VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

VYUŽITÍ ROZHRANÍ HDMI NA VÝUKOVÉM KITU MINERVA

EXPLOITATION OF HDMI INTERFACE ON MINERVA EDUCATIONAL PLATFORM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **JAN MAREK**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. VÁCLAV ŠIMEK**
SUPERVISOR

BRNO 2017

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačových systémů Akademický rok 2016/2017

Zadání bakalářské práce

Řešitel: **Marek Jan**
Obor: Informační technologie
Téma: **Využití rozhraní HDMI na výukovém kitu Minerva**
Exploitation of HDMI Interface on Minerva Educational Platform
Kategorie: Vestavěné systémy

Pokyny:

1. Podrobně se zabývejte aktuálním zněním specifikace rozhraní HDMI pro přenos obrazového signálu a sběrnice I2C.
2. Seznamte se s rodinou FPGA obvodů Spartan-6 a jazykem VHDL pro návrh číslicového hardware
3. Nastudujte obvodové zapojení výukové platformy Minerva, kde se zaměřte především na část související s HDMI rozhraním a obvodem TFP410.
4. Navrhněte koncepci řadiče pro komunikaci s obvodem TFP410, který umožní přenos obrazových dat i konfiguračních parametrů.
5. Proveďte implementaci navrženého řadiče v jazyce VHDL pro FPGA obvod Spartan-6 na výukovém kitu Minerva.
6. Připravte demonstrační aplikaci, která prokáže funkčnost vámi navrženého řešení.
7. Zhodnoťte dosažené výsledky a pokuste se navrhnout případná vylepšení či rozšíření.

Literatura:

- Dle pokynů vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- Splnění bodů 1 až 4 zadání.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Šimek Václav, Ing.**, UPSY FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 17. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačových systémů a sítí
612 66 Brno, Božetěchova 2

L.S.

prof. Ing. Lukáš Sekanina, Ph.D.
vedoucí ústavu

Abstrakt

Cílem této práce je prozkoumat možnosti přenosu obrazu přes rozhraní HDMI. Na základě zjištěných informací je navrhnuto a realizováno ovládání obvodu TFP410 včetně generování obrazových dat pro tento obvod. Cílovou platformou je školní vývojová deska Minerva, která je osazena mikrokontrolérem Kinetis K60 od společnosti NXP (dříve Freescale) a programovatelným logickým hradlovým polem Spartan 6 od Xilinx. Navrženým řešením ovládání rozhraní je řadič, který vytváří komunikační rozhraní mezi mikrokontrolérem a digitální HDMI/DVI vysílačem TFP410 od firmy Texas Instruments. Mikrokontrolér komunikuje s řadičem pomocí sběrnice FlexBus. Řadič s obvodem TFP410 komunikuje pomocí sběrnice I2C. Rozhraní řadiče i obslužné knihovny v mikrokontroleru musí umožňovat jednoduchou použitelnost. Funkcionalita řadiče a knihovny je demonstrována vzorovými aplikacemi (hra Tetris a Sokoban).

Abstract

The aim of this thesis is to explore the possibilities of video transfer via HDMI interface. Based on the results a circuit driver TFP410 will be designed, and both the circuit driver and generator of video data for this circuit will be implemented. The target platform in this thesis is school learning developing kit Minerva with MCU Freescale (NXP) Kinetis K60 and FPGA Xilinx Spartan 6. The designed solution of the controller is a device controller which transfers communication from MCU to HDMI/DVI digital transmitter TI TFP410. MCU communicates with the device controller over FlexBus. All communication between the transmitter and the controller is realized over I2C bus. The device controller interface and MCU driver library must provide easy usage. The functionality of both the controller and MCU library is demonstrated on two demo applications (games Tetris and Sokoban).

Klíčová slova

HDMI, Kinetis, K60, NXP, Minerva, řadič, mikrokontrolér, TFP410, FlexBus, obraz

Keywords

HDMI, Kinetis, K60, NXP, Minerva, driver, microcontroller, TFP410, FlexBus, video

Citace

MAREK, Jan. *Využití rozhraní HDMI na výukovém kitu Minerva*. Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Šimek Václav.

Využití rozhraní HDMI na výukovém kitu Minerva

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Václava Šimka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jan Marek
13. května 2017

Poděkování

Chtěl bych poděkovat panu Ing. Šimkovi za pomoc při řešení problému s nahráváním firmwaru do FPGA a pomoc s hledáním řešení některých problémů.

Obsah

Kapitola 1

Úvod

High-Definition Multimedia Interface (HDMI) je dnes nepostradatelným zobrazovacím rozhraním, které oproti starším rozhraním přináší velké výhody. S postupným vývojem se zvětšuje přenosová frekvence, která umožňuje větší propustnost celého rozhraní. HDMI je plně digitální a nabízí velké množství využití. Například lze kromě video a audio signálů přenášet i ethernet.

Splněným cílem této práce je vytvořit řadič obvodu TFP410 pro ovládání rozhraní HDMI a obslužnou knihovnu pro mikrokontrolér na cílové výukové školní platformě Minerva. Cílová platforma je osazena mikrokontrolérem od společnosti NXP(Freescale) - Kinetis K60 a programovatelným hradlovým polem Spartan 6 od firmy Xilinx. Předpokládáme, že řadič a knihovna bude používaná v dalších studentských projektech, proto se budeme snažit přizpůsobit rozhraní knihovny a řadiče tak, aby bylo možné jednoduše modul použít a nevznikaly tak problémy při použití.

V následujících kapitolách si představíme rozhraní HDMI, hardware, který se nachází na cílové platformě, a blíže se zaměříme na propojení mikrokontroléru, FPGA a obvodu TFP410. Popíšeme si sběrnici Flexbus, kterou v práci používáme pro komunikaci mezi mikrokontrolérem a FPGA, sběrnici I2C pro konfiguraci obvodu TFP410. Podíváme se na vlastnosti a funkcionalitu obvodu TI TFP410. V druhé části práce se podíváme na popis vytvořeného řadiče a knihovny určené k jeho ovládání. Demonstujeme si funkčnost řadiče na několika příkladech (hra Tetris a Sokoban). V závěru práce zhodnotíme dosažené výsledky a navrhneme vylepšení řadiče a knihovny.

Kapitola 2

Rozhraní HDMI

V této kapitole se podrobně podíváme na vlastnosti a popis komunikace rozhraní HDMI (High-Definition Multimedia Interface). Informace jsou shrnuty z více zdrojů, především ale ze specifikace[6].

2.1 Historie

Vývoj HDMI rozhraní probíhá od 16. dubna 2002, kdy došlo k rozhodnutí vytvořit nové rozhraní kompatibilní s DVI, které bylo v té době na většině televizorů a přehrávačů DVD s vysokým rozlišením. Cílem bylo vylepšit HDMI pomocí menšího konektoru s přidanou podporou zvuku. První verze HDMI byla představena 9. prosince 2002. HDMI se brzy těšilo velké úspěšnosti a stalo se tak celosvětovým standardem, zejména díky zastaralosti rozhraní SCART sloužícího u televizní techniky.

2.2 Základní vlastnosti

HDMI slouží zejména k přenosu obrazu a zvuku v jednom kabelu. Kromě této základní vlastnosti umožňuje hot-plug, E.D.I.D (Extended Display Identification Data) pro získání informací o cílovém zařízení, HDCP (High-bandwidth Digital Content Protection) pro ochranu před kopírováním multimediálního obsahu vysoké kvality. Spojení HDMI je jednosměrné, pro obousměrný provoz jsou nutné dva spoje. Data se přenáší nekomprimovaně.

2.2.1 EDID

EDID slouží pro zjištění informací o zobrazovacím zařízení. Poskytuje informace o výrobci a modelu, ale také mnohé důležitější informace, jako je časování signálů nebo podporovaná rozlišení. EDID lze číst přes rozhraní I2C. EDID je uložen v zobrazovacím zařízení v PROM nebo EEPROM paměti a je dostupný na adrese 0x50, respektive 0xA0, když započítáme RW bit. Informace jsou rozložené do bloků po 128 bajtech. Součástí EDID informací jsou informace rozšířené DDC (Display Data Channel).

Na dílčí adrese v rámci EDID informací 0x00 až 0x07 se nachází hlavička EDID. Na adrese 0x08 až 0x11 se nachází identifikační údaje o výrobci a zařízení včetně sériového čísla, roku a týdne výroby. Verze EDID struktury včetně revize se nachází na adrese 0x12 a 0x13. Základní informace o velikosti zobrazovací plochy a základní podpoře se dají přečíst na adrese 0x14 až 0x18. Zajímavé informace o barevném chování zařízení lze vyčíst z adres 0x19 až 0x22.

Základní informace o požadovaném časování lze přečíst na 0x23 až 0x25, kde na adrese 0x25 může výrobce definovat své požadavky. Na adrese 0x26 až 0x35 jsou standardní časování. Podrobnější informace o časování mohou podle verze EDID být na adresách 0x36 až 0x7D. Pokud EDID informace obsahují více než základní strukturu, tak na adrese 0x7E je uložena informace o počtu dalších bloků o velikosti 128 bajtů, které následují. Na adrese 0x7F se pak nachází kontrolní součet celého bloku, tak aby byl roven 0.

2.2.2 Kompatibilita s DVI

Zařízení s DVI výstupem lze připojit díky kompatibilitě těchto rozhraní na vstup HDMI. Děje se tak beze ztráty kvality obrazu, není ale umožněn přenos zvuku nebo signálů z dálkového ovládání. Toto se musí řešit pomocí přídavných kabelů nebo adaptérů, protože například zvuk v rozhraní DVI používá vlastní datové vodiče.

Opačné spojení (HDMI výstup na DVI vstup) nemusí být již vždy funkční. Zvuk se přenáší pomocí stejných datových vodičů jako obraz. Je tedy velmi pravděpodobné, že si zařízení s DVI vstupem neporadí s takto promíchanými daty a nedokáže tedy obraz zobrazit.

2.2.3 Konektory, kabely a zapojení

HDMI využívá 5 typů konektorů. Konektory jsou nezátěžové a tudíž je nelze příliš ohýbat. Ukázka všech konektorů na obrázku 2.1.

Základním konektorem je **typ A**, který obsahuje 19 pinů, a je dostupný od specifikace 1.0. Podporuje většinu režimů (SDTV, EDTV, HDTV i 4K UHD). Zapojením je kompatibilní s DVI-D single-link.

Lze ho tedy používat pro přenos obrazu ve velmi vysokém rozlišení (např. WQUXGA 3840x2400pixelů). **Typ B** má 29 pinů a oproti typu A umožňuje dvojnásobnou šířku přenosového pásma. Zapojením je kompatibilní s DVI-D dual-link. V době psaní této práce se v žádném zařízení nepoužívá, v této kategorii je běžnější DisplayPort, jelikož není zatížen poplatkem za možnost vývoje zařízení s podporou HDMI (poplatek za specifikaci činí cca 10000 amerických dolarů).

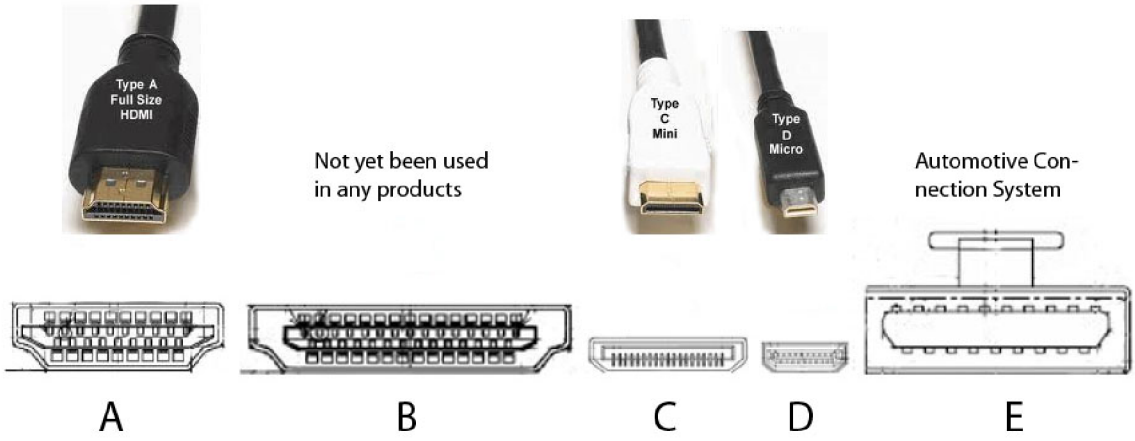
Dalším spíše dříve používaným konektorem u grafických karet je **typ C (mini)**. Konektor je určený spíše pro přenosná zařízení (např. telefony a tablety). Má stejné složení pinů jako konektor typu A jen s poněkud proházeným pořadím, je však v menším pouzdru. Pomocí redukce tedy z něj lze udělat konektor typu A.

Ještě menším konektorem je **typ D**. Velikostně se velmi podobá microUSB konektoru. Opět má 19 pinů, avšak rozložení signálů v konektoru je rozdílné od typu A i typu C.

Poslední minimálně používaný typ je **typ E**. Tento konektor se používá hlavně v automobilovém průmyslu. Je zkonstruován tak, aby odolával vibracím.

