	9991 Hz

5.3.1 Omezení LabVIEW

Omezené možnosti zobrazování grafů: Grafy v LabVIEW mají omezené funkce a přizpůsobitelnost. To vede k omezení možností nastavení os, měřítek, škály a interaktivity s grafem.

Limitovaná výkonnost při příjmu dat: Bylo zjištěno, že aplikace v LabVIEW má velmi omezenou výkonnost při zpracování a zobrazování velkého množství dat. To vede k zpomalení načítání dat a zpoždění aktualizace grafů. Tento problém by z části mohla řešit práce s knihovnou GigaLabVIEW, která se používá pro práci s velkými datovými poli.

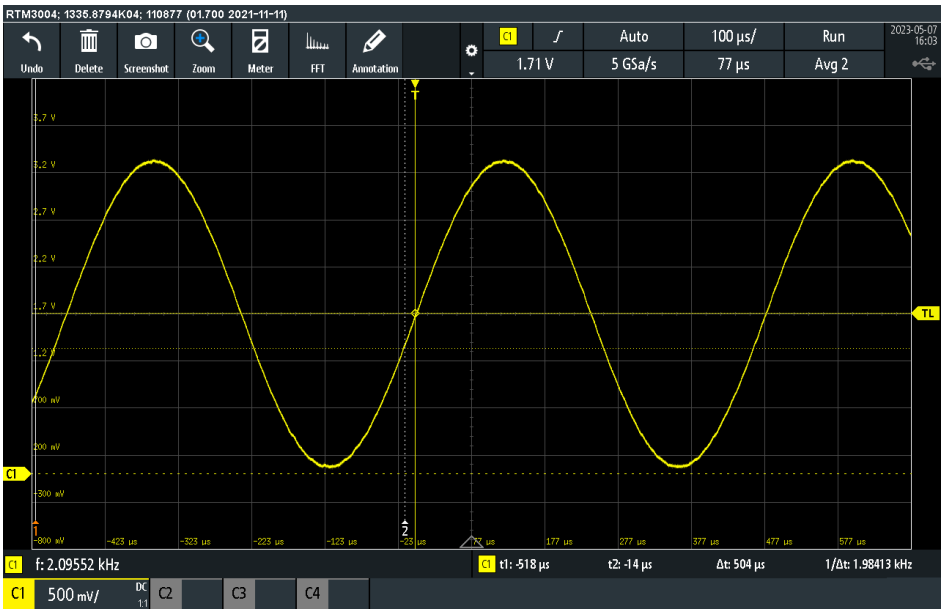
Některé realizované funkce jako například možnost zastavení grafu, velmi zpomalují celkový program ještě více. Toto ovšem může být dáno neznalostí pokročilejších funkcí programu LabVIEW.

Ostatní omezení: Některé funkce osciloskpu se nepovedlo realizovat v LabVIEW. Konkrétně šlo o převod vzorků na čas a to z důvodu pomalého běhu programu. Pro takto časově náročnou aplikaci není LabVIEW vhodným řešením, jelikož nejde získat plnou kontrolu nad celou architekturou programu. Možné vylepšení by tedy spočívalo v navržení samostatného programu, například v prostředí PyQt či samotném QT. Bylo by to ovšem mnohem náročnější na vývoj.

Také je problém s funkcí zachycení a následného replikování signálu. LabVIEW umí zachytit signál do souboru, ale pokus o replikování chování osciloskopu tímto přístupem byl neúspěšný.

5.4 Měření generátoru

U měření generátoru bylo klíčové ověření zda se zobrazuje správně požadovaný signál, vyzkoušen byl sinusový průběh, trojúhelníkový průběh a PWM.



Obr. 5.4: 2kHz sinus na osciloskopu Rhode & Schwarz RTM3004

PWM výstup je zobrazen na obrázku 5.5

Obr. 5.5: PWM signál na osciloskopu Rhode & Schwarz RTM3004, oranžový je ze Zybo Z7 a zelený je z generátoru Rigol DG4162

Více příkladů PWM generování je v příloze A.7

Zde jsou na PWM signálu z navrženého generátoru patrné přechodové jevy. Jednou z možných příčin těchto jevů je, že na Pmod výstupu desky nejsou implementovány adekvátní filtry, které by byly běžně k dispozici u komerčně dostupných generátorů

signálů. Další možností je chybějící impedanční přizpůsobení, kdy má Pmod výstup jinou impedanci než osciloskop.

Absence těchto filtrů může vést k nežádoucím harmonickým složkám a šumům v generovaném PWM signálu. Pro zajištění optimální kvality signálu a minimalizaci přechodových jevů je důležité zvážit použití dodatečných externích filtrů nebo alternativních hardwarových řešení s integrovanými filtry navrženými pro generování PWM signálu. Při návrhu SoC ZYNQ by se tento problém dal zmírnit změnou strmosti náběžné hrany

Při použití děličky frekvence pro změnu frekvence PWM signálu bylo zjištěno, že generované frekvence nejsou zcela přesné. Například, při pokusu o generování signálu s frekvencí 500 kHz byla skutečná frekvence naměřena jako 488,2 kHz. Tento rozdíl v frekvenci je způsoben omezením dělení referenční frekvence 125 MHz děličkou.

