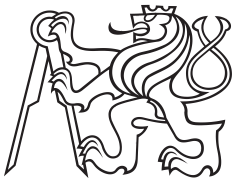
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra mikroelektroniky

Aplikace pro využití PS/2 vstupu a výstupu na LCD displej přípravku Spartan3E v jazyce VHDL

Lukáš Liebzeit

Vedoucí: Ing. Pavel Lafata, Ph.D.
Obor: Elektrotechnika, elektronika a komunikacni technika
Leden 2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Liebzeit** Jméno: **Lukáš** Osobní číslo: **406889**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra mikroelektroniky**

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

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Aplikace pro využití PS/2 vstupu a výstupu na LCD displej přípravku Spartan3E v jazyce VHDL

Název bakalářské práce anglicky:

Creating VHDL Modules and Application of PS/2 Input and LCD Display of Spartan3E Kit

Pokyny pro vypracování:

Seznamte se s přípravkem Xilinx Spartan3E a jeho obsluhou pomocí jazyka VHDL. K přípravku připojte pomocí rozhraní PS/2 standardní klávesnici. Navrhněte a realizujte VHDL kód pro čtení stisknuté klávesy. Doplňte využití vestavěného LCD displeje s řadičem HD44780 na přípravku tak, že na tomto znakovém displeji budou jednotlivé klávesy vypisovány. Vytvořte nezbytné VHDL kódy. Výstupem tak bude jednoduché zobrazení stisknuté klávesy na LCD displeji, eventuálně upravte dle pokynů vedoucího práce.

Seznam doporučené literatury:

[1] Lafata, P. - Hampl, P. - Pravda, M.: Digitální technika. 1. vyd. Praha: Česká technika - nakladatelství ČVUT, 2011. 164 s. ISBN 978-80-01-04914-3.
[2] Pinker, J. - Poupa, M.: Číslicové systémy a jazyk VHDL. Praha : BEN - technická literatura, 2006. 349 s. ISBN 80-7300-198-5.
[3] Digilent: Spartan-3E Reference Manual [online]. Dostupné z: https://digilent.com/reference/programmable-logic/spartan-3e/reference-manual
[4] Popis komunikace a rozhraní PS/2 [online]. Dostupné z: https://www.avrfreaks.net/sites/default/files/PS2%20Keyboard.pdf
[5] Popis řadiče LCD HD44780 [online]. Dostupné z: https://vyvoj.hw.cz//teorie-a-praxe/dokumentace/inteligentni-displeje-a-jejich-pripojeni-k-pc.html

Jméno a pracoviště vedoucí(ho) bakalářské práce:

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

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

Datum zadání bakalářské práce: **24.05.2022** Termín odevzdání bakalářské práce: **10.01.2023**

Platnost zadání bakalářské práce: **19.02.2024**

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

Poděkování

Rád bych poděkoval svému vedoucímu práce za cenné připomínky a za to, že byl k dispozici i ve svém volnu. Dále bych rád poděkoval své babičce a mámě, že mě podporovali po celou dobu studia. Nakonec bych rád poděkoval své přítelkyni Lence a své ještě nenarozené dceři za trpělivost, lásku a motivaci.

Prohlášení

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

V Praze, 2. ledna 2023

Abstrakt

Cílem bakalářské práce je vytvořit aplikaci pro vývojovou desku Spartan3E Starter Kir. Tato aplikace bude uchovávat v paměti znaky napsané na klávesnici. Klávesnice bude připojena přes PS/2 rozhraní. Znaky z paměti budou zobrazeny na externím LCD monitoru připojeném přes VGA rozhraní. Na obrazovce monitoru bude viditelný kurzor, tímto kurzorem půjde posouvat v rámci obrazovky. Při stisku klávesy se daný znak uloží na pozici kurzoru.

Klíčová slova: FPGA, VHDL, PS/2, VGA, BlockRAM

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

Abstract

The goal of my bachelor thesis is to create a standalone application using Spartan3E Starter Kit. This application will allow user to write text on external LCD screen connected by VGA connector. Image on the LCD screen will be generated based on a content of the RAM memory. Content of the memory will be generated by user using keyboard connected by PS/2 interface. There will be a cursor on the screen as well, user will be able to move cursor in the screen boundaries. If user will press a key on the keyboard and it will be a letter, this letter will be written into the cursor position.

Keywords: FPGA, VHDL, PS/2, VGA, BlockRAM

Title translation: Creating VHDL Modules and Application of PS/2 Input and LCD Display of Spartan3E Kit

Obsah

Obrázky

Kapitola 1

Úvod

Jako svoji bakalářskou práci jsem si vybral téma zahrnující FPGA a vývojovou desku Spartan3E neboť s ní mám pár zkušeností. Zadáním bude vytvořit aplikaci zobrazující znaky napsané na klávesnici připojené přes PS/2 rozhraní na externím monitoru připojeného pomocí VGA výstupu. Znaky budou uloženy ve vnitřní paměti RAM vygenerované pomocí „IP cores“ a jejich barva půjde upravovat pomocí přepínačů na desce.

Oficiálně je v zadání použití 2 řádkového LCD displeje, který je součástí vývojové desky Spartan3E, nicméně po konzultaci s vedoucím práce byl jako výstupní displej zvolen VGA výstup s připojeným LCD displejem. Díky tomu lze při zobrazování znaků na externím LCD displeji využít mnohem širší možnosti změny zobrazených znaků, např. jejich barvy a také zobrazovat mnohem větší počet těchto znaků najednou.

Externí LCD displej bude používat pevné rozlišení 800x600 pixelů, bude tedy třeba vytvořit font. Vytvořený font by měl být dost velký, aby šly znaky jasně rozpoznat. Po LCD monitoru bude možno pohybovat kurzorem a kurzor bude odlišitelný od ostatních znaků. Při stisku klávesy s písmenem se písmeno zobrazí na dané pozici kurzoru.

1.1 Motivace

Původně jsem chtěl studovat pouze programování a s tímto cílem jsem se přihlásil na vysokou školu, ačkoliv mé předchozí pokusy o studium nevyšly, získal jsem během nich zápal pro práci s hardwarem. Postupně jsem se začal učit s Arduinem a jinými mikrokontroléry. Později jsem si zkusil i práci s VHDL na desce Basys2. Pracovat s jazykem VHDL a vytvářet tak prototyp hardwaru na této úrovni mi přišlo zajímavé, protože lze vytvářet čip od základů a pro správné fungování je potřeba podrobně rozumět časování signálů a mít potřebné detailní znalosti protokolů.

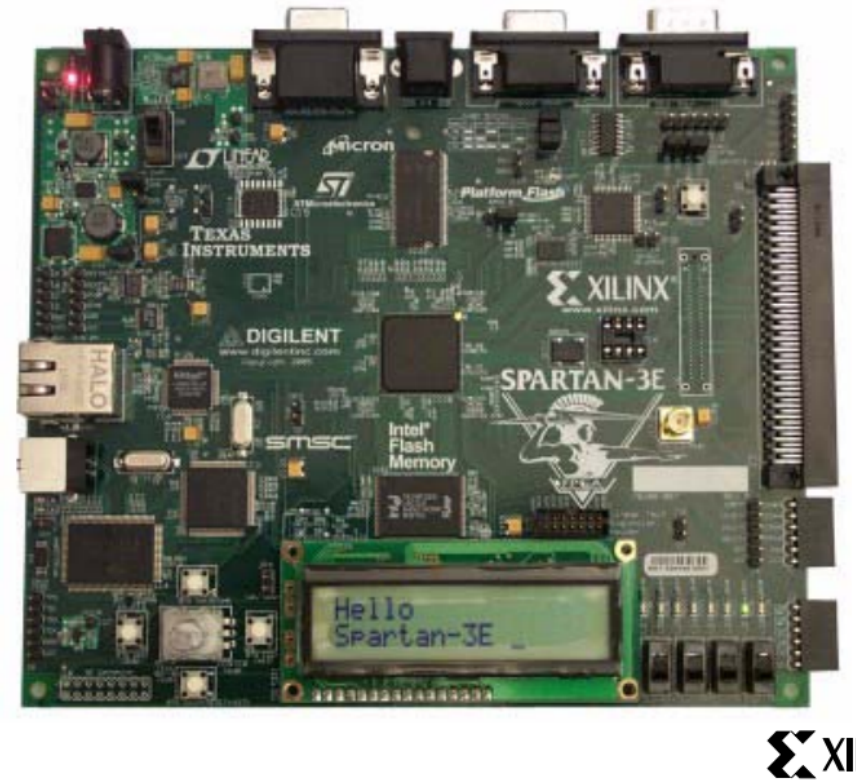
Když jsem tedy přemýšlel, jaké téma si vybrat pro svoji bakalářskou práci a narazil na možnost pracovat s vývojovou deskou Spartan3E, neodolal jsem a dané téma si zapsal. V rámci rešerše jsem měl možnost nastudovat několik prací mých kolegů předchůdců a rozhodl se přinést něco nového do tohoto tématu, proto jsem se rozhodl použít paměť RAM pro ukládání znaků, což předtím nikdo nezkusil.

