VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

EMULÁTOR DOMÁCÍHO POČÍTAČE AMIGA A500 V FPGA

EMULATOR OF AMIGA A500 HOME COMPUTER IN FPGA

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. ZDENĚK BIBERLE**
AUTHOR

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

BRNO 2017

Zadání diplomové práce/19898/2016/xbiber00

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačových systémů Akademický rok 2016/2017

Zadání diplomové práce

Řešitel: **Biberle Zdeněk, Bc.**
Obor: Počítačová grafika a multimédia
Téma: **Emulátor domácího počítače Amiga A500 v FPGA**
Emulator of Amiga A500 Home Computer in FPGA
Kategorie: Počítačová architektura

Pokyny:

1. Podrobně prostudujte architekturu domácího počítače Amiga A500. Připravte krátké shrnutí vystihující jeho podstatné rysy.
2. Seznamte se s jazykem VHDL a vývojovými kity pro FPGA obvody firmy Xilinx, které jsou dostupné na FIT.
3. Navrhněte koncepci emulátoru počítače Amiga A500, který by využíval FPGA. Jako vstupní zařízení uvažujte klávesnici, PC myš (případně joystick) a paměťovou kartu SD.
4. Zvolte vhodnou platformu pro implementaci uvažované koncepce emulátoru. V návrhovém systému desek plošných spojů vytvořte rozšiřující modul pro připojení periferií.
5. V jazyce VHDL implementujte jednotlivé části návrhu. Můžete využít volně dostupných zdrojových kódů (např. z webu Opencores).
6. Vhodným způsobem ověřte funkčnost vytvořeného emulátoru. Zhodnoťte dosažené výsledky a diskutujte možná rozšíření.

Literatura:

- Dle pokynů vedoucího.

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

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci dřívějších projektů (30 až 40% 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: 24. 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

Tato práce popisuje a hodnotí existující hardwarové reimplementace počítače Amiga 500 a prozkoumává možnost realizace podobných řešení na vývojových platformách Minerva a Pipistrello. Výsledkem práce je emulátor počítače Amiga 500 postavený na kombinaci obou platforem.

Abstract

This work takes a look at existing hardware reimplementations of the Amiga 500 home computer and examines the possibility of implementing a similar solution based on the Minerva a and Pipistrello development platforms. The result of this work is an emulator od the Amiga 500 based on a combinator of both platforms.

Klíčová slova

Amiga, Amiga 500, A500, Commodore, FPGA, emulace, reimplementace, Minerva, Pipistrello

Keywords

Amiga, Amiga 500, A500, Commodore, FPGA, emulation, reimplementation, Minerva, Pipistrello

Citace

BIBERLE, Zdeněk. *Emulátor domácího počítače Amiga A500 v FPGA*. Brno, 2017. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Václav Šimek

Emulátor domácího počítače Amiga A500 v FPGA

Prohlášení

Prohlašuji, že jsem tento semestrální projekt vypracoval samostatně pod vedením Ing. Václava Šimka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Zdeněk Biberle
24. května 2017

Poděkování

Děkuji Ing. Šimkovi za zapůjčení kitů Minerva a Pipistrello, poskytnutí materiálů k počítači Amiga 500 a kitu Minerva a realizaci výroby desky plošných spojů.

Obsah

Kapitola 1

Úvod

Amiga 500 (také známý jako A500, interně „Rock Lobster“) je osobní počítač vyvinutý společností Commodore a uvedený na trh v roce 1987. Jedná se o počítač založený na procesoru Motorola MC68000, jde tedy o hybridní 32- a 16-bitový systém.

Počítač disponoval na svou dobu poměrně vysokým grafickým a zvukovým výkonem vzhledem k ceně. Z tohoto důvodu se stal oblíbeným hlavně pro hraní počítačových her, ale byl používán i pro výukové a kancelářské účely.

Cílem emulací a reimplementací tohoto počítače je tedy např. možnost spouštět specifické A500 verze aplikací, přístup k datům ve formátu, pro který je dostupný software pouze pro A500, hraní A500 videoher atd.

Obrázek 1.1: Fotografie počítače Amiga 500[1]

Cílem této práce je popsat a zhodnotit existující řešení, která hardwarově reimplementují tento počítač, a navrhnout vlastní řešení, které na těchto existujících řešeních staví a které ukáže možnosti nové hardwarové platformy Minerva.

[1] Autor: Bill Bertram, dostupné z https://commons.wikimedia.org/wiki/File:Amiga500_system.jpg.

Kapitola 2

Architektura počítače Amiga 500

V této kapitole se budeme zabývat klíčovými vlastnostmi hardwaru počítače Amiga 500. Pokud není uvedeno jinak, tak jsou informace v této části převzaty z [21].

Počítač Amiga 500 má následující komponenty:

- hlavní procesor Motorola MC68000;
- aplikačně specifické integrované obvody pro řízení systému a zpracování obrazu a zvuku. Amiga 500 obsahuje čtyři takové obvody nazvané Agnus, Gary, Denise a Paula;
- 512 KiB operační paměti, rozšířitelné až na 9 MiB;
- 256 KiB či 512 KiB paměti pouze pro čtení obsahující operační systém;
- vestavěná disketová mechanika pro diskety o velikosti 3,5 palce a externí port pro připojení další mechaniky;
- paralelní a sériový port;
- porty pro analogový i digitální grafický výstup;
- porty pro levý a pravý zvukový kanál;
- vestavěná klávesnice.

Blokové schéma počítače Amiga 500 převzaté z [9] je na obrázku 2.1. Zde vidíme hlavně tři integrované obvody, které tvoří tzv. OCS, tedy Original Chip Set. Tyto jsou:

- Agnus či později Fat Agnus – tento integrovaný obvod řídí přístup ostatních komponent (včetně CPU) k operační paměti a překládá adresy pro přístup k registrům.
- Denise – tento integrovaný obvod obsluhuje grafický výstup a sleduje pohyb myši a joysticků.
- Paula – tento integrovaný obvod slouží jako řadič zvuku a pomáhá se sériovou komunikací a řízením disketových mechanik.

Další klíčové komponenty jsou samozřejmě procesor Motorola MC68000, 512 KiB dynamické operační paměti, ROM s operačním systémem, dva integrované obvody MOS Technology 8520 pro řízení sériové a paralelní komunikace, klávesnice a disketových jednotek a konečně Gary, což je další čip specifický pro Amigu. Jeho činnost je zřejmě nezajímavá, Big Book of Amiga Hardware [28] jej popisuje následovně:

> The Gary chip isn't really much more than a gate array which contains some control logic for accessing certain busses.

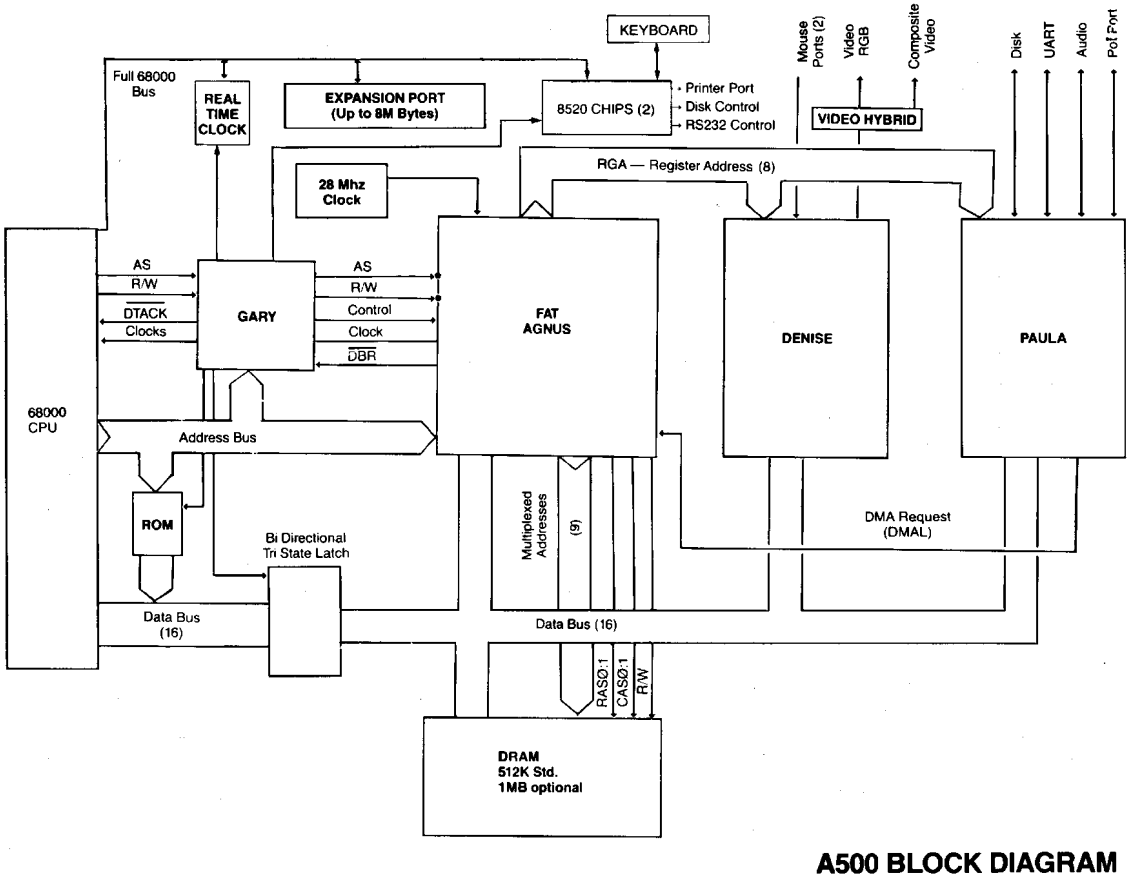
Můžeme si také všimnout, že Amiga 500 může mít dva druhy paměti: Jednu standardní, která je připojena k procesoru přes Agnus a může být sdílena se zbytkem OCS. Tato paměť je běžně nazývána „Chip RAM“. Další paměť může být připojena přímo ke sběrnici procesoru a není tedy přístupná z OCS. Ta je běžně nazývána „Fast RAM“, jelikož přístup k ní není nijak zpomalován přes Gary a Agnus. Tato paměť ovšem pochopitelně není dostupná pro DMA přenosy v rámci OCS.

Nyní se podívejme detailněji na obvody tvořící OCS.

2.1 Agnus

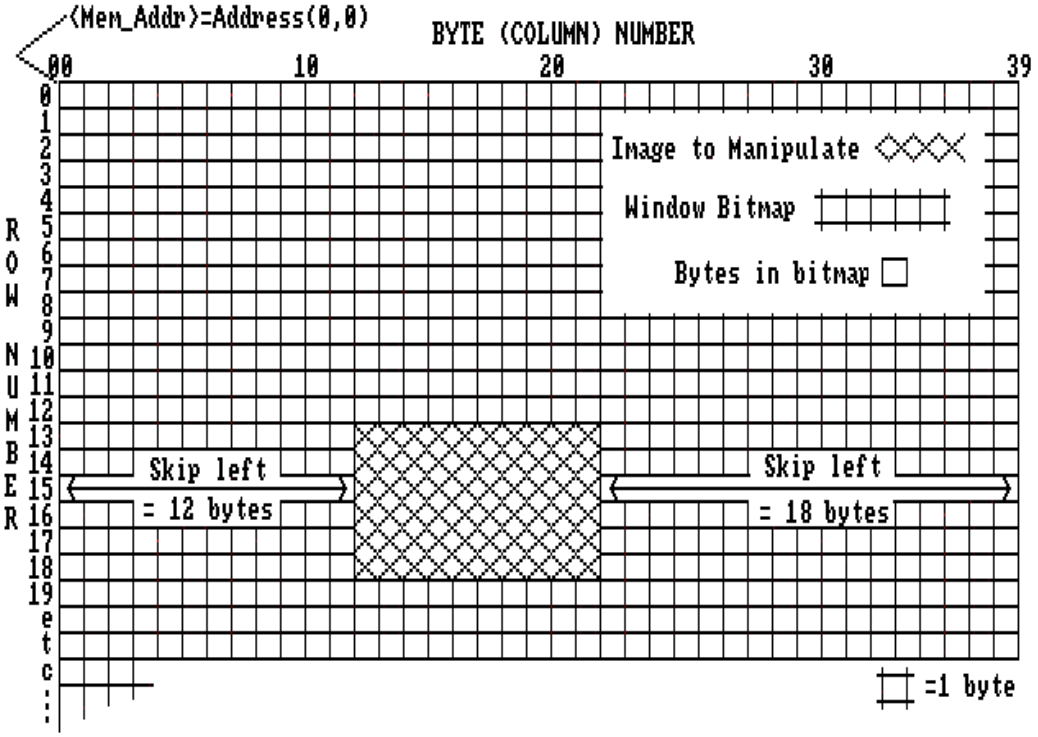
Jak již bylo zmíněno, tak Agnus je integrovaný obvod řídící přístup ostatních komponent k operační paměti („Chip RAM“). Agnus také poskytuje DMA kanály pro ostatní prvky OCS a snaží se střídat využití paměti mezi procesorem a zbytkem OCS a slouží jako generátor hodinových signálů.