Tento problém by mohl být potenciálně řešen použitím jiných metod generování frekvence nebo optimalizací parametrů děličky frekvence. Alternativní řešení by mohlo zahrnovat použití jiných zdrojů hodinových signálů nebo PLL (Phase Locked Loop), která by umožnila přesnější a flexibilnější nastavení frekvencí.

Amplitudu signálu je možné nastavit prostřednictvím UART na hodnoty 1V, 2V a 3,3V. Pokud by byla požadována vyšší amplituda, bylo by nezbytné přidat externí napěťový zdroj a proces generování přesunout na analogovou desku, přičemž řízení by probíhalo stále přes UART.

Jedno z potenciálních vylepšení by mohlo spočívat v přidání možnosti načtení dat přes UART, a následné replikaci a násobením konstantou, což by umožnilo modifikovat amplitudu načteného signálu. Signálový generátor by tak získal možnost vytvářet, libovolný signál.

Výsledný generátor signálu má tedy následující parametry:

Tab. 5.4: Parametry generátoru signálu

Frekvenční rozsah	**Amplituda**	**Kanály**	**Rozlišení**
1 kHz - 500 kHz	0 - 3,3 V	2 (1 PWM)	1

1: Frekvenční skok generátoru signálu je navržen tak, aby byl logaritmicky rozložen. To znamená, že místo lineárního kroku po jednotlivých hertzích je frekvenční skok vytvořen pro rychlé a snadné pokrytí širokého rozsahu frekvencí. Frekvenční skoky jsou rozděleny do následujících kroků: 1 kHz, 2 kHz, 5 kHz, 10 kHz, 20 kHz, 50 kHz, 100 kHz, 200 kHz a 500kHz.

6 Závěr

V této bakalářské práci byla uvedena koncepce digitálního osciloskopu na vývojové desce Zybo Z7-20 s čipem z rodiny Zynq7000. Projekt byl rozdělen na řídicí a zobrazovací část s přidruženým generátorem a na signálovou část, které se věnoval Michal Navrátil.

Tato práce se soustředila na řídicí a zobrazovací část se signálovým generátorem. V zobrazovací části byl úspěšně realizován modul digitálního zpracování signálu pro základní měření parametrů navzorkovaného průběhu. Pro zobrazování naměřených dat a samotného signálu byl vytvořen program v prostředí LabVIEW. V tomto programu jsou dostupné základní funkce pro práci se zobrazeným signálem včetně možnosti zastavení a opětovného spuštění záznamu, nastavení vertikálního i horizontálního měřítka, FFT analýzy a spuštění záznamu po určitou dobu. Dále jsou zobrazovány naměřené parametry z modulu pro digitální zpracování signálu a umožňuje se ovládání parametrů vzorkování signálu a signálového generátoru.

Řídící část se zabývá popisem kódu pro řízení komunikace mezi FPGA a ARM Cortex procesorem, kód je také zodpovědný za UART komunikaci. Důraz je kladen na demonstraci práce s AXI sběrnicí pro komunikaci mezi FPGA a procesorem.

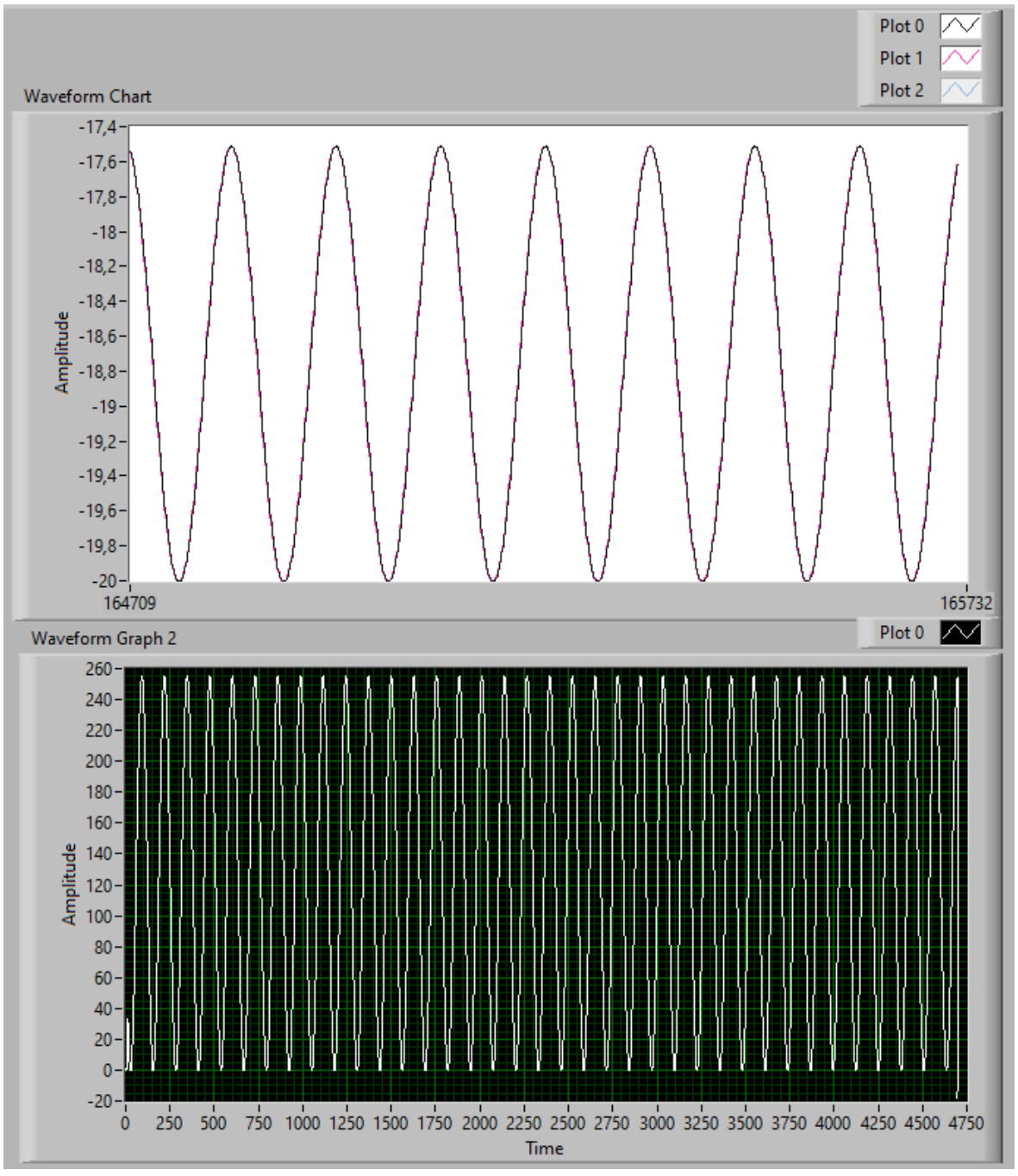
Jako poslední byl vytvořen signálový generátor se sinusovým, trojúhelníkovým, pilovým a PWM výstupem.