V rámci studentského projektu předcházejícího bakalářské práci jsem úspěšně implementoval zobrazování znaků z klávesnice na 2řádkovém LCD displeji, který je součástí vývojové desky, proto jsem po dohodě s vedoucím bakalářské práce rozšířil svoje zadání o zobrazování znaků na externím LCD displeji pomocí rozhraní VGA.

Kapitola 2

Teoretický rozbor

Použitá vývojová deska Spartan3E od výrobce Xilinx obsahuje výstup VGA, vstup PS/2 použitelný pro klávesnici a mnoho dalších periferií, které ovšem nebyly v projektu použity. Vývoj probíhal ve vývojovém prostředí ISE Design Suite 14.7 – novější verze vývojového studia již nepodporuje vývojovou desku Spartan3E.



Obrázek 2.1: Vývojová deska Spartan3E Starter Kit, výrobce Xilinx[7]

Vývojová deska Spartan3E Starter Kit obsahuje[7]:

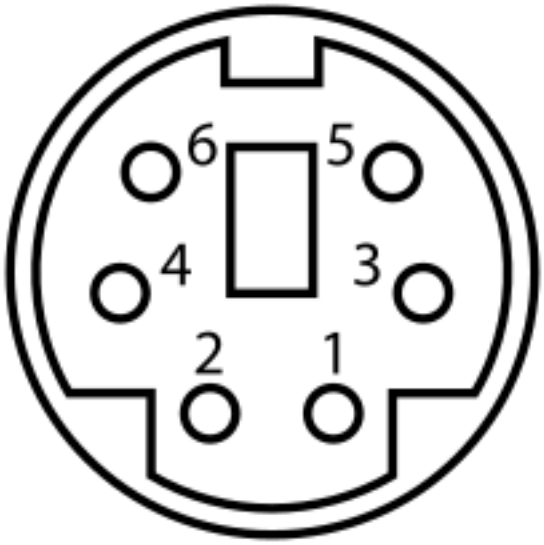
- Čip Xilinx XC3S500E Spartan-3E FPGA;
- Paměť DDR SDRAM, x16 data interface, 100+ MHz, kapacita 64 MByte (512 Mbit);
- paralelní NOR Flash (Intel StrataFlash) s kapacitou 16 Mbyte (128 Mbit);
- SPI serial Flash (STMicro) s rychlostí 16 MBits;
- LCD displej, 2-řádkový, 16-znaků na řádku;
- Port PS/2 pro připojení myši nebo klávesnice;
- Port VGA pro připojení externího monitoru;
- Rozhraní 10/100 Ethernet;
- Rozhraní RS-232, dva 9-pinové konektory (DTE a DCE);
- Rozhraní pro debugování/nahrávání;
- 50 MHz oscilátor;
- EEPROM pro uchování bitstreamu s SHA-1 1-wire rozhraním;
- Rozšiřující konektor Hirose FX2;
- Rozšiřující konecktory Digilent 6-pin (3x);
- Čtyři výstupový Digitálně-Analogový Konvertor (DAC), rozhraní SPI;
- Dvou vstupový, Analogově-Digitální Konvertor (ADC) s programovatelným předzesilovačem, rozhraní SPI;
- Rotační enkodér se stlačitelným tlačítkem;
- LED diody (8x);
- Čtyři posuvné přepínače;

Z tohoto seznamu je vidět, že vývojová deska Spartan3E Starter Kit obsahuje mnoho periferií. Můj projekt bude využívat pouze několik málo z nich.

2.1 PS/2 rozhraní

Ke vstupu znaků je použita starší klávesnice s rozhraním PS/2, toto rozhraní se v dnešní době již moc nepoužívá – bylo nahrazeno rozhraním USB. Po stisknutí klávesy vysílá klávesnice tzv. „scan code“ (některá literatura uvádí „make code“), tento „scan code“ neodpovídá ASCII kódům znaků, proto je potřeba provádět konverzi.

Rozhraní PS/2 se poprvé objevilo v roce 1987 v IBM Personal System/2 – odtud jeho název, poté bylo postupně přejato jakožto standart (v průběhu času prošlo menšími úpravami), v současné době je již zcela vytlačeno USB, které překonalo veškeré jeho výhody (jako poslední byla uváděna implementace ovladačů přímo v BIOSu, ale pro moderní BIOS není problém přehledné grafické rozhraní ovládané myší - natož ovladač USB). Fyzicky je realizováno 6-ti pinovým mini DIN konektorem. Komunikace na tomto rozhraní může probíhat obousměrně. Moje bakalářská práce si vystačila se směrem komunikace klávesnice -> vývojová deska Spartan3E.



Obrázek 2.2: Pin 1 je DATA, Pin 3 je GND, Pin 4 Vcc (+5V), Pin 5 CLK, Piny 2 a 6 jsou nepřipojeny[1]

Pro komunikaci na portu PS/2 jsou v konektoru nejdůležitější vodiče DATA a CLK. Zde je potřeba v implementaci a v simulacích pamatovat, že frekvence CLK klávesnice je mnohem menší, než frekvence CLK desky. Pro simulaci je proto potřeba volit řádově vyšší hodnotu.

Při komunikaci přes PS/2 se přenáší 11-bitová slova skládající se z start bitu, stop bitu, parity bitu a 8-bitových dat. Pokud neprobíhá žádná komunikace je pin DATA a CLK v logickém stavu HIGH. Z toho důvodu je tedy Start-bit roven logické hodnotě LOW, poté následuje 8 bitů tzv. „scan code“, poté následuje bit parity. Typ parity se může lišit v závislosti na typu klávenice (může sejednat o sudou či lichou paritu). Po paritě následuje Stop-bit, ten má opět hodnotu HIGH. Při přijímání dat na rozhraní PS/2 je potřeba pamatovat, že DATA jsou platná na sestupnou hranu signálu CLK z klávesnice.

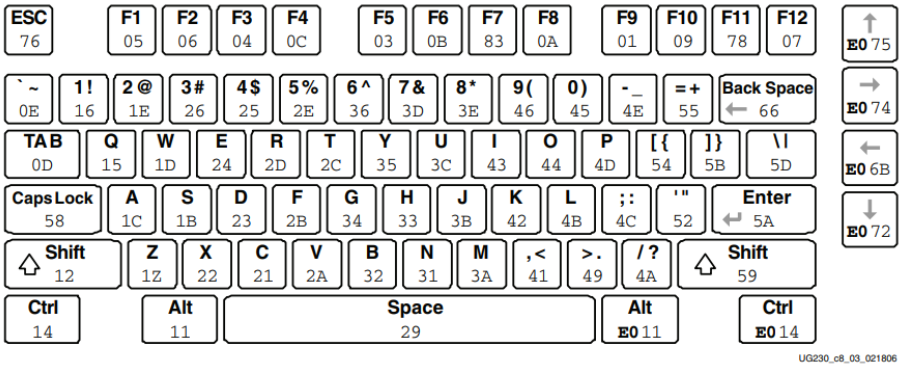
Každá klávesa má jiný “scan code” a to včetně speciálních kláves jako „caps lock“, „num lock“, „shift“... Klávesnice neprovádí žádné jiné operace, kromě posílání „scan code“. Z toho plynou následující omezení:

- Velká/malá písmena mají stejný „scan code“, pro zapsání velkého písmena je potřeba detekovat stisk klávesy shift, popřípadě stisk klávesy „caps lock“ a poté poslat LCD displeji jiný znak pro výpis (ASCII hodnota písmene „a“ je odlišná od písmene „A“);
- Pokud se má rozsvítit LED dioda (například po stisku klávesy „caps lock“), je třeba poslat klávesnici příkaz k rozsvícení dané diody (při stisku klávesy „caps lock“ se pouze pošle příslušný „scan code“, ale příslušná LED dioda se nerozsvítí);

Při uvolnění klávesy pošle klávesnice kód 0xF0 a „scan code“ klávesy která byla puštěna. Většina dohledatelných implementací na internetu a řešení kolegů přede mnou toto zcela ignoruje a stisk klávesy detekují pouze při jejím stisku, kdy je přijat “scan code“ klávesy. Já jsem zvolil řešení pro detekci stisku klávesy při jejím puštění – přijde mi, že oproti jiným implementacím zvládá jeden stisk = jeden znak lépe. Při dlouhém stisku klávesy je „scan code“ posílán opakovaně po určitých intervalech, tím může být stisk klávesy detekován více než jednou.

