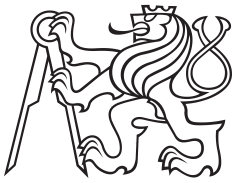
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra kybernetiky

Obsluha rozhraní VGA a PS/2 pomocí FPGA na přípravku Spartan3E v jazyce VHDL
Implementation of VGA Output and PS/2 Input Using Spartan3E FPGA in VHDL

Jiří Hodný

Vedoucí: Ing. Pavel Lafata, Ph.D.
Obor: Kybernetika a robotika
Studijní program: Robotika
Květen 2018

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Jiří H o d n ý

Studijní program: Kybernetika a robotika (bakalářský)

Obor: Robotika

Název tématu: Obsluha rozhraní VGA a PS/2 pomocí FPGA na přípravku Spartan3E v jazyce VHDL

Pokyny pro vypracování:

Vytvořte projekt v jazyce VHDL, který bude obsluhovat výstup VGA a vstup PS/2 na přípravku Xilinx Spartan3E, který je vybaven programovatelným hradlovým polem FPGA.
Výstupem práce bude knihovna v jazyce VHDL, která umožní z běžné klávesnice připojené ke vstupu PS/2 číst stisknuté klávesy a zobrazovat tyto znaky na obrazovce monitoru, připojeného k portu VGA. Tato knihovna tak bude zajišťovat i komunikaci na VGA rozhraní s připojeným monitorem, včetně nastavení korektního rozlišení.
Doplňkově navrhněte a implementujte možnost změny barvy zobrazovaných znaků na monitoru, či zobrazení jednoduché statické grafiky (barevné obrazce apod.). Pro toto ovládání využijte jednoduché přepínače na přípravku Spartan3E, či vstup PS/2 a připojenou klávesnici.

Seznam odborné literatury:

[1] Pinker, J., Poupa, M.: Číslicové systémy a jazyk VHDL. 1. vydání, Vydavatelství BEN, Praha, 2006.
[2] Šťastný, J.: FPGA prakticky. 1. vydání, Vydavatelství BEN, Praha, 2010.
[3] Pong P. Chu: FPGA Prototyping by VHDL Examples: Xilinx Spartan-3 Version, Wiley-Interscience; 1. edition, 2008.

Vedoucí bakalářské práce: Ing. Pavel Lafata, Ph.D.

Platnost zadání: do konce letního semestru 2017/2018

L.S.

prof. Dr. Ing. Jan Kybic
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 25. 11. 2016

Poděkování

Tímto bych chtěl poděkovat svému vedoucímu práce, panu Ing. Pavlu Lafatovi, Ph.D., za příkladné vedení. Rovněž bych chtěl poděkovat přátelům za podporu a jazykovou korekturu. Obzvláště panu Michalovi Gabrielovi.

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, 24. května 2018

Abstrakt

Cílem bakalářské práce je vytvoření knihovny v jazyce VHDL. Knihovna by měla obsluhovat výstup VGA a vstup PS/2 na přípravku Xilinx Spartan3E, respektive Nexys 4. Tyto přípravky jsou vybaveny programovatelným hradlovým polem FPGA. Knihovna bude umožňovat čtení stisknutých kláves na klávesnici a jejich zobrazení na monitoru připojeném přes VGA.

Klíčová slova: FPGA, VGA, PS/2, VHDL

Vedoucí: Ing. Pavel Lafata, Ph.D.
Katedra telekomunikační techniky,
Technická 2,
Praha 6

Abstract

The aim of the bechelor thesis is creation of a VHDL library. The library should operate the VGA output and PS/2 input in Xilinx Spartan3e or Nexys 4 FPGA development kit. The library must read pushed keys and display them on a VGA monitor.

Keywords: FPGA, VGA, PS/2,VHDL

Title translation: Implementation of VGA Output and PS/2

Obsah

Obrázky

Tabulky

Kapitola 1

Úvod

Dostal jsem za úkol navrhnout program v jazyce VHDL, který bude umět číst stisknuté klávesy na běžné počítačové klávesnici. Dále je potřeba ovládat obyčejný monitor, dnes již klasický LCD, avšak standard VGA byl vyvinut již pro CRT monitory. Stisknuté klávesy budu zobrazovat na připojeném monitoru. Později jsme s vedoucím bakalářské práce dohodli na změně přípravku z původního Spartan3e na Nexys 4, který se dá programovat z moderního vývojového prostředí Vivado a také se lépe hodí na danou aplikaci. Dále jsme se rozhodli přidat čtení analogové hodnoty napětí na AD převodníku daného kitu. Proto jsem obrazovku monitoru rozdělil na část textového editoru, nebo řekněme terminálu a část obrazovky "virtuálního"osciloskopu. Terminál zobrazuje stisknuté klávesy a také vykonává různé jednoduché příkazy. Obrazovka osciloskopu zobrazuje průběh napětí na daném AD převodníku v různé vzorkovací frekvenci.

Kapitola 2

Motivace

Již mnoho let se zajímám o matematickou logiku a její aplikaci v digitálních obvodech a počítačových systémech. V druhém ročníku studia jsem se v předmětu Struktury Počítačových Systémů poprvé setkal s programovatelným hradlovým polem Altera, byl jsem v úžasu co vše lze na vývojovém kitu implementovat. Když jsem vybíral téma bakalářské práce, věděl jsem, že VHDL a FPGA je problematika, které budu rád tolik času věnovat. Od zadání studentského projektu, který této bakalářské práci předcházel uběhlo mnoho času, během kterého jsem se s FPGA setkal i v praxi, kde obsluhuje systém hardwarových ochran statického měniče. Programovatelná hradlová pole mají mnoho univerzálních použití v praxi a proto se chci v dané problematice rozvíjet. Můj cíl, zdokonalit své znalosti v oblasti FPGA a VHDL se v této bakalářské práci splnil.

Kapitola 3

Rozbor zadání

Mým cílem je vytvořit projekt v jazyce VHDL. Projekt měl být pro vývojový kit Spartan3e od výrobce Xilinx. Po diskuzi s vedoucím projektu jsme usoudili, že bude vhodnější využít Přípravek Nexys 4 s FPGA Artix7, také od výrobce Xilinx. Projekt je psán v prostředí Vivado 2018.1, avšak před tím jsem testoval i na ISE Design Suite 14.7.

3.1 Obsluha rozhraní PS/2

První klíčová část zadání je čtení kláves z klávesnice připojené přes PS/2, respektive obsluhovat daný interface. Klávesnice vysílá "make code"dané klávesy, který se samozřejmě liší od hodnoty v Ascii tabulce pro daný znak. Proto je potřeba dále implementovat překladač stisknutých kláves (make code) na odpovídající Ascii znak.

3.2 Obsluha rozhraní VGA

Druhá klíčová část je zajištění komunikace s monitorem přes rozhraní VGA. Knihovna musí také nastavit i nějaké korektní rozlišení, volil jsem rozlišení 640x480 při 60ti Hz obnovovací frekvenci. Na monitoru bych měl zobrazovat

stisknuté znaky již jako text psaný určitým fontem. Toto jsem implementovat jako jednoduchý terminál připomínající program PuTTY. Dále bych měl mít možnost měnit barvu textu, případně pozadí.

3.3 Funkce osciloskopu

Dle zadání bych měl na monitoru také zobrazit nějaké obrazce. Já jsem pro zajímavost volil zobrazování průběhu hodnot napětí na vstupu AD převodníku, jehož obsluhu jsem tím pádem také implementoval. Tuto funkci mohu nazvat primitivním osciloskopem, ten se řídí pomocí klávesnice přes řídící klávesu ctrl + klávesa.

Kapitola 4

Teoretický rozbor

4.1 PS/2 rozhraní

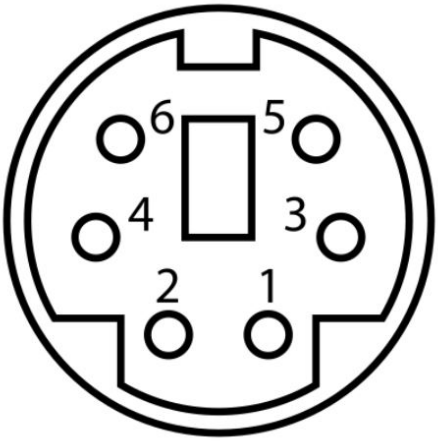
4.1.1 Základní informace

Název PS/2 je označení pro šesti-pinový mini-DIN konektor, kterým se připojuje nejčastěji k počítači klávesnice, myš, případně různá polohovací zařízení. Tento konektor nahradil dříve používaný sériový port a dnes je již plně nahrazen moderním konektorem USB.

4.1.2 Použití rozhraní

Konektor pro připojení myši i klávesnice je mechanicky i elektricky kompatibilní. Avšak komunikační protokoly se pro myš a klávesnici mírně liší. Proto by při opačném zapojení (u starších zařízení, která měla 1 port na myš a 1 na klávesnici) periferie nefungovaly.
Zařízení používající PS/2 mají vlastní elektroniku, která provádí požadované výpočty a předává již kompletní data. Zařízení typu PS/2 proto méně zatěžuje systém, než například USB. Například starší počítače v režimu BIOS neměly problém s PS/2 klávesnicí, jelikož ovladač pro pro ni byl často integrován v BIOSu. Nicméně starší počítače již mnohdy neměly integrovaný ovladač pro USB a tyto vstupní zařízení fungovaly až při běhu operačního systému [3].

4.1.3 Zapojení konektoru



Obrázek 4.1: Rozmístění pinů na konektoru typu PS/2 [3]

- (1) Datový vodič,
- (2) nezapojeno,
- (3) GND,
- (4) +5V DC napětí,
- (5) Hodinový signál,
- (6) nezapojeno.

Zapojení konektoru ke kitu s FPGA

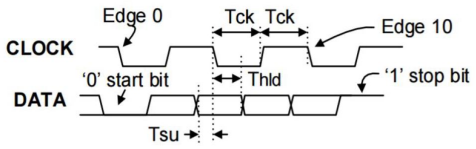
Zemnící a napájecí vodiče jsou přímo připojeny k zemi a k +5V DC na kitu.
Při neaktivitě periferie (klávesnice nebo myši) je hodinový a datový signál ve stavu vysoké impedance ('Z'), proto je možné vyslat na periferii jednoduché řídící instrukce jako ("rozsvit caps lock ledku", případně Numlock a tak podobně). Z důvodu nastavení vysoké impedance je potřeba vodiče hodinového a datového signálu připojit přes PULL-UP rezistor k daným pinům FPGA. PULL-UP rezistor zajistí, že signál je při neaktivitě periferie v logické '1'. Na kitu Spartan3E je běžný PS/2 konektor. Na kitu Nexys4 je pouze konektor USB na kterém funguje rozhraní USB nebo UART. Při připojení klávesnice používající standard PS/2 začne fungovat emulace PS/2, po té konektor USB začne chovat jako konektor PS/2 na kitu Spartan3E.