Po celou práci byl kladen důraz na to, aby všechny části spolu komunikovaly, a aby se maximum parametrů dalo ovládat přes zobrazovací aplikaci

Po spojení této bakalářské práce a práce Michala Navrátila vzniká tedy jednoduchý dvoukanálový osciloskop na desce ZyboZ7 a čipu z rodiny Zynq7000. Vytvořený osciloskop má šířku pásma 500kHz, vzorkovací frekvenci 500KSPS na kanál a přidružený signálový generátor se základními průběhy.

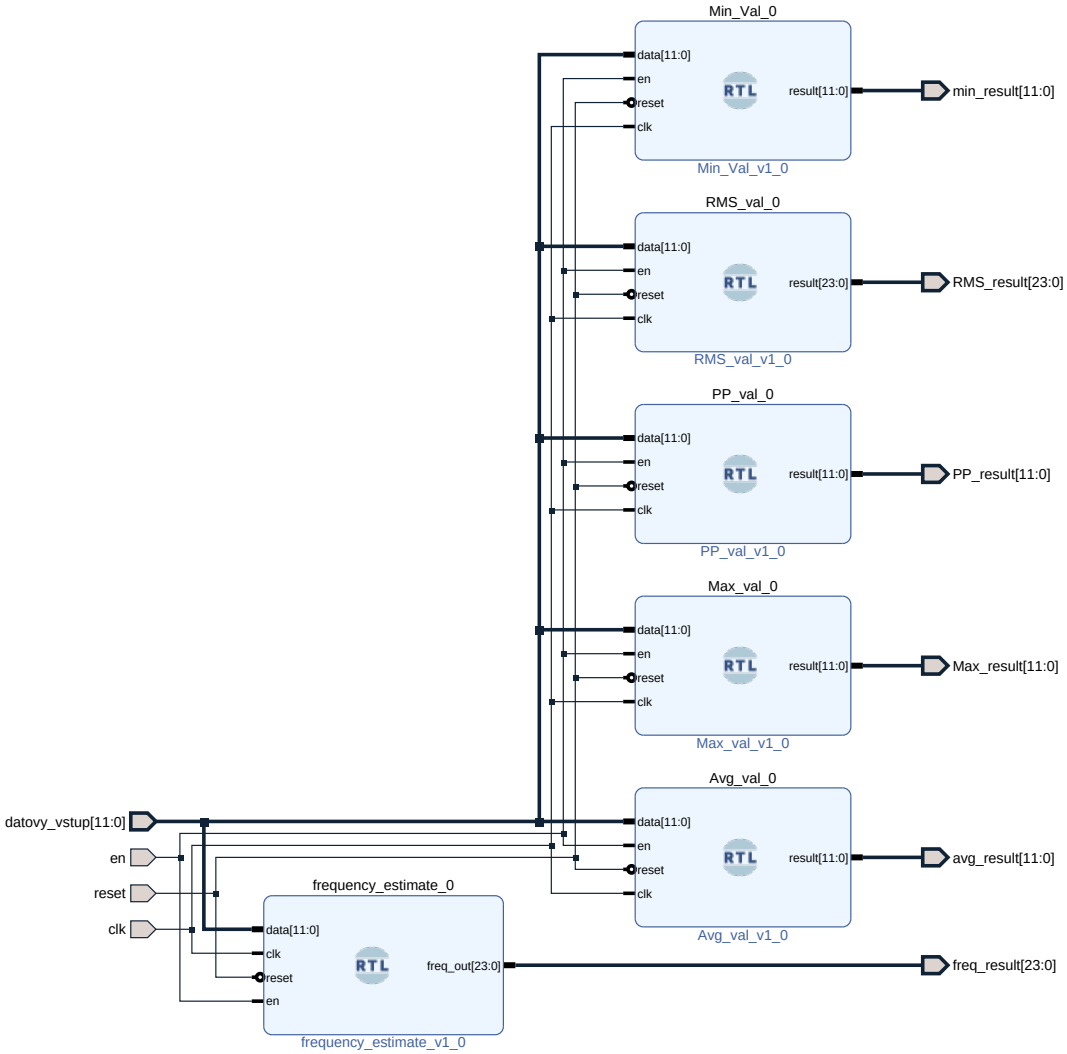
A Přílohy

A.1 Porovnání Waveform Chart a Waveform Graph

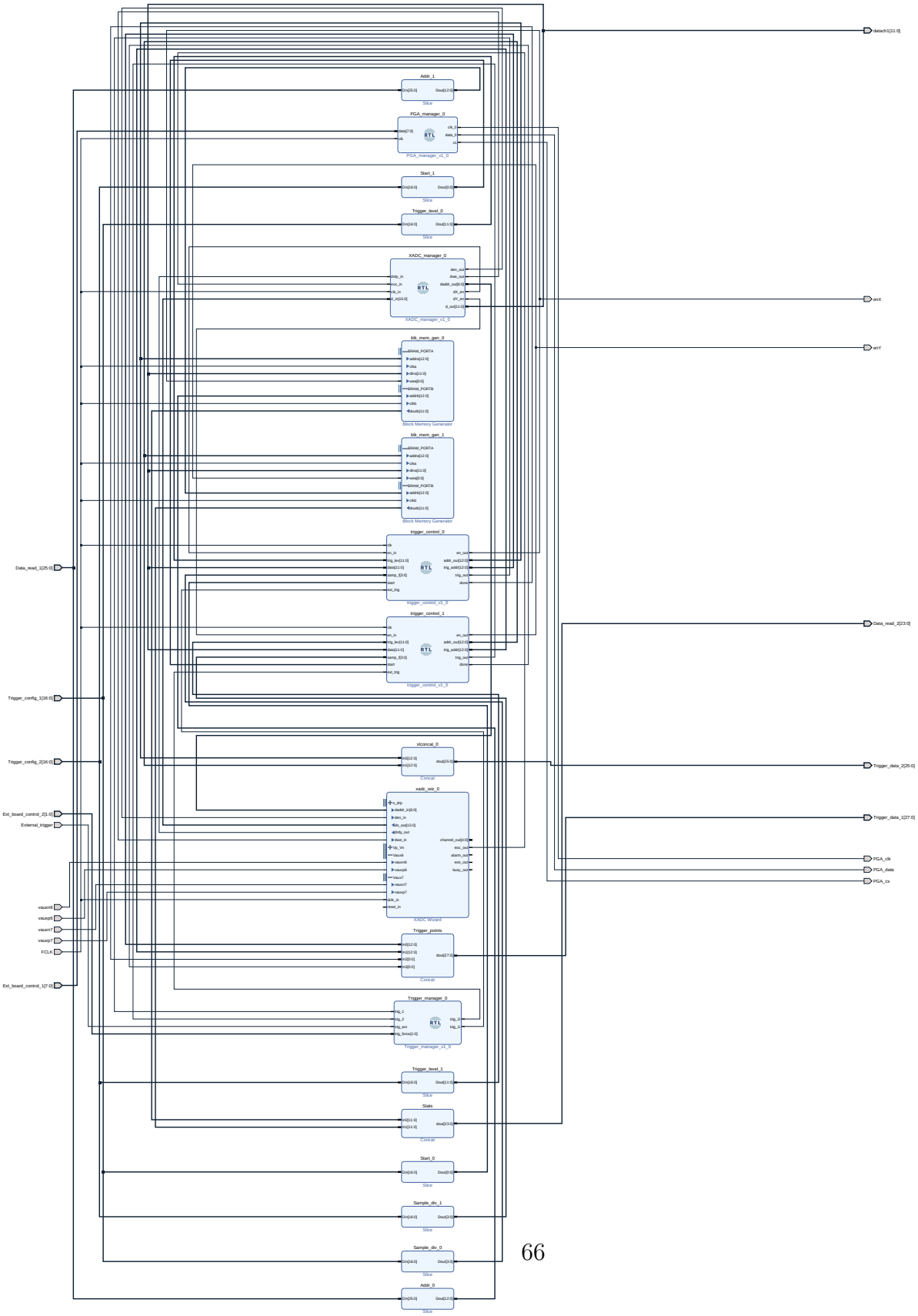