HDMI není dle specifikace omezeno délkou kabelu. Maximální délku omezuje pouze útlum signálu. Záleží tedy především na kvalitě materiálů, které jsou použity, a dále na konstrukci kabelu. HDMI v1.3 definuje dva typy kabelů. Jedná se o certifikaci kategorie 1, která je testována na přenosovou frekvenci 74.5MHz (to znamená přenos 1080i/720p), a kategorii 2, která je testována na frekvenci 340MHz (to znamená přenos 1600p).

Normální kabel HDMI může být použit do délek 12 až 15 metrů. Do 5 metrů může kabel kategorie 1 dosáhnout kvalit kategorie 2. Větší délka může způsobovat nestabilitu, která se bude projevovat blikáním na obrazovce. Existují převodníky a aktivní kabely, které zesílením či převodem na jiný typ přenosu (například pomocí ethernetového spojení přes ethernetový kabel Cat5), mohou prodloužit dosah až na 50 metrů. Existují i převodníky na optické vlákno, které umožňují přenos na vzdálenosti až 100 metrů.



Obrázek 2.1: HDMI konektory. Převzato z [8]

Od roku 2012 se neoznačují kabely číslem, ale slovním popisem: *(Standard nebo High Speed) (Automotive) HDMI Cable (with Ethernet).*

V základní variantě obsahuje každý kabel následující složení vodičů: 3xTMDS skupinu (DATA+, DATA-, stínění DAT) ve verzi B je těchto trojic 6, hodinový signál(CLOCK+, CLOCK-, stínění CLOCK), I2C rozhraní(SDA, SCL), napájení pro vnitřní DDC obvod v zobrazovacím zařízení (+5V), CEC signál pro povolení CEC ovládací kódy, zemnící vodič, detekci Hot-Plug a od verze 1.4 ještě ethernetový kanál nebo kanál zpětné zvukové vazby.

2.3 Rozvoj HDMI

HDMI 1.0 je první specifikace HDMI z 9. prosince 2002. Jednokabelový audio/video konektor s maximální propustností 4.9Gbit/s, v případě kombinace se zvukem až 3.96Gbit/s plus 192kHz/24-bitová kvalita zvuku.

Další verze **HDMI 1.1** byla představena 20. května 2004. Přináší podporu zvuku v kvalitě/formátu DVD-Audio.

Potřeba nového rozhraní u osobních počítačů vedla k vývoji verze **HDMI 1.2** představené 8.srpna 2005. Rozšiřuje podporu zvuku o One Bit Audio u Super Audio CD. Objevují se první HDMI konektory (typ A) pro PC, do této doby využívá konektor DVI. Možnost převodu RGB na YCbCr v PC. Přichází s možností snížení napětí. Později v prosinci byla vylepšena na verzi 1.2a, která přináší podporu CEC.

V závislosti na vývoji zobrazovacích zařízení bylo nutno rozšířit podporu o větší hloubku barev. Toto rozšíření přináší **HDMI 1.3**, která byla uvedena v platnost 22. června 2006. Kromě původní hloubky 24-bitů přibylo ještě 30,36,48-bitů v prostorech barev xvYCC, sRGB, YCbCr. Šířka přenosového pásma byla rozšířena na 340MHz (propustnost 10.6 Gbit/s). Přidána byla možnost automatické zvukové synchronizace (Audio video sync). Byl rozšířen zvukový výstup o Dolby TrueHD a DTS-HD Master Audio pro externí dekódování pomocí AV přijímače. Jedná se o formáty používané na HD DVD a Blu-ray discích.

V této specifikaci došlo ke vzniku kategorií kabelů. Rodina konektorů se rozrostla o typ C (mini) pro přenosná zařízení. Verze 1.3b, 1.3b1 a 1.3c přináší specifikaci testování konektorů a kabelů.

S příchodem popularity 3D zobrazování bylo nutné zařídit přenos dvou rozdílných snímků. Na konci května 2009 byla představena verze **HDMI 1.4**, která přináší podporu 3D obrazu, 100Mbit ethernetového a zpětnovazebného zvukového kanálu. Verze 1.4 rozšiřuje podporu vysokých 4K rozlišení (3840x2160 pixelů při 24/25/30Hz a 4096x2160 pixelů při 24Hz). Opět byla rozšířena podpora barevných prostorů. Konektory byly rozšířeny o typ D a odolný typ E.

Přelomovým krokem byla verze **HDMI 2.0** ze 4. září 2013. Přichází se zvýšenou propustností až 18Gb/s. Podporuje až 32 zvukových kanálů, 4 zvukové stopy a zvyšuje vzorkovací frekvenci audia na 1536kHz. Byla rozšířena podpora pro CEC a byla přidána podpora pro technologii dynamic auto lip-sync (obraz/zvuk). Byla přidána podpora rozlišení 4K s frekvencí 60Hz a formátu stran 21:9. V dubnu 2015 byla přidána podpora pro video s vysokým dynamickým rozsahem (HDR) se statickými metadaty.

Poslední a zároveň čerstvou verzí je **HDMI 2.1**. Oficiální představení proběhlo 4. ledna 2017. Uplatnění této verze je plánováno během 2. až 3. čtvrtiny roku 2017. Přinese rozšíření podpory HDR o dynamická metadata umožňující nastavení pro každou scénu, a dokonce i pro každý snímek a ne pouze staticky pro celé vysílání. Bude podporovat ještě vyšší rozlišení (4K při 120Hz a 8K při 120Hz). Toto si vyžádalo nové kabely (tzv. 48G), které budou mít podporu propustnosti 48Gbit/s a budou i nadále používat konektory typu A,C,D. Teoretická podpora rozlišení bude až do 10K při 120Hz. Přidáno bude rozšíření zvukové zpětné vazby a formáty Dolby Armos a DTS:X.

2.4 Princip generování obrazu

Obraz se generuje stejně jako u staršího rohraní VGA. Narozdíl od VGA jsou data kódována a přenášena přes diferenciální spoje. HDMI komunikace probíhá pomocí packetů. Kromě obrazových packetů existuje spousta dalších packetů. Některé souvisejí s obrazovým přenosem, jiné typy jsou například pro přenos zvuku. V rámci této práce se nebudeme zabývat ostatními typy packetů.

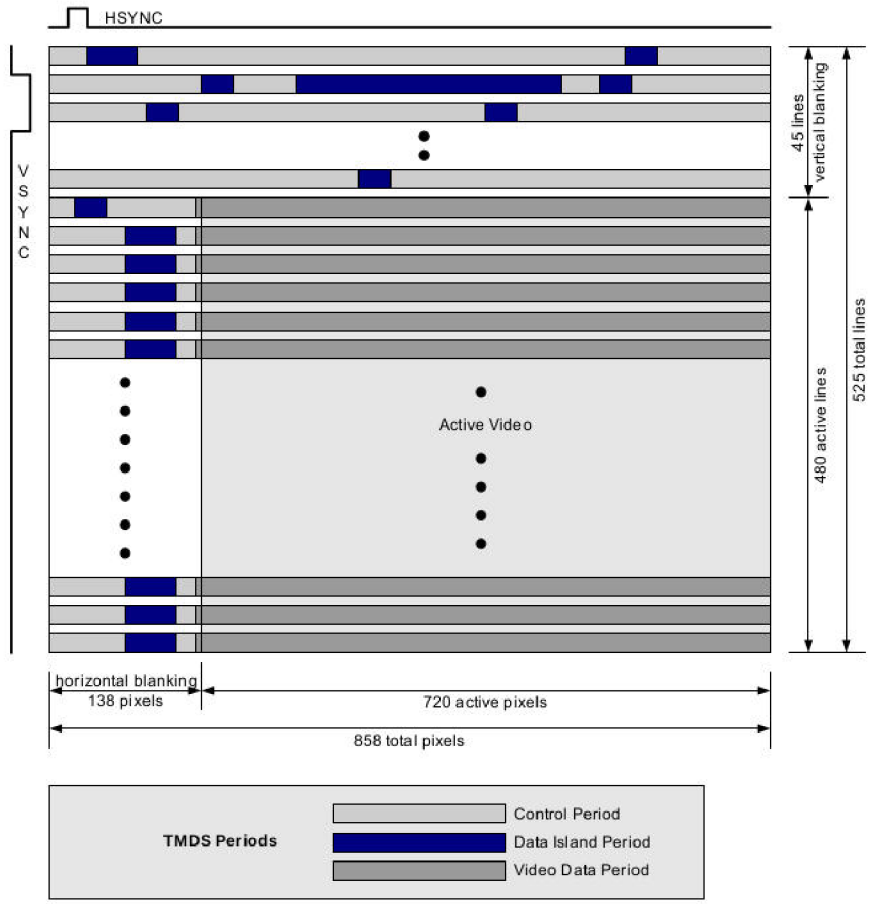
Obrazová data se posílají sériově přes jednotlivé kanály (např. červená přes kanál 0, zelená přes kanál 1 a modrá přes kanál 2). Data se posílají od nejméně významného bitu po nejvíce významný bit. Před posláním dat pixelu se jednotlivá 8-bitová data zakódují nejprve na 9-bitová a následně se všemi ochranami proti rušení na 10-bitová. Přesný způsob kódování je popsán ve specifikaci HDMI 1.3 na diagramu 5-7.

Komunikace probíhá ve třech možný režimech. Ukázka, kdy jsou jednotlivé režimy aktivní nebo je možné je využívat, je znázorněna na obrázku 2.2. Hlavní režim je přenos video dat. V tomto režimu se přenáší pouze data jednotlivých pixelů.

Přenos informačních paketů o zasílaném obrazu nebo zvukové packety jsou posílány v režimu "datových ostrovů"(data island). Během tohoto režimu se přenáší kromě těchto packetů i HSYNC a VSYNC signál. V tomto režimu se data packetů kódují pomocí TERC4 kódování. Toto kódování je popsáno opět ve specifikaci. Zjednodušeně lze říci, že kóduje 4bity na 10bitů.

Poslední režim je řídící. V tomto režimu se také přenáší SYNC signály. Kromě těchto signálů se přenáší preambule paketů. Data těchto paketů se přenáší pomocí kódování 2bitů na 10bitů. Řídící packet musí být vložen mezi každé dva nestejné packety (mezi data island a video data paket).

Veškeré generování komunikace přes HDMI zařizuje v naší práci obvod TFP410. Je tedy nutné tento obvod zásobovat správnými daty. Kromě zobrazovaných dat v hlavní viditelné části je nutné generovat ještě správný overscan.



Obrázek 2.2: Režimy při vykreslování obrazu. Převzato ze specifikace[6].

Overscan je generovaný obraz mimo viditelnou oblast zobrazovacího zařízení. V tomto overscanu se právě posílají packety v režimu datových ostrovů a řídícím režimu. Během overscanu je potřeba generovat správné časování pro synchronizační signály HSYNC a VSYNC. Jedná se konkrétně o časování (délku) SYNC pulzů, mezery před pulzem a mezery za pulzem.

Časování obrazu je možné převzít z VGA dle VESA specifikací rozlišení. Pro jiná rozlišení (mimo poměr 4:3) je nutné vyjít z časovacích požadavků, která poskytují cílová zobrazovací zařízení přes EDID.

Kapitola 3

Cílová platforma

V této kapitole seznámíme čtenáře s platformou, pro kterou jsou řadič a knihovna určeny. Podíváme se na komponenty, které jsou na ní osazeny, a dále na způsob použití platformy.

3.1 Výukový kit Minerva

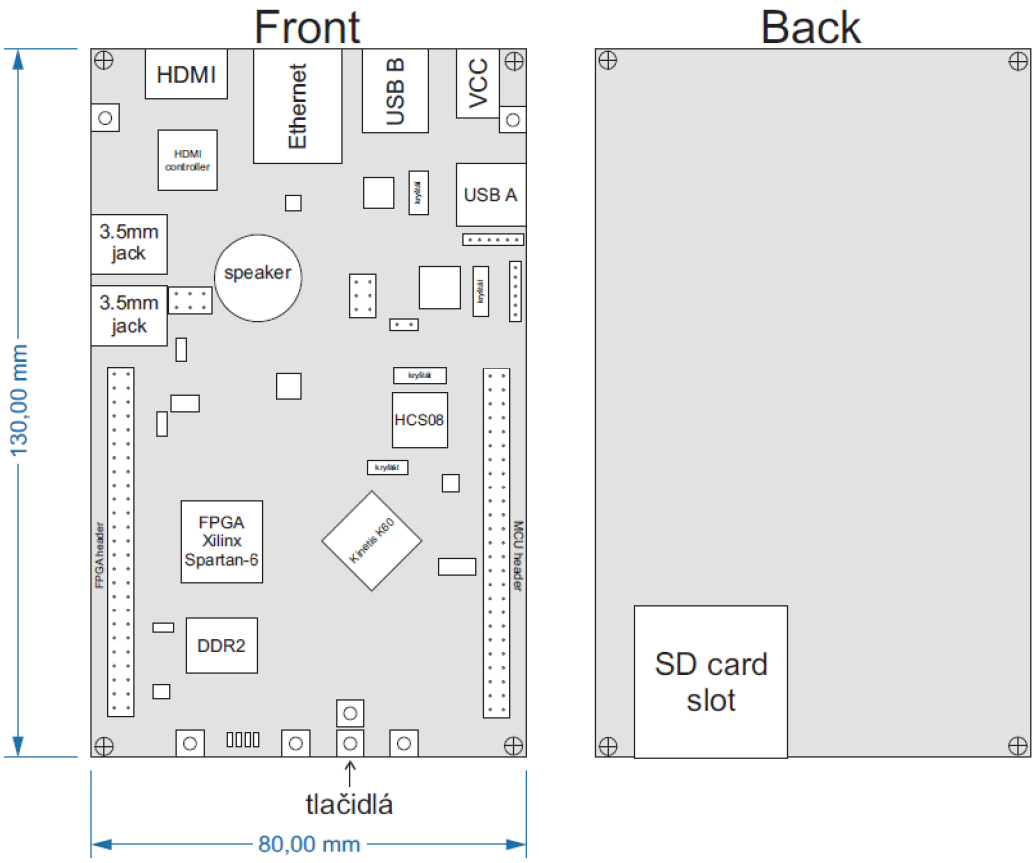
Cílová platforma (dále už jen Minerva) byla vytvořena jako nová a modernější verze stávajících výukových platforem FITkit verze 1.0.,1.2. a 2.0., které začínají být zastaralé a poruchové. Minerva byla vytvořena zaměstnanci fakulty informačních technologií Vysokého učení technického v Brně. Jedním ze spoluautorů Minervy je i vedoucí této práce, pan Ing. Václav Šimek[2]. Vývoj Minervy začal v roce 2012 a o rok později byl již hotový první funkční vzorek.