4.1.4 Popis komunikace na rozhraní PS/2

Přenos dat z klávesnice na vstup systému je řízen hodinovým signálem. Hodinový signál je generovaný přímo v periferii. Data jsou platná vždy na sestupnou hranu signálu. Každé stisknutí klávesy vyvolá přenos jednoho "slova", každé slovo znamená přenést 11 bitů. Jedno takové "slovo"obsahuje:

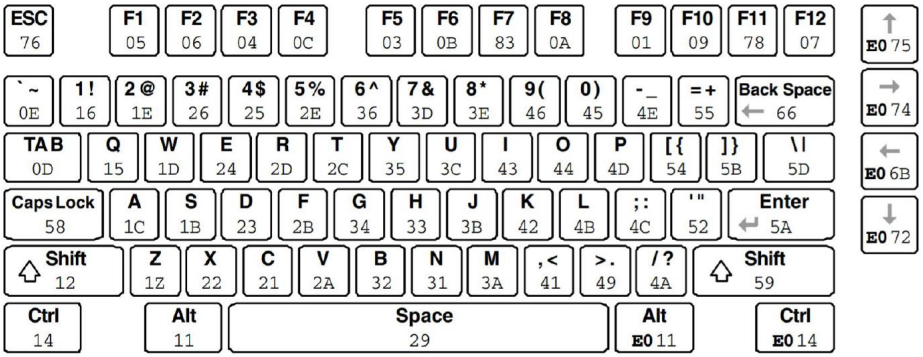
- (0) Start bit(logická '0'),
- (1-8) 8 datových bitů,
- (9) Parity bit,
- (10) Stop bit (logická '1'),

Start přenosu slova je signalizován sestoupením datového vodiče do hodnoty logické '0'. Od této chvíle začne periferie generovat hodinový signál s frekvencí mezi 20ti až 30ti kHz. Na každou sestupnou hranu jsou platná data v pořadí uvedeném výše. Start bit je vždy v logické '0', 8 datových bitů libovolně, následuje paritní bit, ten zde slouží k ověření, zda při komunikaci nedošlo k chybě, respektive, zda se nenačetl některý z datových bitů s opačnou hodnotou. Nastavení paritního bitu záleží na aplikaci. Komunikaci ukončuje načtení stop bitu, který je vždy v logické '1'. Časování je dobře vidět z obrázku níže:



Obrázek 4.2: Časování komunikace PS/2[4]

Je nutné si uvědomit, že je potřeba číst datovou hodnotu jen při sestupné hraně hodinového signálu, jen tak je zaručena bezchybovost načtených hodnot datového signálu.
Data přenesená v jednom slově obsahují 1 byte, ten obsahuje 1 "make code". "Make code"je hodnota klávesy, která byla stisknuta, případně uvolněna. Některé "pokročilé"klávesy jako například pravý ctrl, alt případně šipky vyšlou 2 "make cody"první je vždy 0xE0 a druhý dle stisknuté klávesy. Uvolnění klávesy je signalizováno vysláním "make codu"0xF0 před hodnotou klávesy, která byla uvolněna.
Například stisknutí klávesy 'A' vyšle "make code"0x1C. Uvolnění klávesy pravý ctrl vyšle tři "make cody"a to 0xF0, 0xE0, 0x14. "make cody"kláves jednoduché klávesnice jsou na obrázku níže:



Obrázek 4.3: Vybrané "make cody"pro základní klávesnici[4]

4.2 Výstup VGA

4.2.1 Základní informace

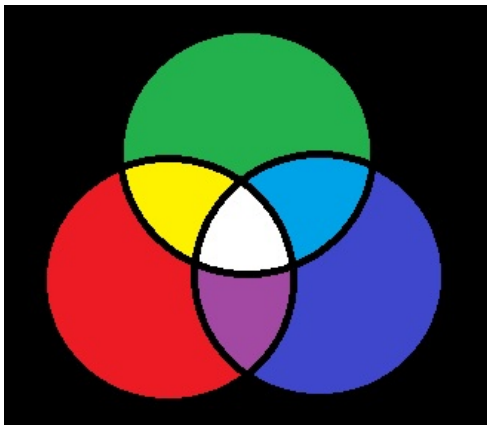
VGA je počítačový standard pro zobrazovací techniku Video Graphics Array). Patří do rodiny nejstarších standardů IBM. Jedná se o nástupce starších standardů EGA nebo CGA. Aktuálně se používá spíše jeho rozšířená verze SVGA, VESA, nebo již modernější digitální DVI, případně HDMI a display port[**?**].

4.2.2 Technické parametry

- Maximálně 720x480 pixelů
- Standardně používá rozlišení 640x480 pixelů při 60ti Hz
- Obnovovací frekvence až 70Hz
- Původně až 256 barev
- Frekvence hlavních hodin 27,175MHz, případně 28,322MHz
- Jedná se o analogový standard, používá napěťovou úroveň 0V až 0.7V

4.2.3 RGB barvy

Standard VGA využívá pouze 3 vodiče pro určení barvy daného pixelu. Jedná se o barvy RGB (červená, zelená, modrá), jejichž intenzita je daná právě hodnotou napětí na daném vodiči, 0V znamená nulová sytost barvy a 0.7V znamená plnou sytost barvy. Každý pixel obrazovky svítí barevně dle hodnot napětí na těchto třech vodičích. Úroveň napětí se tedy mění s frekvencí hlavních hodin (25Mhz).
Bílá barva se docílí plnou sytostí na všech třech vodičích, černá barva nulovou sytostí. Zjednodušený diagram pro 8 barev je znázorněn níže.

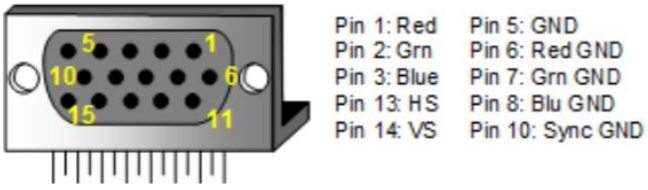


Obrázek 4.4: Diagram RGB pro 8 barev

4.2.4 Elektrické zapojení standardu

Konektor D-SUB DE-15

VGA využívá standardně tento typ konektoru, jeho zapojení níže:

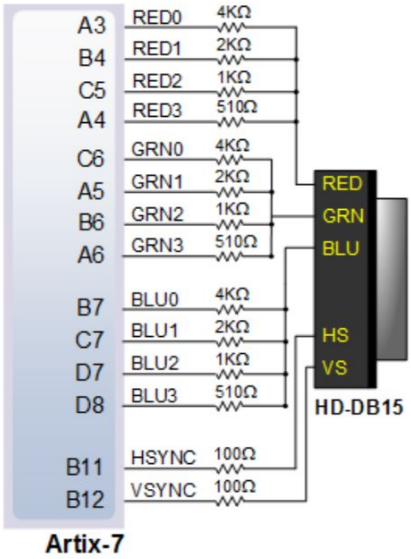


Obrázek 4.5: Zapojení konektoru D-SUB[4]

Konektor obsahuje 3 vodiče určující barvy, vodič pro horizontální synchronizaci, vodič pro vertikální synchronizaci a přirozeně i zemnící vodič GND. Maximální napětí na jednotlivých vodičích proti zemnícímu vodiči je 0,7V. Vertikální a horizontální synchronizace je aktivní v logické '0'.

Připojení konektoru D-SUB k čipu FPGA

Jelikož FPGA je čistě digitální, je potřeba požadovanou digitální hodnotu sytosti barvy převést na hodnotu analogovou. K tomuto účelu postačuje zcela obyčejný DA převodník.
Jedná se o převodník ze tří digitálních čtyřbitových signálů na 3 analogové v rozsahu 0V až 0.7V. Pro jeden DA převod jsou 4 piny FPGA připojeny ke čtyřem různým odporům a to: 510Ω, 1KΩ,2KΩ a 4KΩ. Logická '1' na všech čtyřech pinech vytvoří úbytek napětí 0.7V oproti GND, na daném pinu VGA konektoru viz obrázek níže:



Obrázek 4.6: Zapojení konektoru D-SUB k FPGA Artix-7[4]

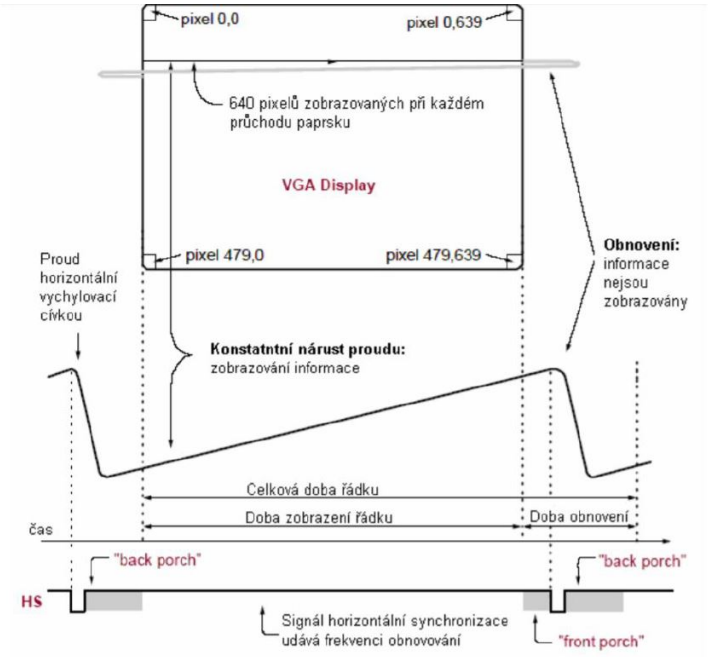
4.3 Časování VGA

Jak jsem uvedl výše, budu používat rozlišení 640x480 pixelů při obnovovací frekvenci 60Hz. Každý videosignál se skládá ze jednotlivých snímků, které

obrazovka vykresluje jeden za druhým ve frekvenci 60Hz (60 snímků za sekundu). Každý snímek se skládá ze 640x480 pixelů. Vykreslení jednoho snímku si můžeme představit jako takový složitější cyklus, respektive 2 cykly vložené "do sebe".