Některé klávesy mají „scan code“ skládající se z 2B – před samotným „scan code“ je ještě byte 0xE0 (zde zapsán hexadecimálně). Po puštění takové klávesy vyšle klávesnice sekvenci bytů 0xE0F0XX, kde XX je „scan code“ klávesy.

Pro správnou detekci kláves je potřeba vytvořit a odladit buffer, ze kterého se dá detekovat, jestli poslední 3 přijatá 11-ti bitová slova od klávesnice obsahují kombinaci 0xF0 + “scan code”, popř: 0xE0F0 + „scan code“. Toto je řešeno pomocí 33-bitového posuvného registru do kterého se zapisuje na sestupnou hranu signálu CLK z klávesnice.

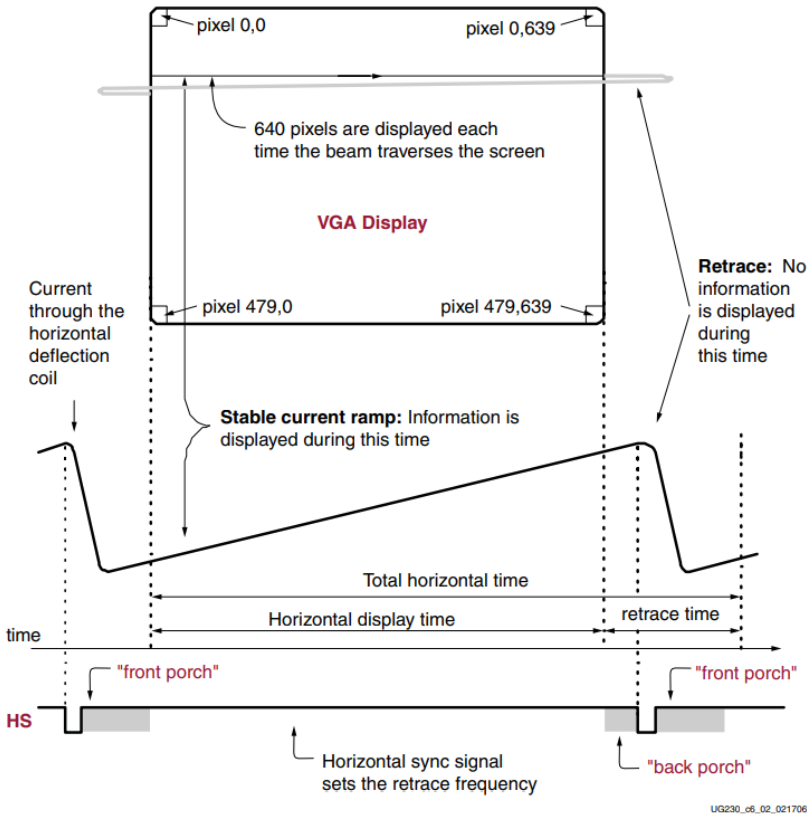


Obrázek 2.3: Klávesnice s příslušnými „scan code“ (zapsány v hexadecimální soustavě)[7]

2.2 VGA rozhraní

Pro rozhraní VGA jsem zvolil pevné rozlišení 800x600 při zobrazovací frekvenci 60Hz, tato frekvence je maximální dosažitelná na vývojové desce Spartan3E při použití vestavěného 50MHz oscilátoru. Pro zobrazení znaku je třeba mít k dispozici i font, který ASCII kód převede na zobrazitelné pixely. VGA rozhraní je další pozůstatek IBM Personal System/2 uveřejněného v roce 1987. I moderní standart HDMI stále používá některé principy, které tento standart přinesl.

V době uvedení VGA se používaly CRT obrazovky, které vykreslovaly obraz pomocí směrovaného paprsku. Pro co nejrychlejší zobrazování začínal paprsek v horním rohu obrazovky a poté vykresloval obraz tak jako jsme zvyklí číst (zleva doprava, shora dolů). Pokud parsek došel na konec řádky, popř. obrazovky, bylo nutné paprsek vrátit. K tomu sloužily signály pro horizontální a vertikální synchronizaci. Tím vzniká jakési „hluché“ místo, které zajistilo, že paprsek měl dost času přesunout se na začátek řádky/obrazovky.



Obrázek 2.4: Průběh paprsku na rozhraní VGA s označenými oblastmi[7]

Z obrázku je patrné, že po a před každým viditelným místem na obrazovce je čas nazývaný „front porch“ a „back porch“, což je v překladu něco jako „přední“ a „zadní veranda“. Během této doby se nic nezobrazuje, jedná se o jakési „hluché“ místo. Používání signálů pro vertikální a horizontální synchrinizaci trvající několik řádků/sloupců přetrval dodnes. Během této doby se nic užitečného neděje, moderní standart HDMI používá tento čas k přenosu audio paketů.

Pro svoji bakalářskou práci jsem chtěl použít jiné rozlišení než základní 640x480 pixelů, které je uváděno v téměř každém dohledatelné ukázce kódu pro vývojovou desku Spartan3E. Po chvíli hledání jsem narazil na tabulku rozlišení a potřebných frekvencí oscilátorů a zde jsem zjistil, že 50MHz oscilátor, který je součástí vývojové desky Spartan3E zvládne rozlišení 800x600 pixelů[3]. Po dalším hledání jsem našel konstanty pro synchronizační pulsy.

Celkové hodnoty pro moje řešení jsou tedy: 1040 bodů na řádce, z toho „front porch“ je 0-176 bodů, poté následuje 800 bodů reprezentující pixely a poté dalších 64 bodů jako „back porch“. Pro vertikální hodnoty máme 43 řádků jako „front porch“, poté 600 řádků viditelného prostoru a poté dalších

21 řádků „back porch“. Tím získáme 692640 bodů pro jeden obraz, to nám při frekvenci hodin 50MHz dovoluje mít obnovovací frekvenci asi 72Hz - to je o malinko více než je třeba, ale monitor Dell na kterém jsem testoval neměl se zobrazením problém.

Pro horizontální synchronizační puls platí, že je v logické hodnotě LOW pro čísla sloupců menší než 120, jinak je ve stavu logické HIGH. Pro vertikální synchronizační puls platí, že je v logické hodnotě LOW pro čísla řádků menší než 6, jinak je ve stavu logické hodnoty HIGH. Na první pohled to vypadá, že synchronizační puls pro vertikální synchronizaci je kratší než ten pro horizontální, ale zde je třeba si uvědomit, že každý řádek obsahuje 1040 pixelů (z nichž některé nejsou vidět). Reálně tak trvá horizontální puls 120 pixelů a vertikální 6*1040=6240, což je řádově větší hodnota.

2.3 Paměť RAM

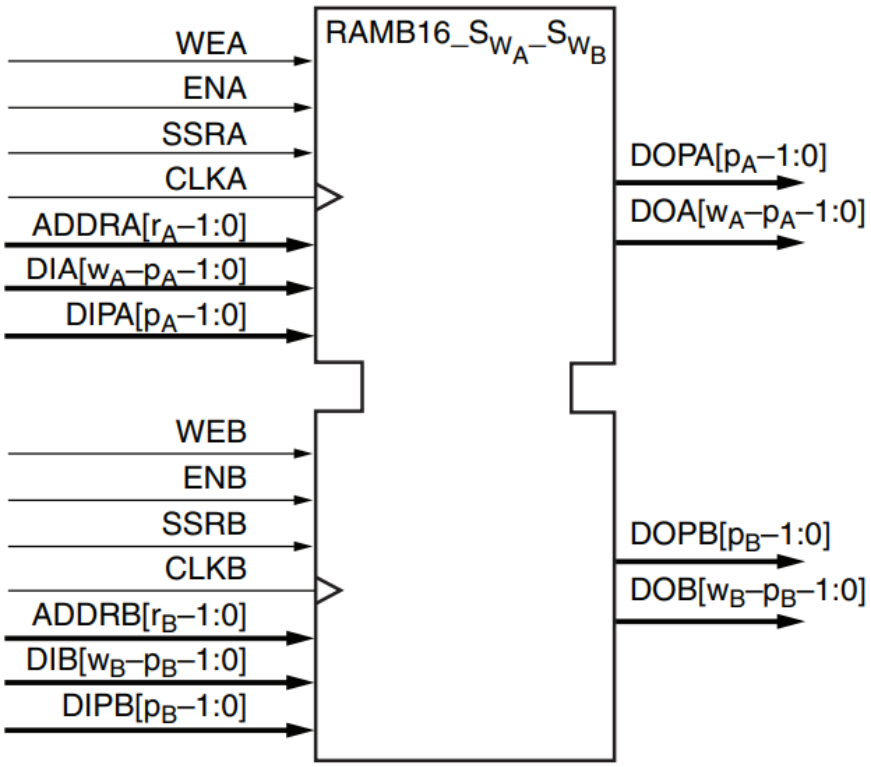
Znaky z klávesnice budou uloženy v paměti RAM, z této paměti budou poté čteny, aby mohly být zobrazeny na LCD displeji. Paměť je generována pomocí „IP cores“, musí mít kapacitu alespoň 7500 znaků (100x75 znaků lze zobrazit při rozlišení 800x600 a fontu 8x8). V paměti jsou uloženy znaky ve formě jejich ASCII kódu, při zobrazení na LCD bude tedy nutné je převádět na pixely dle fontu.