Agnus obsahuje dvě komponenty, které hrají klíčovou roli v zajišťování flexibilního a výkonného grafického výstupu: Blitter a Copper.



Obrázek 2.1: Blokové schéma počítače Amiga 500 [9]

- Blitter je koprocesor určený pro grafické operace. Blitter umí kopírovat obdélníkové části paměti (typicky tedy bitmapové obrázky) a vykreslovat úsečky. Při kopírování paměti je Blitter přibližně dvakrát rychlejší než hlavní procesor a využívá až tři zdrojové DMA kanály, jejichž hodnoty může kombinovat jedním z 256 způsobů. Úsečky je Blitter schopný kreslit rychlostí až jeden milion pixelů za sekundu.
- Copper je velmi jednoduchý koprocesor s pouze třemi instrukcemi, jehož účel je synchronizovat přenos dat z paměti do registrů v rámci OCS s obrazovým paprskem. Copper tak může ovládat řadu parametrů, typicky se používá ke změně spritů, aktualizaci zvukových kanálů či dokonce řízení Blitteru.



Obrázek 2.2: Běžné použití Blitteru — modifikace části bitové roviny [21]

2.2 Denise

Denise je integrovaný obvod zaměřený primárně na grafický výstup, ale i na sledování pohybu myši a joysticků. Nás budou zajímat především grafické schopnosti tohoto čipu.

Denise poskytuje širokou škálu grafických módů kombinováním horizontálního rozlišení, prokládání a různého počtu bitových rovin.

Maximální počet barev na výstupu je 4096 a maximální viditelné rozlišení výstupu je 640×400 pixelů prokládaně u NTSC modelů a 640×512 pixelů prokládaně u PAL modelů.

Denise zvládá vykreslovat jeden nebo dva překrývající se bloky obrazových dat zároveň. Tyto bloky nazýváme „playfield“. Obraz můžeme ještě dodatečně obohatit o sprity z osmi jednotek dedikovaných na jejich kreslení.

2.2.1 Grafické módy

Každý grafický mód Denise je definován dvěma hlavními parametry: horizontálním rozlišením a vertikálním prokládáním.

Horizontální rozlišení může být 320 („low-resolution“) či 640 („high-resolution“) pixelů. Vertikální prokládání může být zapnuté a dosahovat zdánlivého vertikálního rozlišení až 400 pixelů u NTSC modelů či 512 pixelů u PAL modelů.

Horizontální rozlišení a prokládání lze libovolně kombinovat. Dodatečně je možné i měnit módy během zobrazování jednoho snímku a dosáhnout tak různých rozlišení na různých částech obrazovky. K tomuto účelu lze využít Copper, viz 2.1.

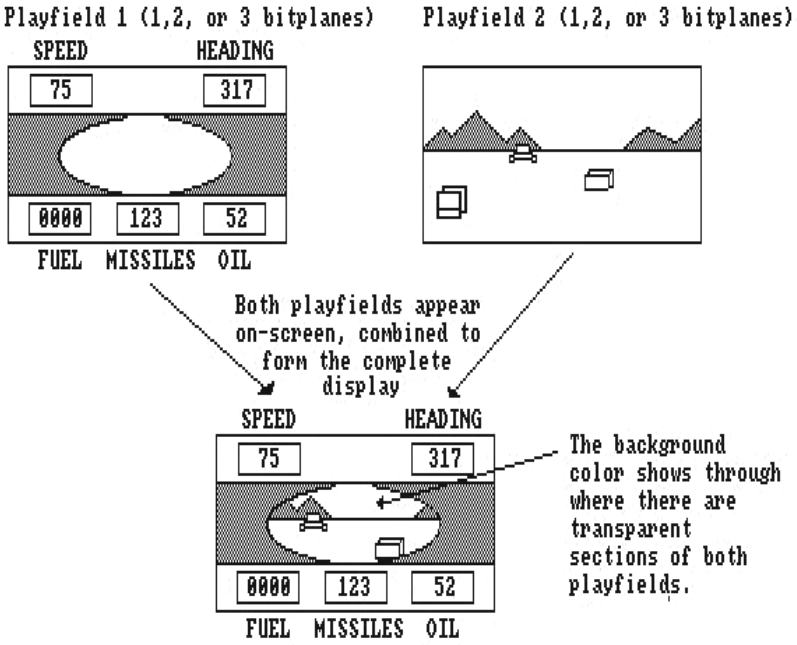
2.2.2 Playfieldy

Každý playfield je obdélníkový blok obrazových dat. Ten může být zobrazen celý a nebo pouze částečně a s libovolným posunem oproti počátku obrazovky. Barevná data playfieldu mohou pocházet až z šesti bitových rovin, ovšem jejich počet je omezen počtem playfieldů a použitým horizontálním rozlišením, viz tabulka 2.1.

Počet playfieldů	1	2
Low-resolution mód	6	3
High-resolution mód	4	2

Tabulka 2.1: Tabulka maximálního použitelného počtu bitových rovin

V případě, že je použito méně jak šest bitových rovin na jeden playfield, tak jsou barvy získávány z třiceti dvou registrů s barvami jednoduchou indexací registru dle hodnoty složené z jednotlivých bitových rovin. Každý z barevných registrů má velikost dvanáct bitů a obsahuje tedy jednu z dostupných 4096 barev.



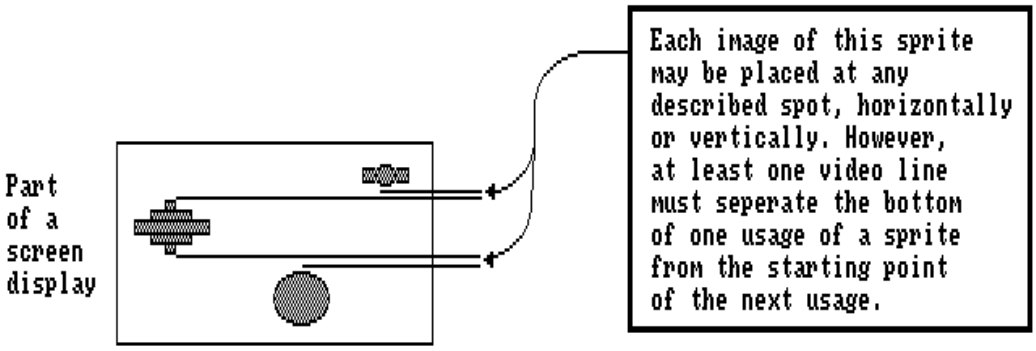
Obrázek 2.3: Demonstrace použití dvou playfieldů [21]

V případě použití šesti bitových rovin lze použít tzv. hold-and-modify mód, u kterého je barva pixelu určena buď dle jednoho z šestnácti barevných registrů nebo dle barvy předchozího pixelu s jednou barevnou složkou změněnou na přesně danou hodnotu. Tento mód umožňuje zobrazit všech 4096 možných barev v jednom snímku, ale pro dosažení konkrétní barvy může vyžadovat až tři pixely za sebou. Následkem této limitace je nemožnost zobrazit na řádku dvojice vedle sebe umístěných pixelů, které se mezi sebou výrazně liší ve více než jedné barevné složce, např. černá a bílá barva. Tento mód tedy můžeme považovat za formu ztrátové komprese obrazu oproti obrazu se dvanácti bity na pixel.

Pokud jsou navíc použity dva playfieldy, tak je jedna z dostupných barev průhledná, viz 2.3.

2.2.3 Sprity

Denise obsahuje osm speciálních DMA kanálů, které mohou být použity pro čtení spritů z paměti a jejich umístění na obrazovku. Každý sprite je široký šestnáct pixelů, libovolně vysoký a skládá se ze tří barev získaných z registrů s barevnou paletou a jedné průhledné barvy. Alternativně je možné spojit dva kanály v jeden a získat tak patnáct barev z barevné palety.



Obrázek 2.4: Typické použití jednoho kanálu pro vykreslení více spritů [21]

Každý kanál je schopný zobrazit libovolný počet spritů během jednoho snímku s jednou podmínkou: Všechny sprity jednoho kanálu se musí nacházet striktně nad sebou s vertikální mezerou o velikosti alespoň jeden řádek (viz obrázek 2.4). Tato funkcionalita je implementovaná přímo v Denise čtením pole s definicí spritů v Chip RAM tak dlouho, dokud toto pole není ukončeno dvěma nulovými slovy. Není tedy nutné pro tuto funkcionality využívat Copper.

2.3 Paula

Poslední z integrovaných obvodů tvořících OCS je Paula. Paula slouží primárně jako čtyřkanálový DA převodník pro výstup zvuku, ale obsluhuje i datové linky sériového portu, přerušení a společně s čipem Gary i disketové mechaniky. My se zaměříme na zvukové schopnosti tohoto čipu.

Paula obsahuje čtyři zvukové kanály, každý s osmibitovým DA převodníkem, šestibitovým registrem s hlasitostí a DMA kanálem pro získávání zvukových dat z Chip RAM.

Každý kanál může generovat zvuk nebo modulovat parametry jiného kanálu. Pokud kanál generuje zvuk, tak je napevno připojen na levý nebo pravý výstupní kanál.

Každý kanál může číst libovolně dlouhá zvuková data z libovolného místa v Chip RAM s téměř libovolným intervalem mezi vzorky. Zvukový výstup je synchronizován s grafickým výstupem a každý kanál má přidělený jeden DMA slot (tedy možnost přenést dva osmibitové vzorky) na jeden řádek obrazového výstupu. To limituje maximální vzorkovací frekvenci na teoretických 31 469 vzorků za sekundu. Praktické maximum je pak 28 867 vzorků za sekundu.

Pokud využijeme možnost kanálu modulovat jiný kanál, tak si můžeme zvolit modulaci hlasitosti, intervalu mezi vzorky nebo obou parametrů střídavě. Právě díky modulaci hlasitosti jiným kanálem je možné dosáhnout čtrnáctibitového zvuku (tj. osm bitů ze samotných zvukových vzorků a šest bitů z modulovaného registru s hlasitostí).