4.3.1 Horizontální synchronizace

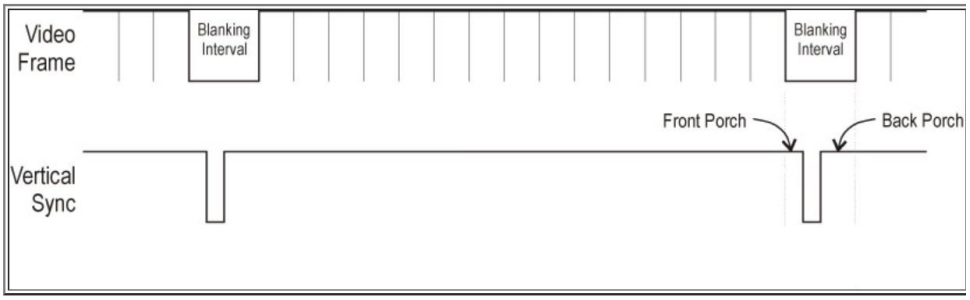
Vnitřní cyklus se stará o zobrazení jednotlivých řádků (640 pixelů). Každý řádek začíná první dobou klidu "back porch", následuje 640 pixelů a po nich se objevuje "front porch", což je druhá doba klidu. Tato doba klidu byla u starých CRT monitorů využívaná k návratu paprsku z pravého okraje zpět na levý (doba obnovení paprsku - retreace time). Po skončení druhé doby klidu dochází k horizontální synchronizaci (vodič VS v logické '0'). Tato synchronizace značí konec řádku. Úplně identicky se zobrazí všech 480 řádků.



Obrázek 4.7: Znázornění principu horizontální synchronizace[4]

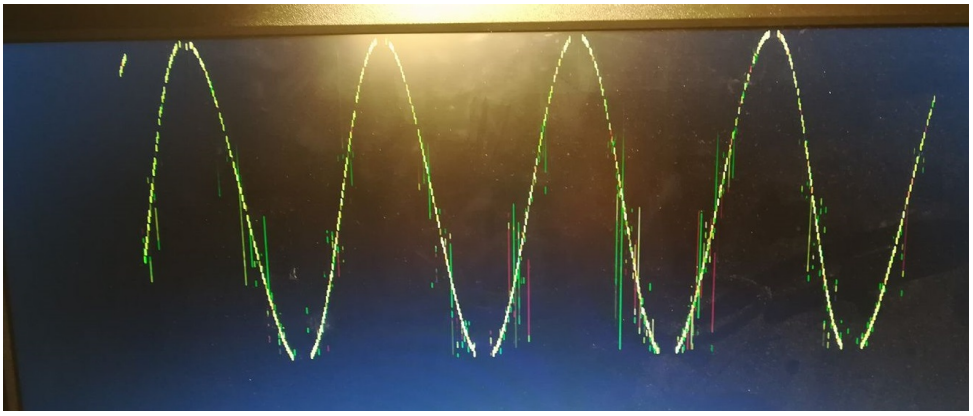
4.3.2 Vertikální synchronizace

Podobně pro vnější cyklus, ten se stará o zobrazení celého snímku a skládá se z 480ti řádků. Každý snímek začíná první dobou klidu "back porch". Následuje 480 řádků, pak druhá doba klidu "front porch"a po ni již vertikální synchronizační puls (vodič VS v logické ’0’) značící konec snímku. Zde se doby klidu využívaly pro vrácení paprsku z pravého dolního rohu obrazu zpět na levý horní roh.



Obrázek 4.8: Znázornění principu vertikální synchronizace[6]

Z vysvětlení výše je patrné, jak funguje vykreslení jednoho snímku. Začínáme v levém horním rohu. Vykreslují se řádky od shora dolů zleva doprava. Pro každý pixel nastavujeme zvlášť barvu. V dobách obnovení (klidu) je potřeba nechat vodiče RGB na hodnotě 0V. Pokud se snažíme "obarvit"pixely mimo obrazovku, může dojít (v mé aplikaci také došlo) ke značnému rozladění obrazu viz obrázek níže:



Obrázek 4.9: Rozladění obrazu VGA

Z obrázku je patrné "probarvení"některých pixelů jinou barvou. Sinusovka měla být "bez šumu"a žlutou barvou na modrém podkladu. Podklad je černý

a některé pixely jsou mylně probarvené červeně, nebo zeleně.
Nicméně monitory odpustí drobné nepřesnosti v časování. Například při nižší frekvenci, než je požadovaných 25.175Mhz upraví obnovovací frekvenci z 60Hz na 59Hz, ale obraz vykreslují.

4.3.3 Hodnoty časování pro 640x480 pixelů

	Horizontální	Vertikální
Back porch	48 pixelů	33 řádků
Viditelný obraz	640 pixelů	480 řádků
Front porch	16 pixelů	10 řádků
Šířka synchronizačního pulsu	96 pixelů	2 řádky
Celý snímek	800 pixelů	525 řádků

Tabulka 4.1: Hodnoty časování[7]

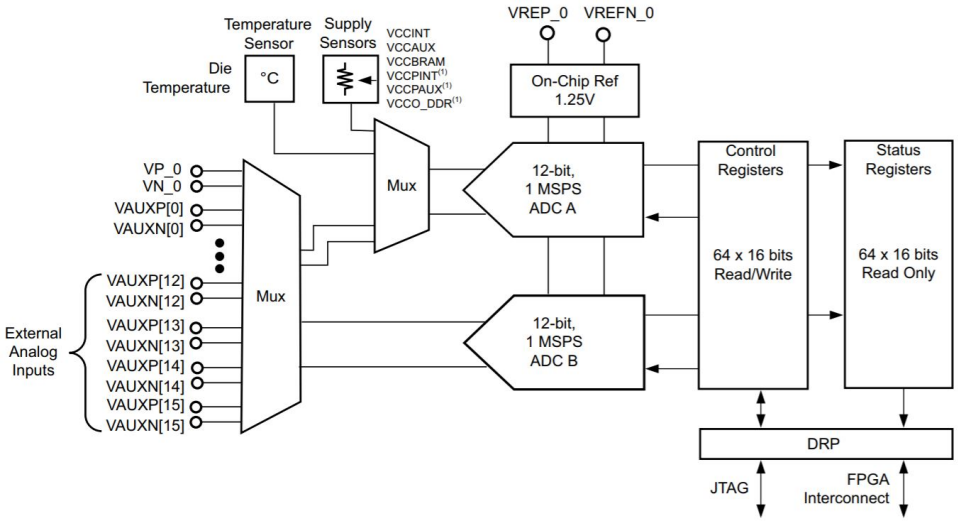
	Frekvence	čas
Hlavní hodiny(pixely)	25.175MHz	0.04us
Řádkový kmitočet	31.469 kHz	0.032ms
Obrazový kmitočet	60Hz	17ms

Tabulka 4.2: Hodnoty synchronizačních frekvencí[7]

4.4 AD převodník

Vývojový kit Nexys4 obsahuje na čipu FPGA Artix7 dvoukanálový 12-ti bitový AD převodník. Převodník disponuje vzorkovací frekvencí až 1MSPS (jeden milion vzorků za sekundu). Převodník lze připojit ke konektoru Pmod XADC a to inicializací skrze periferii DRP (Dynamic Reconfiguration Port [8]). Tato periferie je také připojena k teplotnímu senzoru a také monitoruje napětí všech interních zdrojů napětí (1.8V, 3.3V ,..).
Převodník se dá používat buď v bipolárním, nebo unipolárním módu. Já používám unipolární mód.
V tomto módu se převádí úbytek napětí mezi piny VAUXP(n) a VAUXN(n) kde n je číslo vstupu na kitu. Úbytek by měl být od 0V do 1V. Vyšší napětí by nemělo být na piny přikládáno. Já používám piny č. 3, tedy VAUXP(3) a VAUXN(3).

4.4.1 Schéma připojení převodníku a DRP k FPGA



Obrázek 4.10: Schéma zapojení ADC a DRP[8]

Jak je vidět, FPGA komunikuje s AD převodníkem pouze pomocí periferie DRP, která obsluhuje status i control registr AD převodníku. Připojení externích analogových vstupů se u různých vývojových kitů liší. DRP je potřeba optimálně nakonfigurovat.

4.4.2 Inicializace AD převodníku

Pro inicializaci AD převodníku je potřeba do designu vložit knihovní blok XADC od Xilinxu a správně jej připojit do schématu. Ukázková inicializace je popsána v Uživatelské Příručce [8], nebo [9]. Nám to takto plně postačuje . Vstupy a výstupy bloku jsou zobrazeny na obrázku 4.11.

4.4.3 Unipolární mód AD převodníku

Obrázek 4.12 ukazuje přenosovou funkci AD převodníku v unipolárním módu. Nominální napětí na vstupu do ADC je od 0V do 1V. ADC převodník generuje hodnotu 0x000 při hodnotě 0V a 0xFFF při hodnotě 1V.
Hodnota LSB (Least Significant Bit) v napětí je $\frac{1V}{2^{12}} = 244\mu V$[8]

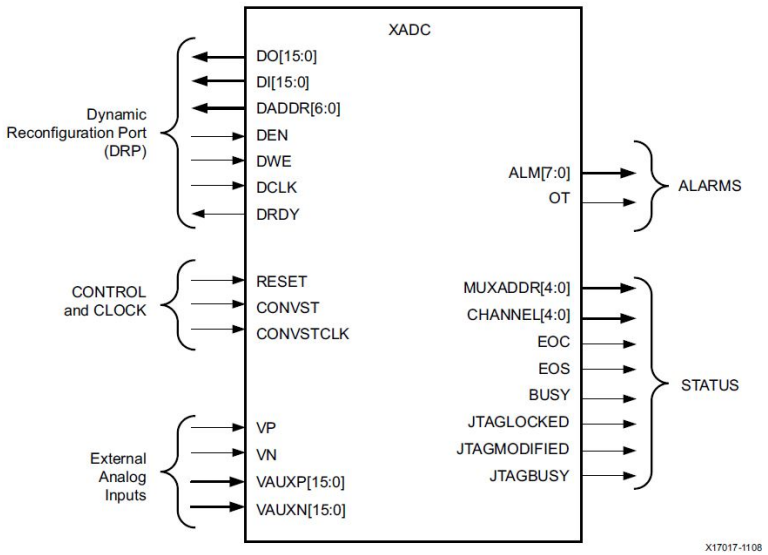
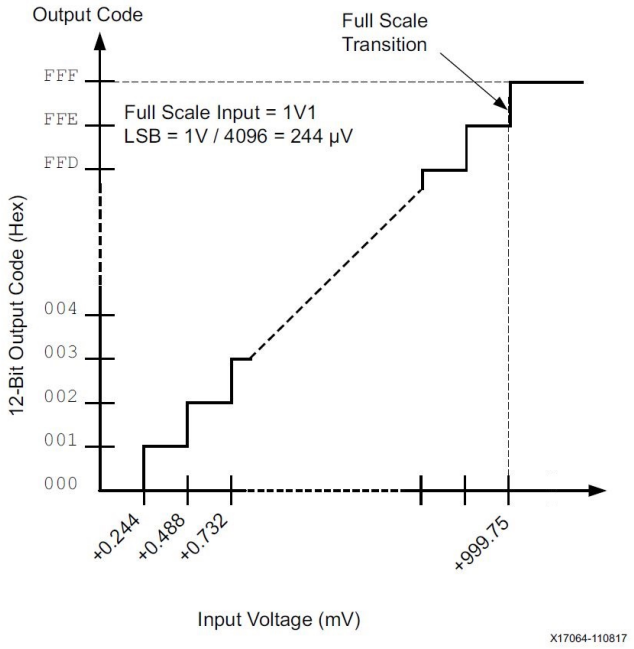