Původní plán byl vytvořit kompletní GRAM, tedy grafickou paměť, která by udržovala barvu všech pixelů. Po přečtení příslušné dokumentace jsem zjistil, že vývojová deska Spartan3E, obsahuje FPGA čip s přesným označením „Spartan3EXC3S500E“[6]. Tento FPGA čip má maximální kapacitu BlockRAM 360 000 bitů. Pro rozlišení 800x600 pixelů je potřeba 800x600x3 (každý pixel má RGB hodnotu), čímž získáme potřebu paměti 1 440 000 bitů. Je na první pohled zřejmé, že kapacita BlockRAM u čipu je nedostačující. I pokud bych vytvořil jednobarevný displej a uchovával pouze hodnotu zda je pixel vybarven nebo nikoliv, tak bych potřeboval 480 000 bitů, což je stále o mnoho více, než kapacita BlockRAM umožňuje.

Další možností realizace GRAM bylo použití DRAM externího čipu, který je součástí vývojové desky Spartan3E, bohužel se mi nepovedlo čip včas zprovoznit a bylo tedy nutné zvolit jiné řešení. Kapacita 360000 bitů odpovídá 45 000B, jeden znak v ASCII kódování má velikost 1B. Při rozlišení 800x600 pixelů a fontu 8x8 tak dostáváme 100x75 znaků, tedy potřebujeme kapacitu paměti 7500B, což je dosažitelné.

Generování RAM paměti proběhlo pomocí nástore „IP cores generator“, který je součástí vývojového studia „ISE Design Suite“. „IP cores“ je zkratkou pro „intelektual property core“ a jedná se o funkční blok logiky nebo dat, který byl vytvořen výrobcem hardwaru a je jím licencován (a otestován).

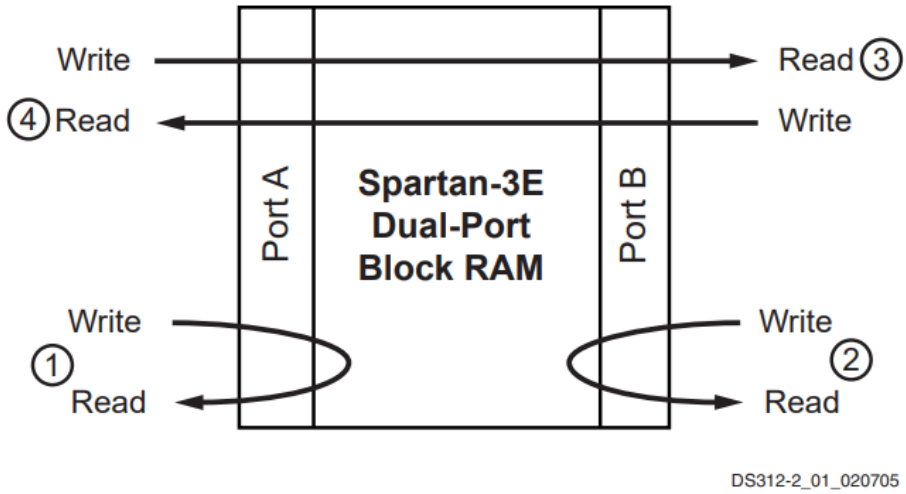
Pro práci s pamětí jsem zvolil paměť typu DualPort, výhodou tohoto typu je snažší práce s pamětí při zápisu a čtení. Jeden port jsem určil jako zapisovací a druhý jako čtecí. Portu pro čtení nastavíme pin „write enable“ na logickou LOW, portu pro zápis nastavíme pin „write enable“ na logickou HIGH. Tyto piny jsou na následujícím obrázku označeny jako „WEA“ a „WEB“. Další užitečná vlastnost při použití DualPort paměti je též viditelná z obrázku níže - každý port má vlastní vstup/výstup a vlastní adresu. To umožňuje kompletně oddělit logiku čtení a zápisu do paměti.



Obrázek 2.5: Prototyp dvoubránové paměti s označenými piny[6]

Následující obrázek ukazuje rozdíl přístupu DualPort (dvoubranová) paměti a SinglePort (jednobranová) paměti. Pro SinglePort paměť se provádí současně zápis i čtení, je tedy nutné správně nastavit pin „write enable“ (přeloženo jako „povolení zápisu“, dále jen jako WE), aby nedošlo k přepsání paměti – při signálu hodin se provede zápis na danou adresu (je-li bin WE v logické HIGH) a současně čtení dat z dané adresy. Chování v případě zápisu jiné hodnoty na vstup než je uložena v paměti lze nastavit – například lze nastavit

aby se paměť přečetla a až poté přepsala nebo přepsala a až poté přečetla. Paměť DualPort umožňuje logicky oddělit čtení a zápis.



Obrázek 2.6: Rozdíl v práci DualPort/SinglePort paměť[6]