Minerva má velikost o něco málo větší než dvě kreditní karty. Rozložení důležitých komponent a zajímavých periferií na platformě je vyobrazeno na obrázku 3.1. Je založena na mikrokontroléru firmy NXP (v době vytvoření Minervy firma Freescale) řady K60. Jedná se 32-bitový jednojádrový ARM mikrokontrolér MK60DN512VMD10, který je postaven na vysoce výkonném jádru Cortex-M4 s nízkou spotřebou. Pro program má 512kB flash paměti a pracuje na frekvenci až 100MHz.

Mikrokontrolér obsahuje velké množství modulů, které umožňují jeho široké použití. Mezi nejzajímavější moduly patří RTC (modul reálných hodin), ethernet (připojení k sítím LAN), CRC (modul výpočtu kontrolních součtů pro detekci chyb), USB (hostitelský řadič) a SDHC (SD hostitelský řadič). Mimo tyto moduly obsahuje i základní periferie a komunikační rozhraní (např. ADC, DAC, GPIO, UART, I2C).

Minerva dále disponuje, stejně jako její předchůdce, FPGA(programovatelné hradlové pole) od firmy Xilinx. Konkrétně se jedná o výkoný model Spartan-6. Pro potřebu rychlé paměti je na FPGA připojena DDR2 paměť o kapacitě 64MB. Oproti předchůdcům je navíc vybavena pro tuto práci důležitým HDMI rozhraním s řadičem TFP410.

V dnešní době se mikropočítače těší velké oblibě (v čele s úspěšným projektem Raspberry Pi). Narozdíl od Raspberry Pi zatím Minerva nepodporuje možnost spuštění s operačním systémem. Firma Freescale v době vytvoření portovala pouze FreeRTOS systém pro tuto platformu.



Obrázek 3.1: Podoba vývojového kitu Minerva. Převzato z [9]

3.2 Mikrokonroler Kinetis K60 - Cortex M4

Na výukovém kitu je použit mikrokontrolér rodiny Kinetis K60 od firmy Freescale. Série K je řada výkonných a úsporných mikrokontrolérů. Řada K6 nabízí rozhraní USB, ethernet a navíc podporu hardwarového šifrování a detekci vniknutí. Podkategorie K60 je nejuniverzálnější z celé řady K6. Nemá dostupnou pouze detekci vniknutí.

Na výukové platformě se jedná o model K60DN512[3] v pouzdru BGA144 s 512kB flash paměti pro program, 12kkB SRAM paměti a 100 vstupně/výstupními piny. Jedná se o model bez FPU jednotky. Jádro mikrokontroléru může pracovat až na frekvenci 100MHz. Na kitu je mikrokontrolér taktován hodinami poskytujícími 50MHz.

Kromě USB a ethernetového rozhraní nabízí další komunikační rozhraní jako je 6xUART rozhraní, 3xSPI, 2xI2C, dvěmi audio rozhraní I2S, dvěmi průmyslovými sběrnicemi CAN, námi použitou sběrnicí FlexBus a sběrnicí SDIO s SDHC kontrolérem. Pro zpracování analogových signálů je dostupný 16-bitový AD a 12-bitový DA převodník, vysokorychlostní komparátor a napěťová reference.

Mikrokontrolér umožňuje řízení spotřeby pomocí až 10 nízkoodběrových režimů, které ocení hlavně aplikace napájené z baterií. Jedná se například o režim clock-gating, který odpojí nepotřebné periferie. Dále disponuje vlastním generátorem náhodných čísel, případně pro urychlení šifrovacích algoritmů důležitou hardwarovou jednotkou[5].

Jádrem kontroléru je 32bitový mikrokontrolér Cortex-M4 od firmy ARM. Mikrokontrolér je Harvardské architektury, která má oddělenou paměť pro program a pro data. Tímto rozdělením může dojít ke zvýšení výsledného výkonu. Jádro disponuje i základní technikou paralelizace v podobě instrukční sady SIMD (single instuction multiple data), která umožňuje provádět jednu instrukci nad více daty zároveň. Tyto instrukce najdou uplatnění hlavně při zpracování signálu (typicky zvuk a obraz).

Cortex-M4 v sobě neskrývá pouze mikrokontrolér, ale i důležité periferie pro běh systému. Například obsahuje řadič přerušení, jednotku pro ochranu paměti (MPU), výpočetní jednotku pro výpočty v plovoucí řádové čárce (FPU) nebo ladící rozhraní a mnoho dalších. Výhodou této architektury je, že tyto obvody jsou součástí architektury a výrobce je není nucen nahrazovat vlastní jednotkou.

Je tak docíleno standardního rozhraní, které usnadňuje práci s mikrokontroléry od různých výrobců, založených na stejné architektuře jádra. U řad kontrolérů ARM Cortex-M se jedná o standardní rozhraní CMSIS (Cortex Microcontroller Software Interface Standard) zpřístupňující funkcionalitu komponent této architektury. Díky této snaze vývojářů ARM je možné psát přenositelnější aplikace nezávislé na výrobci mikrokontroléru.

3.3 FPGA Xilinx Spartan-6

Vývojový kit, stejně jako předchozí verze, je osazen programovatelným hradlovým polem od firmy Xilinx řady. Na Minervě se jedná o model Spartan-6, konkrétně o model XC6SLX9[13]. Podle katalogového listu se jedná o druhý nejslabší model z této řady.

Obsahuje 9152 logický buňěk, 1430 slice buňek, kde každá obsahuje čtyři šestivstupé LUT tabulky a 8 klopných obvodů typu flip-flop. Celkem je možné udělat 90kb distribuované paměti spolu s 11440 bistabilních klopných obvodů. FPGA obsahuje 32 vestavěných 18kb RAM bloků. Pole obsahuje také 16 DSP bloků, které obsahují 18-bitovou násobičku, sčítačku a akumulační registr. Z pouzdra je vyvedeno celkem 200 uživatelsky konfigurovatelných vstupně/výstupních pinů.

Konfigurovat FPGA lze v jednom případě pomocí připojené konfigurační paměti, v případě druhém přímým bitovým konfiguračním streamem pomocí rozhraní JTAG. FPGA na kitu je nastaveno tak, aby po zapnutí načetlo konfiguraci z připojené paměti.

Konfigurační paměť je tvořena obvodem M25P40 o velikosti 512kB připojeným přes SPI.

3.4 Vývojové prostředky

Společnost NXP (dříve Freescale) dodává ke svým mikrokontrolérům řady Kinetis integrované vývojové prostředí Kinetis Design Studio (dále jen KDS) založené na vývojovém prostředí Eclipse, díky kterému je možné vyvíjet i na operačních systémech Linux[10]. Jedná se o prostředí zaměřené přímo na vývoj pro řadu Kinetis a nahrazuje tak starší prostředí CodeWarrior.

KDS bylo představeno v roce 2014. Bylo to hlavně z důvodu, že CodeWarrior nevyhovuje standardu CMSIS (Cortex Microcontroller Software Interface Standard), ale také proto, že ARM jádra nabízí velké množství nových funkcí a možností, a bylo proto nutné rozšířit podporu ve vývojových prostředích. V době psaní práce je již ve verzi 3.3.0.

Na Minervě je komunikace vývojového prostředí a mikrokontroléru zajištěna pomocí vyzkoušeného open-source rozhraní OSBDM, které je založeno na mikrokontroléru společnosti

NXP (Freescale) HCS08. Mikrokontrolér HCS08 je zapojen na ladící rozhraní mikrokontroléru a do počítače přenáší informace pomocí USB.

KDS obsahuje nástroj ProcessorExpert, který slouží pro jednoduchý a rychlý vývoj aplikací. Obsahuje předpřipravené interaktivní ovladače (LDD bloky) pro moduly, které vygenerují zdrojový kód podle nastavení. V některých případech vygenerovaný kód bývá nepřehledný a nesrozumitelný.

V KDS je možné psát aplikace v jazyce C i C++. Ty se následně překládají pomocí GCC (GNU Compiler Collection). Ladit aplikace je možné pomocí GDB (GNU Debugger). Předpřipravené ovládací bloky pro ProcessorExpert je možné doinstalovat z balíku Kinetis SDK (Software Development Kit).

Pro vytváření aplikace pro FPGA slouží prostředí Xilinx ISE. Nevýhodou tohoto prostředí je ukončený vývoj firmou Xilinx. Společnost Xilinx vyvíjí nové prostředí Vivado, které je určeno pro nové FPGA rodin Virtex, Kintex. Poskytuje i základní podporu pro novější generaci Spartan 7.

Kapitola 4

Systémové komunikační sběrnice

V této kapitole se podíváme na dvě sběrnice používané v rámci této práce. Jedná se o sběrnici FlexBus, která slouží pro komunikaci mezi mikrokontrolérem a obvodem FPGA, a sběrnici I2C, která slouží pro přenos konfigurace z obvodu FPGA do HDMI obvodu TFP410.

4.1 FlexBus

Použitý mikrokontrolér Kinetis K60 disponuje rozhraním FlexBus, které slouží pro komunikaci s externími obvody jako jsou paměti typu ROM, FLASH, programovatelná hradlová pole či ostatní obvody založené na podobném principu. FlexBus je paralelní synchronní sběrnice. Konfigurace umožňuje měnit mnoho parametrů a tak se přizpůsobit velké škále cílových slave zařízení. Na našem vývojovém kitu je použita pro komunikaci mezi mikrokontrolérem a FPGA obvodem.

FB_CLK	Hodinový signál sběrnice generovaný vnitřním děličem dle nastavení v SIM_CLKDIV1[OUTDIV3]
FB_AD[31:0]	V nesdíleném režimu nesou data v sdíleném se střídají data s adresou
FB_CS[5:0]	Chip-select pro výběr slave zařízeni
$\overline{FB\_TA}$	Signál určující dokončení přenosu
FB_ALE	Signál potvrzující platnost nastavené adresy a dat
FB_R/$\overline{W}$	Signál určující typ komunikace 0 = zápis, 1 = čtení

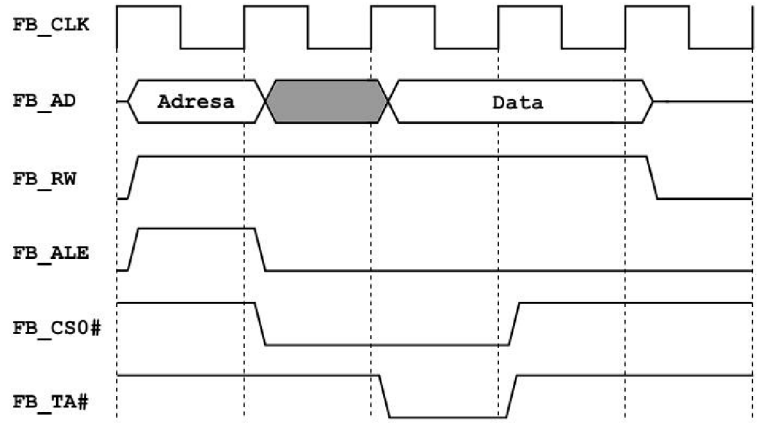
Tabulka 4.1: Popis hlavních FlexBus signálů

FB_A[31:0]	V nesdíleném režimu přenáší adresu
$\overline{FB\_BE/BWE}$[3:0]	Signály oznamují, že data jsou posílána na konkrétní část sběrnice
$\overline{FB\_OE}$	Signál sloužící pro povolení čtení připojené paměti
$\overline{FB\_TBST}$	Signál oznamující dávkový přenos
FB_TSIZ[1:0]	V případě dávky oznamuje velikost přenášených dat

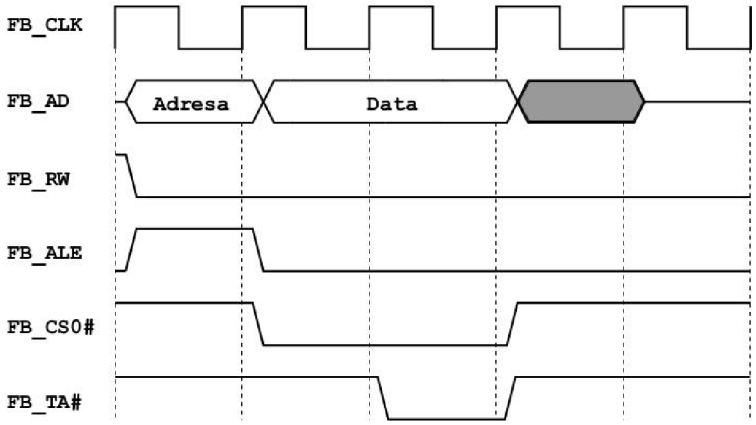
Tabulka 4.2: Popis dalších FlexBus signálů

Sběrnice disponuje až 6 nezávislými programovatelnými chip-select (CS) signály (FB_CS[5:0]). Sběrnice může fungovat s šířkou dat 8,16,32-bitů ve sdíleném nebo nesdíleném režimu datových a adresních signálů.