Figure 1-3: **XADC Primitive Ports**

Obrázek 4.11: Knihovní komponenta obsluhující AD převodník[8]

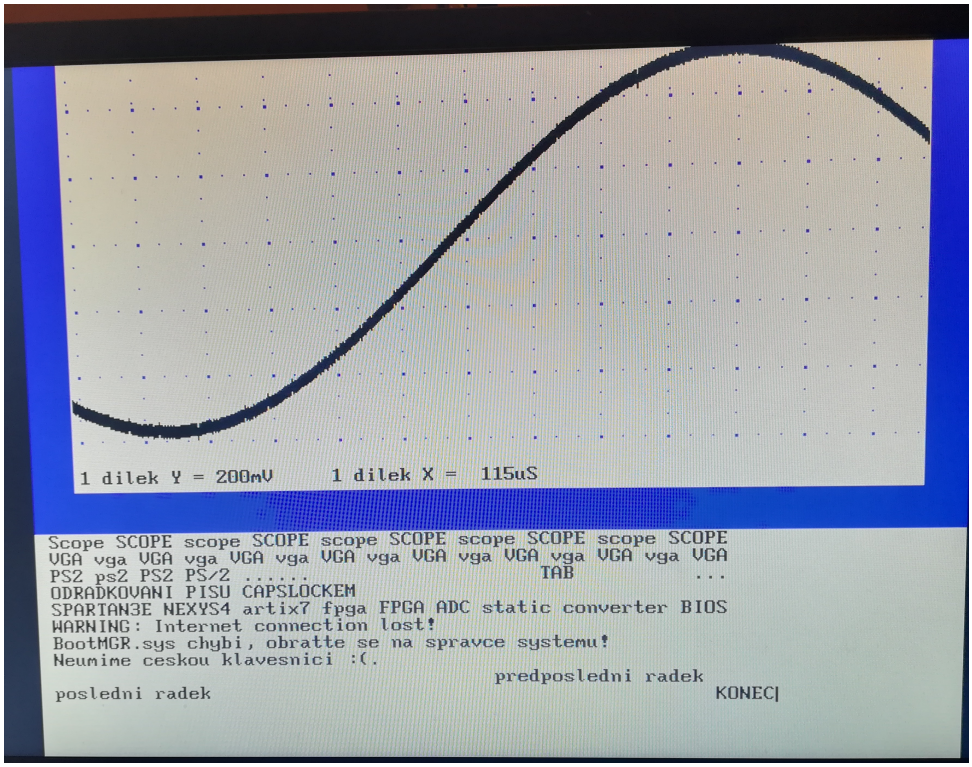


Obrázek 4.12: Přenosová funkce AD převodníku v unipolárním módu[8]

Kapitola 5

Implementace

5.1 Popis výstupu programu



Obrázek 5.1: Ukázka výstupu programu na VGA monitor

Jak již bylo řečeno výše, jediným výstupem programu je počítačový monitor.

Monitor je při rozlišení 640x480 pixelů rozdělen vodorovně na část osciloskopu (nahoře) a na část terminálu (dole).

5.1.1 Osciloskop

Horní část obrazovky(420 horizontálně a 260 vertikálně) slouží k zobrazení průběhu napětí na vstupu AD převodníku při dané vzorkovací frekvenci. Rozsah osciloskopu je 0-1V vertikálně a 1088 hodnot v čase.
Dále se zobrazuje mřížka široká a vysoká 15 pixelů. Vertikální osa osciloskopu je pevná, jeden větší dílek je 200mV. Pro přehlednost je každá větší dílek rozdělen na tři menší dílky.
Vzorkovací frekvence je jak jsem již uvedl variabilní. Kombinací kláves ctrl+Q se vzorkovací frekvence snižuje, respektive zvyšuje pro kombinaci ctrl+R. Nejnižší vzorkovací frekvence osciloskopu je 3KS/s a nejvyšší 780KS/s. Do paměti osciloskopu se vejde vždy jen 1088 hodnot, tedy časově od 1.4mS do 360mS.
Dále je zde funkce posunu po horizontální ose. Stisknutím kombinací kláves ctrl+S se posouvá obraz osciloskopu doleva, respektive kombinací ctrl+T doprava. Na obrazovce osciloskopu je vždy uvedena velikost dílku mřížky. Níže seznam příkazů v přehledně.

- Nový záznam (ctrl+F V zadané vzorkovací frekvenci načte nový průběh dat z ADC)
- Zvětšení časovky na dvojnásobek (ctrl+Q)
- Zmenšení časovky na polovinu (ctrl+R)
- Posun časovky doleva (ctrl+S)
- Posun časovky doprava (ctrl+T)

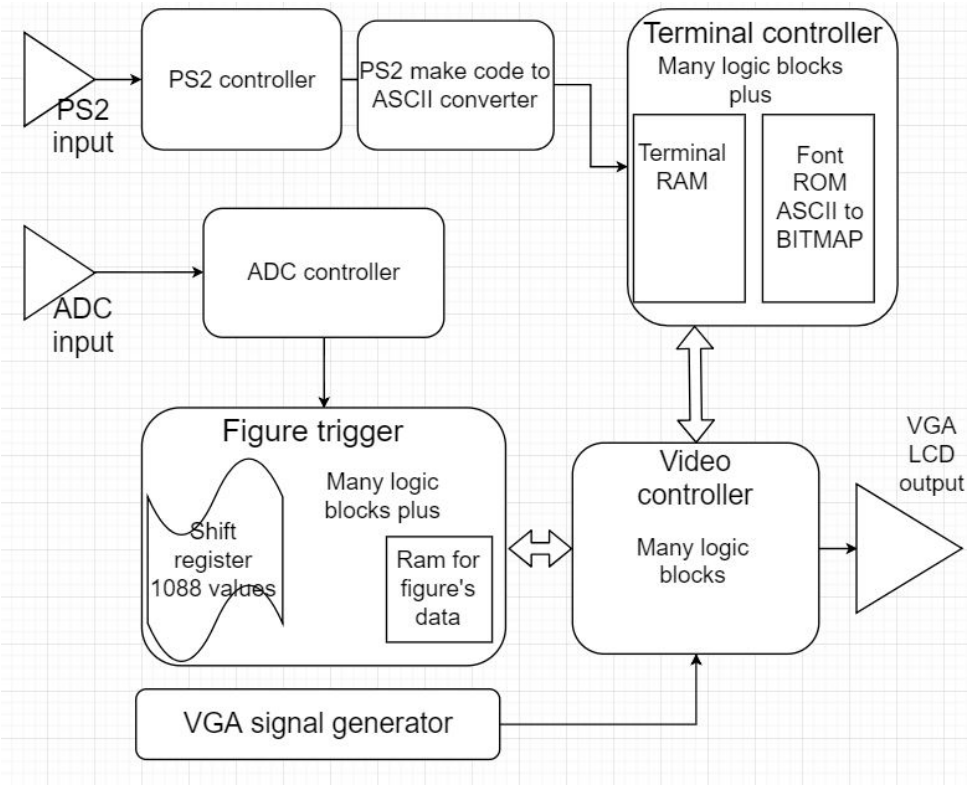
5.1.2 Terminál

Spodní část obrazovky slouží jako výpis jednoduchého terminálu. na terminál se vejde celkem 640 znaků rozdělených do deseti řádek. Na terminál lze vypsat libovolný znak z klávesnice, respektive všechny znaky, které se vyskytují v jednoduché sadě ascii kódů. Uplatňuje se i stisknutí klávesy shift, nebo caps-lock. Dále funguje odstavení klávesou TAB, mazání klávesou backspace,

nebo odřádkování klávesou enter. Barva zobrazení jak terminálu tak osciloskopu je definovaná v 16ti různých předvolených barevných kombinacích (text + průběh na osciloskopu, mřížka + rámeček, pozadí). Změna barevné kombinace je popsaná níže v seznamu příkazů.
Funguje zde několik příkazů a to:

- Nový list (Přesun kurzoru na začátek stránky, vymazání textu - ctrl+L)
- Kurzor na začátek stránky (bez smazání stránky - ctrl+B)
- Kurzor na začátek řádky (bez mazání řádky - ctrl+J)
- Změna barev (ctrl+N další barva/ctrl+O poslední barva)

5.2 Blokové schéma programu



Obrázek 5.2: Blokové schéma hlavní entity programu

5.2.1 Popis schématu

Využitá vstupní zařízení

Nejvyšší entita programu se skládá z mnoha dílčích entit. Jako vstupní zařízení je použita klávesnice s USB portem, dále jeden kanál AD převodníku. Jako další vstupy jsem často využíval 16 přepínačů na vývojovém kitu, ale jen diagnosticky a ve výsledném programu nejsou potřeba.

Využitá výstupní zařízení

Jediný výstup využitý v aplikaci je VGA port pro připojení monitoru. Jako užitečné rozšíření by bylo možné připojit RS232 a to buď pomocí interního USB/UART převodníku, nebo pomocí externího obvodu, například ADM232, který za pomoci pár přidaných kondenzátorů převede 5V logiku na úrovně napětí používané v standardu RS232. Dále jsem pro diagnostické účely používal 16 led na vývojovém kitu, ani ty nejsou ve finální aplikaci využity.