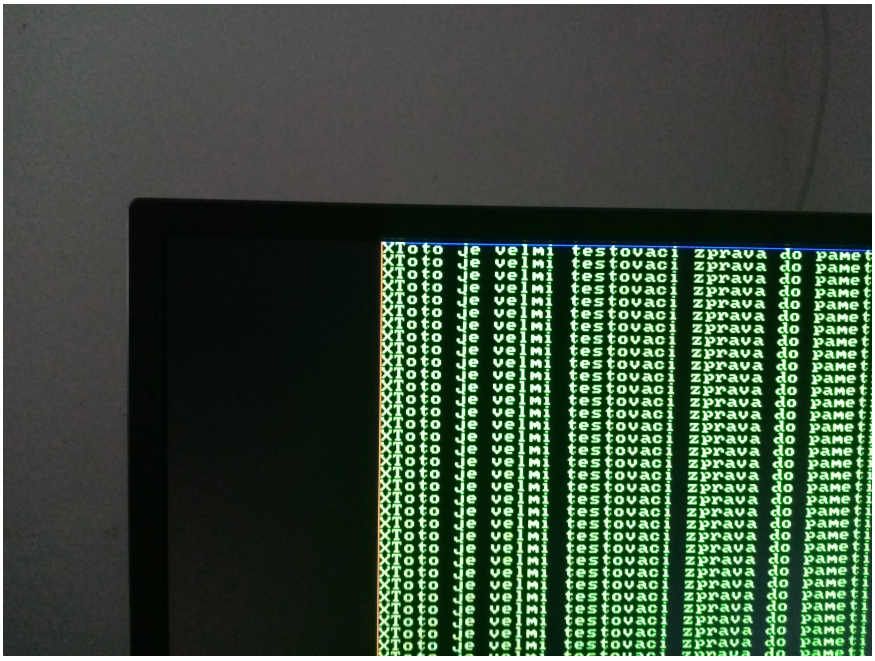
Kapitola 3

Realizace

3.1 VGA rozhraní

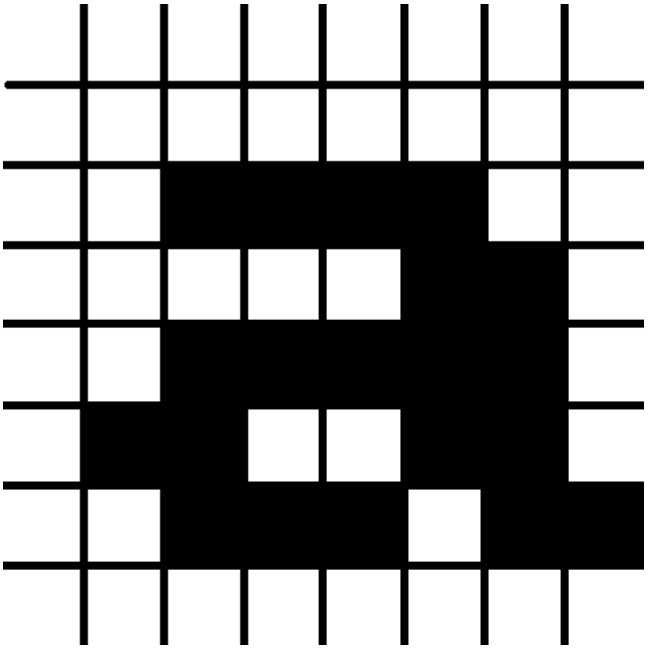
Modul VGA byla první komponenta, kterou jsem pro účely práce vyvíjel. Prvním krokem bylo vykreslit pouze barvu v aktivní oblasti (aktivní oblastí myslím část, která je viditelná na obrazovce). Pro snazší práci jsem si založil v daném procesu proměnné, které udržují x a y souřadnici na obrazovce, tím získáme abstrakci, kde bod [0, 0] na obrazovce bude odpovídat hodnotám x = 0 a y = 0. Tento druh abstrakce o mnoho ulehčil další práci s obrazem.

Dále jsem potřeboval zkontrolovat, zda se správně zobrazuje viditelná část obrazu na monitoru. Toho jsem docílil tak, že jsem na krajních hodnotách vykreslil jednobarevnou čáru o tloušťce 1 pixel. Po nahrání na přípravek jsem poté provedl pozicování na monitoru a v momentě kdy byly všechny čáry viditelné jsem věděl, že obraz bude správně zarovnán (po nastavení na monitoru jsem zkusil desku restartovat a krajní čáry byly viditelné bez nastavování).



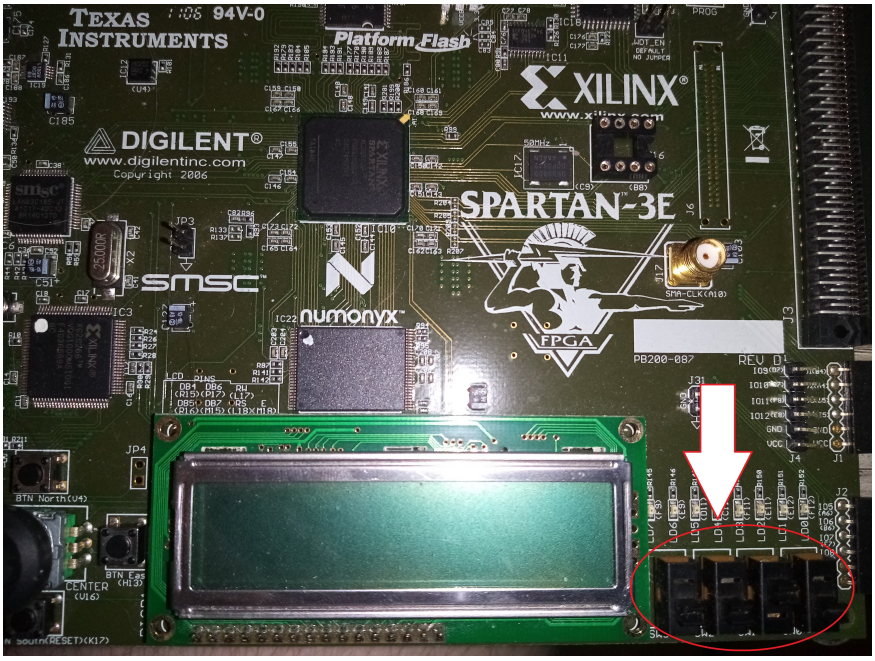
Obrázek 3.1: Ukázka výpisu znaků na externí LCD obrazovku, zde jsou stále vidět čáry pro zarovnání

Dalším krokem bylo vytvoření a vykreslení alespoň jednoho znaku. Zde jsem začal rozmýšlet font, kterým se budou znaky zobrazovat. V první fázi jsem vyzkoušel, jestli font 8x8 bude dostatečný oproti například 10x10. Oba fonty se zdály slibné, takže jsem je zatím ponechal a začal jsem řešit přepočet pixelu na hodnotu pozice vykreslovaného znaku. Zde jsem objevil, že syntéza VHDL nedovoluje provádět operaci dělení a zbytek po dělení pro jiná čísla než mocniny 2, tím bylo rozhodnuto, že bude použit font 8x8. Zobrazení malého „a“ ve fontu je vidět na dalším obrázku.



Obrázek 3.2: Ukázka fontu 8x8, zde malé „A“

Font je implementován jako pole std_logic_vektor, tím jsem získal možnost adresovat přímo kódem ASCII znaku a odpadla tak nutnost řešit daný problém jinak (například podmínkou, která by patrně vytvořila složitější obvod). Pokud se má vykreslit pixel, komponenta se podívá do fontu a pokud je na daném místě logická hodnota HIGH vypíše barvu. Toto pole std_logic_vectorů zabírá bohužel mnoho řádků a tak jsem font převedl do nové komponenty, která funguje jako kombinační logika. Barva zobrazení je do komponenty předávána jako std_signal z nadřazené komponenty, nadřazenou komponentou je v našem případě top_module.vhd (bude představen dále), do něj jsou předávány hodnoty RGB z přepínačů umístěných na desce.



Obrázek 3.3: Umístění posuvných přepínačů na desce (hned vedle je umístěný LCD displej 2x16 znaků)

Další fází bylo zobrazit celý řádek textu, kde řádek bude obsahovat různé znaky tak jak budou později předány načtené znaky z paměti. Zde jsem zjistil, že je nejvýhodnější předat modulu String o kapacitě 100 znaků. Tento vstup do komponenty sice zabere 100B místa, ale poté se díky němu zobrazí 8 řádek bez nutnosti čtení paměti a je I dosaženo synchronizace: komponenta dostane celý řádek a ten vykreslí.

Samotná implementace je provedena pomocí dvou procesů. První proces pouze počítá hodnoty pro horizontální a vertikální pozici. Na jejich základě se provede přiřazení do výstupů horizontální a vertikální synchronizace. Druhý proces provádí vykreslování pixelu. Pokud jsou hodnoty horizontální a vertikální pozice v rozmezí odpovídající viditelné ploše na monitoru, tak se provede abstrakce do proměnných x, y. Pomocí hodnot x, y se poté spočítá, jaký znak a jaký pixel v něm chceme aktuálně zobrazit. Toho se docílí tak, že se dané hodnoty porovnají s fontem. Z pozice znaku víme, kolikátý znak na řádce se zobrazuje (to funguje jako adresa do pole s fontem) a dále víme jaký pixel ve znaku chceme zobrazit. Pokud chceme daný pixel zobrazit, tak do výstupních signálů pro RGB přiřadíme vstupní signály RGB (nastavitelné z posuvných přepínačů). Poslední co se porovnává je zda daná pozice x, y je pozicí kurzoru, u ní se zvolí jiné zbarvení.