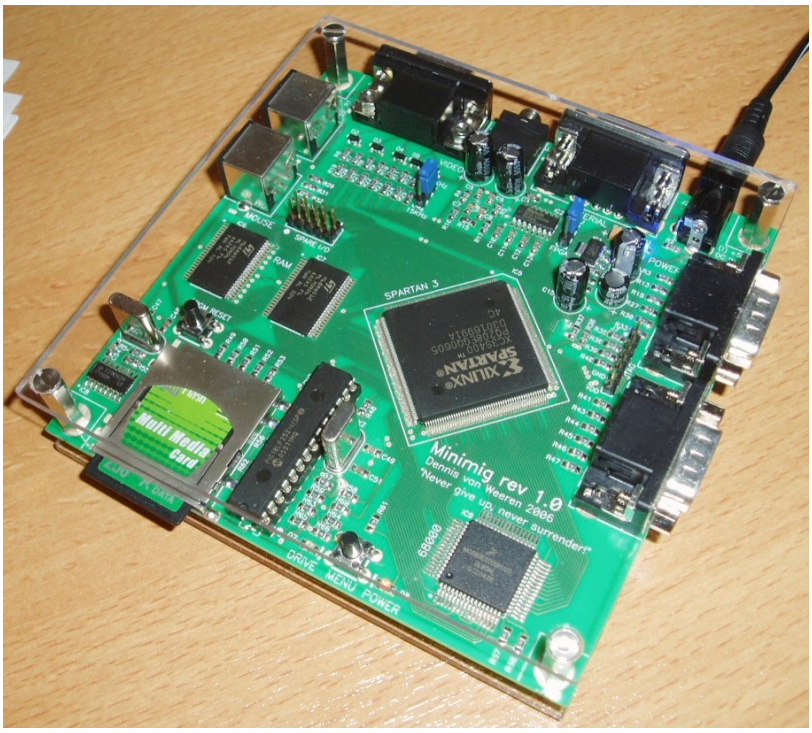
Kapitola 3

Existující reimplementace A500

V současnosti jsou k dispozici dva projekty, které reimplementují počítač Amiga 500 v hardware. Tyto projekty jsou Minimig a MIST. Reimplementace A500 u projektu MIST je založena na projektu Minimig.

3.1 Minimig

Minimig, jehož autorem je Dennis van Weeren, je projekt vzniklý v roce 2005 s cílem implementovat čipovou sadu počítače Amiga 500 v FPGA a s použitím několika externích komponent tak dosáhnout funkčního klonu A500 [26].



Obrázek 3.1: Fotografie sestaveného Minimigu[1]

[1] Autor: Dennis van Weeren, dostupné z https://en.wikipedia.org/wiki/File:Minimig_rev1.jpg.

3.1.1 Popis

Minimig je založen na FPGA Xilinx Spartan-3, konkrétně XC3S400-4PQ208. V tomto FPGA je implementována sada OCS. Minimig dále využívá procesor Freescale[2] MC68SEC000 místo původního Motorola MC68000. Tento procesor má výhodu v tom, že jeho pracovní napětí je 3.3 V. Díky tomu jej lze přímo připojit k použitému FPGA bez nutnosti převádět logické úrovně[3].

Dále se na Minimigu nachází mikrořadič PIC18LF252, který slouží pro programování FPGA, čtení paměťové karty a řízení menu na obrazovce. Zmíněná paměťová karta musí obsahovat minimálně bitstream pro FPGA a firmware pro A500, tzv. Kickstart.

Jako paměť byly na Minimigu původně použity dva integrované obvody STMicroelectronics M68AW512M, ty již ale nejsou dostupné, jejich vhodnou náhradou je ISSI IS62WV51216BLL-55TLI. Tyto paměti dohromady poskytují 2 MiB prostoru. 512 KiB je použito pro uložení firmwaru, který se na počítači Amiga 500 nacházel v ROM. Zbytek paměti je pak použit jako sdílená paměť a jako paměť pouze pro procesor.

Pro výstup obrazu je v Minimigu použito běžné VGA rozhraní s jednoduchým čtyřbitovým DA převodníkem pro každý barevný kanál. Pro zvukový výstup jsou pravděpodobně použity dva PWM signály filtrované dolní propustí[4].

Pro připojení periferií má Minimig k dispozici dva PS/2 porty pro připojení myši a klávesnice, dva porty pro připojení původních A500 joysticků a jeden sériový port.

3.1.2 Dostupnost

Projekt Minimig je pod licencí GPL verze 3, jsou tedy volně dostupné zdrojové soubory, schéma zapojení a návrh desky plošných spojů.

Existuje také malé množství zahraničních prodejců[5], kteří mají Minimig v nabídce, ovšem v době psaní tohoto textu (tj. začátek roku 2016) se mi nepodařilo najít prodejce, který by měl Minimig skladem.

Díky popularitě Minimigu jsou k dispozici i různá příslušenství pro něj, počínaje pouzdry a konče náhradou původního PIC mikrořadiče za mikrořadič Atmel AT91SAM7S256[6], který poskytuje cachování. U těchto se dostupnost liší.

3.1.3 Rozšířitelnost

Projekt Minimig se stal de facto standardním základem pro hardwarovou emulaci nejen počítače A500, ale i jiných osobních počítačů řady Amiga. Minimig se tak dočkal několika portů na jiné HW platformy, několika vylepšení ve formě verzí od Jakuba Bednarskiho, šířených na fórech www.amiga.org a dostupných také z domovských stránek projektu Minimig, a také se stal základem pro emulaci počítačů řady Amiga v projektu MIST, kterému se věnuje další část.

[2] Nyní NXP Semiconductors NV

[3] Proces známý jako level-shifting.

[4] Tento odhad byl proveden na základě schématu zapojení [25] a zdrojového souboru modulu `Paula`.

[5] Viz např. http://amigakit.leamancomputing.com/catalog/product_info.php?products_id=777 a https://acube-systemsbiz.serversicuro.it/shop/en/minimig/42-minimig-2mb-without-pic.html

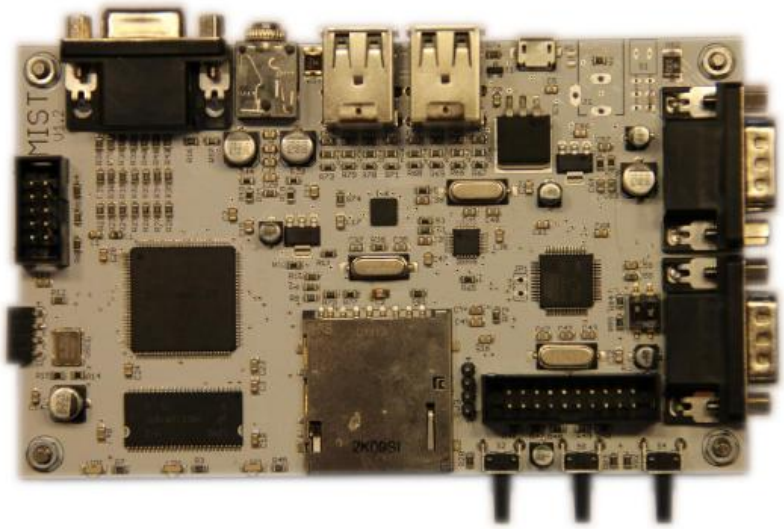
[6] Viz např. http://amigakit.leamancomputing.com/catalog/product_info.php?products_id=1132

3.1.4 Zhodnocení

Za výhody Minimigu lze označit např. podporu poměrně nových klávesnicí a myší díky portům PS/2 a současnou podporu původních A500 joysticků. Za nevýhodu můžeme považovat použití dnes již méně častého rozhraní VGA pro grafický výstup.

3.2 MIST

MIST (někdy také MiST) je projekt na první pohled podobný projektu Minimig, ovšem zatímco cílem Minimigu je emulovat pouze počítač Amiga 500, tak cílem projektu MIST je reimplementovat značný počet různých klasických šestnáctibitových počítačů, herních konzolí a videoherních automatů[4].



Obrázek 3.2: Fotografie osazené desky plošných spojů projektu MIST [4]

3.2.1 Popis

Svého cíle dosahuje MIST poměrně jednoduše: deska neobsahuje žádné příliš specifické komponenty, jako je tomu například u Minimigu a jeho Freescale MC68SEC000. Místo toho obsahuje MIST pouze FPGA Altera Cyclone EP3C25 a malé množství dalších podpůrných komponent. Hardware specifický pro zvolené emulované zařízení je poté implementován čistě v FPGA. Při startu je bitstream pro FPGA (který je u projektu MIST nazýván „core“) nahrán z SD karty.

Na desce MIST se kromě zmíněného FPGA nachází ještě tyto klíčové komponenty:

- Statická paměť Micron MT48LC16M16A2 sloužící jako operační paměť pro emulovaný hardware.
- Procesor Atmel AT91SAM7S256, který slouží jako V/V řadič pro paměťovou kartu a USB rozhraní.
- Řadič USB Maxim MAX3421E.

- USB hub TUSB2046B.

Pro grafický výstup MIST využívá, podobně jako Minimig, tradiční VGA rozhraní s jednoduchým DA převodníkem s šesti bity na barevný kanál. Pro zvukový výstup jsou použity dva PWM signály (jeden pro levý kanál a druhý pro pravý kanál) filtrované dolní propustí.

K desce MIST lze připojit klávesnici a myš pomocí čtyř portů USB, které obsluhuje zmíněný řadič Maxim MAX3421E. Dále je možné připojit dva joysticky pomocí dvou D-Sub 9 konektorů. Tyto jsou obsluhovány přímo přes FPGA. Deska také obsahuje sériové rozhraní, které je určené pro připojení rozšíření.

Počítač Amiga 500 lze emulovat použitím tzv. Amiga core [3], což je konfigurace FPGA, která vychází z projektu Minimig. Amiga core dodatečně využívá klon procesoru Motorola MC68000, jehož autorem je Tobias Gubener [8].

3.2.2 Dostupnost

Schéma zapojení, návrh desky plošných spojů a zdrojové soubory pro firmware a core pro MIST jsou volně k dispozici [2][7][6]. Licence těchto prostředků ovšem není jasná. V některých zdrojových souborech lze najít GPL verze 3, ovšem u již nepoužívaného repozitáře pro MIST lze najít GPL verze 2 [5].

MIST prodává hned několik zahraničních prodejců[7] a obecně se dá říci, že jsou skladem. Dostupná jsou i různá příslušenství, např. rozšiřující modul s MIDI rozhraním[8].

3.2.3 Zhodnocení

MIST je zjevně projekt z technického hlediska velice podobný projektu Minimig. Asi nejvýraznějším rozdílem je absence procesoru Freescale MC68SEC000. Navzdory tomu (či spíše díky tomu) je cíl projektu MIST výrazně odlišný.

Jasnou výhodou projektu MIST je velký počet emulovaných počítačů, herních konzolí a videoherních automatů a jednoduchá budoucí rozšiřitelnost. Výhodou je také použití novějšího USB HID rozhraní pro připojení myši a klávesnice.

Nevýhodou je, stejně jako u projektu Minimig, použití staršího rozhraní VGA pro grafický výstup.

[7] Viz např. http://lotharek.pl/product.php?pid=96 a http://amigastore.eu/en/318-mist-fpga-computer.html

[8] Viz http://lotharek.pl/product.php?pid=151

Kapitola 4

Návrh architektury implementace

S informacemi nabytými v kapitole 3 se můžeme pokusit vytvořit vlastní reimplementaci počítače Amiga 500. Nejprve si ujasněme, co od výsledného řešení očekáváme:

- Moderní rozhraní pro připojení klávesnice, myši a zobrazovacího zařízení. V ideálním případě tedy široce dostupné USB a DVI, HDMI či DisplayPort.
- Možnost připojení klasických joysticků[1].
- Běžný analogový zvukový výstup a dodatečně případně i digitální zvukový výstup, např. přes rozhraní HDMI.
- Emulace disketové mechaniky na základě obrazů disket uložených na paměťové kartě.
- OCS založené na ověřeném základu, tedy na projektu Minimig.

Pokud se tedy inspirujeme návrhem projektu Minimig rozšířeného o tyto požadavky, tak dojdeme přibližně k architektuře dle obrázku 4.1. Zde vidíme OCS ve spojení s procesorem Motorola MC68000 a operační pamětí emulující jádro počítače A500. Tomuto jádru jsou poskytovány vstupy z hostitelského procesoru, který realizuje komunikaci s periferiemi a emulaci disketové mechaniky.