Stručný popis bloků programu

Blok PS2 controller převádí data vyslané sériově z klávesnice na 8mi bitový "make code". Vstupuje tedy datový a hodinový signál z PS2(cca 25KHz). Výstupem je 8-bitová sběrnice obsahující "make code"stisknuté klávesy, dále výstup pro indikaci přijatého "make code", která po správném přijmutí hodnoty vyšle krátký puls v logické ’1’.
Dále se zde nachází blok pro převod "make code"na hodnotu v Ascii kódu (PS2 make code to ASCII converter). Na vstup bloku jsou připojeny veškeré výstupy z předchozího bloku. Výstupem je 8mi bitová sběrnice obsahující Ascii kód stisknutého znaku (rozlišující zda byla stisknuta klávesa caps-lock, nebo zda je stále stisknuta klávesa control atp.). Dále je zde podobně výstup signalizující dokončený převod "make code"na ascii, respektive sděluje, že byla stisknuta klávesa.
Blok ADC controller slouží k obsluze DRP (Dynamic Reconfiguration Port). Vstupem bloku jsou jak vstupy AD převodníku, tak jeho obslužné registry (status registr, control registr). Výstupem převodníku je 12ti-bitová datová

sběrnice obsahující hodnotu 0 až 1 volt z prvního vstupu do AD převodníku. Blok VGA signal generator slouží jen ke generování vertikálních a horizontálních synchronizačních pulsů. Dalším výstupem jsou dvě 10ti-bitové sběrnice pro horizontální a vertikální pozici pixelu monitoru, který je právě "obarvován"hodnotami na třech RGB vodičích monitoru.
Samozřejmě je do všech bloků zaveden vstup pro asynchronní reset a 100MHz signál hlavních hodin.
Blok figure trigger zaznamenává naměřenou hodnotu na AD převodníku v reálném čase. Při příkazu k provedení triggeru naměří ještě cca 250 hodnot a následně hodnoty uloží do paměti ram. Do bloku vstupují data z ADC a vstupní vzorkovací frekvence, dále řídící, adresní a datové signály z bloků VideoController a Terminal controller.
Blok Terminal Controller obsahuje paměť, ve které drží informaci, na které pozici je který znak textu. Dále obsahuje paměť rom, která převede adresu (číslo asci znaku) na data bitmapového obrázku daného znaku k vypsání. Do bloku vstupuje signál z PS2/ascii convertoru, dále několik řídících a datových signálů.
Blok video controller slouží k přiřazení určitých barev ke konkrétní pozici pixelu na monitoru. Horní část monitoru používá data uložená v bloku video controller. Spodní část terminálu využívá data poskytovaná blokem terminal controller, který generuje veškerý text na obrazovce. Blok Video controller dále generuje různé doplňkové obrazce, jako například mřížku osciloskopu, nebo rámeček. Do bloku vstupují řídící, adresní a datové signály jak z bloku Figure trigger, tak z bloku Terminal controller. Výstup bloku jsou barevné kombinace na VGA monitor.

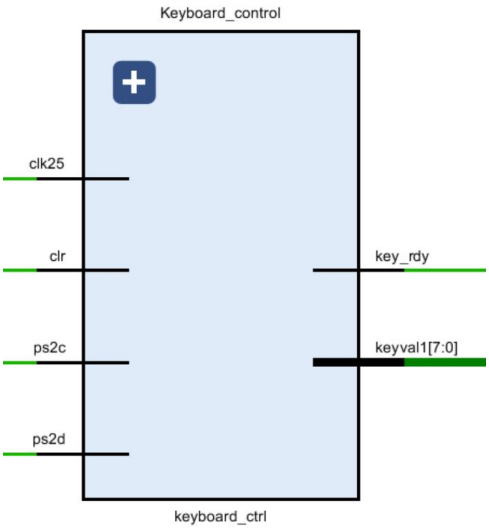
5.3 Detailní rozbor funkce jednotlivých bloků

5.3.1 PS/2 controller

Na vývojové desce Spartan3E se nachází běžný PS/2 konektor. Na desce Nexys 4 už se tento konektor nenachází, ovšem je zde konektor USB, který funguje také v režimu emulace PS/2, takže ovládání takového vstupu je identické, jako u kitu Spartan3E. Pokud FPGA identifikuje zapojení klávesnice podporující PS/2, funkce emulace je automaticky zapnuta. Signál PS/2 clock je pak zapojen na pin F4 a signál PS/2 data je zapojen na pin B2 čipu FPGA.

Popis bloku

Blok má 4 vstupní piny a to hodinový signál clk25 (25Mhz hodinový signál), asynchronní reset aktivní v logické ’0’ a dva potřebné signály ps/2(clock a data). Výstup bloku je datový (8bit) signál obsahující "make code"stisknuté klávesy. Dále signál "key-rdy", který signalizuje stisknutí některé klávesy.



Obrázek 5.3: Blok realizující příjem dat z klávesnice

Použití filtru

Kvůli možnému rušení jsem se rozhodl použít jednoduchý číslicový filtr. Filtr stabilizuje přijatou hodnotu jak z hodinového tak z datového signálu z klávesnice.
Jako jednoduchý filtr jsem použil osmibitový posuvný registr. Hodnota hodin, a dat na výstupu filtru se překlopí z ’0’ na ’1’ až v případě, že všechny bity ve filtru jsou v logické ’1’, identicky pro překlopení z ’1’ na ’0’.

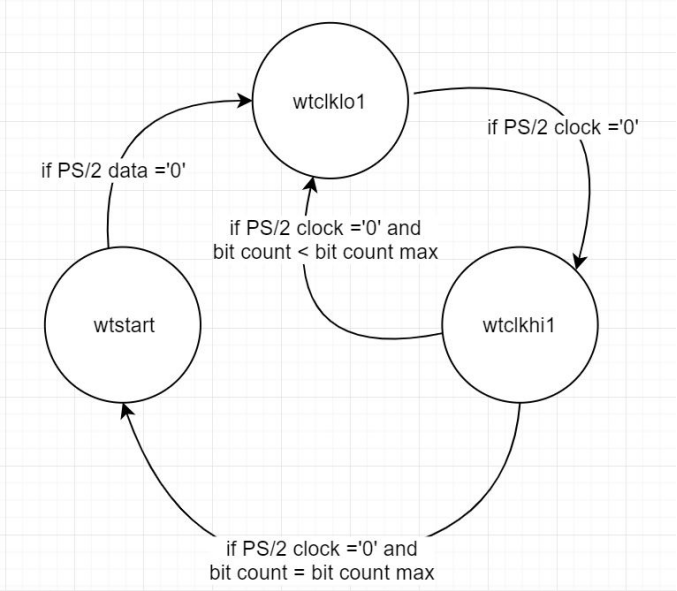
```
elsif clk25'event and clk25='1' then
      ps2c_filter(7) <= ps2c;
      ps2c_filter(6 downto 0) <= ps2c_filter(7 downto 1);

      if ps2c_filter = X"FF" then
            ps2cf<='1';
      elsif  ps2c_filter=X"00" then
            ps2cf<='0';
      end if;
```

Obrázek 5.4: Ukázka části kódu obsahující filtr

Process provádějící příjem dat z klávesnice

Čtení kláves je realizováno v procesu pomocí stavového automatu. Jednotlivé stavy a možnosti přechodu do jiného stavu jsou popsané v tabulce níže:



Obrázek 5.5: Diagram stavového automatu bloku obsluhující PS2

- wtstart
  Jedná se o počáteční stav. V tomto stavu automat čeká na start bit zahajující přenos slova z klávesnice do FPGA. Pokud je signalizován start bit (data v '0'), počká se na sestupnou hranu hodinového signálu a automat se přetočí do stavu wtclklo1.

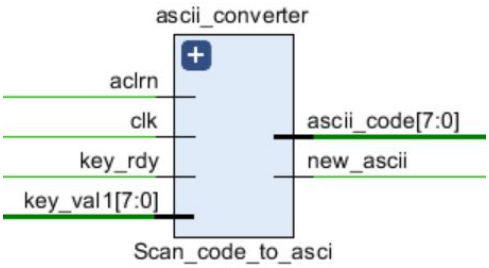
- wtclklo1
  V tomto stavu se čeká na sestupnou hranu hodinového signálu (platnost dat na datovém signálu). Při identifikaci sestupné hrany se načte hodnota na datovém signálu do posuvného registru. Dále se automat přetočí do stavu wtclkhi1.

- wtclkhi1 V tomto stavu se čeká na vzestupnou hranu hodinového signálu. Při naplnění posuvného registru (přijmutí celého slova - "make codu"včetně paritního a stop bitu) se "make code"uloží do výstupního registru. Dále se vymaže posuvný registr, automat se přetočí do stavu wtstart a také se vyšle signalizace přijetí nově stisknuté klávesy na výstupu key-rdy.

5.3.2 Scan code to ascii converter

Popis bloku

Tento blok má vstup pro hodinový signál (25MHz), vstup pro asynchronní reset, řídící vstupní signál key-rdy připojený na výstup bloku PS2 controller a signál key-val1 obsahující "make code"stisknuté klávesy. Výstup bloku je datový signál obsahující přeloženou hodnotu ascii znaku, respektive znak, který byl na klávesnici stisknut. Blok drží v paměti informaci o stisknuté klávese shift, capslock, případně ctrl a alt. Tyto informace jsou využity při překladu "make codu"na ascii znak. Je zřejmý rozdíl mezi malými a velkými písmeny a nebo rozdíl mezi klávesou 1 a vykřičníkem (na anglické klávesnici).

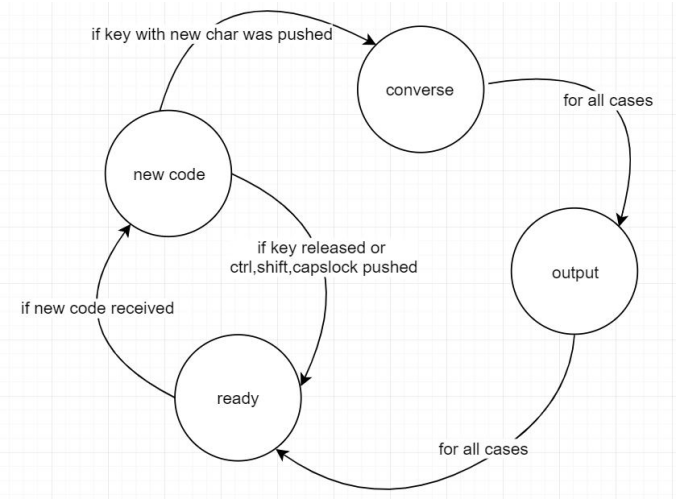


Obrázek 5.6: Blok realizující překlad stisknuté klávesy na ascii znak

Proces převádějící překlad

Překlad "make codu"na daný ascii znak je prováděn pomocí stavového automatu v procesu této entity.

- Stav ready
  V tomto stavu se čeká na příjem nového "make codu"ke zpracování. Při přijmutí náběžné hrany na vstupu key-rdy (signalizující požadavek na překlad) se automat přetočí do stavu new-code.
- Stav new-code
  Zde se ošetřují situace, kdy je stisknuta klávesa shift, caps-lock, ctrl, nebo alt. Dále se zde ošetřuje situace kdy je některá klávesa uvolněna. V případě, že se jedná o "make code"klávesy, která není ošetřena v tomto stavu, automat se přetočí do stavu converse.