3.2 PS/2 rozhraní

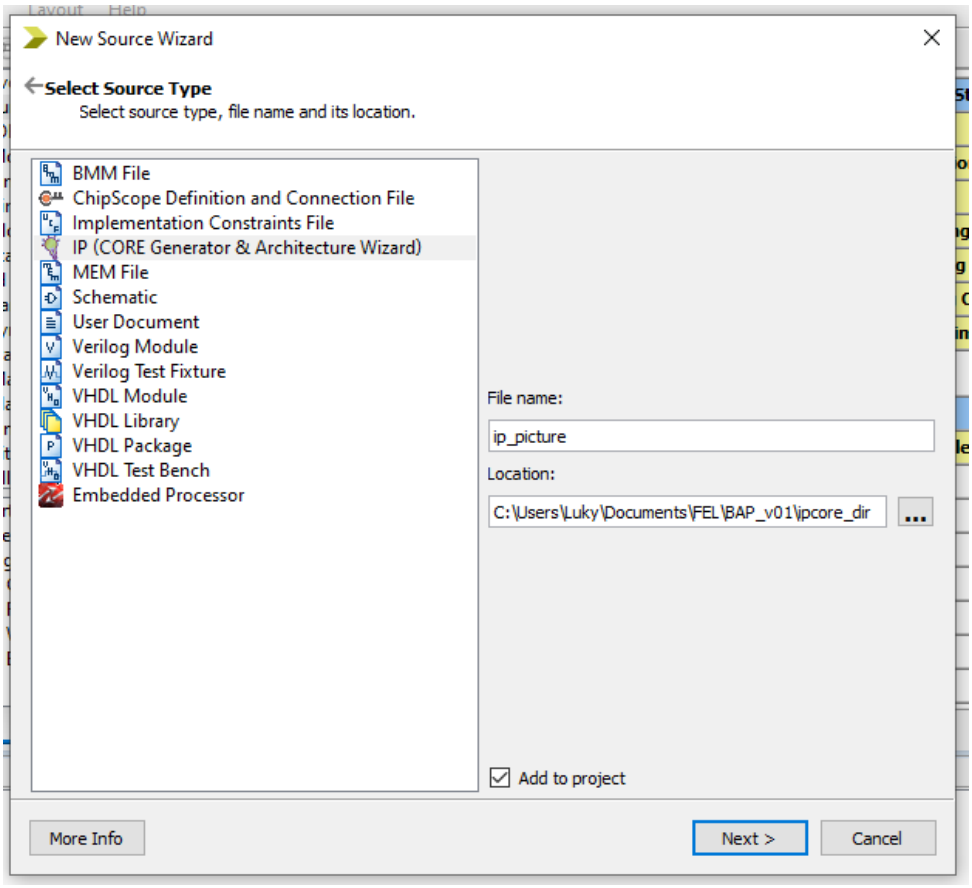
Jak již bylo uvedeno v teoretickém úvodu, rozhodl jsem se pro detekování stisku klávesy pomocí jejího "release code". Některé klávesy mají "release code" skládající se z 33 vyslaných bitů, protože každý oktet má svůj start bit, stop bit a parity bit. Pro toto přijímání byl implementován posuvný buffer s kapacitou 33 bitů.

Rozhraní PS/2 je řádově pomalejší než oscilátor na vývojové desce Spartan3E, proto je v komponentě proces, který počítá počet pulsů CLK z klávesnice a po každých přijatých 11 bitech provede vyhodnocení bufferu, pokud buffer obsahuje na správném místě 0xF0 byte a všechny start bity/stop bity souhlasí, přiřadí se do 16bitového výstupu kód stisknuté klávesy a nastaví se danou dobu výstupní signál „platnosti". Tento signál zajistí, že nadřazený modul nebude chybně detekovat stisknutí klávesy několikrát. V případě, že kód klávesy je pouze 8bitový, tak se zbývající bity vynulují. Tuto funkcionalitu implementují dva procesy: jeden provádí počítání CLK pulsů z ps2 rozhraní a posouvání registru. Druhý proces nastaví výstupní vektor, pokud při 11. cyklu CLK z ps2 rozhraní detekuje v bufferu „release code" 0xF0 na očekáváném místě a všechny start i stop bity mají očekávanou hodnotu (ochrana proti poškozenému signálu).

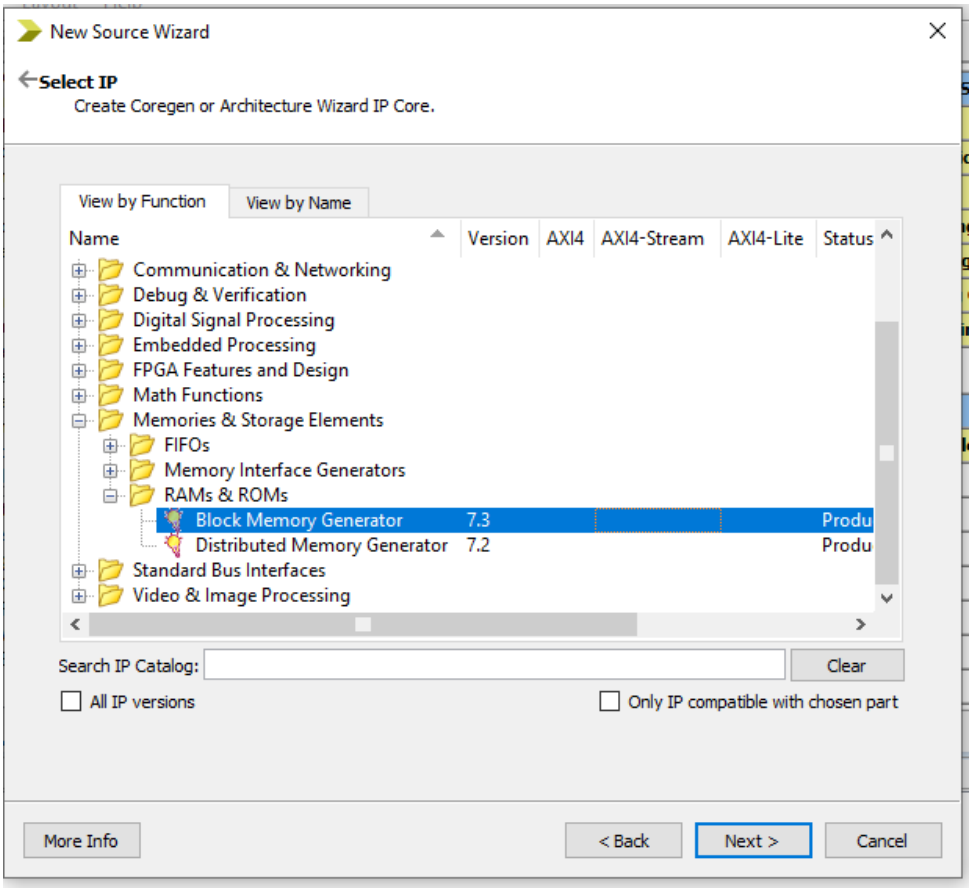
Jeden z výstupních signálů je určen pro detekování platnosti stisku klávesy. Tento signál je aktivní 2000 pulsů oscilátoru. Jedná se o 50MHz oscilátor, který je umístěn na vývojové desce Spartan3E StarterKit. Jeden puls trvá 0,00000002 s, to je 20ns. Celková doba trvání platnosti klávesy je tedy 2000*20=40 000 ns, to je 40 mikrosekund. Toto je dostatečně krátká doba aby se stačila nějaká klávesa stisknout dvakrát a zároveň dostatečně dlouhá doba, aby klávesu stačila nadřazená komponenta zpracovat. Tato doba platnosti je počítána separátním procesem.

3.3 Memory controller

Vytvoření "IP core" se provádí stejně jako přidávání jakékoliv jiné komponenty, zvolíme "Add new source" a vybereme možnost "IP (CORE Generator TODO: amperesand Architecture Wizard)" – viz obrázek, následně se otevře okno, ve kterém je potřeba vybrat ve stromové struktuře jaký typ "IP core" chceme vytvořit, pro naše účely zvolíme "Memories & Storage Elements" > "RAMs & ROMs" > "Distributed Memory Generator".