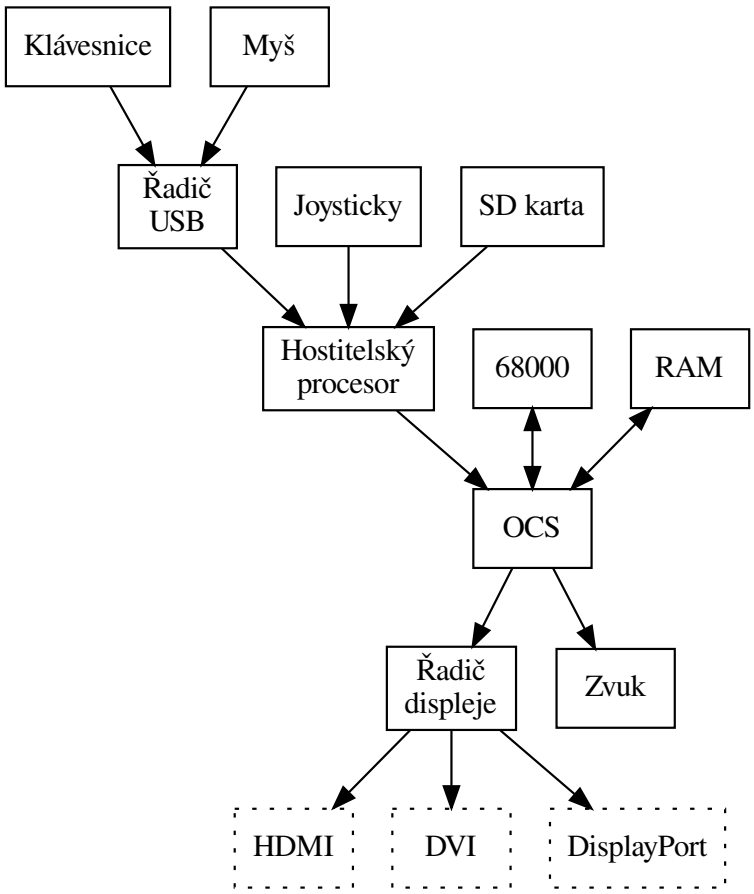
Na výstupu OCS je pak vhodný řadič zobrazovacího zařízení a vhodný zvukový výstup.

Rozhodně se nejedná o jediný možný návrh systému. Např. komunikace s joysticky by mohla být realizována stejně jako u projektu Minimig, tj. mít joysticky připojené přímo k OCS. Možnými alternativními řešeními se bude tato kapitola zabývat později.

Na základě tohoto hrubého návrhu již můžeme dělat určité závěry. Je například zjevné, že logiku OCS budeme, stejně jako u projektu Minimig, implementovat v FPGA. Další části této architektury už tak zjevné nejsou, jelikož značnou část návrhu (např. procesor Motorola MC68000, řadič USB, hostitelský procesor a řadič zobrazovacího zařízení) můžeme umístit do FPGA či je mít přítomny fyzicky. Kterou variantu u jednotlivých komponent použijeme je pak hlavně otázkou dostupnosti prostředků na FPGA a složitosti implementace v FPGA oproti použití hotového řešení.

Jako inspiraci pro konkrétní řešení mi posloužily vývojové kity Pipistrello od Saanlima Electronics a Minerva, nový vývojový kit vyvinutý na FIT VUT, který by v budoucnosti měl nahradit současný FITKit 2.0. Nejprve se tedy seznámíme s možnostmi těchto kitů a následně se budeme zabývat variantami umístění, propojení a komunikace klíčových komponent celého systému.

[1] Repliky těchto jsou dnes stále k dostání, viz např. http://amigastore.eu/en/307-competition-pro-joystick-retro.html



Obrázek 4.1: Architektura navrhované implementace

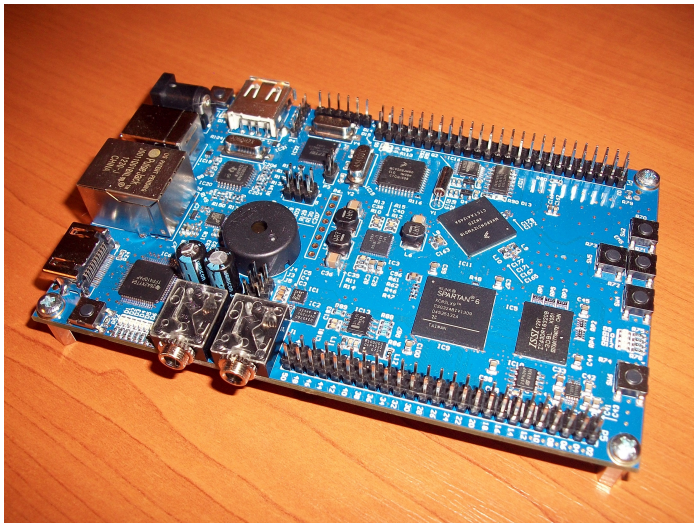
4.1 Kit Minerva

Kit Minerva se hned na první pohled jeví jako takřka ideální pro naše účely. Kit obsahuje značné množství komponent, které jsou schopny plnit funkce popsané v obrázku 4.1. Jedná se především o následující[2]:

- Xilinx XC6SLX9-2CSG324C – FPGA srovnatelné s XC3S400-4PQ208, které se nachází v Minimigu.
- Freescale PK60N512VMD100 – Tento mikrořadič může vykonávat podpůrnou činnost, např. inicializaci systému či dekódování souborového systému na použitém datovém úložišti, podobně jako u projektů Minimig a MIST.
- ISSI IS43DR16320B-3DBL – Paměť typu DDR2 organizovaná jako 32 M × 16 b [16] připojená přímo k V/V bance 1 FPGA a lze tedy využít Xilinx Memory Interface Generator pro zajištění komunikace s touto pamětí.

[2]Informace převzaty ze schématu kitu Minerva, revize 0.92, jehož autorem je Ing. Václav Šimek.

- FTDI Vinculum-II VNC2-48Q1B – Programovatelný hostitelský řadič USB, který může být využit k připojení klávesnice a myši. Na desce Minerva se sice nachází pouze jeden USB konektor typu A samice. Připojení obou periférií je tak možné přes USB rozbočovač.
- Freescale SGTL5000XNAA3 – Zvukový kodek, připojený přes I^2S k FPGA.
- Texas Instruments TFP410PAP – Řadič pro obrazové DVI rozhraní, připojený přes 24 datových linek (osm bitů na každý ze tří barevných kanálů) k FPGA. Díky tomuto řadiči tedy získáme moderní DVI rozhraní (na desce je ovšem použit konektor HDMI) [23].
- Slot pro paměťovou kartu – Tento je připojen přímo k mikrořadiči a může sloužit jako jednoduše vyměnitelný zdroj firmware a obrazů disket.



Obrázek 4.2: Fotografie kitu Minerva

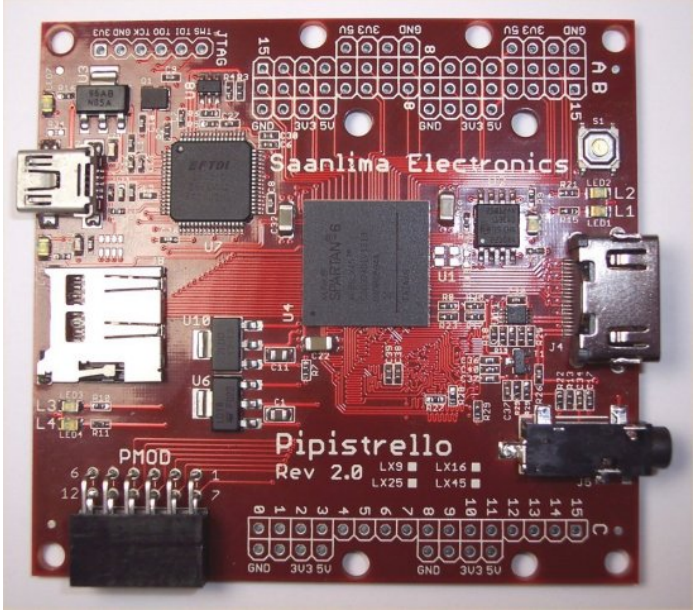
K dispozici je také 44 pinů FPGA vyvedených přímo na rozšiřující konektor P5 na desce a také rozšiřující konektor P1, který poskytuje přístup k několika pinům mikrořadiče a k USB řadiči Vinculum-II. Tyto můžeme využít pro připojení joysticků.

4.2 Kit Pipistrello

Kit Pipistrello je vývojová platforma od Saanlima Electronics a disponuje následujícím hardware:

- Xilinx XC6LX45-3CSG324C
- Micron N25Q128A13ESE40G – 128 Mb SPI flash paměť pro programování FPGA
- Micron MT46H32M16LFBF-5 – Paměť typu LPDDR organizovaná jako 32 M × 16 b [17] připojená přímo v V/V bance 3 FPGA a lze tedy využít Xilinx Memory Interface pro zajištění komunikace s touto pamětí.

- FTDI FT2232H – Dvoukanálový převodník mezi USB 2.0 a sériovými rozhraními. Na desce Pipistrello je jeden kanál konfigurován jako JTAG rozhraní pro programování FPGA a druhý kanál je konfigurován jako RS-232 rozhraní připojené přímo k FPGA použitelný k libovolné komunikaci.
- HDMI či DVI výstup prostřednictvím HDMI konektoru typu A.
- Stereo zvukový výstup prostřednictvím 3,5 mm konektoru připojeného přes pasivní dolní propust přímo k FPGA.
- Slot pro paměťovou kartu připojený přímo k FPGA.



Obrázek 4.3: Fotografie kitu Pipistrello[3]

Kit Pipistrello samotný je méně vhodný pro implementaci emulátoru než kit Minerva z důvodu absence vhodné komponenty pro řízení systému (jako např. mikrořadič PIC18LF252 v Minimigu) a uživatelský vstup (jako např. USB řadič FTDI Vinculum-II na kitu Minerva). Oba tyto nedostatky lze obejít za cenu značného množství prostředků na FPGA implementací USB rozhraní a podpory čtení SD karet a dekódování souborového systému. Pro oba tyto účely se ovšem více hodí programovatelný procesor, jako je tomu v Minimigu a na kitu Minerva.[4]

Výhodou kitu Pipistrello oproti kitu Minerva je podstatně schopnější FPGA (více než čtyřikrát větší počet slices [30, s. 2] a větší počet dostupných V/V pinů FPGA (48 či 56 pokud počítáme i konektor PMOD).

[3] Fotografie dostupná z http://pipistrello.saanlima.com/index.php?title=File:Pipistrello_v2_25.jpg.

[4] Zde se lze inspirovat např. z varianty Minimigu od Tobiase Gubenera pro vývojovou platformu Terasic Altera DE1, která využívá v FPGA implementovaný procesor z projektu OpenRISC 1200, viz https://github.com/rkrajnc/minimig-de1/blob/master/rtl/ctrl/ctrl_top.v.

4.3 Procesor MC68000

Procesor Motorola MC68000 je nepochybně klíčovou součástí počítače Amiga 500. Později také uvidíme, že se jedná o složitou komponentu a emulace tohoto procesoru v FPGA vyžaduje značné množství prostředků. To je jistě jeden z důvodů, proč projekt Minimig využívá externí procesor Freescale MC68SEC000. Na druhou stranu projekt MIST, který disponuje výrazně schopnějším FPGA, si může dovolit procesor emulovat.

Podívejme se tedy na možnosti implementace procesoru v FPGA kitů Minerva a Pipistrello a také na možnosti připojení externího procesoru k oběma kitům.

4.3.1 Implementace procesoru v FPGA

Alternativou k použití externího procesoru je implementace procesoru v rámci FPGA, podobně jako u projektu MIST a jeho Amiga core. Toto řešení má oproti externímu procesoru hlavní výhodu v tom, že propojení procesoru a OCS je triviální. Nevýhodou je pochopitelně vyšší využití prostředků na FPGA.