Se signály vypsanými v tabulce 4.1 budeme v této práci dále pracovat. Jedná se o nejmenší možné zapojení sběrnice pro úspěšnou komunikaci. Mimo tyto signály však FlexBus disponuje dalšími signály (některé ze zajímavějších vypsány v tabulce 4.2).



Obrázek 4.1: Úkázka operace čtení na sběrnici FlexBus. Převzato z [1]



Obrázek 4.2: Ukázka operace zápisu na sběrnici FlexBus. Převzato z [1]

Podívejme se tedy jak probíhá komunikace. Znázornění čtení dat je na obrázku 4.1. Synchronizační signál FB_CLK generuje mikrokontrolér. Když master zahajuje komunikaci, vystaví při sdíleném režimu na port FB_AD[31:0] adresu slave zařízení a nastaví signál FB_ALE, kterým informuje všechna zařízení na sběrnici o této změně. K tomu je nastaven signál FB_RW do log.1, kterým jsou informována zařízení o směru komunikace (v tomto případě o čtení).

Dobu, po kterou je vystavena adresa na sběrnici (Adress Hold Time), je možné pomocí parametrů sběrnice změnit. Po uplynutí této doby je nastaven konkrétní signál FB_CS[5:0], pro vybrání konkrétní periferie, se kterou bude probíhat komunikace (pokud je více periferií majících stejnou adresu). V našem případě je použit pouze signál FB_CS0, jelikož

na sběrnici máme připojený jen obvod FPGA. Slave zařízení informuje o vystavení dat pomocí signálu FB_TA nastavení log.0. Master tímto dostává pokyn, že data může přečíst. Transakce se ukončí nastavením příslušného signálu FB_CS do neutrální úrovně (log.1).

Zápis na sběrnici probíhá zcela analogicky. Znázornění zápisu přes sběrnici na obrázku 4.2. Rozdílem je nastavení signálu FB_RW, a dále to, že signál FB_TA oznamuje, že data jsou přečtena. Po potvrzení dat specifikace nedefinuje v jakém stavu budou datové signály drženy.

Pro naši potřebu je komunikace na kitu nastavena jako sdílená 16-bitová. Adresní prostor pro naši aplikaci je nastaven od 0x60000000 až po adresu 0x6FFFFFFF, avšak sběrnice může na našem mikrokontroléru pracovat až s adresou 0xDFFFFFFF.

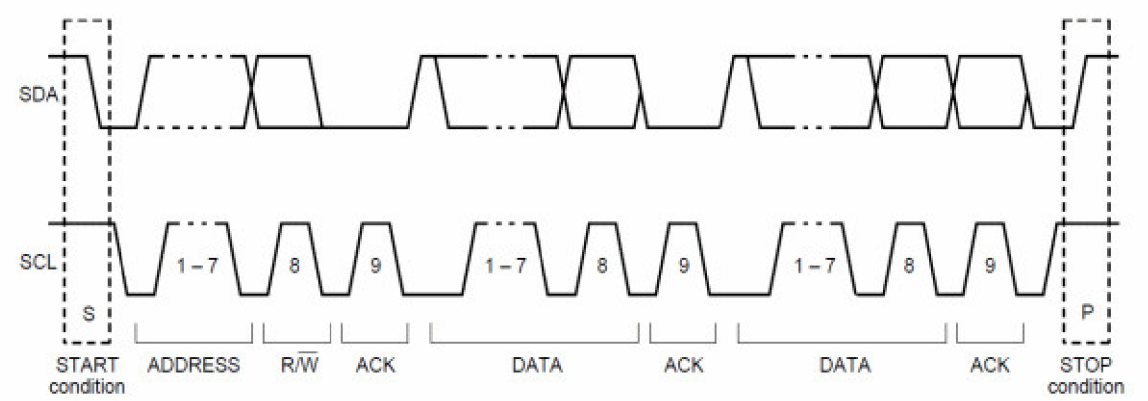
4.2 I2C

Sběrnice I2C je multi-masterová synchronní sériová half-duplexová (lze přenášet data jen jedním směrem) sběrnice vyvinutá firmou Philips. Obdobou I2C je i sběrnice Two Wire Interface (TWI) od společnosti Atmel, která vznikla jen kvůli potřebě vyhnout se zpoplatněné sběrnici I2C. Aktuální stav je takový, že se platí za přidělování adres pro slave zařízení, implementace je již nezatížená poplatky. Sběrnice se používá pro komunikaci s nízkorychlostními periferiemi procesoru. Komunikace na sběrnici může probíhat až rychlostí 400kHz.

Sběrnice využívá dva vodiče. První je SCL neboli hodinový signál sběrnice a druhý SDA neboli datový vodič. Oba jsou zapojeny jako otevřený kolektor (tzn. vodič je připojen přes pull-up k napětí sběrnice a ovládací pin zařízení je tvořen pouze jedním tranzistorem, který spíná výstup ke společnému potenciálu (zemi)). Maximální délka sběrnice je dána maximální přípustnou kapacitou vodiče 400pF.

Zařízení na sběrnici se rozdělují na řídící (master, je v aktivní době pouze jeden, zahajuje a ukončuje komunikaci, generuje hodinový signál) a řízené (slave, zařízení adresované masterem na sběrnici).

Na sběrnici je možné adresovat 128 různých zařízení. Osmi bitová adresa se skládá ze 7bitů adresy zařízení a jednoho RW-bitu, který rozhoduje zda chceme zapisovat nebo číst ze slave zařízení. Sběrnice podporuje i 10bitovou adresu. Převážná většina implementací si vystačí se 7bitovou. Lze vysílat i broadcast packety a to na adresu 0.



Obrázek 4.3: Schéma I2C komunikace. Převzato z [7]

Komunikace (ukázka na obrázku 4.3) probíhá zahájením **START** podmínky, kdy je nejdříve SDA signál stažen na log.0, a teprve poté je stažen signál SCL na log.0. Následně

je odvysílána adresa cílového zařízení. Signál SDA se po *start* podmínce mění pouze v okamžiku, kdy je signál SCL v log.0. Pokud cílové zařízení najde shodu s adresou, je mu určeno potvrdit tento fakt zasláním ACK bitu po posledním bitu adresy.

Po ACK bitu může začít vysílat nebo přijímat data dle typu přenosu. Přenos každého bajtu potvrzuje při zápisu slave zařízení a při čtení master zařízení. Vysílací stanice ACK signál vytváří ponecháním SDA v log.1, přijímací v log.0. Přenos končí po zaslání posledního bajtu signálem **STOP**, který nastaví master. Jedná se o zvednutí SDA do log.1. když je SCL v log.1. Pokud kdykoliv během komunikace vysílací stanice **neobdrží ACK** signál, ukončí komunikaci signálem **STOP**.

Zařízení může zahájit komunikaci kdykoliv, kdy je sběrnice volná (tzn. je-li sběrnice v klidovém stavu, neprobíhá žádná komunikace). Řízení komunikace se provádí pomocí metody detekce kolizí. Každé zařízení (master), které vysílá data na sběrnici, kontroluje, zda stav sběrnice odpovídá tomu, co posílá. Pokud neodpovídá, došlo ke kolizi. Toto dokáže poznat jen pouze vysílá-li stav log.1, a jiné zařízení mu mezitím vysílá stav log.0, tím pádem na sběrnici bude číst hodnotu log.0. Zařízení, které toto detekuje, okamžitě ukončí komunikaci.

Kapitola 5

Řadič rozhraní HDMI

V této kapitole se podíváme podrobně na obvod TFP410, který používáme ke generování HDMI signálů, a na řadič implementovaný v rámci této práce v jazyce VHDL pro obvod TFP410.

5.1 Obvod TFP410 od Texas Instruments

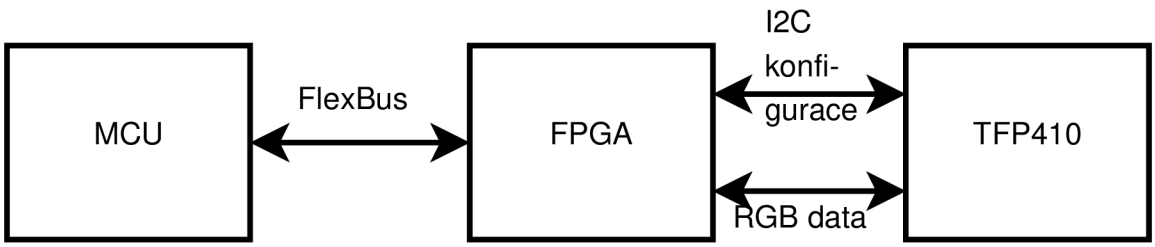
Obvod TFP410 slouží pro generování DVI/HDMI obrazového výstupu. Obvod podporuje generování výstupu až do rozlišení WUXGA (1080p) s maximální frekvencí pixelů 165MHz. Obvod je možné konfigurovat pomocí pinů a nebo pomocí sběrnice I2C.

Vstupní barevná data jsou kódována do tří sériových diferenciálních spojů T.M.D.S. pro přenos přes kroucenou dvoulinku. Data mohou být dodávána v plné šířce 24-bitů, nebo v poloviční šířce 12-bitů při použití sestupné i vzestupné hrany hodinového signálu. Obvod umožňuje pouze barevný prostor RGB s 24-bitovou hloubkou.

Obvod poslouchá na I2C adrese 0x70+A[3:1]*2. V našem případě jsou adresové piny připojeny přes pull-up na 3.3V a tudíž poslouchá na adrese 0x7E.

5.2 Návrh řadiče

Vytvoření vhodného řadiče je klíčové pro ovládání HDMI výstupu pomocí mikrokontroléru. Diagram 5.1 představuje zapojení celého námi používaného schématu na fitkitu. Diagram 5.2 představuje vnitřní zapojení bloků řadiče v fpga. Komunikaci s mikrokontrolérem přes flexbus zajišťuje řadič flexbusu z diplomové práce od pana Buchty[1]. Výstup tohoto řadiče jde přímo do řadiče TFP410.

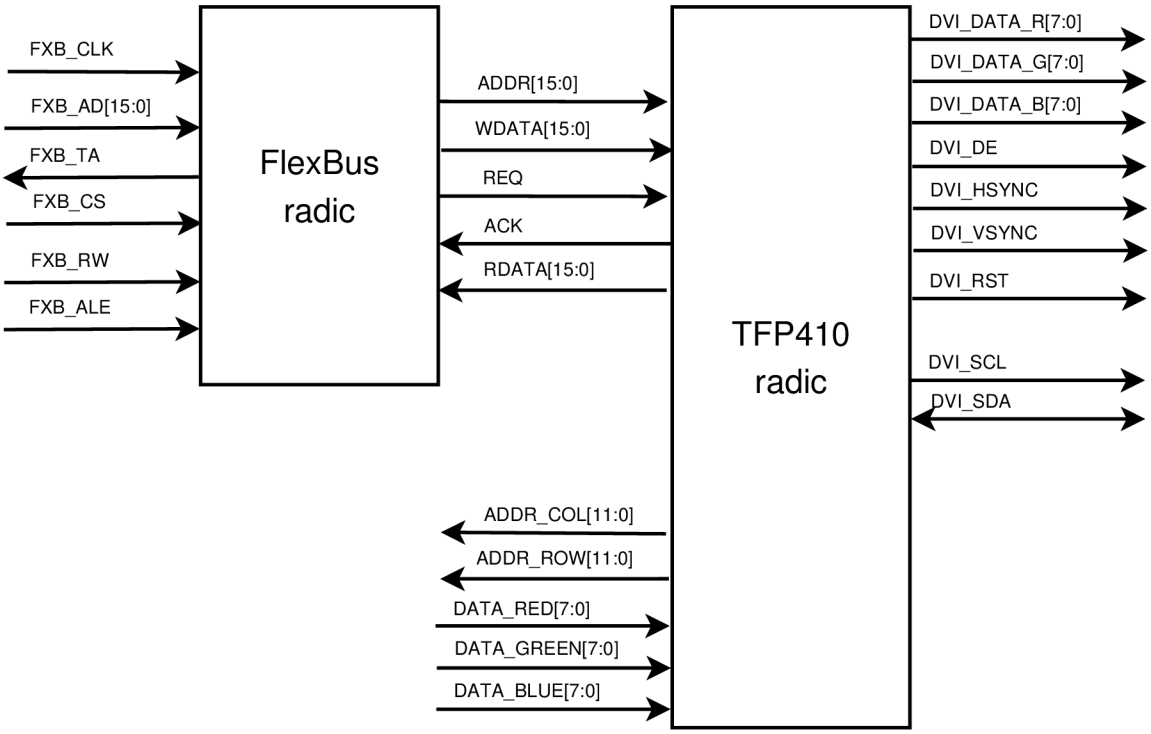


Obrázek 5.1: Zapojení komponent na cílové platformě

Řadič TFP410 tedy musí zajišťovat zpracování komunikace od FlexBus řadiče, komunikovat přes I2C s TFP410 a generovat zobrazovaná data. Důležité je, aby řadič generoval data se správnou frekvencí a dal tak dohromady podporované rozlišení zobrazovacího zařízení.