Obrázek 5.7: Diagram stavového automatu bloku provádějící překlad

- Stav converse
  Ve stavu converse dochází ke generování ascii znaku dle přijatého "make codu". V případě, že je stále držena klávesa ctrl, vyšle se některý "řídící"znak, jako například STX, ETX, atp. ze začátku seznamu znaků ascii. Tyto "řídící"znaky jsou využity k přenosu příkazu do bloku terminal controller. Například znaky DC1 a DC2 (device controll) se mění vzorkovací frekvence osciloskopu.
  Pokud je uvolněna klávesa ctrl, zjišťuje se stisknutí klávesy shift, případně caps-lock. Dle kombinace stisknutí těchto kláves se generuje ascii znak buďto pro malé, nebo velké písmeno, respektive pro tečku, nebo dvojtečku.
  Po překladu se automat přetočí na stav output.

- Stav output Ve stavu output se zapíše hodnota ascii znaku do výstupního registru bloku. Dále se výstup new-ascii překlopí do logické '1'. Tím je vyslán signál, že došlo k překladu. Nyní se automat překlopí opět do stavu ready, výstup new-asci se překlopí zpět do logické '0'.

5.3.3 Terminal controller

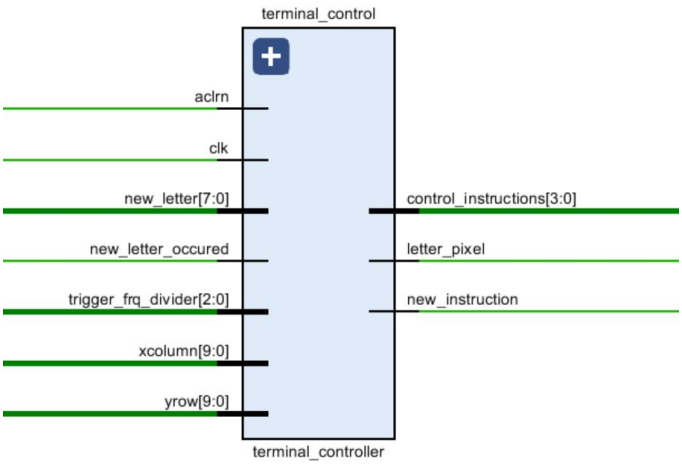
Popis bloku

Tento blok řídí výpis textu na monitor. Jeho vstupem je hodinový signál (clk), asynchronní reset (aclrn), pozici vykreslujícího se pixelu (xcolumn a yrow). Dále vstupuje hodnota ascii znaku naposledy stisknuté klávesy (new

letter) a řídící signál signalizující stisknutí klávesy (new letter occured). Posledním vstupem bloku je signál obsahující informaci o vzorkovací frekvenci osciloskopu (trigger freq divider), která je následně také vypisována na monitor.
Z bloku vystupuje řídící signál pro nadřazenou logiku (control instruction) - daná instrukce a signál signalizující novou instrukci (new instruction). Dalším výstupem je signál obsahující informaci, zda se má daný pixel textového pole obarvit, nebo zůstat neobarvený - barva pozadí textu (letter pixel).
Uvnitř jsou dva paměťové bloky a to:

- terminal text ram Jedná se o paměť typu ram, do které se zapisují jednotlivé stisknuté znaky, paměť je široká 8 bitů a vejde se do ni 640 znaků (do terminálu se vejde 64 písmenek na řádku a 10 řádků), čili v paměti je informace, na které pozici terminálu (adresa paměti) je uložen jaký znak (data paměti).

- letter rom V této paměti ROM jsou zapsaná data o bitmapovém obrázku každého znaku. Tato paměť má 128 buněk pro každý znaky základní ascii tabulky. Každá buňka (každý znak) je vlastně obrázek složený z 8mi pixelů na šířku a z 12ti pixelů na výšku. Inicializační soubor této paměti jsem po dohodě s vedoucím práce převzal a pro svojí potřebu poupravil z materiálu uvedených ve zdrojích [1].
  Výsledná paměť má tedy 11ti bitovou adresu pro 128(znaků) x 12 (12 řádek každého obrázku znaku). Data paměti je vždy daný řádek znaku, tedy 8 bitů.



Obrázek 5.8: Blok řídící výpis textu na monitor

Proces bloku

- Aktualizace stringu popisku osciloskopu
  V procesu se při každé náběžné hraně vždy aktualizuje string popisky os osciloskopu (text zůstává stejný, aktualizuje pouze časová základna osciloskopu, jedná se o výpočet času jednoho dílku a následně o jednoduchý BCD převod z binárního čísla do čísla v desítkové soustavě. Výsledná hodnota je v mikrosekundách, případně v milisekundách.

- Obsluha přijatých dat z klávesnice
  Pokud je přijat ascii znak s hodnotou 32 až 127 (písmena a jiné znaky vypisující se na klávesnici), je daný znak uložen do paměti ram na adresu aktuálního místa kurzoru. Dále se kurzor posune o jedno políčko dál.
  Pokud je přijat řídící znak s hodnotou 0 až 31, nebo znak DEL, je vykonán příkaz příslušející k dané instrukci. Tyto instrukce se dělí do tří skupin a to do instrukcí pro terminál (backspace, delete, enter, tabulátor, nová stránka, posun kurzoru -na začátek stránky nebo řádku), do instrukcí pro osciloskop (nový záznam dat, změna vzorkovací frekvence, posun po ose x) a do instrukce pro ovladač displeje (změna barevného motivu textu a osciloskopu).
  Instrukce pro terminál jsou vykonány lokálně danou úpravou dat v paměti ram (vymazání znaku, vymazání celé stránky atp.). Instrukce pro osciloskop jsou předány na výstup control instruction. Dále je vyslán krátký puls na výstup new instruction, který signalizuje potřebu výkonu instrukce v nadřazené logice.

- Výpis na terminál
  Pokud nejsou přijata žádná nová data, proces se stará o korektní výpis popisku osciloskopu a o výpis dat uložených v paměti ram. Protože paměť ram sdílí adresovou sběrnici jak pro ukládání tak pro čtení dat, je potřeba při zápisu dat adresovou sběrnici "odpojit". Výkon instrukcí tedy na krátký čas odpojí adresovou sběrnici a nedojde k vykreslení některých pixelů znaků terminálu. Nicméně při používané obnovovací frekvenci monitoru si tohoto "problému"lidské oko nevšimne.
  Výpis jednotlivých znaků funguje tak, že dle pozice kurzoru (vstup bloku) je vypočteno, který znak z paměti ram je třeba vypsat. Hodnota daného znaku se vynásobí 12ti(každá buňka paměti rom má 12 řádek) a dále se přičte číslo řádky daného znaku. Tímto dostanem adresu pro paměť ROM, z ní pak vezmeme ten bit, který odpovídá danému sloupci znaku. Zde byly jisté problémy s časováním (obnovení výstupu registrů a pamětí trvá vždy jednu náběžnou hranu), které sem vyřešil vhodným nastavením konstant ve výpočtech používající horizontální pozici pixelu.

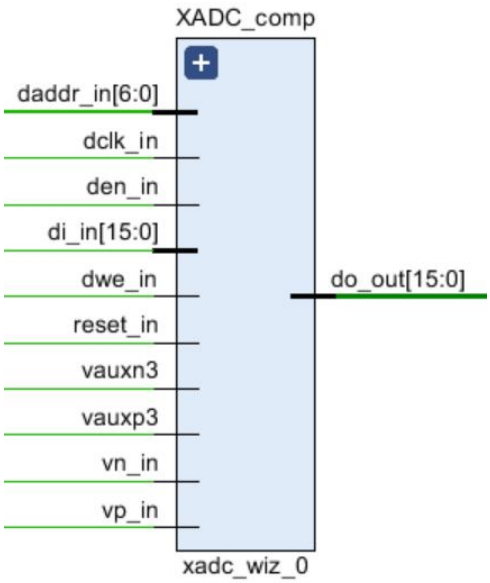
5.3.4 ADC controller

Popis bloku a jeho připojení do schématu

Blok ADC controller obsluhuje DRP (Dynamic Reconfiguration Port). Tento blok jsem vygeneroval pomocí funkce XADC wizard v IP Catalogu ve vývojovém prostředí Vivado. Zde se po nastavení požadovaných parametrů vygeneruje blok, který již stačí připojit do schématu následovně:

- daddr-in[6:0] Jedná se o adresní sběrnici DRP. Na tento vstup v mém případě stačí zapsat hodnotu "0010011". Tato konstanta označuje připojení třetího vstupu konektoru PMOD na námi použitý kanál AD převodníku.
- dclk-in Toto je vstup hodinového signálu do DRP. Já připojil hlavní hodiny (100MHz).
- den-in Vstup povolující vstup signálu do DRP (zapisuji konstantně '1').
- di-in[15:0] Datová sběrnice pro DRP (zapisuji šestnáctkrát logickou '0').
- dwe-in Write enable port pro DRP, nepotřebuji nic zapisovat, proto je konstantně v logické '0'.
- do-out Výstupní datová sběrnice obsahující převedenou hodnotu z AD převodníku. Z tohoto signálu používám horních 8 bitů.
- reset-in Vstup resetující AD převodník. Aktivní v logické '1'.
- vauxn3 a vauxp3 Toto jsou mnou používané vstupy do AD převodníku, k nim připojím odpovídající vstup PMOD konektoru.
- vp-in a vn-in Vstupy pro připojení referenční hodnoty pro AD převodník. Jelikož používám interní referenční hodnoty, připojím oba vstupy k zemi (dle popisu v aplikačním dokumentu[8])

Takto nastavený převodník plně postačuje pro moji aplikaci.



Obrázek 5.9: Blok řídící AD převodník na čipu Artix 7

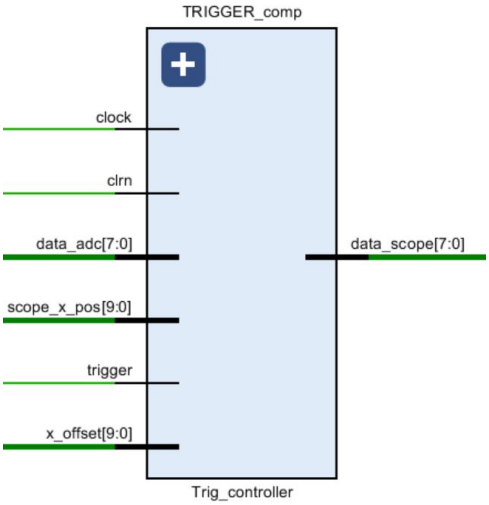
5.3.5 Figure trigger

Popis bloku

Jak již jsem uvedl, tento blok slouží k zaznamenání průběhu napětí na AD převodníku při požadované vzorkovací frekvenci.
Do bloku vstupuje požadovaná vzorkovací frekvence (clock), požadavek na asynchronní reset componenty (clrn), datová sběrnice (data-adc), adresní sběrnice pro požadovaná data uložené v paměti ram (scope-x-pos), sběrnice obsahující posun dat osciloskopu po horizontální ose a kontrolní vstup pro žádost o provedení záznamu dat(trigger).
jediným výstupem jsou data o požadované hodnotě z paměti (data-scope).