Pokud uvažujeme použití stejné softcore implementace Motorola MC68000 jako MIST, tj. TG68k od Tobiase Gubenera, tak musíme počítat s využitím cca 90 % prostoru na současném FPGA na kitu Minerva. Přibližně stejný prostor vyžaduje i implementace OCS počítače Amiga 500 z projektu Minimig. Je tedy zřejmé, že je prakticky nemožné obě tyto komponenty umístit na současné FPGA.

Potenciálním řešením tohoto problému by bylo nahrazení FPGA za schopnější model. Na kitu Minerva se v současnosti nachází XC6SLX9 v pouzdře CSG 324. Ve stejném pouzdře jsou k dispozici i XC6SLX16, XC6SLX25 a XC6SLX45 [30]. XC6SLX25 by mělo být dostatečné pro TG68k, Minimig a další podpůrnou logiku dohromady.

V případě nahrazení by tak došlo k podobné situaci jako v současnosti u kitu FITkit, kde některé FITkity jsou osazeny FPGA Xilinx XC3S50 a některé FPGA Xilinx XC3S400 [24].

Výhodou tohoto řešení je ponechání rozšiřujícího portu P5 na kitu Minerva volným a tedy umožňuje jednoduché připojení např. joysticků či zařízení na lince RS-232 přímo k FPGA. Za výhodu lze považovat také to, že umožňuje budoucím studentům s přístupem k patřičné variantě kitu Minerva jednoduše emulovat počítač Amiga 500 bez nutnosti dodatečného hardware ve formě popsané v předchozí části.

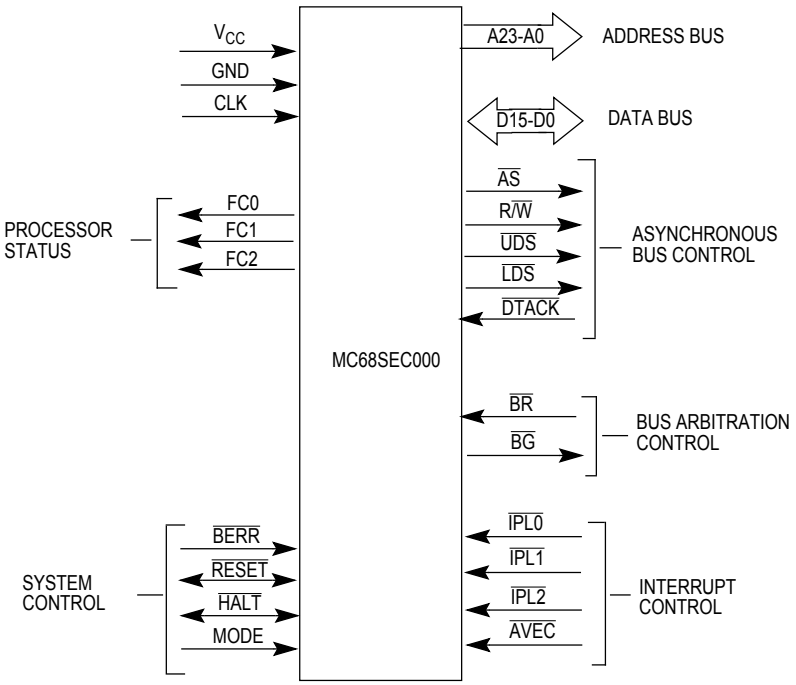
4.3.2 Použití externího procesoru

Pokud se rozhodneme použít externí procesor, tak musíme odpovědět na dvě klíčové otázky:

1. Jaký zvolíme konkrétní model procesoru?
2. Jak procesor připojíme ke zbytku systému?

Odpověď na první otázku je poměrně jednoduchá: Můžeme použít stejný procesor, jako je použit v projektu Minimig, tedy Freescale MC68SEC000. Taktéž využijeme výhody, že pracovní napětí tohoto procesoru je 3.3 V a můžeme jej tedy připojit přímo na FPGA kitu Minerva či Pipistrello.

Odpověď na druhou otázku již tak jednoduchá není. Podívejme se nejprve na schématické znázornění procesoru Freescale MC68SEC000 na obrázku 4.4. Z tohoto schématu je zjevné, že procesor Freescale MC68SEC000 má 59 signálů, které potenciálně musíme připojit do systému. Ideálně chceme všechny tyto signály přivést do FPGA a zapojit tak procesor do OCS.



Obrázek 4.4: Schématické znázornění procesoru Freescale MC68SEC000

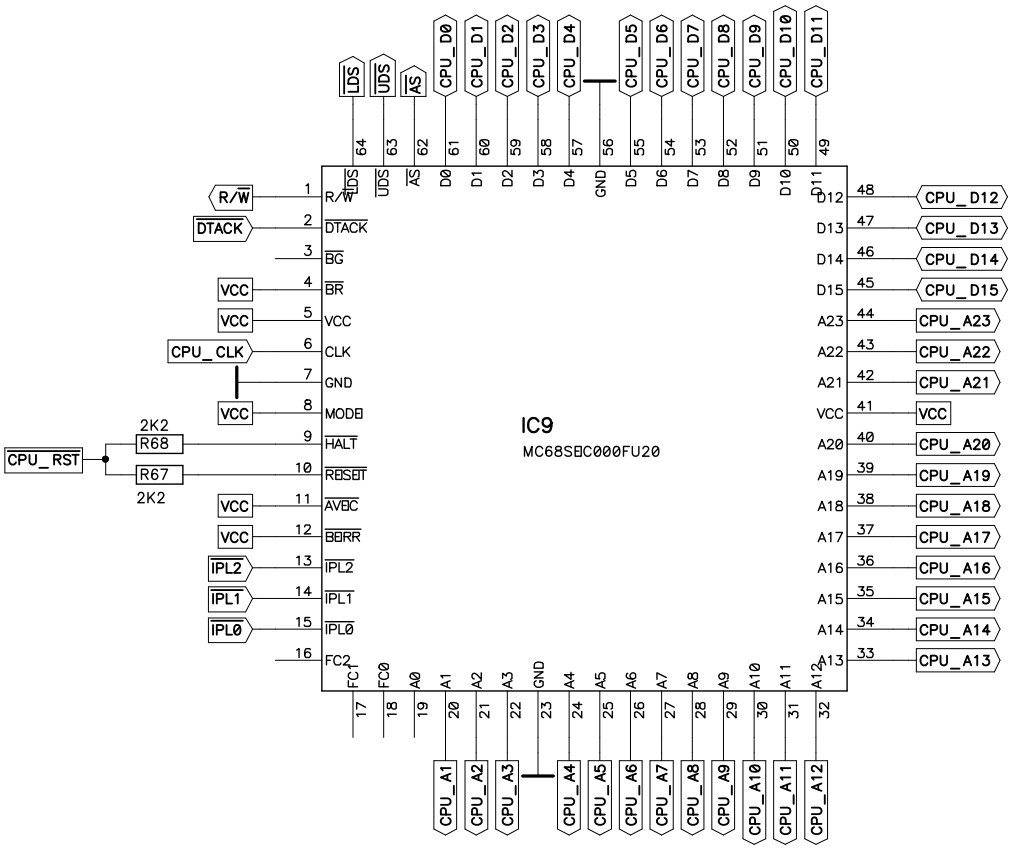
Na kitu Minerva je, jak již bylo zmíněno v kapitole 4.1, k dispozici pouze 44 pinů FPGA vyvedených na rozšiřující konektor na desce. Z tohoto důvodu nemáme žádný jednoduchý způsob, jak všechny signály z procesoru k FPGA připojit. Podívejme se tedy na to, jak je procesor zapojen v projektu Minimig na obrázku 4.5. Zde si všimneme následujících faktů:

- Stavové signály FC0, FC1 a FC2 jsou nezapojeny.
- Nejnižší bit adresové sběrnice, A0, je taktéž nezapojen. To se shoduje se schématem A500 [9, příloha B].
- Signály MODE, $\overline{\text{BERR}}$, $\overline{\text{AVEC}}$, $\overline{\text{BR}}$ a $\overline{\text{BG}}$ jsou buď nezapojeny a nebo jsou napojeny k napájecímu napětí.
- Signály $\overline{\text{HALT}}$ a $\overline{\text{RESET}}$ můžeme spojit do jednoho.

Díky těmto poznatkům můžeme snížit počet použitých signálů z 59 na 49. To je stále o pět více, než si můžeme dovolit. Jelikož potřebujeme snížit počet použitých signálů ještě mnohem výrazněji, tak musíme hlouběji zasáhnout do komunikace mezi procesorem, OCS a zbytkem systému.

Druh multiplexorů	2:1	4:1	8:1
Přidané řídící signály	1	2	3
Počet ušetřených signálů	11	16	18
Celkový počet signálů mezi CPU a OCS	38	33	31

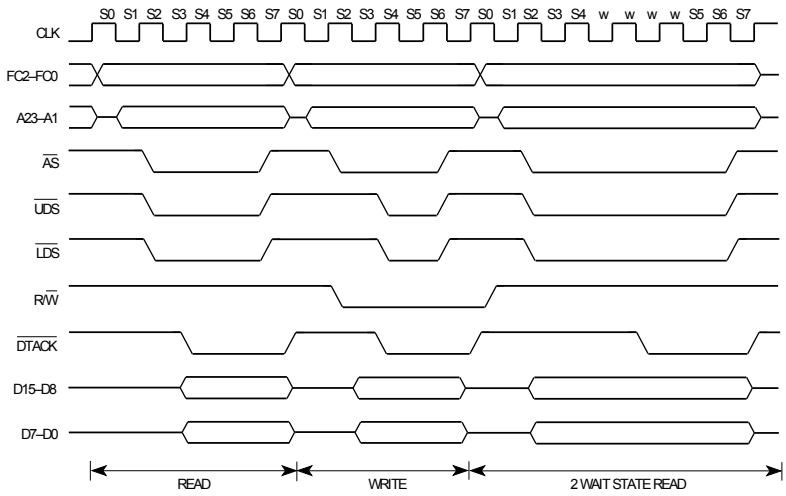
Tabulka 4.1: Počty ušetřených signálů při zavedení multiplexování adresové sběrnice v závislosti na počtu vstupů použitých multiplexorů



Obrázek 4.5: Schéma zapojení procesoru Freescale MC68SEC000 v projektu Minimig[25]

Jako zřejmě nejjednodušší možnost se jeví multiplexování adresové sběrnice, u které jsme díky její poměrně velké šířce schopni výrazně ušetřit signály. Na straně OCS v FPGA pak není problém přidat klopné obvody a rekonstruovat tak stav adresové sběrnice. Počty ušetřených signálů jsou znázorněny v 4.1.

Nesmíme ovšem zapomenout, že přidáním multiplexování adresy zpožďujeme platnost hodnoty na adresové sběrnici v OCS. Podívejme se tedy na časový diagram čtecího a zápisového cyklu procesoru Motorola MC68000 na obrázku 4.6. Zde vidíme, že při čtení dojde po vystavení adresy na adresovou sběrnici k oznámení její platnosti signálem $\overline{\text{AS}}$. Následně proběhne čtení a poté adresované zařízení oznámí platnost dat na datové sběrnici signálem $\overline{\text{DTACK}}$. Při zápisu je situace podobná.



Obrázek 4.6: Časový diagram čtecího a zápisového cyklu procesoru Motorola MC68000 [11]

Jelikož vystavení adresy mírně předchází aktivaci signálu $\overline{\text{AS}}$, tak je zcela reálné provádět přenos během fáze S1. Při dostatečné rychlost multiplexovací logiky bude tato činnost pro zbytek systému zcela transparentní.

Bohužel při použití multiplexorů s poměrem 2:1 či 4:1 přijdeme o možnost přímého připojení dodatečných joysticků, jelikož na rozšiřujícím konektoru nezůstane dostatek volných signálů (pro dva joysticky je potřeba 12 signálů). Nejedná se ovšem o příliš velký problém a můžeme je připojit k mikrořadiči na desce Minerva, který pak může předávat informace do FPGA.