5.3 Implementace řadiče

Řadič je vnitřně řešen jako tři procesy. První proces má na starosti zpracování flexbus komunikace. Další dva procesy se starají o generování správné posloupnosti horizontálních a vertikálních výstupních dat. Dále jsou tady dva procesy počítání správného rozlišení. Poslední proces řeší přechody v konečných automatech na základě hodin pro jednotlivé pixely. Pro konfiguraci obvodu TFP410 je potřeba komunikovat s ním přes I2C. Z tohoto důvodu řadič obsahuje master řadič pro sběrnici I2C převzatý od Scotta Larsona[11].



Obrázek 5.2: Zapojení bloku řadiče uvnitř fpga

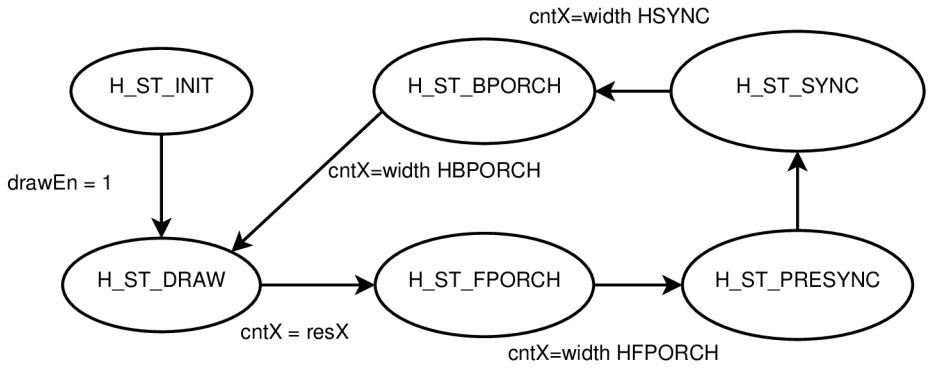
Základní adresa pro tento řadič je zvolena jako 0x1000. Na dílčích adresách 0x1000 až do 0x103E je určena pro přístup k registrům v obvodu TFP410. K těmto registrům se přistupuje vždy po 16 bitech, aby se zefektivnila (snížila se režie) komunikace mezi fpga a mcu. Toto však nese i nevýhodu. Když je sběrnice FlexBus nastavena na datovou šířku jinou než 8bitů, není možné adresovat jednotlivé bajty. Je nutné dodržovat zarovnání adresových bloků (1bajt, 2bajty, 4bajty).

Konečný automat, který se stará o obsluhu komunikace přes flexbus, nejdříve ověřuje shodu na základní adresu. Pak podle příchozí adresy zahájí komunikaci přes I2C (adresy 0x00 až 0x3E), případně operuje s vnitřními proměnnými řadiče (ostatní adresy 0x40 až 0xFF). Na adrese 0x40 je možné nastavit výstupní rozlišení v ose X, na adrese 0x42 v ose Y a na adrese 0x44 lze povolit nebo zakázat vykreslování (zablokovat vykreslovací automat). Rozlišení si řadič drží jako 12bitové hodnoty.

Výchozí rozlišení řadiče je možné definovat v rámci generických syntetizačních parametrů řadiče. Výchozí nastavení těchto parametrů je pro rozlišení 640x480pixelů při obnovovací frekvenci 60Hz.

Kromě těchto parametrů je nutné správně nastavit frekvenci vstupního hodinového signálu pomocí *input_clk* (výchozí hodnota 50MHz), výstupní frekvenci hodin na konfigurační sběrnici I2C pomocí *i2c_scl_frq* (výchozí hodnota 400kHz). Toto nastavení slouží pro správnou konfiguraci obvodu I2C master. Je možné měnit i I2C adresu, na které poslouchá obvod TFP410, pomocí parametru *i2c_addr* (výchozí hodnota je dle zapojení obvodu TFP410 na Minervě 0x7E).

Během vykreslování řadič vystavuje adresu aktivního řádku a sloupce, kterou aktivní aplikace používá pro krmení HDMI výstupu daty. Pro toto krmení řadič disponuje 8-bitovými vstupy pro jednotlivé barvy RGB (červená, zelená, modrá).



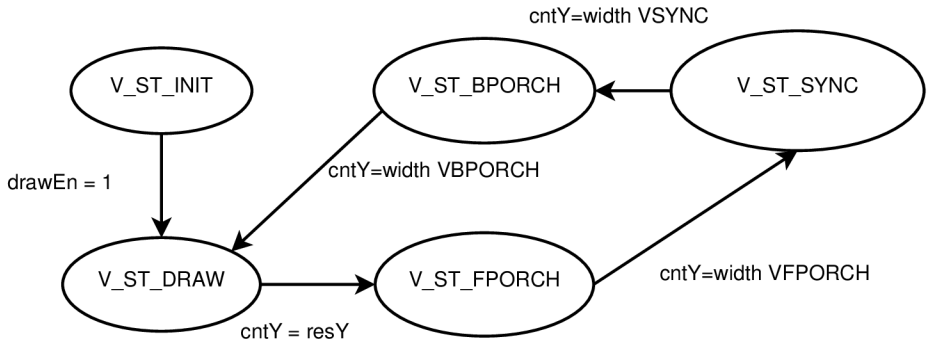
Obrázek 5.3: Stavový automat generující horizontální data

Generování výstupního obrazu probíhá pomocí dvou stavových automatů, kde první řídí generování horizontálních dat a druhý vertikálních. Princip těchto automatů vychází z principu generování obrazu popsaného v kapitole 2.4. Grafické znázornění těchto automatů je na diagramech 5.3 a 5.4.

Oba automaty začínají po restartu ve stavu H/V_ST_INIT. Do tohoto stavu se dostanou i pokud kdykoliv během generování je vypnuto vykreslování pomocí signálu *drawEn*. V tomto stavu čekají na povolení vykreslování a drží reset stav čítačů horizontálních i vertikálních signálů.

Oba automaty setrvávají většinu času ve stavu H/V_ST_DRAW, kde pouze počítají zobrazené pixely (nebo řádky) a drží informaci o výstupu horizontálních nebo vertikálních dat. Když je napočítán požadovaný počet pixelů (řádků), je potřeba zaslat synchronizační pulzy. Proto přechází oba automaty do stavu H/V_ST_FPORCH, kde nevykreslují data a drží neutrální stav výstupu po minimální potřebnou mezeru před synchronizačním pulzem.

Před horizontálním synchronizačním pulzem je potřeba informovat druhý automat o zobrazení celého řádku. K tomu v našem řešení slouží signál *cntY_inc*, který nastavíme na hodnotu 1. Aby byla tato změna platná, je potřeba v horizontální automatu počkat jeden stav. K tomu slouží přechod skrze stav H_ST_PRESYNC. Následně můžeme přejít do stavu H/V_ST_SYNC. V tomto stavu držíme signály synchronizace v požadované logické úrovni. Po dosažení potřebné délky je nutné vygenerovat mezeru za synchronizací. K tomu dojde ve stavech H/V_ST_BPORCH. Po vygenerování této mezery se vracíme zpět ke generování obrazu v H/V_ST_DRAW.



Obrázek 5.4: Stavový automat generující vertikální data

Pro správné generování výstupních dat je důležitý hodinový signál pixelů. Frekvenci tohoto signálu získáme pomocí rovnice

$$f_{pixel} = f_{frame} * ((res_x + porch_{hf} + porch_{hb} + sync_h) * (res_y + porch_{vf} + porch_{vb} + sync_v)).$$

V rovnici f_{frame} představuje obnovovací frekvenci obrazovky, res_x horizontální rozlišení, res_y vertikální rozlišení. Délku synchronizačního pulzu označuje $sync_h$ a $sync_v$. Při generování obrazu je nutné vkládat mezeru v řádku před data a před horizontální synchronizačním pulzem. Délka těchto mezer je označena $porch_{hf}$ a $porch_{hb}$. Totéž ve vertikální ose před odesíláním řádků a za řádky před vertikálním synchronizačním pulzem (označení $porch_{vf}$ a $porch_{vb}$).

Příklad pro rozlišení 640x480pixelů při obnovovací frekvenci 60 snímků za vteřinu, vertikální mezera před řádky 10 taktů, za řádky 33 taktů, horizontální mezera před řádkem 16 taktů a za řádkem 48 taktů, délka horizontální synchronizace 96 taktů a vertikální 2 takty. Výsledkem je frekvence pro pixely 25.2MHz.

Ve výsledné implementaci řadiče je nutné generovat hodinový signál pixelů mimo řadič. Vstupní hodinový signál je potřeba prohnat vstupním hodinovým bufferem. Výstup bufferu se přivede na DCM blok (hodinový manažer), který nastavíme dle požadavků výstupní frekvence pixelů. Generovaný signál přivedeme na vstup hodinového bufferu. Totéž uděláme s negovaným výstupním hodinovým signálem z DCM bloku. Výstup obou bufferů přivedeme na vstup ODDR2 bloku, který slouží k předcházení problému *clock forwarding* při syntéze generovaného hodinového signálu, který je přiváděn na výstupní pin FPGA. Výstup tohoto bloku je naším hodinovým signálem pixelů.

Rozlišení	f_{PIXEL}	hfp	hs	hbp	vfp	vs	vbp	hpol	vpol
640x480 60Hz	25.2MHz	16	96	48	10	2	33	0	0
800x600 60Hz	40MHz	40	128	88	1	5	23	1	1
1280x1024 60Hz	108MHz	48	112	248	1	3	38	1	1
1368x768 60Hz	85.86MHz	72	144	216	1	3	23	0	1
1024x768 70Hz	75MHz	24	136	144	3	6	29	0	0
1280x720 60Hz	74.25MHz	72	80	216	3	5	22	1	1
1920x1080 60Hz	148.5MHz	88	44	148	4	5	36	1	1

Tabulka 5.1: Testovaná rozlišení a nastavené příslušné časování

Vytvořený řadič umožňuje použití maximální frekvence hodinových signálů 115MHz při vybalancované strategii syntézy a umísťování. Při nastavení strategie na optimalizaci časování je možné dosáhnout maximální frekvence 145MHz. Toto omezení je limitující pro výsledná možná rozlišení. Data pro zobrazení musí být většinou připravována pomocí hodinového signálu s minimálně dvakrát vyšší frekvencí. Není tedy možné dosáhnout rozlišení FullHD (1920x1080px při 60Hz), jelikož vyžaduje frekvenci pro pixely 148.5MHz.

Řadič byl otestován s rozlišeními a příslušným časováním v tabulce 5.1 pomocí generátoru čtvercové sítě. Rozlišení FullHD se nepovedlo vygenerovat.

5.4 Problémy s generováním

Při testování zobrazení jsme narazili na zajímavý problém s reakcí monitorů na signál. U digitálního rozhraní jsme očekávali, že není nutné generovat mezery před a za sync signály a řešit délku sync signálu, jelikož LCD zobrazování nepotřebuje čas na návrat světelného paprsku. Tato doměnka byla vyvrácena při spuštění s délkou prodlev a sync signálů jednoho hodinového taktu. Některé monitory pak detekovaly zasílaný signál, avšak nezobrazily jej. Některé z těchto monitorů zobrazily informaci o nepodporovaném signálu. Po přečtení několika EDID informací je zřejmé, že i u moderních monitorů je potřeba dodržovat nějakou minimální prodlevu kolem sync signálu a jeho délku. U malých rozlišení tento problém nenastává, ale u vyšších rozlišeních, kde je frekvence pixelů vysoká, se projevuje.

Další zajímavé zjištění nastalo kolem obnovovací frekvence. Některé monitory akceptují i obnovovací frekvence mimo jejich rozsah (například hlavní testovací monitor BenQ E2220HD). Při jednom z testů byly zkráceny mezery a délky na jeden takt, nastaveno rozlišení 854x480pixelů a hodinový signál pro pixely na 25MHz. Takto byla vygenerována obnovovací frekvence kolem 90 snímků za vteřinu. Monitor i takovou frekvenci zobrazil. Pravděpodobně vzorkuje dostatečně rychle HDMI rozhraní a pouze zahazuje přebytečné snímky.