Obr. A.1: Porovnání Waveform Chart a Waveform Graph

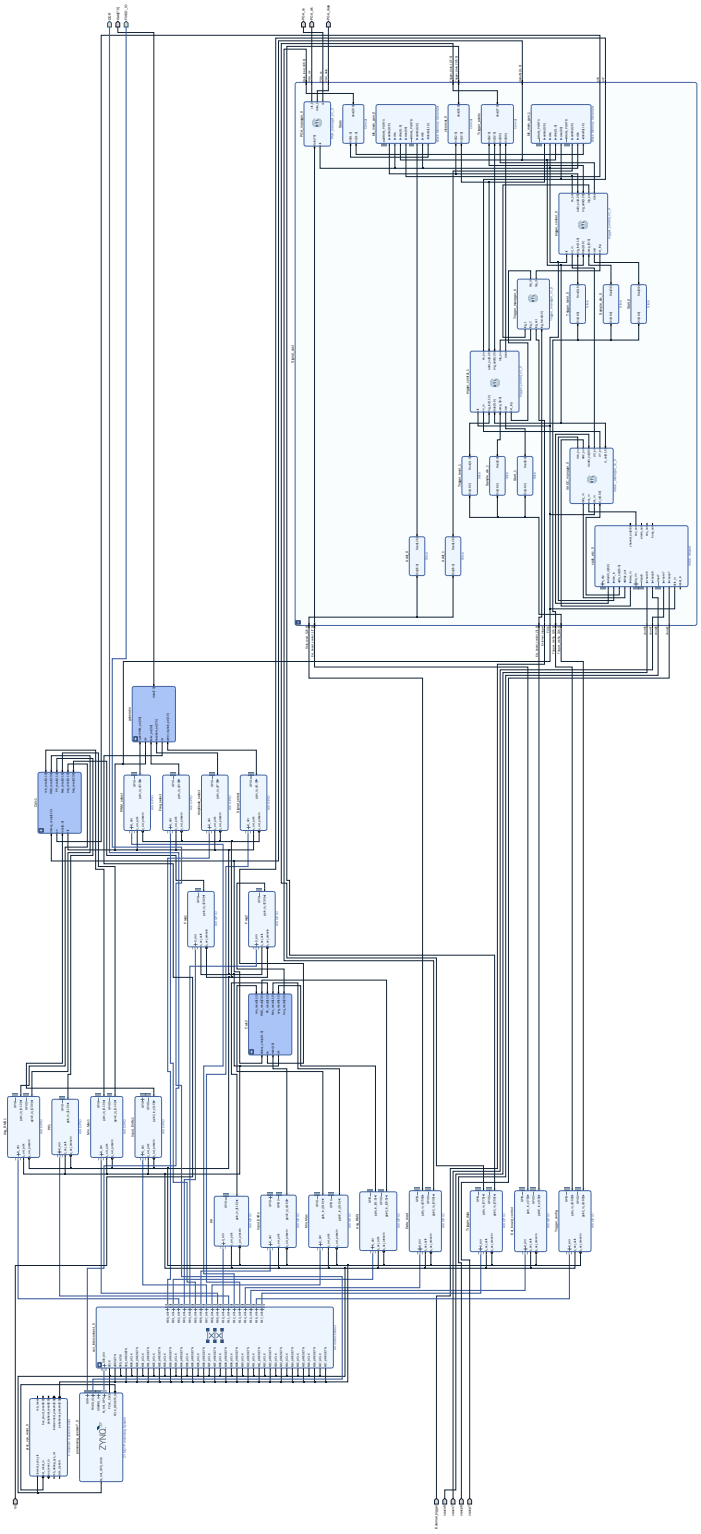
A.2 Blokové schéma modulu pro zpracování signálu ve Vivadu



A.3 Blokové schéma signálového generátoru ve Vivadu



A.4 Blokové schéma celého společného projektu ve Vivadu



A.5 Ukázkové Matlab skripty

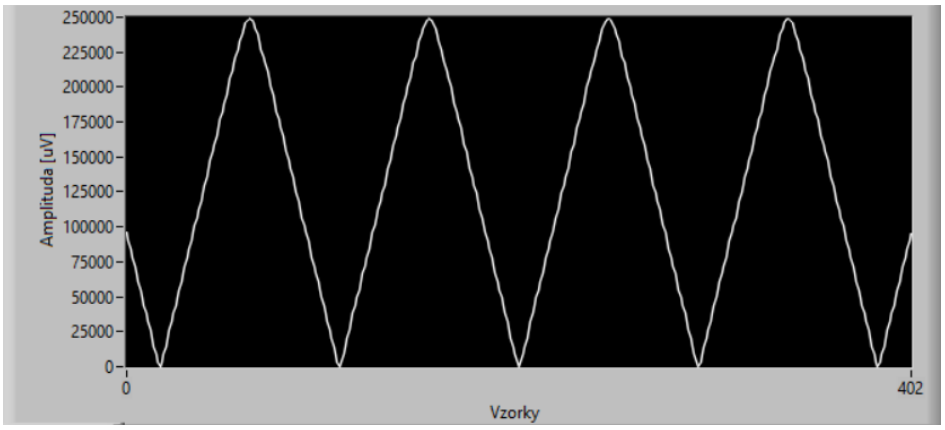
```
% Parametry sinusové funkce
amplitude = 127; % Maximální amplituda (0-255 rozsah -> 127 max)
offset = 128;    % Posun (0-255 rozsah -> 128 střed)
frequency = 1;   % Frekvence sinusové vlny
phase = 0;       % Fázový posun

% Počet vzorků
numSamples = 256;

% Vytvoření vektoru s hodnotami x
x = linspace(0, 2*pi*frequency, numSamples);

% Výpočet hodnot sinusové funkce pro všechny hodnoty x
y = round(amplitude * sin(x + phase) + offset);

% Převod hodnot y na hexadecimální řetězce
hexValues = cell(1, numSamples);
for i = 1:numSamples
    hexValues{i} = dec2hex(y(i), 2); % Převod na 2-místný hexadecimální řetězec
end

% Uložení hexadecimálních hodnot do souboru .hex
filename = 'sin.hex';
fileID = fopen(filename, 'w');
for i = 1:numSamples
    fprintf(fileID, '%s\n', hexValues{i});
end
fclose(fileID);
```