V bloku jsou použity dva paměťové bloky a to:

- Scope ram
  Tato paměť typu ram slouží k uchování celého průběhu napětí, zde jsou uložená data, která zobrazuje osciloskop na monitoru. Má desetibitový adresní vstup, osmibitový datový vstup a osmibitový datový výstup. Dále obsahuje vstup pro hodinový signál a vstup povolující zápis do paměti (write enable).
- Shift register

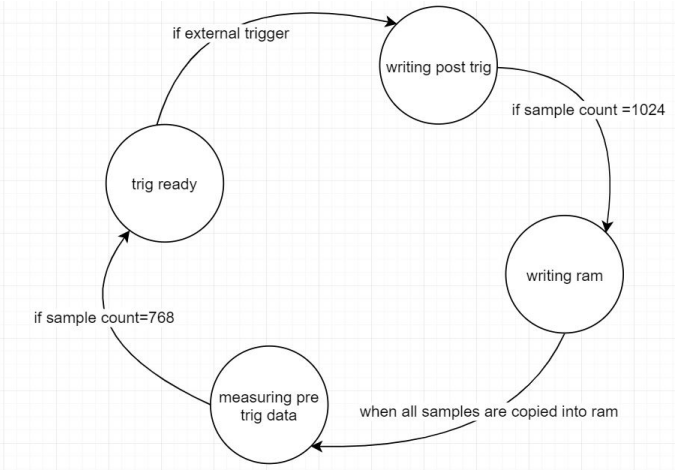


Obrázek 5.10: Blok provádějící záznam průběhu napětí na externí požadavek

Jedná se o posuvný registr (struktura typu fronta - FIFO), obsahuje 1024 hodnot po 8mi bitech. Do této paměti jsou v reálném čase zapisovány data z ADC převodníku. Obsahuje vstupní a výstupní datovou sběrnici, vstup pro hodinový signál a vstup pro synchronní reset (mazání na náběžnou hranu).

Funkce procesu v bloku

Činnost bloku je realizována stavovým automatem s následujícími stavy:



Obrázek 5.11: Diagram stavového automatu bloku provádějící záznam průběhu

- measuring pre trig data
  Jedná se o počáteční stav automatu, do tohoto stavu se přetočí automat i při resetu.
  V tomto stavu se do posuvného registru načte 768 hodnot. Po té se automat přetočí do následujícího stavu. Data v registru jsou ovšem stále aktualizovaná. V tomto stavu je adresní sběrnice paměti ram používána blokem zobrazující aktuální zaznamenaná data na monitor.

- trig ready
  V tomto stavu se jen čeká na pokyn k uložení záznamu. Při identifikaci náběžné hrany na řídícím vstupu (požadavek na trigger), automat se přetočí do stavu writing post trig. I v tomto stavu je adresní sběrnice paměti ram používána blokem zobrazující aktuální zaznamenaná data na monitor.

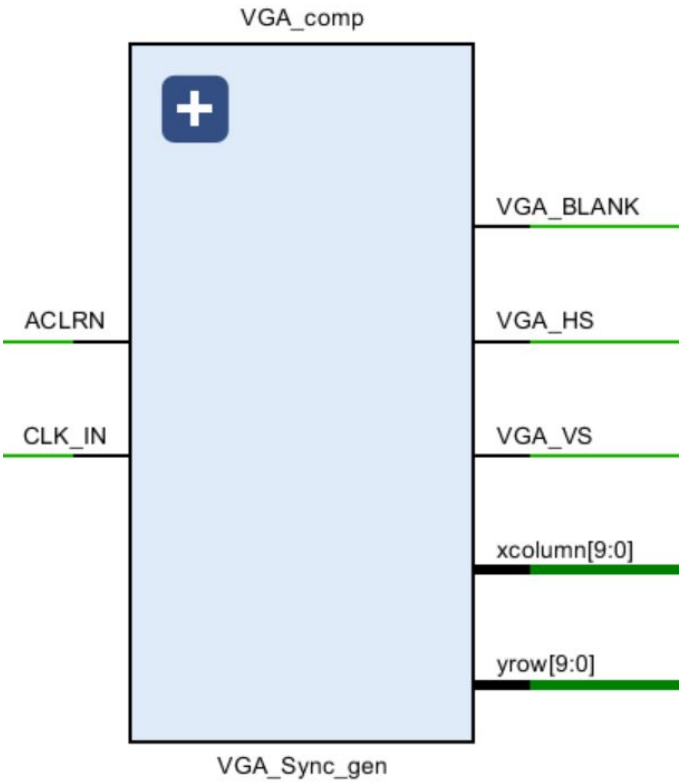
- writing post trig
  Zde se naměří zbývajících 256 hodnot. Po naměření máme potřebných 768 hodnot před požadavkem a 256 po požadavku. Automat se nyní přetočí do stavu writing ram. Zde dojde k odpojení adresní sběrnice ram od bloku zobrazující zaznamenaná data na monitor jelikož se následně budou data přepisovat aktuálními.

- writing ram
  Nyní se sekvenčně všech 1024 hodnot z posuvného registru zapíše do paměti ram. Po zapsání všech dat se opět připojí adresní sběrnice k bloku zobrazující data na monitor a automat se přetočí do počátečního stavu.

5.3.6 VGA signál generátor

Popis bloku

Tento blok slouží ke generování horizontálních a vertikálních synchronizačních pulzů. Generované rozlišení je 640x480 pixelů při 60ti Hz, proto je na hodinovém vstupu bloku připojen hodinový 25MHz signál (přes děličku 1/4. Dále je připojen vstup pro asynchronní reset.
Blok má výstupy pro vertikální, horizontální synchronizaci a výstup povolující barvení pixelů (ten je aktivní v logické '1'). Další výstupy jsou souřadnice pixelu (pozice v řádce - xcolumn a řádka yrow), který se má zrovna "obarvit".



Obrázek 5.12: Blok generující signál pro obsluhu VGA rozhraní

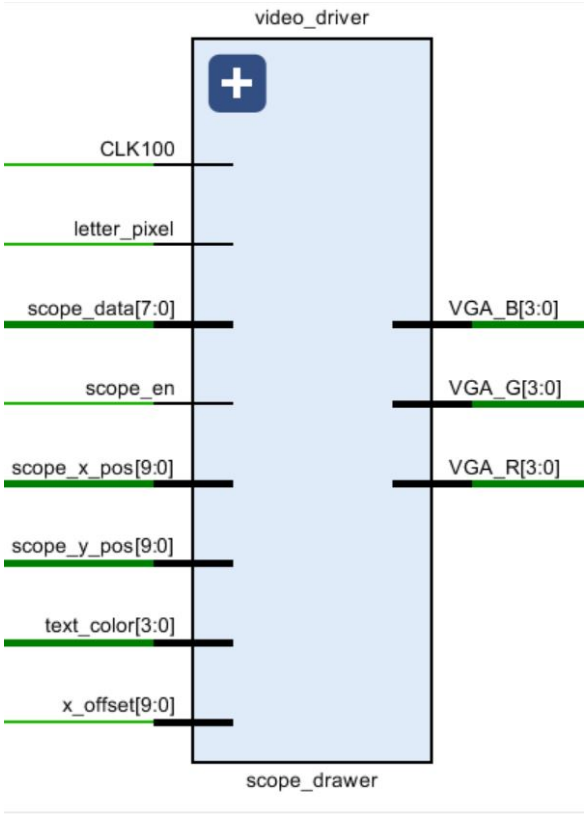
Činnost procesu bloku

Blok má nastavené konstanty pro časování dle tabulky 4.1 v kapitole 4.3.3 V architektuře jsou dva procesy, jeden pro horizontální a druhý pro vertikální synchronizaci. Čítač pro výpočet horizontální synchronizace čítá s každou náběžnou hranou hodin. Horizontální synchronizace je od začátku čítání v logické '0'. Po načítání hodnoty 96 (délka synchronizačního pulzu se HS překlopí do logické '1'. Při hodnotě 144 se dostáváme do viditelné části obrazu. Při hodnotě 784 (640+144) se dostáváme pryč z viditelné části obrazu. Nyní se již časově dostáváme do nezobrazující se části obrazu (front porch). Při načítání hodnoty 800 se čítač opět vynuluje, vyšle se 1 takt krátký puls (vs enable), který slouží pro čítání vertikální synchronizace.
Čítání (při vs enable - jednou za dobu zobrazení jedné řádky) pro vertikální synchronizaci funguje obdobně. Délka synchronizačního pulsu je jen 2 řádky, viditelná část obrazu začíná na řádku 35 a trvá 480 řádků. Při hodnotě 515 se dostáváme zpět do nezobrazované oblasti (front porch pro VS). Čítač se nuluje při dosažení hodnoty 525 řádků.
Hodnota obou čítačů je i výstupem bloku (souřadnice vykreslujícího se pixelu).

5.3.7 Display controller

Popis bloku

Tento blok programu ovládá tři výstupní čtyř-bitové signály RGB, respektive určuje, jakou barvu bude daný pixel na monitoru mít. Do bloku vstupuje hodinový signál, signál povolující zobrazování (logická '1' v případě, že pozice vykreslujícího se pixelu je ve viditelné části obrazu), souřadnice vykreslujícího se pixelu, 4 bitový signál určující barevný font (text/graf, mřížka/rámeček, pozadí). Dále data osciloskopu (8mi-bitový signál digitální hodnoty načtené AD převodníkem a uložené do paměti RAM na daném sloupci obrazu). Posledním vstupem bloku je signál určující obarvení pixelu v textu (logická '1' v případě, že se jedná o pixel znaku a '0' pokud se jedná jen o pozadí znaku).