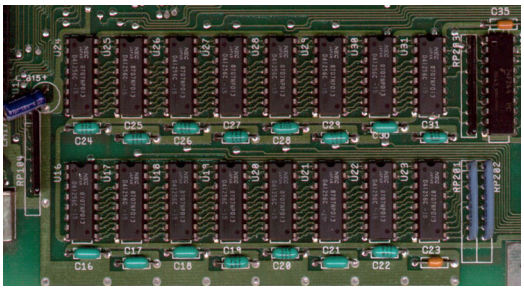
4.4 Operační paměť

Kity Minerva a Pipistrello disponují dynamickou pamětí o dostatečné velikosti pro implementaci operační paměti a ROM pro firmware emulovaného počítače. Podívejme se tedy na detaily paměti počítače Amiga 500, jak je tato paměť replikována v projektu Minimig a jak ji můžeme replikovat na kitech Minerva či Pipistrello.

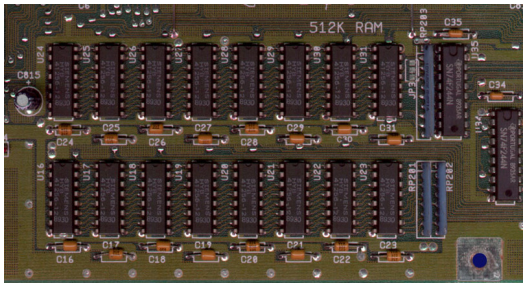
Informace o paměti použité v počítači Amiga 500 nejsou bohužel příliš rozsáhlé. Dokumenty [21] a [9] do detailů paměti nezasahují. Malé množství informací můžeme získat z fotografií různých revizí základní desky základních desek počítače Amiga 500, viz 4.7. Zde vidíme několik rozdílných konfigurací s odlišnými paměťovými integrovanými obvody:

- Revize 3 obsahuje 16 paměťových IO NEC μPD41256. Každý z těchto IO je vnitřně organizována jako 256 k × 1 b [18].
- Revize 5 používá stejnou konfiguraci jako revize 3, ovšem za použití Siemens HYB 41256-12, které jsou prakticky shodné s NEC μPD41256 [22].
- Revize 6a se odlišuje použitím pouze čtyř pamětí organizovaných jako 256 k × 4 b, a to Fujitsu MB81C4256-12 [13].

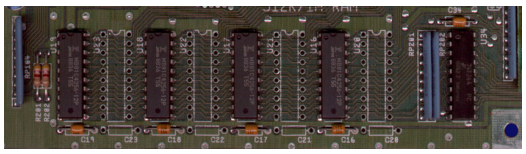
[5] Viz http://amiga.resource.cx/mod/a500.html.



(a) Revize 3



(b) Revize 5

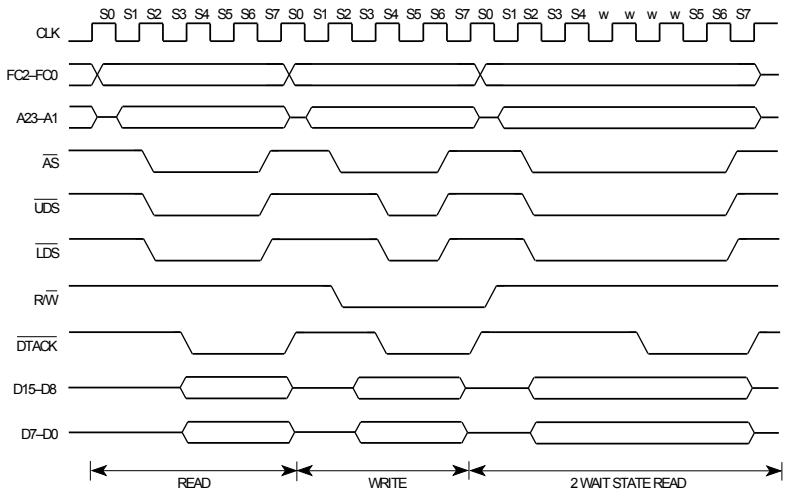


(c) Revize 6a

Obrázek 4.7: Rozdílné konfigurace operační paměti počítače Amiga 500 na různých revizích jeho základní desky[5].

Všechny tyto použité paměti mají jednu společnou vlastnost, jedná se o dynamické paměti. Je tedy nasnadě, že Agnus, který koordinuje přístup procesoru a zbytku čipové sady k paměti, bude zároveň sloužit i k řízení těchto pamětí, včetně např. obnovování.

V projektu Minimig je ovšem operační paměť řešena jinak — paměť je tvořena dvěma integrovanými obvody STMicroelectronics M68AW512M, každý z nich je organizován jako 512 k × 16 b, celkem tedy 2 MiB. Čtecí i zapisovací latence těchto pamětí je 55 ns, čehož Minimig využívá. Experimentálně jsem zjistil, že paměť na Minimigu nepodléhá běžnému paměťovému cyklu procesoru MC68000. Místo toho se Minimig spoléhá právě na nízkou latenci použité statické paměti a adresu a prováděnou operaci vystavuje paměti pouze jeden takt procesoru před dokončením paměťového cyklu.



Obrázek 4.8: Časový diagram čtecích a zápisovvých cyklů procesoru Motorola MC68000

Porovnáme-li např. běžný čtecí paměťový cyklus procesoru MC68000 znázorněný v levé části obrázku 4.8 s přístupem do paměti tak, jak je realizován na Minimigu, tak zjistíme, že Minimig očekává, že celé čtení proběhne během fáze S6.

Pokud uvažujeme hlavní hodinový takt rovný 7,093 79 MHz, tak jedna polovina taktu (tj. čas ekvivalentní trvání fáze S6 čtecího cyklu) trvá přibližně 70,5 ns. Odečteme-li latenci pamětí, tj. 55 ns, tak získáme pžibližně 14,5 ns pro propagaci signálů mezi FPGA a pamětmi. V praxi bude tento čas o něco delší, jelikož paměťové operace začínají mírně před začátkem fáze S6, zřejmě synchronizovaně s hodinovým signálem `qclk` Minimigu, což je pouze hlavní hodinový signál s fázovým posunem 90°.

Nyní se tedy zaměřně na dosažení stejného či lepšího časování s dynamickou pamětí na kitech Minerva a Pipistrello. Předpokládejme použití nástroje Memory Interface Generator pro FPGA Spartan-6. Dle [29, s. 57] můžeme očekávat latenci čtení až 85 ns pro paměti s CAS latencí rovnou pěti taktům. Ačkoliv jsou paměti na kitech Minerva a Pipistrello schopny pracovat s CAS latencí nejméně 2 takty, tak k výraznému zkrácení celkové latence bohužel nedojde.

Abych se ujistil, že tomu tak vskutku je, provedl jsem malé množství testů s OCS z projektu Minimig, emulovaným procesorem MC68000 a jednoduchým ovladačem paměti zaměřené na měření právě latence paměti. Výsledky testů byly zachyceny integrovaným logickým analyzátorem pro nástroj ChipScope. Podmínky testů byly následující:

- Takt OCS a procesoru byl roven přibližně čtvrtině nominální frekvence počítače Amiga 500.
- Takt ovladače paměti byl v různých testech 100 MHz, 200 MHz a 400 MHz. Níže uvedené závěry jsou z varianty s taktem 200 MHz. Celkový vliv změny taktu ovladače paměti je minimální — v rámci celé operace je mu věnováno pouze malé množství taktů.
- Takt logického analyzátoru byl vždy roven taktu ovladače paměti.
- Sledovány byly primárně signály značící začátek a konec operace paměti.

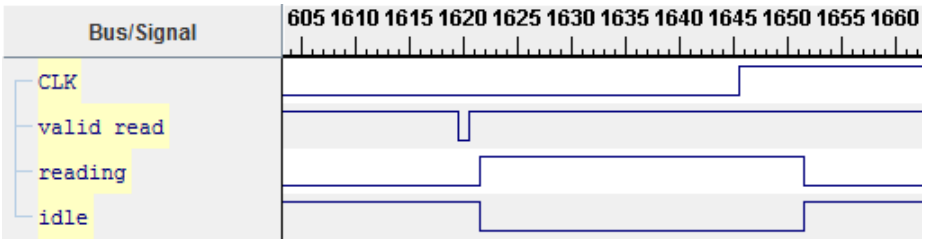
Na obrázku 4.9 je vidět časový diagram jedné čtecí operace provedené při taktu analyzátoru i ovladače paměti rovném 200 MHz. Z diagramu jsme schopni vyčíst následující:

- Začátek čtecí operace (sestupná hrana signálu `valid_read`) předchází hodinový signál OCS a procesoru (`CLK`) o 125 ns, tedy o méně než čtvrtinu periody signálu `CLK` (560 ns).
- Celá čtecí operace začíná vzorkem 1620 a končí vzorkem 1652. Celková latence čtecí operace je tedy 160 ns.

Příčinu takto vysoké latence paměti se mi bohužel nepodařilo zjistit, což ovšem znamená, že paměť kitu Pipistrello není schopna splnit časování OCS Minimigu.

Poměrně jednoduchým řešením této situace je použití dodatečné externí statické paměti, stejně jak je tomu u projektu Minimig. Paměť vhodná pro použití s jádrem Minimigu musí splňovat následující charakteristiky:

- Celková kapacita alespoň 2 MiB a organizace jako 1 M × 16 b.
- Možnost zápisu pouze horních či spodních osmi bitů nebo všech šestnácti bitů zároveň. Čtení může být vždy prováděno



Obrázek 4.9: Zachycený časový diagram čtecí operace paměti.

- Dostatečně nízká latence splňující požadavky Minimigu.

Jako vhodná volba se jeví např. IS62WV102416DBLL-45TLI od Integrated Silicon Solution s latencí 45 ns [15] či IS61WV102416BLL-10TLI s latencí až 8 ns [14], ale jistě i mnohé další.

Pro připojení paměti jsou zpravidla nutné následující signály:

- 20 adresních,
- 16 datových,
- signály výběru nižšího a vyššího bajtu,
- signál aktivující zápis,
- případně i signál aktivující výstupní budiče paměti.

Celkem se tedy jedná o 39 či 40 signálu. Kity Minerva a Pipistrello poskytují 44 a 56 pinů FPGA v podobě pinů na desce plošných spojů a připojení je tedy v obou případech realizovatelné bez problému.

4.5 Vstupní periferní zařízení

Počítač Amiga 500 disponoval vestavěnou klávesnicí a dvěma porty použitelnými pro připojení myší, joysticků, či jiných periférií. V projektu Minimig je komunikace s těmito vstupními zařízeními realizována přímo v FPGA. Myš i klávesnice jsou připojeny přes rozhraní PS/2 a tradiční joysticky jsou připojeny přímo konektorem D-sub 9.

Na kitech Minerva a Pipistrello lze jistě realizovat podobné řešení, které by navíc mohlo nahradit rozhraní PS/2 za rozšířenější rozhraní USB. Na kitu Minerva je pochopitelně možné použít dostupný USB řadič FTDI Vinculum-II. Na kitu Pipistrello je ovšem nutné řídit USB komunikaci z FPGA.

Alternativou k přístupu použitému v Minimigu je připojení všech periferií k hostitelskému procesoru a následně předávání této komunikace do OCS. Na kitu Pipistrello se jedná prakticky o jedinou možnost, jelikož řízení USB komunikace je úkol vhodný pro programovatelný procesor. Na kitu Minerva tato povinnost odpadá, jelikož samotný řadič USB FTDI Vinculum-II je plně programovatelný [12]. Tato alternativa ovšem nabízí i další výhody:

- Minimig posílá informaci o stisku některých klíčových kláves do hostitelského procesoru kvůli obsluze menu na obrazovce. Přesunem obsluhy vstupních zařízení do hostitelského procesoru se zbavíme této komunikace a zároveň logice pro menu umožníme přístup ke všem vstupům.

- Budoucí rozšířitelnost ve formě např. USB joysticků či jiných vstupních zařízení bude realizovatelná zcela v rámci hostitelského procesoru.