Kód. A.1: MATLAB skript pro sinus

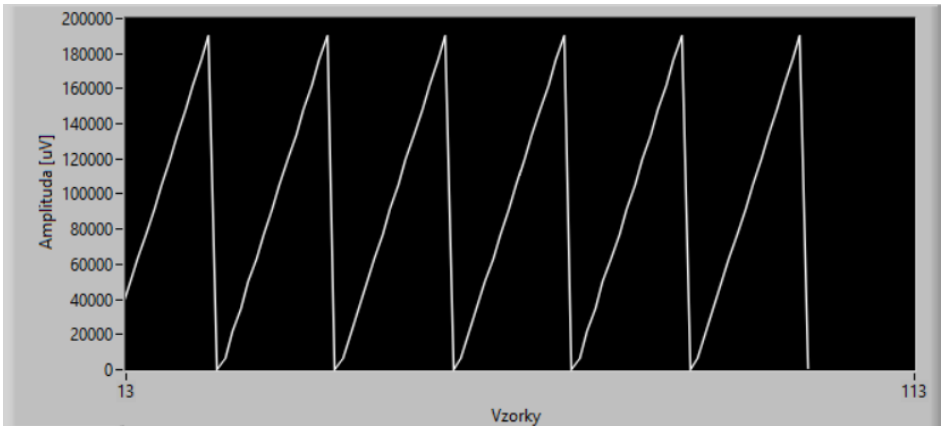
```
% Parametry pilové funkce
amplitude = 127; % Maximální amplituda (0-255 rozsah -> 127 max)
offset = 128;    % Posun (0-255 rozsah -> 128 střed)
frequency = 1;   % Frekvence pilové vlny

% Počet vzorků
numSamples = 256;

% Vytvoření vektoru s hodnotami x
x = linspace(0, 2*pi*frequency, numSamples);

% Výpočet hodnot pilové funkce pro všechny hodnoty x
y = round(amplitude * sawtooth(x) + offset);

% Převod hodnot y na hexadecimální řetězce
hexValues = cell(1, numSamples);
for i = 1:numSamples
    hexValues{i} = dec2hex(y(i), 2); % Převod na 2-místný hexadecimální řetězec
end

% Uložení hexadecimálních hodnot do souboru .hex
filename = 'saw.hex';
fileID = fopen(filename, 'w');
for i = 1:numSamples
    fprintf(fileID, '%s\n', hexValues{i});
end
fclose(fileID);
```

Kód. A.2: MATLAB skript pro pilu

A.6 Průběhy zachycené pomocí zobrazovacího programu v LabVIEW



Obr. A.2: Trojúhelníkový průběh o amplitudě 250 mV a frekvenci 1 kHz na frontpanelu v LabVIEW