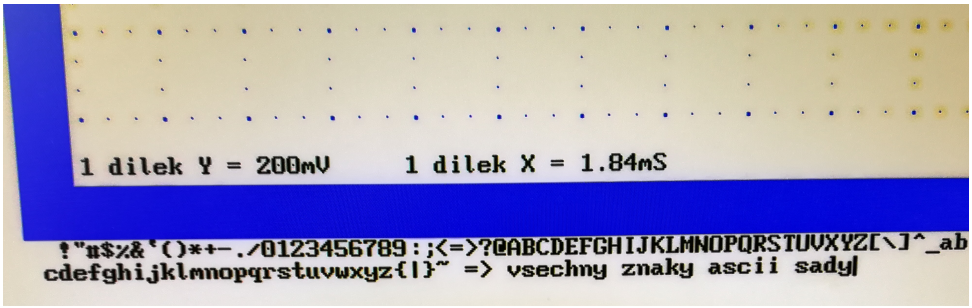
Obrázek 5.13: Blok generující barvy obrazu na monitoru

Činnost procesu bloku

V tomto bloku mám definované vlastní proměnné, jako třeba barvy (červená, modrá, zelená, žlutá, černá, bílá,..), nebo výčtové typy zobrazovací se barevné kombinace.
Při každé náběžné hraně (pokud se pixel nachází ve viditelné části obrazu) se dle série podmínek určí, zda se vykreslí průběh grafu (pokud se vertikální souřadnice vykreslujícího pixelu rovná příslušné hodnotě průběhu - vstup "scope data"), případně mřížka, nebo pozadí osciloskopu.
V případě vykreslování textu se bere v úvahu hodnota na vstupu "letter-pixel", který je ovládán blokem "terminal controll".

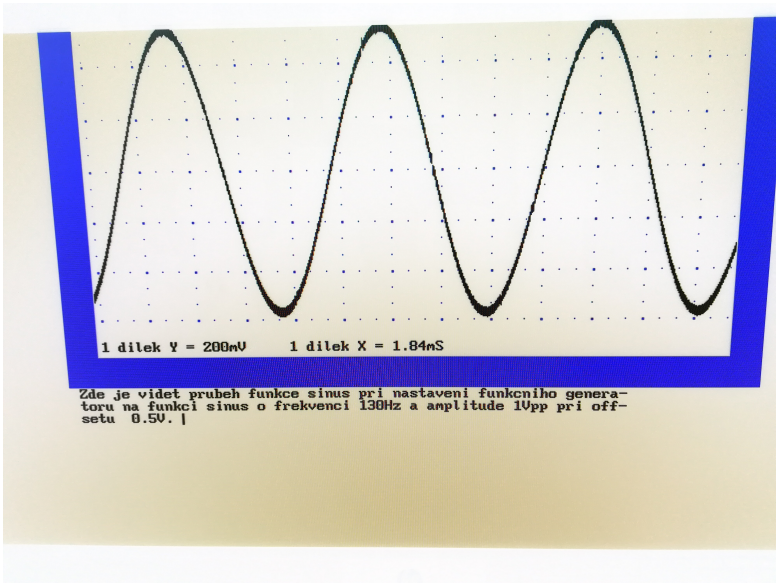
5.4 Ukázka funkčnosti programu

Pro ukázku funkčnosti výpisu znaků jsem vypsal všechny znaky ascii tabulky na terminál monitoru:



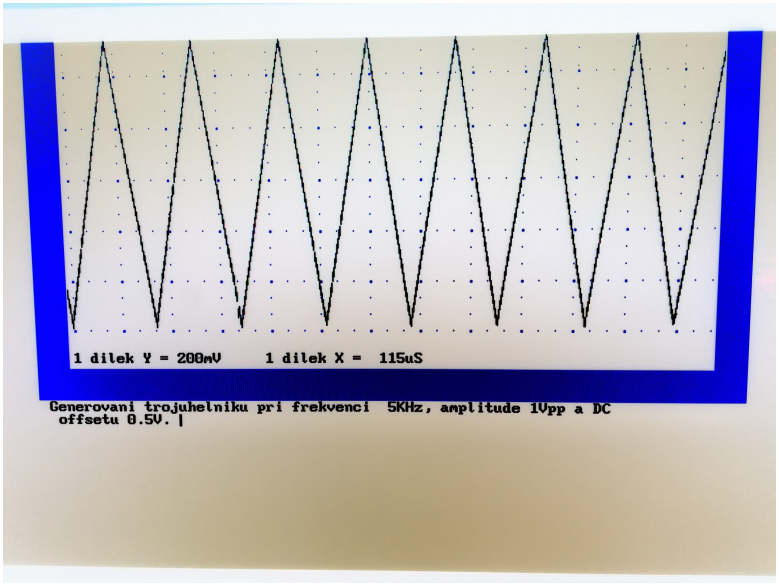
Obrázek 5.14: Ukázka výpisu všech ascii znaků

Pro ukázku funkčnosti osciloskopu jsem použil generátor funkcí GWINSTEK AFG-2225. Na něm nastavoval různé druhy signálu (sinus, trojúhelník, obdélník) při různých frekvencích.
Na obrázku níže je vidět sinusový signál o generované frekvenci 130Hz. Když z obrázku odečtu dobu jedné periody (4 dílky, tj 4x1.84ms) a z ní udělám převrácenou hodnotu, dostanu frekvenci 136Hz, vychází mi chyba cca 4,5 procenta pro vypočtenou frekvenci signálu.



Obrázek 5.15: Ukázka vykreslení sinusového signálu o frekvenci 130Hz

Na dalším obrázku je vidět průběh trojúhelníkového signálu o generované frekvenci 5000Hz. Frekvence vypočtená z odečtených hodnot z obrázku je rovna 5072Hz. Nyní vychází chyba cca 1,5 procenta.



Obrázek 5.16: Ukázka vykreslení trojúhelníkového signálu o frekvenci 5KHz

Tyto chyby odečtené frekvence budou nejspíše způsobeny jak nepřesností měření, tak nepřesností zobrazení a odečtení daného signálu z monitoru. Mnoho dalších zaznamenaných průběhů na osciloskopu jsou k dispozici na přiloženém CD společně s digitální verzí BP.

5.5 Návrhy na vylepšení programu

Program by se dal určitě vylepšit rozšířením osciloskopu na 2 kanály, případně navrhnout hardwarovou děličku napětí pro měření průběhů vyššího napětí, než je 1V. Případně by šlo příkazy z klávesnice měnit režim AD převodníku z unipolárního na bipolární pro měření záporných hodnot napětí. Dále bych rozšířil paměť osciloskopu, zaznamenat by šlo klidně několik průběhů po sobě a po té mezi naměřenými průběhy přepínat. Také by bylo zajímavé přidat funkci pro posun po vertikální ose a zoom pro vertikální osu. Zkrátka by se dal na tomto fpga vytvořit zcela funkční osciloskop pro reálné použití. Bohužel by zřejmě nestačila maximální vzorkovací frekvence převodníku která je jen 1 milion vzorků za vteřinu. Moderní osciloskopy jsou dnes totiž mnohokrát výkonnější.
Co se týče terminálu, bylo by zajímavé implementovat komunikaci po RS232

a jak text (stisknuté znaky), tak naměřená data osciloskopu by šlo posílat do počítače, nebo do jiného zařízení používající RS232, respektive by se dal na osciloskopu zobrazit průběh přijatý z jiného zařízení. Také by šlo zobrazit text přijatý po RS232 na terminálu, případně vykonávat požadované instrukce.
Data by šla také ukládat do statické paměti, například na SD kartu.
Terminál by šel vylepšit rozšířením paměti a vytvořením nějakého adresáře, kam by se dala data napsaná na terminálu, respektive data průběhu osciloskopu ukládat.
Určitě by také šlo měnit barvu jednotlivých znaků, nebo zavést nějaké formátování textu.
Dále by šlo rozšířit ascii tabulku o další znaky (rozšířená tabulka ascii znaků má 256 znaků). Případně přidat volbu klávesnice (česká/anglická).
V neposlední řadě by šlo rozšířit volbu používaného rozlišení a veškeré zobrazované komponenty danému rozlišení přizpůsobit. Mě se podařilo implementovat volitelné rozlišení (od původních 640x480 až po 1440x900). Bohužel zřejmě nestačil výkon PLL bloku, který při implementaci Vivada vykazoval mnoho varování a při zapojení do celkového schématu program nešel implementovat vůbec. Chybu jsem se pokoušel několik hodin opravit avšak neúspěšně.

Kapitola 6

Závěr

Cílem práce bylo napsat program pro čtení stisknutých znaků na klávesnici používající standard PS2. Dále bylo cílem zobrazit tyto znaky na monitoru pomocí standardu VGA.

V této práci se mi podařilo splnit veškeré body zadání. Z klávesnice umí program přečíst veškeré stisknuté klávesy a napsat tak jakýkoliv znak z jednoduché ascii tabulky na monitor. Lze dokonce měnit barvu textu, pozadí i rámečku. Veškeré příkazy jsou zadávané klávesnicí a tlačítka kitu tedy nejsou potřeba. Po dohodě s vedoucím práce jsem volitelně implementoval funkci osciloskopu při proměnné vzorkovací frekvenci. Osciloskop funguje bezchybně ač má různá omezení.

Pro splnění úkolu bylo potřeba se dokonale naučit jazyk VHDL a obsluhu vývojového prostředí Vivado. Také bylo nutné pochopit proces syntézy a implementace pro daný čip FPGA (při opravě chyb syntézy, případně implementace).

Jazyku VHDL a programovatelným polím se také zabývám v praxi. Znalosti nabyté při plnění úkolů této bakalářské práce budu dále rozšiřovat.

Seznam příloh

- Příloha A, seznam literatury
- Příloha B, CD obsahující:
  Materiály/nexys4_rm.pdf
  Materiály/s3estarter_ug.pdf
  Materiály/ug480_7Series_XADC.pdf
  Nexys_VGA_PS2_final - Složka projektu BP Vivado
  Přílohy fotky z měření - Složka s fotkami měřených průběhů
  Přílohy kody - Složka s mnou napsanými kódy VHDL
  bthesis_hodny.pdf - Bakalářská práce v digitální podobě
  Nexys VGA PS2 final.xpr.zip - archivovaný projekt BP z Vivada

Příloha A

Literatura

[1] PS/2 to ascii converter, online,
https://eewiki.net/pages/viewpage.action?pageId=28279002#PS/2KeyboardtoASCIIConverter(VHDL)-Introduction.

[2] Spartan3e Starter Guide, online,
https://reference.digilentinc.com/_media/s3e:s3estarter_ug.pdf.

[3] Standard PS/2, online,
https://en.wikipedia.org/wiki/PS/2_port.

[4] Manual Nexys4, online,
https://reference.digilentinc.com/_media/reference/programmable-logic/nexys-4/nexys4_rm.pdf.

[5] Standard VGA, online,
https://en.wikipedia.org/wiki/Video_Graphics_Array.

[6] Časování VGA, online,
http://www.eecg.toronto.edu/~tm4/rgbout.html.

[7] Časování VGA2, online,
http://www.tinyvga.com/vga-timing/640x480@60Hz.

[8] Manual XADC, online,
https://www.xilinx.com/support/documentation/user_guides/ug480_7Series_XADC.pdf.

[9] Minimal XADC design, online,
http://hamsterworks.co.nz/mediawiki/index.php/Minimal_XADC_design.

[10] Hundreds of examples in VHDL, online,
https://www.youtube.com/user/LBEbooks/feed.