Ve všech případech je ovšem nutné adaptovat Minimig tak, aby mohl přijímat informace o klávesnici a myši např. přes rozhraní SPI, nezávisle na tom, zda bude komunikovat přímo s USB řadičem na kitu Minerva a nebo s hostitelským procesorem. Podívejme se tedy jak projekt Minimig v současnosti komunikuje s jednotlivými vstupními zařízeními a jak lze dosáhnout navrhované alternativy.

4.5.1 Komunikace s myší

Projekt Minimig emuluje jednu myš přes rozhraní PS/2, ovšem Amiga 500 umožňuje připojení dvou myší. Každá myš má vlastní osmibitvá počítadla horizontální a vertikální pozice v registrech (`JOY0DAT` a `JOY1DAT`). Pro čtení tlačítek myší slouží registr `CIAAPRA` pro přístup k levým tlačítkům obou portů, registr `POT0DAT` pro přístup k pravému a prostřednímu tlačítku portu 0 a registr `POT1DAT` pro přístup k pravému a prostřednímu tlačítku portu 1 [21, s. 217-219].

V zásadě máme dvě možnosti, jak zařídit komunikaci OCS z projektu Minimig s myší připojenou k vhodnému USB řadiči. Můžeme emulovat rozhraní PS/2 a nebo můžeme zasáhnout hlouběji do zdrojového kódu projektu Minimig a komunikace přes rozhraní PS/2 se zcela zbavit. Jelikož je rozhraní myši poměrně triviální, tak je zřejmě druhá možnost jednodušší a také méně náročná na použité zdroje.

Projekt Minimig bohužel není v tomto ohledu příliš dobře dokumentován a tudíž nevíme, jak je tento hardware emulován. Naštěstí lze tuto informaci lehce dohledat ve zdrojových souborech. Pokud následujeme signály pro připojení myši, `msdat` a `msclk` na modulu nejvyšší úrovně, tj. `Minimig1`, tak dojdeme k modulu `userio`, v němž se dále nachází instance modulu `ps2mouse`, což je zjevně modul právě pro práci s PS/2 myší. Při pohledu na jeho rozhraní vidíme:

- Signály `clk` a `reset` pro řízení.
- Signály `ps2mdat` a `ps2mclk`, které jsou napojeny přímo na PS/2 port a tedy přímo na myš.
- Výstupní vektory `ycount` a `xcount`, které slouží jako výše zmíněná počítadla pohybu myši.
- Výstupní signály `_mleft`, `_mright` a `_mthird`, které slouží jako invertované hodnoty tlačítek myši.

Pro nahrazení tohoto modulu nám tedy stačí dodat vlastní hodnoty na signálech `ycount`, `xcount`, `_mleft`, `_mright` a `_mthird`, nejspíše tedy budeme tyto hodnoty získávat z vlastního SPI modulu.

4.5.2 Komunikace s klávesnicí

Situace s klávesnicí je podobná situaci s myší. Minimig emuluje klávesnici počítače A500 pomocí klávesnice připojené k dalšímu PS/2 rozhraní.

Stav klávesnice ovšem není v počítači A500 reprezentovaný stejně jednoduše, jako například stav myši či joysticků. V tomto případě řadič klávesnice aktivně řídí komunikaci

s jedním integrovaným obvodem 8520 pomocí jednoduchého sériového rozhraní postaveného pouze na dvou signálech [21, s. 245]:

- `SP` – datový signál;
- `CNT` – hodinový signál. Při náběžné hraně je obvodem 8520 zachycena logická úroveň na signálu `SP` a ta je vsunuta do vnitřního posuvného registru.

Při stisku či uvolnění klávesy posílá řadič klávesnice přes toto rozhraní jeden bajt, jehož nižších sedm bitů zpravidla identifikuje klávesu a jeho nejvyšší bit určuje, zda byla klávesa stisknuta či uvolněna. Následně je vyvoláno přerušení procesoru, který tak může přijatou informaci zpracovat.

V případě Minimigu je komunikace s klávesnicí, stejně jako v případě myši, realizovaná pomocí rozhraní PS/2. Modul `ciaa` poskytuje signály `kbddat` a `kbdclk`, které jsou připojeny přímo ke klávesnici. Tyto jsou pak uvnitř připojeny k modulu `ps2keyboard`, který se stará o řízení a dekódování PS/2 komunikace. Se zbytkem Minimigu pak komunikuje primárně pomocí následujících signálů:

- Výstupní vektor `keydat` – Emulace bajtu přijateého přes rozhraní klávesnice Amigy.
- Výstupní signál `keystrobe` – Signál indikující příjem kódu klávesy. Slouží k vyvolání přerušení procesoru.
- Výstupní signál `kbdrst` – Signál sloužící k resetu celého systému. Tento signál je generován v případě, že jsou stisknuty klávesy Ctrl, Alt a Delete současně.
- Vstupní signál `keyack` – Signál indikující zpracování stisku či uvolnění klávesy procesorem. Po přijetí klávěsy čeká modul `ps2keyboard` na aktivaci tohoto signálu a nebo vypršení časového intervalu před přijímáním dalších událostí.
- Vstupní signály `leda` a `ledb` – Tyto signály nahrazují LED ‘Power’ a ‘Drive’ Amigy LED umístěnými na připojené klávesnici.
- Výstupní vektor `osdctrl` – Signály pro řízení OSD Minimigu. Jedná se právě o signály přenášené do hostitelského procesoru přes rozhraní SPI v modulu `userio` pro řízení menu na obrazovce. Rozšíření Minimigu od Jakuba Bednarskiho poté přidává ještě dva další signály, z nich je jeden nepoužitý a druhý slouží pro řízení nově přidaného modulu `ActionReplay`.

Pokud tedy chceme nahradit chování modulu `ps2keyboard`, tak musíme primárně replikovat chování na signálech `keydat`, `keystrobe` a `keyack`. Signály `kbdrst`, `leda` a `ledb` pak nejsou kritické, jejich chování ovšem můžeme taktéž replikovat. Vektor `osdctrl` pak není nutný v případě, že komunikace s periferiemi bude probíhat přes hostitelský procesor. Pouze v případě, že založíme systém na Minimigu od Jakuba Bednarskiho, je vhodné replikovat bit řídící modul `ActionReplay`.

Pokud použijeme pro přenášení událostí klávesnice z hostitelského procesoru do Minimigu další SPI rozhraní, podobně jako v případě komunikace s myší, tak můžeme v nejjednodušším případě přenášet pouze hodnotu pro vektor `keydat` a po jejím příjetí můžeme replikovat chování na signálech `keystrobe` a `keyack`. Pokud ovšem bude žádoucí toto rozhraní replikovat zcela, tak bude nutné komunikaci rozšířit o možnost pravidelného získávání hodnot signálů `leda` a `ledb`, řízení signálu `kbdrst` a případně i řízení modulu `ActionReplay`.

4.5.3 Komunikace s joysticky

Počítač Amiga 500 disponuje dvěma konektory D-sub 9, které byly oba použitelné pro připojení myši nebo joysticku. Projekt Minimig taktéž disponuje těmito dvěma konektory, jsou ovšem použitelné pouze pro připojení joysticků. Myš může být k Minimigu připojena pouze jedna, a to přes rozhraní PS/2, jak je zmíněno výše. V Minimigu je pak implementováno automatické přepínání mezí myší či prvním joystickem na základě aktivity těchto zařízení.

Amiga Hardware Reference Manual definuje zapojení zmíněných konektorů takto:

- Pin 1 – Spínač pohybu vpřed.
- Pin 2 – Spínač pohybu vzad.
- Pin 3 – Spínač pohybu vlevo nebo primární akční tlačítko.
- Pin 4 – Spínač pohybu vpravo nebo sekundární akční tlačítko.
- Pin 5 – Potenciometr na horizontální ose.
- Pin 6 – Primární akční tlačítko.
- Pin 7 – Pětivoltové napájení joysticku.
- Pin 8 – Zemnění joysticku.
- Pin 9 – Sekundární akční tlačítko nebo potenciometr na vertikální ose.

Toto rozhraní je tedy uzpůsobeno pro připojení joysticků digitálních, kde směr pohybu páky je zachycen čtyřmi spínači (vpřed, vzad, vlevo a vpravo), a joysticků analogových, kde pohyb páky je zachycen dvěma potenciometry (pro vertikální a horizontální pohyb současně). Tyto potenciometry tvoří napěťový dělič, jehož výstupní napětí lze měřit pomocí AD převodníku vestavěného do integrovaného obvodu Denise.

U digitálních joysticků slouží jako akční tlačítka piny 6 a 9. U analogových joysticků slouží ke stejnému účelu piny 3 a 4.

Projekt Minimig nedisponuje žádnými AD převodníky použitými pro analogové joysticky a umožňuje tedy použití pouze digitálních joysticků.

O přístup k joystickům se, stejně jako v případě PS/2 myši, stará modul `userio`. Do tohoto modulu vstupuje přímo všech šest směrových a akčních signálů obou joysticků.

Pokud požadujeme nahrazení tohoto rozhraní za takové, které bude, podobně jako u myši a klávesnice, komunikovat s hostitelským procesorem, tak pouze potřebujeme do FPGA z hostitelského procesoru přenášet ve vhodný okamžik těchto dvanáct hodnot. V opačném případě můžeme toto rozhraní ponechat beze změny.

4.6 Hostitelský procesor

V projektu Minimig se mimo procesoru Freescale MC68SEC000 nachází i PIC18LF252, jehož účelem je programování FPGA, nahrávání firmware, čtení obrazů disket pro emulaci disketové mechaniky a řízení menu na obrazovce. Jeho činnost je tedy kritická k běhu systému a je nutné ji zachovat.

Nahrávání firmware a čtení obrazů disket je u Minimigu realizováno čtením paměťové karty se souborovým systémem FAT16 za použití knihovny FatFS (viz zdrojové soubory pro hostitelský procesor Minimigu). Komunikace s OCS je pak prováděna přes běžně rozhraní SPI [27]:

> The PIC communicates with the FPGA through an SPI interface. [...] In it's current form, the FPGA has 2 SPI "addresses". Address #0 is selected by the `fpga_sel0` signal and control the floppy emulator. `fpga_sel1` controls the on-screen-display. `fpga_sel2` is currently unused.

Základní požadavky na hostitelský procesor vyplývají z jeho funkce a jsou následující:

- možnost komunikace s OCS v FPGA, ideálně pomocí rozhraní SPI;
- přístup k paměťové kartě;
- komunikace s periferními vstupními zařízeními v případě, že je žádoucí implementovat komunikaci s těmito periferiemi dle části 4.5.

Na kitu Minerva může pozici hostitelského procesoru pochopitelně zastávat mikrořadič Freescale PK60N512VMD100, jelikož všechny tyto vlastnosti splňuje:

- Pro komunikaci s FPGA jsou k dispozici hned tři rozhraní SPI, čtyři asynchronní sériová rozhraní a dvě rozhraní I^2C.
- Paměťová karta je připojena přímo k MCU na rozhraní SDHC0.
- MCU může komunikovat s řadičem USB přes rozhraní SPI2 nebo UART4 a disponuje dostatečným množstvím vstupních pinů pro připojení joysticků.

Na kitu Pipistrello je situace zcela odlišná — kit žádným vhodným procesorem nedisponuje a je tudíž nutné jej implementovat v FPGA či připojit externě. Při implementaci v FPGA pochopitelně nenastává problém s komunikaci s OCS na stejném FPGA a komunikace s paměťovou kartou, která je na kitu Pipistrello připojena přímo k FPGA, je taktéž bezproblémová. Pro komunikaci s periferiemi jsou pak k dispozici všechna tři křídla a konektor PMOD. To umožňuje připojit periferie přímo k FPGA či použít externí USB řadič, podobně jako je tomu na kitu Minerva.

Tato konektivita je zřejmě dostatečná i pro připojení externího procesoru a to i v případě implementace USB řadiče v FPGA, ovšem použití externího řadiče USB se s použitím externího procesoru nijak nevylučuje. K tomuto účelu by bylo reálné použít kupříkladu i kit Minerva.