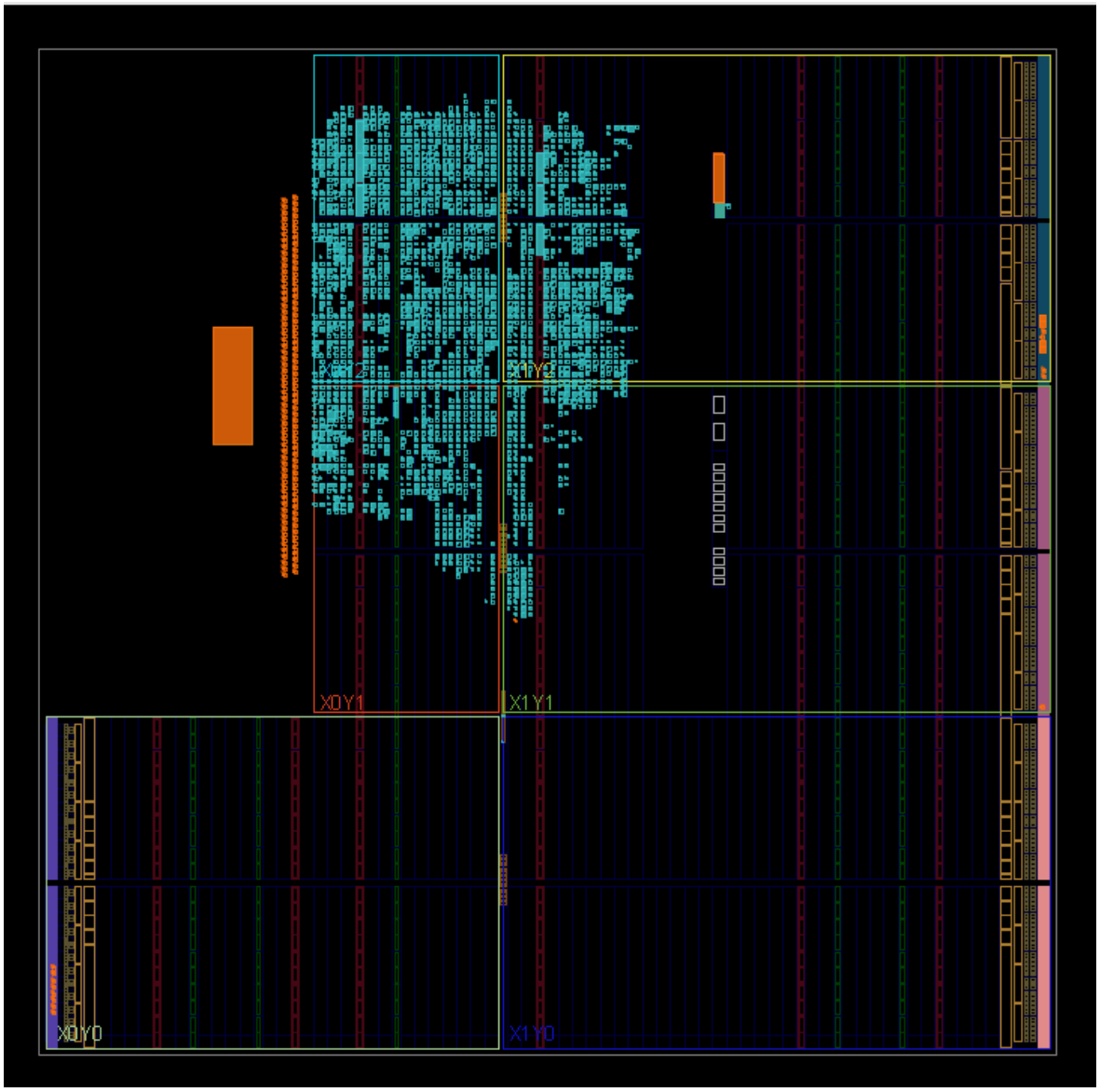


Obr. A.3: Pilový průběh o amplitudě 190 mV a frekvenci 100 kHz na frontpanelu v LabVIEW

A.7 Naměřená data z PWM generátoru

Obr. A.4: Naměřené PWM signály na osciloskopu Rhode & Schwarz RTM3004

A.8 Grafické znázornění využití čipu Zynq7000



Obr. A.5: Grafické znázornění využití čipu Zynq7000

Použité zdroje

1. NAVRÁTIL, Michal. *Digitální vzorkovací osciloskop na bázi SoC – Signálová část*. Praha, 2023. Bakalářská práce. ČVUT FEL. Vedoucí práce CSc. prof. ING. PAVEL HAZDRA. Signálová část této práce.
2. KULARATNA, Nihal. *Digital and analogue instrumentation: testing and measurement*. 11. vyd. London, United Kingdom: IET, 2003. ISBN 978-0-85296-999-1.
3. HICKMAN, I. *Digital Storage Oscilloscopes*. Newnes, 1997. ISBN 9780750628563.
4. HAASZ, Vladimír; HOLUB, Jan; JANOŠEK, Michal; KAŠPAR, Petr; PETRUCHA, Vojtěch. *Elektrická měření: přístroje a metody*. 3. přepracované vydání. Praha: Česká technika - nakladatelství ČVUT, 2018. ISBN 978-80-01-06412-2.
5. HICKMAN, Ian. *Digital storage oscilloscopes*. 1. vyd. Oxford: Newnes, 1997. ISBN 07-506-2856-1.
6. Digital Phosphor Oscilloscope, DPO: What is it. *Electronics Notes* [online]. 2023 [cit. 2023-05-01]. Dostupné z: `https://www.electronics-notes.com/articles/test-methods/oscilloscope/digital-phosphor-oscilloscope-dpo.php`.
7. Mixed Signal Oscilloscope: What is an MSO? *Electronics Notes* [online]. 2021 [cit. 2023-05-01]. Dostupné z: `https://www.electronics-notes.com/articles/test-methods/oscilloscope/mixed-signal-oscilloscope-mso.php`.
8. *Why Would You Want a Mixed Signal Oscilloscope?* [online]. Keysight Technologies, 2021. [cit. 2023-05-01]. Dostupné z: `https://edadocs.software.keysight.com/kkbopen/why-would-you-want-a-mixed-signal-oscilloscope-583415750.html`.
9. *Mixed Domain Oscilloscope* [online]. Tektronix, 2021. [cit. 2023-05-01]. Dostupné z: `https://www.tek.com/en/oscilloscope/mixed-domain-oscilloscope`.
10. XYZs of Signal Generators. In: [online]. [B.r.] [cit. 2023-02-09]. Dostupné z: `https://download.tek.com/document/76W_16672_7_XYZ-Signal-Gen_PR.pdf`.
11. CHURIWALA, Sanjay; HYDERABAD, I. *Designing with Xilinx® FPGAs*. Springer, 2017. ISBN 978-3-319-42438-5.
12. CROCKETT, L.H.; ELLIOT, R.A.; ENDERWITZ, M.A.; STEWART, R.W. *The Zynq Book: Embedded Processing Withe ARM® Cortex®-A9 on the Xilinx® Zynq®-7000 All Programmable SoC*. Strathclyde Academic Media, 2014. ISBN 9780992978709. Dostupné také z: `https://books.google.cz/books?id=9dfvoAEACAAJ`.
13. MARTIN, G.; CHANG, H. System-on-Chip design. In: *ASICON 2001. 2001 4th International Conference on ASIC Proceedings (Cat. No.01TH8549)*. 2001, sv. 1, s. 12–17. Č. 1. Dostupné z DOI: `10.1109/ICASIC.2001.982487`.
14. HADEX. *DSO138*. n.d. Dostupné také z: `%7Bhttps://www.hadex.cz/m460-osciloskop-200khz-dso138-sestaveny-modul/%7D`.