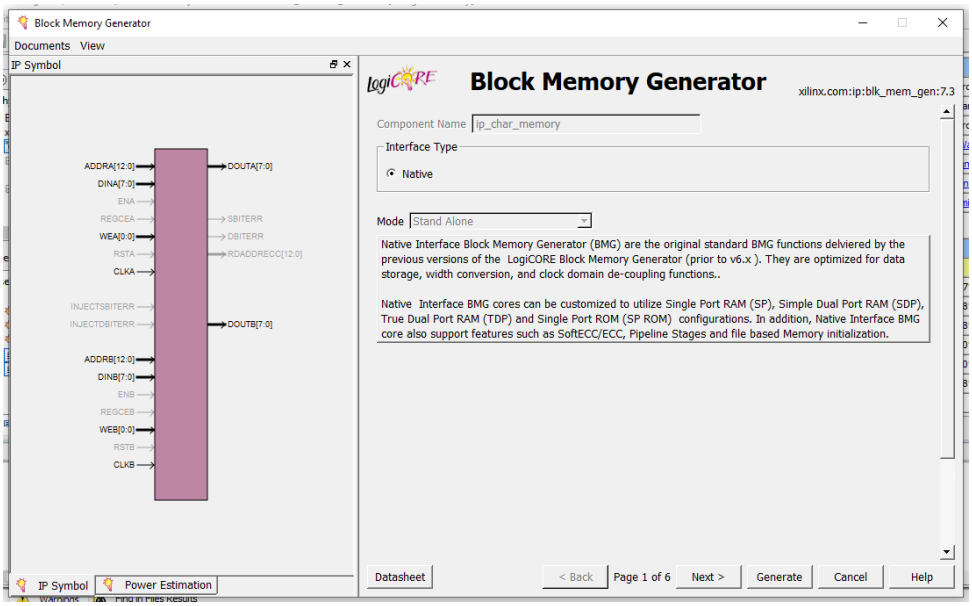


Obrázek 3.4: Přidání nového „IP Core“ do projektu

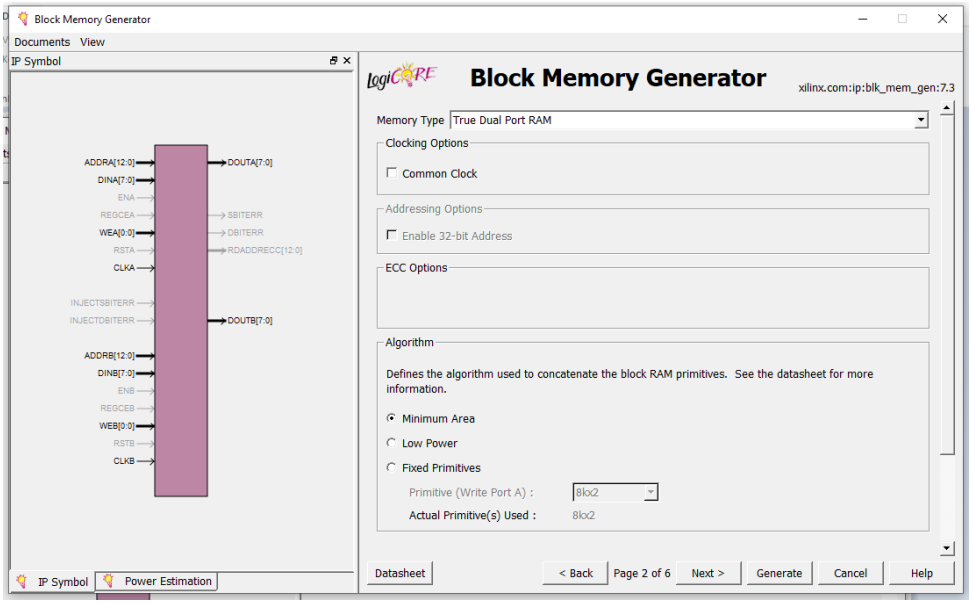


Obrázek 3.5: Výběr typu generovaného „IP Core“

Typ paměti jsem zvolil jako distribuovaný (to znamená, že větší paměť se poskládá z více menších bloků), protože původně měla paměť sloužit jako paměť pro pixely a proto jsem počítal s její mnohem větší kapacitou. Jakmile je „IP core“ vytvořeno, otevře se průvodce, ve kterém lze nastavit všechny požadované parametry generovaného „core“ – v našem případě paměti. Následující sekvence obrázků ukazuje nejdůležitější kroky.

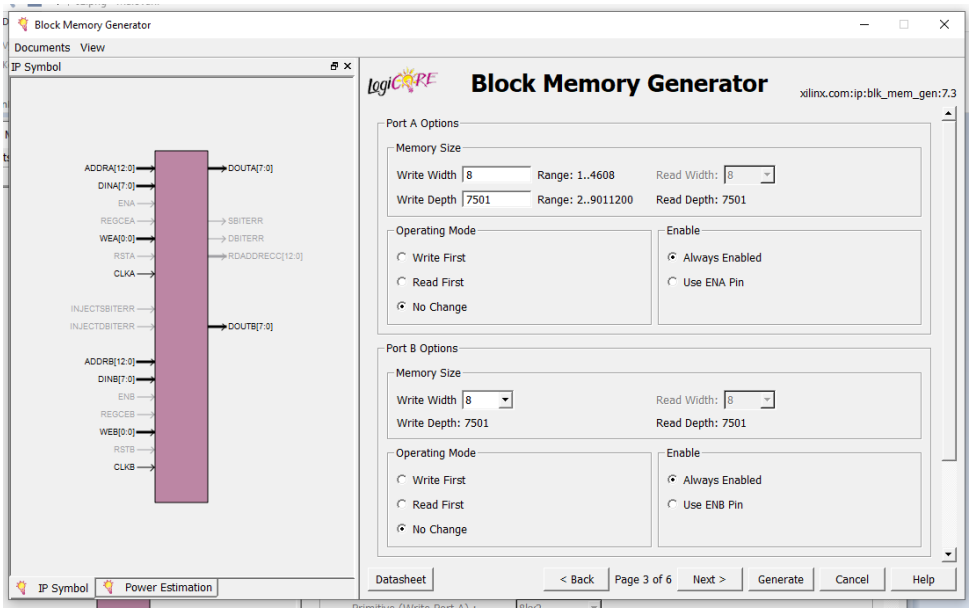


Obrázek 3.6: Volba parametrů pro generovanou paměť – nastavení pomocí „click-through“



Obrázek 3.7: Volba parametrů pro generovanou paměť (zde se vybírá zda bude paměť DualPort/SinglePort)

První krok je samozřejmě zvolení typu paměti, zde jako “True Dual Port RAM”. Všimněme si obrázku v levé části – ten ukazuje jaké piny budou použity podle aktuálního nastavení, pokud se jedná o sběrnici, tak ukazuje i její velikost.



Obrázek 3.8: Volba parametrů pro generovanou paměť – volba velikosti buňky a paměti

Zde lze nastavit velikost jedné buňky a "hloubku" paměti, tedy počet buněk. Volba velikosti jedné buňky (v bitech) ovlivňuje velikost signálů DIN a DOUT. Volba hloubky paměti zase ovlivňuje velikost signálu ADDR. Hloubka paměti se volí jako počet buněk – velikost signálu ADDR je poté nejbližší vyšší mocnina 2.

3.4 Komponenta pro obsluhu paměti

Po vygenerování "IP core" nabízí vývojové prostředí template pro použití daného "IP core", tento template má stejný formát jako inicializace jakékoliv jiné komponenty.

```
ip_char_memory_0 : ip_char_memory
  port map (
    clka => clk50_in,
    wea => "1",
    addra => addr_write_in,
    dina => data_in,
    douta => douta,
    clkb => clk50_in,
    web => "0",
    addrb => addr_read,
    dinb => dinb,
    doutb => data_out
  );
```

Ukázka inicializace Takto vypadá inicializace vygenerovaného "IP core", jak je vidět – neliší se od inicializace jakékoliv jiné komponenty, pro srovnání uvádím dále inicializaci komponenty pro vykreslení obrazu:

```
vga_module_0 : vga_module
port map(
clk50_in    => clk50,
line_text   => text_from_mem,
        red         => vga_red,
       green       => vga_green,
       blue        => vga_blue,
       hs_out      => vga_hs_inner,
       vs_out      => vga_vs_inner,
```

```
red_in      => btns(3),
blue_in     => btns(2),
green_in    => btns(1),
line_number => line_number,
cur_pos_x   => pos_x,
cur_pos_y   => pos_y
);
```

Komponenta dále obsahuje 2 procesy, jeden pro zápis do paměti a druhý pro čtení z paměti. Proces pro zápis do paměti provádí přepočet pozice znaku na adresu. Adresa v paměti pro dané pozice x, y se spočte jako: adresa = (y * 100) + x + 1, každý řádek má 100 znaků, proto y (které reprezentuje řádek, na kterém je znak) musíme vynásobit 100, poté musíme přičíst x (pozice znaku na dané řádce) a poté ještě musíme přičíst 1, toto přičtení je nutné, protože adresa 0 je patrně rezervovaná – během implementace jsem nebyl schopen z adresy ani číst, ani na ni zapsat.

Proces pro čtení z paměti obsahuje opět výpočet adresy, zde ovšem vystačíme pouze s číslem řádku (celý řádek se načítá z paměti a předává se komponentě pro zobrazení obrazu na monitoru). Čtení z paměti se provádí pouze pokud je číslo řádku dělitelné 8 beze zbytku. Tím provádím jistý druh optimalizace, protože font má 8 řádků a provádět nové čtení z paměti během doby kdy se daný řádek zobrazuje by mohlo způsobit problémy.