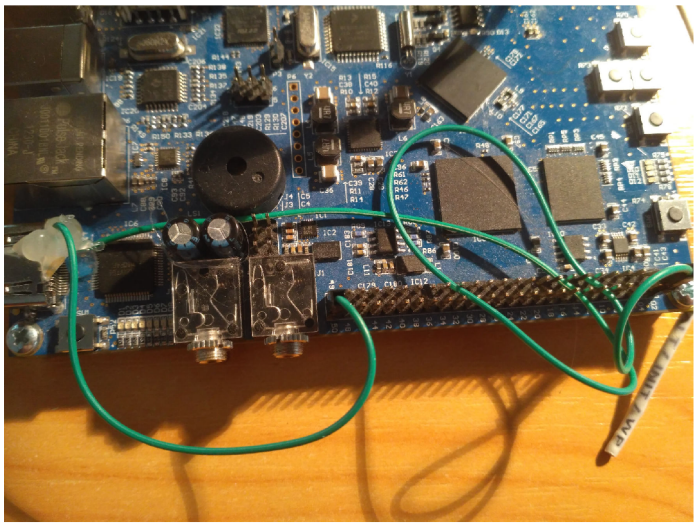
Jediný problém, který byl pozorován, bylo problikávání posledního vykresleného řádku. V tomto případě však nevíme, zda máme problém přisuzovat nepodporovanému rozlišení, vysoké frekvenci, případně tomu jak se monitor vyrovnává s poskytnutým signálem.

5.5 Nutné úpravy Minervy

V rámci vývoje řadiče jsme narazili na několik problémů v zapojení HDMI, které způsobují problémy se zobrazováním na některých zařízeních. Některá zobrazovací zařízení potřebují zapojení napájení 5V a zemnícího potenciálu GND_CEC v konektoru HDMI.

Toto napájení slouží u některých monitorů k napájení vnitřních obvodů DDC (Display Data Connection), které slouží pro dekódování komunikace proudící přes rozhraní HDMI. Zobrazovací zařízení v takovém případě detekuje, že je připojeno k nějakému zařízení, ale již není schopný detekovat proudící signál. Tímto problémem trpěly některé testovací monitory. Některé nechávají černou obrazovku, ale detekují, že je nějaké zařízení připojené. Několik testovaných televizí nedetekovalo zařízení v žádném z případů a hlásily se jako nepřipojené (jedna od Samsungu, jedna od LG).

Oproti tomu některá zařízení detekují připojení i bez napájení 5V, detekují pouštěný signál, ale oznamují ho jako nepřítomný. Pro zobrazení jim pak stačí připojit zemnící potenciál GND_CEC, který jim slouží jako reference k datovým signálům. Ukázka nápravy tohoto problému pomocí připojení signálů přímo na konektor je zachycena na fotce 5.5.



Obrázek 5.5: Nutná úprava platformy pro podporu většiny monitorů

Pro lepší řadič by bylo vhodné připojit ještě komunikační sběrnici I2C v konektoru HDMI. Tím by bylo možné lépe generovat výstupní signál i nižší frekvencí hodin pro pixely. K tomu je potřeba zjistit podporované časování (problém v podkapitole 5.4). Dále by bylo možné zjistit vlastnosti monitoru jako je maximální rozlišení, maximální obnovovací frekvence, podporovaná rozlišení a maximální frekvenci pixelů. Další zajímavou informací by mohla být velikost zobrazovací plochy. Ta by mohla umožnit aplikace s potřebou zobrazování reálných velikosti.

U aktuálně vyrobených kitů by mohl být tento problém řešen pomocí speciálního nástavce. Tento nástavec by byl tvořen jako prodlužovací kabel HDMI. Nástavec by byl na malé desce plošných spojů s jedním mužským konektorem HDMI pro zapojení do kitu, druhým ženským konektorem HDMI pro připojení kabelu vedoucího do zobrazovacího zařízení a potřebným počtem pinů v hřebínku pro možnost připojení chybějících signálů.

U revize by bylo dobré připojit všechny zemnící nepřipojené signály v HDMI konektoru na společnou zem, napájení na +5V a I2C vyvést na piny, které pomocí jumperu bude takto možné připojit na I2C sběrnici vedoucí z FPGA do obvodu TFP410, za účelem ušetření IO pinů na FPGA, případně by se výše zmíněné I2C mohlo připojit právě na rozšiřující port vyvedený z fpga či mikrokontroléru.

5.6 Implementace dalších pomocných řadičů

Pro ukázkové aplikace bylo nutné vytvořit vhodný řadič pro virtuální RAM blok v FPGA. Protože se jedná o přenos dat pro RAM, je vhodné využít také FlexBus. Řadič je implementován na základě obsluhy flexbusové komunikace. Tento řadič poslouchá na adrese určené obecným parametrem entity. Výchozí je 0x2000. Řadič obsahuje dvě funkce. První na adrese 0x00 slouží pro nastavení adresy, na kterou chceme v rámci RAM přistupovat, druhá na adrese 0x02 pro čtení nebo zápis dat.

Kromě řadiče VRAM vznikl také jednoduchý řadič pro registr v FPGA aplikaci. Tento řadič poslouchá na výchozí adrese 0x3000. Výchozí šířka registru, se kterým tento řadič pracuje, je 8bitů. Řadič nemá žádné další dílčí adresy, a proto je možné další takový řadič zavěsit na adresu 0x3002.

Kapitola 6

Obslužná knihovna rozhraní HDMI

Řadič HDMI je potřeba ovládat z mikrokontroléru, kde běží hlavní aplikace. K tomuto účelu byla vytvořena obslužná knihovna, která zpřístupňuje rozhraní řadiče.

Knihovna byla tvořena tak, aby byla jednoduchá na použití a integraci do aplikace.

6.1 Konfigurace řadiče

Před zahájením práce s řadičem je potřeba inicializovat rozhraní FlexBus v mikrokontroléru. K tomu slouží funkce *flexbus_init*. Tato funkce byla převzata z knihovny pro komunikaci přes FlexBus z diplomové práce pana Buchty[1]. Tato funkce slouží k přípravě rozhraní. Využívá funkci *flexbus_preinit*, která nastavuje porty mikrokontroléru pro FlexBus komunikaci a základní parametry komunikace (jako je datová šířka, nastavení zdroje hodinového signálu).

Po korektní inicializaci sběrnice je nastavena proměnná *FPGA_ConfOK* na hodnotu 1. Tato proměnná slouží pro oznámení všem komponentám (knihovnám), které využívají sběrnici FlexBus, že je sběrnice nastavena pro použití. Proměnná je tedy zpřístupněna všem knihovnám. Knihovna v inicializaci nastavuje komunikaci jako neblokující. To znamená, že pokud na příslušné adrese nebude poslouchat žádné zařízení (neodpoví na operaci), bude hlavní program pokračovat ve vykonávání dalšího kódu. Na konci inicializace se nastaví přenos jako blokující. V případě, že bude v FPGA nahrán firmware, který nebude obsahovat zařízení, která bude vyžadovat hlavní aplikace, dojde k zamrznutí hlavní aplikace na operaci čtení či zápisu do tohoto zařízení.

Knihovna pana Buchty poskytuje dále i základní nastavení přerušení od FPGA a jeho jednoduchou obslužnou funkci.

Tuto knihovnu používá i naše knihovna. Před zahájením práce s řadičem je nutné provést jeho inicializaci pomocí funkce *tfp410_init*. Tato funkce zkontroluje, jestli je rozhraní FlexBus již nastaveno, pokud není provede jeho inicializaci. Poté si stáhne informaci o nastaveném výchozím rozlišení v řadiči v FPGA.

Veškerá komunikace s řadičem probíhá pomocí dvou základních funkcí. První funkce *tfp410_read16* slouží pro čtení 16-bitových dat z řadiče. Čtení probíhá pomocí pointeru na základní adresu řadiče (0x60001000) s přičtenou adresou registru, který chceme z řadiče číst. Nad touto funkcí je postavena ještě jedna pomocná funkce *tfp410_read8*, která umožňuje číst pouze 8-bitová data. Jedná se pouze o zjednodušující obal. Funkce provede 16-bitové čtení adresy registru s vymaskovaným nejnižším bitem a následně vybere podle vymaskovaného bitu z dat horní nebo dolní bajt podle požadované adresy.

Druhou funkcí je zápis 16-bitových dat *tfp410_write16*. Funguje stejně jako čtení pouze se do ukazatele zapisuje místo čtení. Stejně jako čtení má i funkci pro práci s 8-bitovými daty. Tato funkce však navíc provádí čtení celých 16bitů dat na vymaskované adrese. Pak zapisovaná data vloží na příslušný horní nebo spodní bajt a provede zpátky zápis celých 16bitů dat.

Konfigurace řadiče TFP410 se provádí pomocí registrů na sběrnici I2C. Tyto registry náš řadič zpřístupňuje na sběrnici FlexBus (viz. kapitola 5.3). Mapování registrů řadiče je znázorněno v příloze A.

Knihovna kromě základních funkcí pro čtení nebo zápis přes sběrnici FlexBus do řadiče poskytuje funkce pro čtení a nastavování všech konfiguračních parametrů. Funkce zápisu jsou dostupné pouze pro parametry, které je skutečně možné měnit. Všechny tyto funkce provádí maskování přijatých nebo poslaných dat tak, aby autor aplikace využívající tuto knihovnu nemusel znát rozložení registrů a jejich obsah. V případě, že by autorovi nestačily tyto funkce, jsou v hlavičkovém souboru k dispozici všechny adresy registrů a masky jednotlivých parametrů.

Kromě obsluhy konfiguračních registrů knihovna poskytuje ovládání dalších parametrů řadiče. Mezi tyto parametry patří práce s registrem povolujícím nebo zakazujícím generování obrazových dat, registry rozlišení výstupního obrazu (horizontální a vertikální).

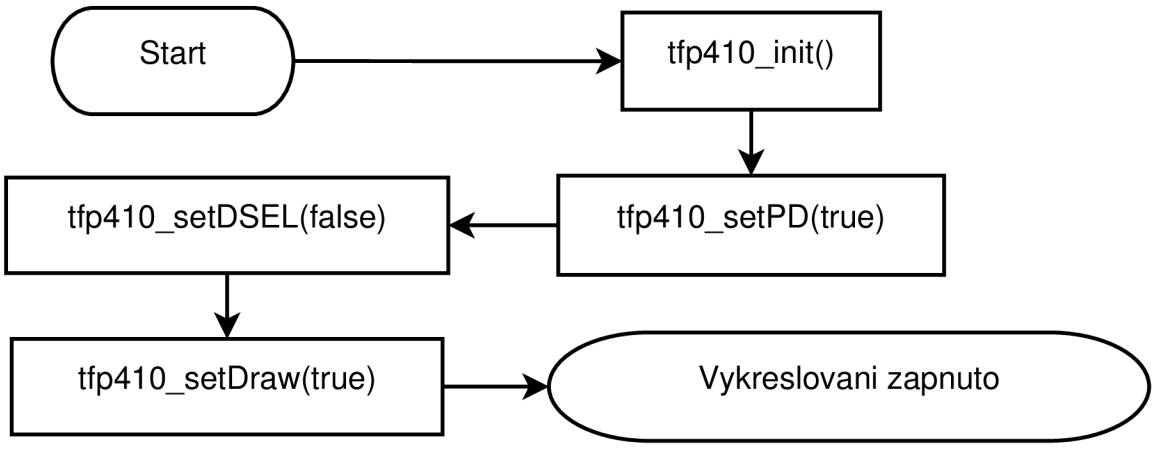
Podívejme se teď na některé parametry, které je možné měnit a jsou vždy potřebné pro spuštění vykreslování.

Pro minimální spuštění vykreslování je potřeba zapnout obvod TFP410. Toto se provede funkcí *tfp410_setPD*. Jako parametr očekává nastavovaný stav tohoto bitu. Dále je potřeba hodiny nastavit do stavu jednoduchého vstupu. Výchozí stav po restartu je diferenciální hodinový signál. K nastavení slouží funkce *tfp410_setDSEL*.

Další parametry obvodu TFP410 je možné měnit nebo číst podle vzoru funkce *tfp410_(set|get)JMENOREGISTRU*.

K vypnutí nebo zapnutí generování výstupu (ovládání generovacího procesu v řadiči, nikoliv nastavení obvodu TFP410) slouží funkce *tfp410_setDraw*. Nastavené rozlišení je možné zjistit pomocí funkcí *tfp410_getResX* a *tfp410_getResY*.

Popis ovládání řadiče pro spuštění vykreslování je znázorněn na diagramu 6.1.



Obrázek 6.1: Diagram spuštění vykreslování

6.2 Práce s podpůrnými řadiči

Pro práci s řadiči VRAM knihovna nabízí set funkcí. Řadiče jsou dostupné na adresách 0x60002000 pro vlastní zobrazovaná data, 0x60002100 pro paměť textur a 0x60002200 pro paměť fontů.