15. BAL, Gurpreet. *DLO-138: Implementation of Digital Logic Design Experiments* [online]. GitHub, 2021. [cit. 2023-05-22]. Dostupné z: https://github.com/ardyesp/DLO-138.

16. *Rigol DS1052E Datasheet* [online]. Gilching: All about circuits, 2009 [cit. 2023-02-10]. Dostupné z: https://www.allaboutcircuits.com/electronic-components/datasheet/DS1052E--Rigol/.

17. *TBS1000C Datasheet* [online]. -: Tektronix, - [cit. 2023-02-10]. Dostupné z: https://www.tek.com/en/datasheet/digital-storage-oscilloscope-tbs1000c-series-datasheet.

18. *TBS2000B Datasheet* [online]. -: Tektronix, - [cit. 2023-02-10]. Dostupné z: https://www.tek.com/en/datasheet/digital-storage-oscilloscope-tbs2000b-series-datasheet.

19. *RTB2000 Datasheet* [online]. -: Rhode-Schwarz, - [cit. 2023-02-10]. Dostupné z: https://scdn.rohde-schwarz.com/ur/pws/dl_downloads/dl_common_library/dl_brochures_and_datasheets/pdf_1/RTB2000_bro_en_3607-4270-12_v0700.pdf.

20. *MSO22 Datasheet* [online]. -: Tek, - [cit. 2023-02-10]. Dostupné z: https://www.tek.com/en/datasheet/2-series-mso-mixed-signal-oscilloscope-datasheet.

21. *R&S RTM3000 Oscilloscope* [online]. Rohde & Schwarz, n.d. [cit. 2023-05-25]. Dostupné z: https://www.rohde-schwarz.com/us/products/test-and-measurement/oscilloscopes/rs-rtm3000-oscilloscope_63493-427459.html.

22. DIGILENT.COM. *Analog discovery* [online]. n.d. [cit. 2022-12-04]. Dostupné z: https://digilent.com/shop/analog-discovery-2-100ms-s-usb-oscilloscope-logic-analyzer-and-variable-power-supply/.

23. DIGILENT.COM. *Eclypse Z7 Reference manual* [online]. n.d. [cit. 2022-12-04]. Dostupné z: https://digilent.com/reference/programmable-logic/eclypse-z7/reference-manual.

24. DIGILENT. *Zmod moduly* [online]. Pullman, WA: Digilent, - [cit. 2022-12-04]. Dostupné z: https://digilent.com/reference/zmod/start.

25. DIGILENT. *Eclypse Z7: Zynq-7000 SoC Development Board with SYZYGY-compatible Expansion* [online]. 2023. [cit. 2023-05-15]. Dostupné z: https://www.youtube.com/watch?v=wc-nkeWz9S8. Video.

26. DIGILENT. *Zybo Z7 Reference manual* [online]. Pullman, WA: Digilent, 2016 [cit. 2022-12-04]. Dostupné z: https://digilent.com/reference/programmable-logic/zybo-z7/reference-manual.

27. XILINX. *SDSoC Development Environment* [online]. 2023. [cit. 2023-05-12]. Dostupné z: https://www.xilinx.com/products/design-tools/legacy-tools/sdsoc.html.

28. DIGILENT. *Pmod Interface Specification* [online]. 2023. [cit. 2023-05-15]. Dostupné z: https://digilent.com/reference/_media/reference/pmod/pmod-interface-specification-1_3_1.pdf.

29. DIGILENT. *Pmod R2R: Reference Manual* [online]. 2023. [cit. 2023-05-15]. Dostupné z: https://digilent.com/reference/pmod/pmodr2r/start?redirect=1.

30. NATIONAL INSTRUMENTS. *LabVIEW Fundamentals* [National Instruments]. 2006. Dostupné také z: %7Bhttps://neurophysics.ucsd.edu/Manuals/National%20Instruments/LV_Fundamentals.pdf%7D.

31. NATIONAL INSTRUMENTS. *NI-VISA*. 2023. Dostupné také z: https://www.ni.com/cs-cz/shop/software/products/ni-visa.html.

32. NATIONAL INSTRUMENTS. *NI-VISA Programmer's Manual*. 2011. Dostupné také z: http://scs.etti.tuiasi.ro/scslabs/Aparatura/NI-VISAProgrammersMan.pdf.

33. XILINX. *AXI General Purpose Input/Output v2.0 PG144* [online]. 2016. [cit. 2023-05-15]. Dostupné z: https://docs.xilinx.com/v/u/en-US/pg144-axi-gpio.

34. XILINX. *7 Series FPGAs and Zynq-7000 SoC Xilinx Processing System IP* [online]. 2023. [cit. 2023-05-15]. Dostupné z: https://docs.xilinx.com/v/u/en-US/pg082-processing-system7.