Čtení z paměti poté trvá 102 cyklů, kdy postupně načítáme znaky do bufferu a pokud je všech 100 znaků načtených, tak obsah bufferu přiřadíme do výstupu. Při čtení znaků do bufferu musíme počítat s prodlevou během které se znak čte z paměti (není dostupný hned). Experimentálně jsem zjistil, že počet cyklů nutných pro čtení je 2. První obrázek ukazuje obsah paměti, včetně adres buněk. Na druhém obrázku je vidět, jaká adresa se nastavuje a z jaké reálné adresy jsou dostupná data. Písmeno „X“ během prvních dvou cyklů značí, že tato data mohou být jakákoliv (odpovídají předchozímu čtení).

clk	0	1	2	3	4	5	6	7
setting address	1	2	3	4	5	6	7	8
output of memory	X	X	H	E	L	L	O	

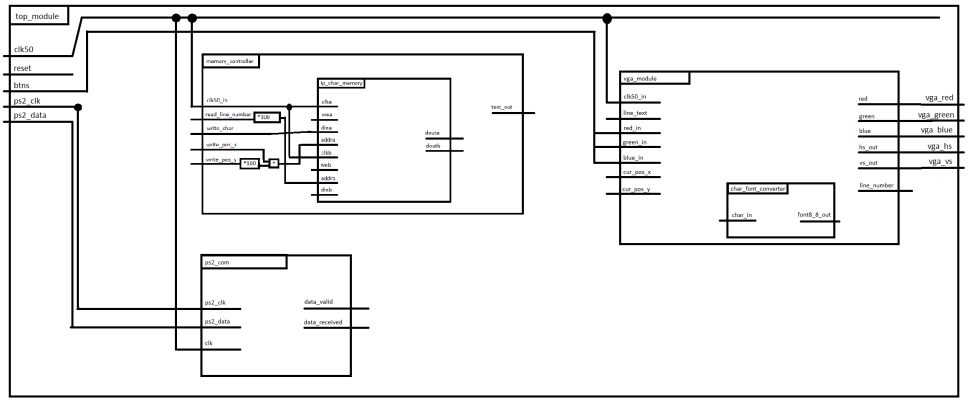
Obrázek 3.9: Obsah paměti pro ukázku čtení

content of memory							
address	1	2	3	4	5	6	7
value	H	E	L	L	O		M

Obrázek 3.10: Ukázka čtení z paměti, „X“ je neznámý stav

3.5 Komponenta top_module.vhd

Komponenta top_module je nejvyšším modulem celého řešení, spojuje dohromady komponentu pro generování obrazu na externí LCD monitor, komponentu pro příjem „scan code“ z klávesnice a komponentu pro obsluhu paměti. Zároveň je zde implementována řídící logika.



Obrázek 3.11: Diagram všech komponent, nepřipojené vstupy/výstupy jsou připojeny na některých z vnitřních signálů

Jediný proces, který tato komponenta má, čeká na stisk klávesy, pokud je klávesa stisknuta a nebyla zpracována již předtím, tak stisknutou klávesu proces vyhodnotí a podle toho provede akci.

Pokud byla stisknuta šipka, provede se příslušný posun kurzoru na monitoru změnou jeho souřadnic. Změna souřadnic obsahuje podmínky, aby nebylo možné s kurzorem vyjet mimo zobrazovanou plochu.

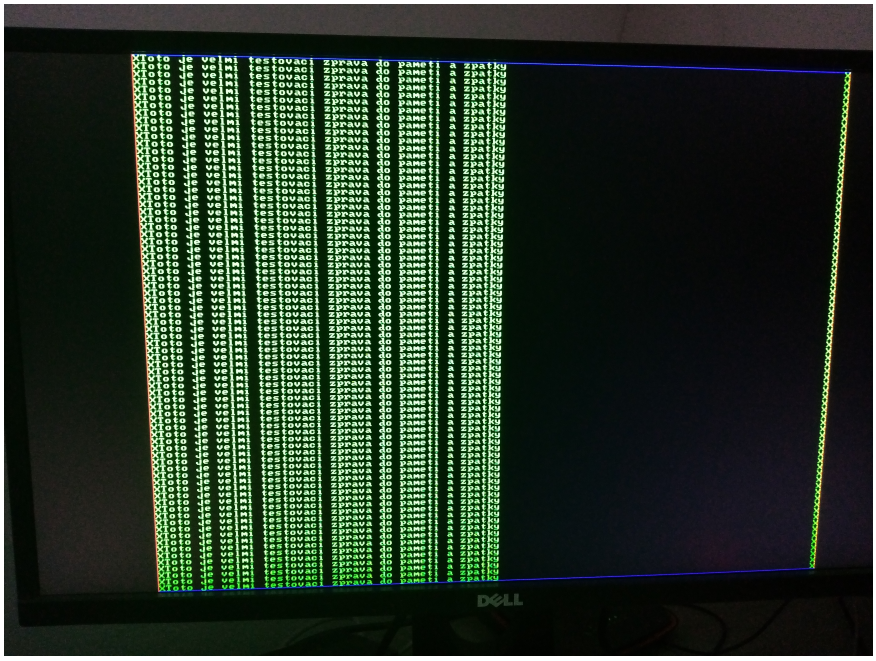
Pokud byl stisknut enter, tak se do kurzorové souřadnice x přiřadí 0 a souřadnice kurzoru y se zvedne o 1. Toto simuluje stisknutí enteru v konzolových aplikacích. Díky takto implementovanému řešení by bylo snadné rozšířit aplikaci práci o zpracování jednoduchých příkazů a vytvořit tak jednoduchý terminál. Pokud byla stisknuta klávesa a-z, provede se převedení jejího „scan code“ na ASCII znak a poté se provede zápis do paměti. Jakmile

je zápis hotov, k souřadnici kurzoru x je přičtena 1. Pokud by souřadnice kurzoru měla přesáhnout maximální hodnotu, tak ji nastavíme na 0 a přičteme 1 k souřadnici y. Toto simuluje automatické odřádkování při psaní dlouhého textu.

Kapitola 4

Závěr

Během mé bakalářské práce se mi povedlo s pomocí FPGA čipu vytvořit jednoduché psaní na externím LCD monitoru a zprovoznit RAM paměť na čipu Spartan3E.



Obrázek 4.1: Ukázka čtení z paměti, zde ještě nebylo řešeno odřádkování

Bohužel se nepovedlo zprovoznit dostupný čip DRAM na vývojové desce. Zprovoznění tohoto čipu a vytvoření rozhraní pro další využití této rychlé 512MB paměti bude zcela jistě cílem mé další práce, neboť na internetu není dostupný žádný tutoriál, který by ukazoval jak danou paměť použít, což

představuje výzvu.

Dalším rozšířením by zcela určitě mělo být vylepšení práce s klávesnicí. V momentálním stavu není odezva ideální – při testování bylo zjištěno, že klávesu je nutné zmáčknout i několikrát, než se stisk projeví na externím LCD monitoru. Také by bylo dobré zprovoznit oboustrannou komunikaci s klávesnicí. Momentální implementace umožňuje získat z klávesnice stisknutí klávesy, avšak neumí rozsvítit LED diody „CapsLock“ a „Scroll lock“, dalším vylepšením na klávesnici by mohla být detekce funkčních kláves Shift, Ctrl a implementace reagování na klávesové zkratky.

Co se týče vylepšení VGA výstupu, je třeba si uvědomit, že i rozlišení 800x600 je dnes dávno překonané, zde ovšem narážíme na hodnotu oscilátoru, kterým je vývojová deska Spartan3E Starter Kit osazena. Pokud by se dalo pomocí PLL vytvořit generátor hodin o vyšší frekvenci – mohlo by se zvýšit rozlišení a tím i počet zobrazovaných znaků. Toto rozšíření by stálo do budoucna za zvážení.

Jak je vidět z dosažených výsledků, tak zadání práce jsem splnil – dokonce i mírně překonal, doufám, že tato práce pomůže budoucím kolegům s podobným zadáním.

Příloha A

Literatura

[1] PS/2 port, online
Dostupné z: https://en.wikipedia.org/wiki/PS/2_port.

[2] VGA connector, online
Dostupné z: https://en.wikipedia.org/wiki/VGA_connector.

[3] VGA timing, online
Dostupné z: https://projectf.io/posts/video-timings-vga-720p-1080p/#svga-800x600-60-hz.

[4] LAFATA, Pavel, Petr HAMPL a Michal PRAVDA. Digitální technika.
V Praze: České vysoké učení technické, 2011. ISBN isbn978-80-01-04914-3.

[5] KRÁL, Jiří. Řešené příklady ve VHDL: hradlová pole FPGA pro začátečníky.
Praha: BEN - technická literatura, 2010. ISBN isbn978-80-7300-257-2.

[6] Spartan-3E FPGA Family Data Sheet, online
Dostupné z: https://hwlab.fit.cvut.cz/_media/pripravky/fpga/basys2/ds312.pdf.

[7] Spartan-3E Starter Kit Board User Guide, online
Dostupné z: https://digilent.com/reference/_media/s3e:s3estarter_ug.pdf.