Pro práci s pamětí dat jsou zde funkce *setPixel* a *getPixel*. Obě funkce nejdříve nastaví adresu, na které se pixel nachází, voláním funkce *vram_setAddr*, kde adresa je 12-bitová. K této funkci existuje i *vram_getAddr*, která slouží pro získání aktuálně nastavené adresy. Operace s adresou se provádí na interní adrese 0 řadiče VRAM. Práce s daty v této paměti se odehrává na adrese 2.

Adresování v rámci virtuální paměti probíhá na jednobajtová data. Tímto způsobem je možné pomocí tohoto řadiče adresovat až 4kB paměti.

Vytváření ukazatele na tyto registry (parametry) se sestavují stejně jako u řadiče HDMI.

Knihovnu je možné zkompilovat pro VRAM nebo DDR RAM. V aktuálním stavu obslužné funkce pro DDR RAM nic nedělají a jsou připravené pro rozšíření s řadičem DDR2.

Pro potřeby ukázkových aplikací je knihovna rozšířena i o pár funkcí pro použití textur a fontů. Funkce *loadTexture* slouží pro nahrávání 16bitových dat do paměti textur. Obdobně *loadFont* slouží pro nahrávání 8bitových dat do paměti fontu. Tato data se pak v aplikaci adresují pomocí indexu do uvedených pamětí.

Kapitola 7

Ukázkové aplikace

Funkčnost navrženého řadiče je testována ukázkovýma aplikacemi. V našem případě jsou zvolenými aplikacemi hry Tetris a Sokoban. Pro tyto hry bylo nutné vytvořit několik podpůrných knihoven. Videoukázku obou příkladů a detailní návod na použití řadiče a knihovny naleznete u této práce na přiloženém CD.

7.1 Podpůrné knihovny

Pro ukázkové hry bylo nutné vytvořit obslužnou knihovnu pro ovládání tlačítek a signalizačních diod. Pro případné lepší vykreslování byla vytvořena knihovna pár grafických funkcí.

Knihovna pro tlačítka kromě inicializace příslušných pinů poskytuje ještě obslužnou rutinu přerušení portu E. Tato rutina volá callback funkcí jednotlivých tlačítek. Tyto callbacky se nastavují pomocí *btn_setFnSW{2,3,4,5,6}*. Jako argument požaduje ukazatel na funkci (*void (*fnPtr)(void)*). Pro aplikace, které nevyžadují reakci na tlačítka pomocí přerušení a stačí jim detekce pomocí cyklického testování, knihovna poskytuje *btn_getState*, která jako parametr požaduje pin portu, na který je tlačítko připojeno. Návratovou hodnotou je informace, zda je tlačítko stisknuté (false) nebo nestisknuté (true). Tlačítka je nutné inicializovat pomocí funkce *btn_init*.

Knihovna grafických funkcí poskytuje několik funkcí, které zapisují pomocí funkce *tfp410_setPixel* přímo do RAM. Nejzákladnější funkcí je *drawLine*, která vykreslí přímku. Pak je možné vykreslit obdélník a vyplnit ho pomocí funkcí *drawRectangle* a *fillRectangle*.

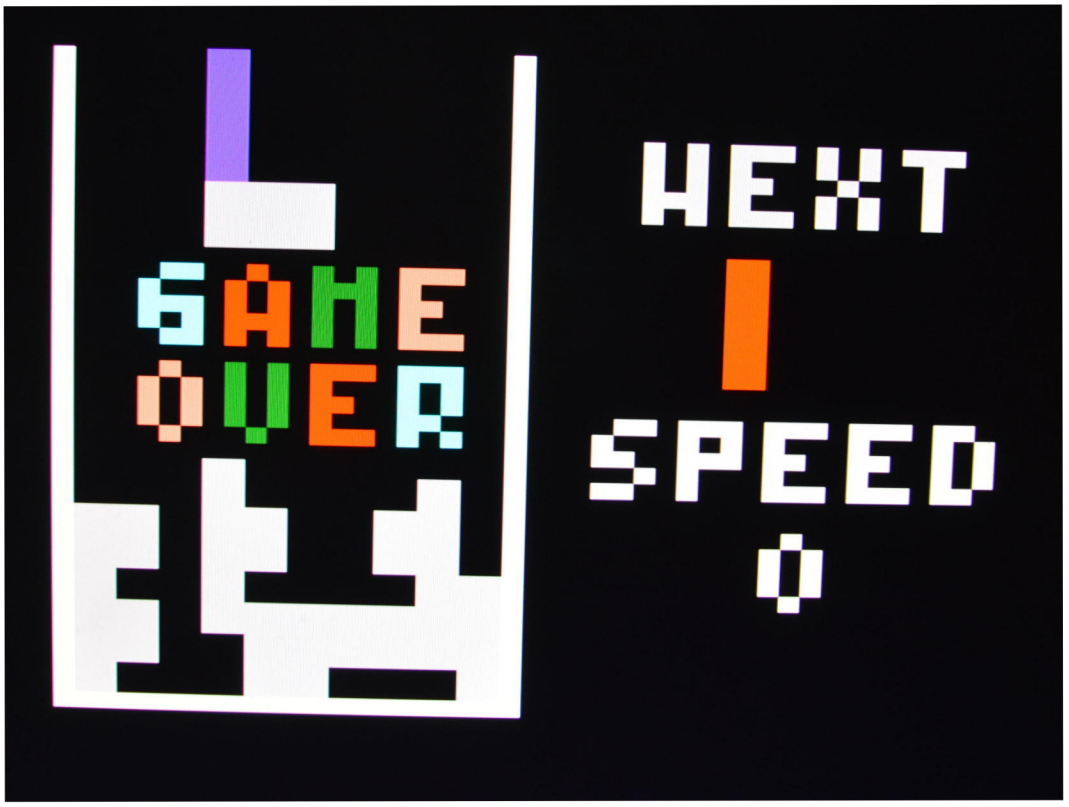
7.2 Společný základ

Hlavní kód obou ukázkových her byl převzat z repozitáře SVN staršího výukového FITkitu. Pro jednodušší sestavování her v těle funkce *main* se volá před hlavní smyčkou funkce *prepareGame* a pak v hlavní smyčce se volá funkce *mainGameLoop*. Tyto funkce musí být definované ve zdrojových souborech konkrétní aplikace. Stačí tedy naimportovat správné hlavičkové soubory konkrétní hry. Tímto by bylo možné hry dělat i jako knihovny a při kompilování celé aplikace linkovat jen se správnou knihovnou.

V převzatém kódu aplikací byly upraveny funkce závislé na mikrokontroléru (detekce tlačítek, časovače) a všechny funkce pracující s grafickým výstupem na naši knihovnu. Herní jádro aplikací zůstalo původní.

7.3 Tetris

První ukázková aplikace vytvořeného ovládání rozhraní HDMI. Ukázka zobrazení hry na obrázku 7.1. Tato aplikace využívá pro generovaná data jednu VRAM na adrese 0x2000 (adresa řadiče na sběrnici flexbus) s datovou šířkou 4bity. Jednotlivá 4-bitová data jsou dekódována na předem danou 24-bitovou barvu v rámci aplikace v FPGA. Zobrazování těchto dat se provádí pomocí sestavené adresy.

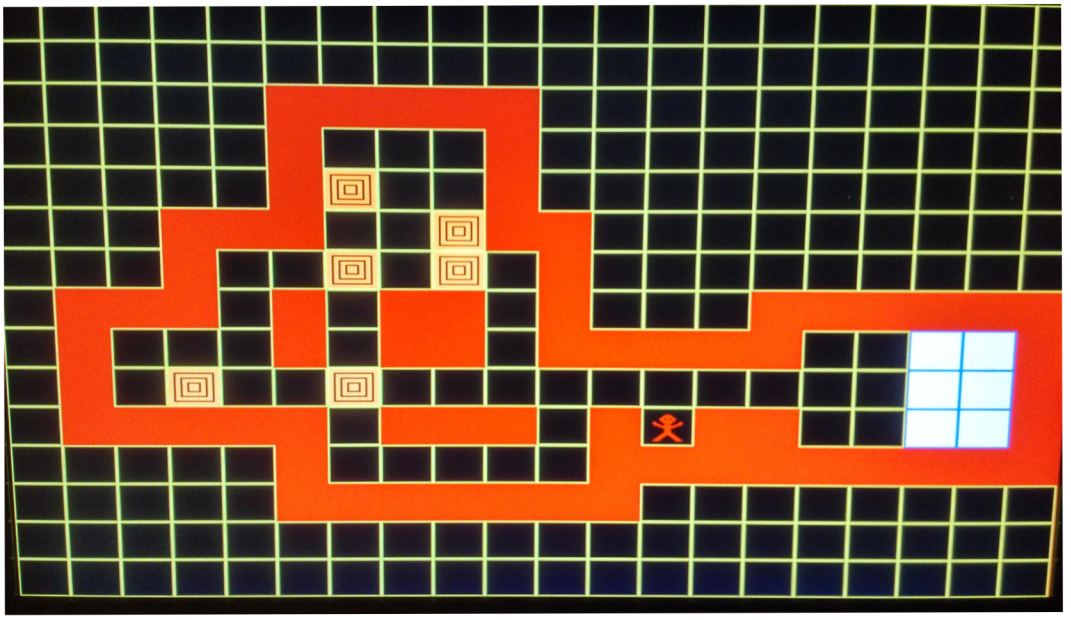


Obrázek 7.1: Ukázka z aplikace Tetris

Adresa dat je pro vykreslování sestavována z adresy aktivního pixelu dle pravidla $ram\_adresa = aktivni\_radek[8:3]aktivni\_sloupec[8:3]$. Adresa je tak 10 bitová, kdy na vyšších 5 bitech je 8. až 3. bit z adresy aktivního řádku a na nižších 5 bitech je 8. až 3. bit z adresy aktivního sloupce. Výsledný vykreslovaný pixel má tak reálnou velikost 8x8 pixelů. Tím, že se nevyužívají všechny bity aktivního pixelu, je potřeba nastavit pro tyto přebytečné pixely vykreslení výchozí barvy, aby se obraz neduplikoval.

Funkce *prepareGame* v tetrisu není potřeba. Je sice deklarovaná a definovaná, ale prázdná, nic tedy nedělá.

Funkce *mainGameLoop* detekuje stav kláves před zahájením hry. V případě zahájení hry je volána funkce překreslení zobrazovací plochy a hlavní herní funkce *play*, která vykonává všechny herní interakce.



Obrázek 7.2: Ukázka z aplikace Sokoban

7.4 Sokoban

Druhou ukázkovou aplikací je hra Sokoban. Tato hra pro zobrazování využívá dva řežimy zobrazení. První režim je textový. K tomuto se využívá VRAM s načteným fontem. Druhý režim je zobrazování textury. Textura velikosti 64x32x2B dat je načtena do druhé VRAM textury. Ukázka z texturového režimu na obrázku 7.2.

Mezi těmito režimy je přepínáno pomocí mode, který je ovládán pomocí flexbus jednotky na práci s registrem v FPGA. Tento registr je simulován *std_logic_vector*. Tuto jednotku je možné vysyntetizovat pro práci se skoro jakkoliv velkým registrem (až 16bitů). Vlastní zobrazovaná data jsou indexy do paměti textur nebo fontu podle *mode* a jsou uložena v dvou pamětích VRAM, kde jedna obsahuje horní polovinu zobrazovaných dat a druhá spodní polovinu obrazových dat. Toto se rozlišuje dle 8.bitu z adresy aktivního řádku.

Přípravná funkce *prepareGame* zde slouží pro načtení fontu a textury do příslušné VRAM. Toto zařídí aplikační funkce *Font_Flash_FPGA* a *Texture_Flash_FPGA*, které berou data z proměných definovaných v hlavičkovém souboru *sokoban_data.h*. Tímto je nahrazena flash paměť, která byla v rámci MCU na předchozím vývojovém kitu. Kromě načtení dat provede inicializační restart celé hry.

Hlavní smyčka v *mainGameLoop* zjišťuje stav tlačítek a na jejich základě rozhoduje o vykonání herní funkce.

Kapitola 8

Závěr

V této práci jsem úspěšně implementoval řadič rozhraní HDMI včetně obslužné knihovny pro mikrokontrolér. Tento řadič funguje s více rozlišeními, která není možné měnit za běhu, jelikož není možné změnit frekvenci hodin pro pixely. Řadič byl úspěšně otestován na ukázkové aplikaci Tetris a Sokoban, které byly v rámci této práci převedeny ze starší výukové platformy FITkit.

Zobrazování bylo testováno na monitorech BenQ E2220HD, Acer V243H, DELL SE2416H, BenQ BL2420PT, televizích LG 50LF561V a Samsung UE40K5602. Byla otestována rozlišení 640x480 60Hz, 800x600 60Hz, 1280x1024 60Hz, 1368x768 60Hz, 1024x768 70Hz a 1280x720 60Hz. Kromě čistého HDMI-HDMI rozhraní bylo testováno i rozhraní HDMI-DVI, kde jsem otestoval, že vytvořený řadič je kompatibilní s rozhraním DVI a je tak možné použít i zobrazovací zařízení bez HDMI vstupu.

V rámci práce bylo zjištěno a navrhnuto řešení problému v aktuálním zapojení HDMI rozhraní na platformě. Toto řešení by bylo dobré aplikovat do revize výukové platformy. Testovací úpravy platformy pomocí doplnění drátových propojek na konektor HDMI není jednoduché a ani stabilní řešení.

V budoucnu plánuji vylepšit řadič o vnitřní integraci generování hodinového signálu pro pixely na základě parametrů nastavených řadiči. Nejlépe vylepšit řadič tak, aby se dal vysyntetizovat univerzální s možností kompletní změny rozlišení (tím i frekvence hodin pixelů) za běhu bez nutnosti syntetizovat celou aplikaci znovu. V případě, že by byla dostupná sběrnice I2C z HDMI konektoru pro čtení EDID informací, by bylo možné libovolně v závislosti na zobrazovacím zařízení měnit rozlišení a obnovovací frekvenci.

Dále by bylo dobré mít možnost využít pamět DDR2, která je osazena na platformě, jako grafickou paměť, ze které se bude vykreslovat obraz. Tím bude možné využívat plné rozlišení 1920x1080 pixelů bez zvětšování (roztahování) pixelů do matic pixelů. Toto by vyžadovalo u textového módu řešit vykreslování na straně MCU, případně vytvořit řadič v rámci FPGA, který bude mít možnost vykreslovat do grafické paměti. Dalo by se tak řešit i velikost písma u textového režimu.

Obslužnou knihovnu pro MCU by bylo vhodné upravit pro přeložení do podoby knihovny, kterou bude možné jen linkovat při překladu a nebude potřeba zapojovat do nových projektů celou strukturu knihovny. Toto by přineslo programátorský komfort.

I bez těchto další úprav je už možné toto řešení použít pro další aplikace z původního FITkitu a další projekty vyžadující zobrazovací rozhraní.

Literatura

[1] Buchta, P.: *Application Development Framework for the ARM Platform, diplomová práce*. Brno: FIT VUT v Brně, 2015.

[2] FIT VUT Brno: *Produkty, prototypy a software [online]*. FIT VUT v Brně, 2012 [cit. 2017-03-28], [Online; navštíveno 28.3.2017].
URL http://www.fit.vutbr.cz/research/prod/index.php.cs?id=290&notitle=1

[3] Freescale Semiconductor: *K60 Sub-Family Reference Manual[online]*. Freescale, 2012 [cit. 2016-10-11], [Online; navštíveno 11.10.2016].
URL http://www.nxp.com/assets/documents/data/en/reference-manuals/K60P144M100SF2V2RM.pdf

[4] Freescale Semiconductor: *Using FlexBus Interface for Kinetis Microcontrollers[online]*. Freescale, 2012 [cit. 2017-01-04], [Online; navštíveno 4.1.2017].
URL
http://www.nxp.com/assets/documents/data/en/application-notes/AN4393.pdf

[5] Freescale Semiconductor: *Kinetis Peripheral Module Quick Reference[online]*. Freescale, 2014 [cit. 2016-10-11], [Online; navštíveno 11.10.2016].
URL http://cache.freescale.com/files/32bit/doc/quick_ref_guide/KQRUG.pdf

[6] HDMI Licensing Administrator, Inc.: *HDMI :: Manufacturer :: Specification [online]*. HDMI Licensing Administrator, Inc., 2017 [cit. 2017-04-22], [Online; navštíveno 22.4.2017].
URL http://www.hdmi.org/manufacturer/specification.aspx

[7] I2C Info: *I2C Bus Specification[online]*. I2C Info, 2017 [cit. 2017-04-22], [Online; navštíveno 22.4.2017].
URL http://i2c.info/i2c-bus-specification

[8] Jan deno: *File:HDMI Connector.jpg - Wikimedia Commons[online]*. Jan deno, 2013 [cit. 2017-04-22], [Online; navštíveno 22.4.2017].
URL https://commons.wikimedia.org/wiki/File:HDMI_Connector.jpg

[9] Nemček, O.: *Knihovna pro komunikaci mikrokontroleru Kinetis K60 s SD kartou, bakalářská práce*. Brno: FIT VUT v Brně, 2015.

[10] NXP Semiconductors: *Kinetis Design Studio Integrated Development [online]*. NXP Semiconductors, 2017 [cit. 2017-03-28], [Online; navštíveno 28.3.2017].
URL http://www.nxp.com/products/software-and-tools/run-time-software/

kinetis-software-and-tools/ides-for-kinetis-mcus/kinetis-design-studio-integrated-development-environment-ide:KDS_IDE

[11] Scott Larson: *I2C Master (VHDL) - Logic - eewiki[online]*. 2015 [cit. 2016-08-20], [Online; navštíveno 20.08.2016].
URL https://eewiki.net/pages/viewpage.action?pageId=10125324

[12] Texas Instruments, Inc.: *TFP410 TI PanelBus^{TM} Digital Transmitter[online]*. Texas Instruments, Inc., 2016 [cit. 2016-08-17], [Online; navštíveno 17.08.2016].
URL http://www.ti.com/lit/ds/symlink/tfp410.pdf

[13] Xilinx, Inc.: *Spartan-6 Family Overview[online]*. Xilinx, Inc., 2011 [cit. 2016-09-02], [Online; navštíveno 02.09.2016].
URL https://www.xilinx.com/support/documentation/data_sheets/ds160.pdf

Přílohy

Příloha A

Adresový prostor Flexbus

0x60000000		Počátek adresního bloku FlexBus
0x60001000		Počátek bloku pro ovládání řadiče
0x60001000	R/W	Počátek bloku pro přístup k I2C registrům obvodu TFP410
⋮		Adresy registrů dodrženy dle datasheetu obvodu TFP410
0x6000103E	R/W	Konec bloku pro přístup k I2C registrům obvodu TFP410
0x60001040	R	Rozlišení řadiče v **x** ose
0x60001042	R	Rozlišení řadiče v **y** ose
0x60001044	R/W	Ovládání vykreslovacího automatu
⋮		*Nepoužité adresy v řadiči*
0x600010FF		Konec bloku pro ovládání řadiče
⋮		*Volné adresy*
0x60002000		Počátek bloku pro Sokoban/Tetris VRAM
0x60002000	R/W	Adresa do VRAM
0x60002002	R/W	Přenos dat pro nastavenou adresu
⋮		*Nepoužité adresy ve VRAM řadiči*
0x600020FF		Konec bloku pro Sokoban/Tetris VRAM
0x60002100		Počátek bloku pro Sokoban texturové VRAM
0x60002100	R/W	Adresa do VRAM
0x60002102	R/W	Přenos dat pro nastavenou adresu
0x600021FF		Konec bloku pro Sokoban texturové VRAM
0x60002200		Počátek bloku pro Sokoban fontové VRAM
0x60002200	R/W	Adresa do VRAM
0x60002202	R/W	Přenos dat pro nastavenou adresu
0x600022FF		Konec bloku pro Sokoban fontové VRAM
0x60003000	R/W	Pro sokoban ovladani mode registru vykreslovani
0x60003032		*Volné adresy*
0x6FFFFFFF		Konec adresního bloku FlexBus

Tabulka A.1: Adresní prostor MCU pro FlexBus

Příloha B

Popis funkcí knihovny pro MCU a komponenty řadiče pro FPGA

Funkce	Návratová hodnota	Parametry	Popis
tfp410_init	void	void	Úvodní inicializace Flexbus a řadiče
tfp410_read16	uint16_t	uint16_t addr	Čtení 16b dat z "addr"adresy v řadiči
tfp410_read8	uint8_t	uint8_t addr	Čtení 8b dat z "addr"adresy v řadiči
tfp410_write16	void	uint16_t addr, uint16_t data	Zápis 16b dat "data" na "addr"adresu v řadiči
tfp410_write8	void	uint16_t addr, uint8_t data	Zápis 8b dat "data" na "addr"adresu v řadiči

Tabulka B.1: Základní funkce knihovny

Funkce	Návratová hodnota	Parametry	Popis
tfp410_getResX	uint16_t	void	Získání rozlišení X
tfp410_getResY	uint16_t	void	Získání rozlišení Y
tfp410_setDraw	void	bool draw	Nastaví vykreslování ON/OFF
tfp410_getDraw	bool	void	Získání stavu vykreslování
vram_getAddr	uint16_t	void	Získání nastavené adresy z VRAM
vram_setAddr	void	uint16_t	Nastavení adresy VRAM
vram_getData	uint8_t	void	Získání dat z VRAM
vram_setData	void	uint8_t	Zaslání dat do VRAM

Tabulka B.2: Funkce pro ovladaní řadičů

Funkce	Návratová hodnota	Parametry	Popis
tfp410_getVendor	uint16_t	void	Získání ID výrobce
tfp410_getDevice	uint16_t	void	Získání ID zařízení
tfp410_getRevision	uint8_t	void	Získání revize zařízení
tfp410_getTDIS	uint8_t	void	Získání stavu TDIS
tfp410_getVEN	uint8_t	void	Získání stavu VEN
tfp410_getHEN	uint8_t	void	Získání stavu HEN
tfp410_getDSEL	uint8_t	void	Získání stavu DSEL
tfp410_getBSEL	uint8_t	void	Získání stavu BSEL
tfp410_getEDGE	uint8_t	void	Získání stavu EDGE
tfp410_getPD	uint8_t	void	Získání stavu PD
tfp410_getVLOW	uint8_t	void	Získání stavu VLOW
tfp410_getMSEL	uint8_t	void	Získání stavu MSEL
tfp410_getTSEL	uint8_t	void	Získání stavu TSEL
tfp410_getRSEN	uint8_t	void	Získání stavu RSEN
tfp410_getHTPLG	uint8_t	void	Získání stavu hot plug detect
tfp410_getMDI	uint8_t	void	Získání stavu MDI
tfp410_getDK	uint8_t	void	Získání stavu DK
tfp410_getDKEN	uint8_t	void	Získání stavu DKEN
tfp410_getCTL	uint8_t	void	Získání stavu CTL
tfp410_getCFG	uint8_t	void	Získání stavu CFG
tfp410_getDE_DLY	uint16_t	void	Získání stavu DE_DLY
tfp410_getDE_GEN	uint8_t	void	Získání stavu DE_GEN
tfp410_getVS_POL	uint8_t	void	Získání stavu VS_POL
tfp410_getHS_POL	uint8_t	void	Získání stavu HS_POL
tfp410_getDE_TOP	uint8_t	void	Získání stavu DE_TOP
tfp410_getDE_CNT	uint16_t	void	Získání stavu DE_CNT
tfp410_getDE_LIN	uint16_t	void	Získání stavu DE_LIN
tfp410_getH_RES	uint16_t	void	Získání stavu H_RES
tfp410_getV_RES	uint16_t	void	Získání stavu V_RES

Tabulka B.3: Funkce pro čtení registrů obvodu TFP410

Funkce	Návratová hodnota	Parametry	Popis
tfp410_setTDIS	void	bool set	Získání stavu TDIS
tfp410_setVEN	void	bool set	Získání stavu VEN
tfp410_setHEN	void	bool set	Získání stavu HEN
tfp410_setDSEL	void	bool set	Získání stavu DSEL
tfp410_setBSEL	void	bool set	Získání stavu BSEL
tfp410_setEDGE	void	bool set	Získání stavu EDGE
tfp410_setPD	void	bool set	Získání stavu PD
tfp410_setMSEL	void	uint8_t set	Získání stavu MSEL
tfp410_setTSEL	void	bool set	Získání stavu TSEL
tfp410_setMDI	void	bool set	Získání stavu MDI
tfp410_setDK	void	uint8_t set	Získání stavu DK
tfp410_setDKEN	void	bool set	Získání stavu DKEN
tfp410_setCTL	void	uint8_t set	Získání stavu CTL
tfp410_setDE_DLY	void	uint16_t set	Získání stavu DE_DLY
tfp410_setDE_GEN	void	bool set	Získání stavu DE_GEN
tfp410_setVS_POL	void	bool set	Získání stavu VS_POL
tfp410_setHS_POL	void	bool set	Získání stavu HS_POL
tfp410_setDE_TOP	void	uint8_t set	Získání stavu DE_TOP
tfp410_setDE_CNT	void	uint16_t set	Získání stavu DE_CNT
tfp410_setDE_LIN	void	uint16_t set	Získání stavu DE_LIN

Tabulka B.4: Funkce pro zápis do registrů obvodu TFP410

Příloha C

Obsah CD

bp_xmarek56.pdf	Text bakalářské práce
src_text	Složka se zdrojovými soubory technické zprávy
src_fpga	Složka s projekty pro prostředí Xilinx ISE
src_mcu	Složka s projekty pro prostředí KDS 3.2.0
skeleton_fpga	Složka se soubory potřebnými k novému projektu pro Xilinx ISE
skeleton_mcu	Složka se soubory potřebnými k novému projektu pro KDS
build_demo	Složka s přeloženým firmware pro FPGA i MCU pro demo aplikace
demo_video	Složka s video nahrávkou demo aplikací
manual	Složka s návodem na použití řadiče a knihovny

Tabulka C.1: Obsah přiloženého CD