Kapitola 5

Implementovaná varianta

Na základě informací popsaných v předchozí kapitole a několika experimentů jsem pro implementaci emulátoru zvolil jako hlavní metodu implementace OCS i procesor umístěné v FPGA kitu Pipistrello, a to především z důvodu nedostatku prostředků v FPGA na kitu Minerva — výsledné řešení vyžaduje 2636 slices na FPGA, což je přibližně dvojnásobný počet, než je dostupné na FPGA kitu Minerva.

Jako operační paměť bude sloužit statická paměť ISSI IS61WV102416BLL-10TLI připojená přímo k FPGA. Roli hostitelského procesoru a tedy i komunikaci s periferními zařízeními a čtení paměťové karty pak bude zastávat kit Minerva, a to díky svému MCU Freescale PK60N512VMD100 a řadiči USB FTDI Vinculum-II. Data budou mezi hostitelským procesorem a FPGA předávána pomocí rozhraní SPI. Grafický a zvukový výstup bude poskytován přímo z kitu Pipistrello s obrazem na HDMI konektoru a zvukem na 3,5 mm konektoru.

Externí statická paměť bude umístěna na nové desce plošných spojů, která bude zároveň realizovat napájení kitu Pipistrello, komunikaci mezi oběma kity a připojení joysticků.

Samotné OCS je pak postaveno na Minimigu od Jakuba Bednarskiho, který obsahuje řadu opravených chyb, obsahuje lepší řízení komunikace s emulátorem disketové mechaniky a od původního Minimigu se příliš neodlišuje. To ho dělá jednoduše zabudovatelným do jiného systému, narozdíl např. od varianty Minimigu použité v projektu MIST.

Následující části se zabývají klíčovými komponentami v Minimigu i mimo něj. Zde prezentované informace jsou nutné pro dosažení stanovených cílů.

5.1 Změny v jádru projektu Minimig

Minimig disponuje poměrně jednoduchým vnějším rozhraním, jedná se v zásadě o několik rozhraní pro komunikaci s jednotlivými klíčovými komponentami systému, které nejsou či nemohou být umístěny v FPGA:

- Rozhraní pro připojení procesoru, které je přímo připojené k procesoru Freescale MC68SEC000. Jedinou výjimkou je resetovací signál, který je k procesoru připojen přes dva rezistory na signály $\overline{\text{RESET}}$ a $\overline{\text{HOLD}}$;
- rozhraní pro připojení paměti, které je taktéž připojeno přímo k paměťovým IO;
- SPI rozhraní připojené přímo k hostitelskému procesoru;
- vstupní PS/2 rozhraní pro myš a klávesnici;

- vstupní rozhraní pro joysticky;
- výstupní obrazové rozhraní, které je složené z několika signálů pro barvy a synchronizačních signálů. Signály s barvou jsou pak na desce připojeny k jednoduchému rezistorovému DA převodníku pro generování finálního výstupního signálu;
- digitální výstupní signály pro levý a pravý zvukový kanál;
- dériové rozhraní;
- různé systémové signály, jako je např. vnější krystalový oscilátor či resetovací tlačítko.

Většinu těchto rozhraní by bylo možné ponechat v jejich stávající podobě a komunikaci zařídit entitami mimo Minimig, které by adaptovali námi požadované rozhraní na rozhraní Minimigu. Např. u periferních zařízení by tak vznikla vrstva překládající USB komunikaci na komunikaci na rozhraní PS/2. Tento přístup je ovšem nepraktický a zvyšuje nároky na prostředky v FPGA. Lepším řešením je jistě modifikace jádra Minimigu tak, aby vyhovovalo potřebám implementované varianty.

Podívejme se tedy detailně na rozhraní, u kterých bude nutné provádět změny, a jak budou tyto změny realizovány.

5.1.1 Připojení procesoru

Procesor MC68SEC000 použitý v Minimigu komunikuje s datovou sběrnicí poměrně tradičním způsobem, a to pomocí třístavových budičů na pinech 45 až 55 a 57 až 61. Tyto signály jsou připojeny do Minimigu na piny na FPGA využívající taktéž třístavové budiče. Tato konfigurace umožňuje buzení sběrnice z Minimigu či FPGA a tudíž předávání dat z FPGA do Minimigu či naopak.

Pro účely připojení externího procesoru je tento přístup ideální, ovšem při použití emulovaného procesoru umístěného ve stejném FPGA narážíme na problém se syntetizovatelností těchto obousměrných portů. Řešením je použití odlišných vstupních a výstupních portů na relevantních komponentách.

Použitá komponenta emulující procesor Motorola MC68000 od Tobiase Gubenera, TG68, již takové rozhraní poskytuje pomocí následujících signálů:

- `data_in` – data vstupující do procesoru;
- `data_out` – data vystupující z procesoru;
- `drive_data` – signál indikující stav, kdy by běžný procesor MC68000 aktivoval svoje výstupní budiče a budil tedy datovou sběrnici.

Nyní tedy zbývá rozšířit rozhraní Minimigu podobným způsobem. Abychom toho mohli docílit, tak se nejprve podívejme, jak dochází k buzení současného třístavového výstupu Minimigu. Dokumentace těchto vlastností Minimigu je bohužel prakticky neexistující a musíme se tedy uchýlit k získávání informací ze zdrojového textu. Budeme-li tedy následovat signál datové sběrnice procesoru `cpudata` v modulu `Minimig1`, tak dojdeme k následujícím třem přiřazením v modulu `m68k_bridge`:

```
// dataout multiplexer and latch
always @(posedge clk)
ldataout<=data;
```

```
// generate t
assign t=(~r_w) | (_lds&_uds);

// data tristate buffers
assign data[15:0]=(t)?16'bz:ldatain[15:0];
```

Signály `ldataout` a `ldatain` reprezentují právě data z procesoru vystupující a data do procesoru vstupující. Dále vidíme, že signál `cpudata` (v tomto modulu pojmenovaný pouze `data`) je buzen pouze v případě, že signál R/$\overline{\text{W}}$ procesoru značí právě probíhající čtecí paměťový cyklus (což je běžný klidový stav) a zároveň je aktivní alespoň jeden ze signálů LDS a UDS.

Připojení emulovaného procesoru je tedy pouhou záležitostí propojení signálů `ldatain` a `ldataout` modulu `m68k_bridge` se signály `data_in` a `data_out` modulu TG68.

5.1.2 Připojení operační paměti

Jelikož tato varianta používá statickou paměť s rozhraním téměř identickým pamětem v Minimigu, tak jsou změny v tomto rozhraní minimální. Hlavní změna spočívá v adresaci paměti — v Minimigu je paměťový prostor realizovaný dvěma 1 MiB paměťovými IO a je rozdělen na čtyři části, každá o velikost 512 KiB. Tyto části jsou adresovány čtyřmi signály `aen1` až aen4, jejich význam je komentován ve zdrojovém souboru `Minimig1.v` takto:

- `aen1` – first 512Kbyte of chipram
- `aen2` – second 512Kbyte of chipram
- `aen3` – 512Kbyte of slow ram or Action Replay ROM
- `aen4` – 512Kbyte of kickstart rom (write enabled when boot asserted)

Modul `sram_bridge` pak na základě těchto signálů generuje nejvyšší bit adresové sběrnice pamětí a provádí výběr jednoho ze dvou paměťových IO. Tuto logiku je nutné mírně pozměnit:

- Je třeba rozšířit adresovou sběrnici o jeden další bit.
- Logiku určující hodnotu nejvyššího bitu adresové sběrnice (po rozšíření tedy druhého nejvyššího) je nutno ponechat.
- Je nutno doplnit logiku určující nyní nejvyšší bit adresové sběrnice a nahradit tak vybírání paměťových IO z Minimigu.

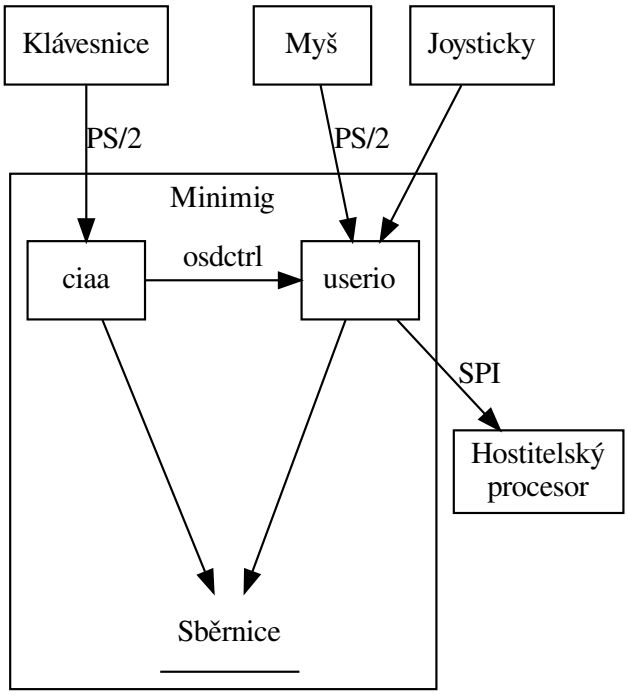
Tohoto cíle dosáhneme následující přiřazením v modulu `sram_bridge`:

```
assign ram_address = {aen4|aen3, aen4|aen2, address_in};
```

Signál `ram_address` zde reprezentuje výstupní dvacetibitovou adresu použitou přímo pro adresování paměti. Signál `address_in` je pak adresa v rámci jedné 512 KiB části celého adresového prostoru pamětí, tj. spodních 18 bitů.

5.1.3 Rozhraní pro periferní zařízení

Jak již bylo zmíněno v kapitole 4.5, o komunikaci se vstupními perfierními zařízeními se u Minimigu starají moduly `ciaa` a `userio`, ke kterým jsou tato zařízení přímo připojena. Tyto dva moduly jsou pak připojeny na sběrnici systému, kde s nimi může komunikovat hlavní procesor. Modul `ciaa` pak dodatečně zachytává i klávesy pro ovládání menu na obrazovce, které předává do modulu `userio`, který je poskytuje dále přes rozhraní SPI hostitelskému procesoru. Tato architektura je znázorněna na obrázku 5.1.



Obrázek 5.1: Diagram komunikace hostitelského procesoru a OCS Minimigu se vstupními periferními zařízeními.

Tato architektura ovšem pokulhává v případě, že chceme minimálně s klávesnicí a myší komunikovat pomocí rozhraní USB. Řízení komunikace na rozhraní USB je podstatně složitější než je tomu u rozhraní PS/2 a hodí se více pro programovatelný procesor než pro FPGA. Je tedy zjevné, že mezi vstupními zařízeními a OCS v FPGA chceme mít další stupeň, který přetransformuje USB komunikaci do jednodušší formy použitelné v OCS. Na kitu Minerva je tímto stupněm zcela jasně řadič USB Vinculum-II od FTDI. Tento obsahuje vestavěný programovatelný procesor (viz [12]), který může sloužit k libovolnému účelu a v našem případě tedy může řídit komunikaci s připojenými periferiemi a výsledky předávat do OCS.

Nyní tedy proveďme rozhodnutí, jak bude toto předávání probíhat. Na desce kitu Minerva k tomuto účelu můžeme použít následující metody:

- SPI komunikace na pinech 41, 42, 43 a 44 VNC2 – Tyto piny jsou vyvedeny přímo na rozšiřující konektor P1 a tudíž máme možnost je připojit přímo na některé z pinů FPGA na kitu Pipistrello. V rámci tohoto FPGA pak realizujeme podřízené SPI zařízení.
- SPI komunikace na pinech 31, 32, 33 a 34 jako zařízení podřízené hostitelskému procesoru. Hostitelský procesor pak může informace předávat do OCS pomocí již existujícího SPI rozhraní pro řízení emulované disketové mechaniky a menu na obrazovce.
- Stejný postup jako v předchozí variantě, ovšem za použití asynchronní sériové komunikace na pinech 35 a 36.

Jako přímá cesta se jeví varianta první, jelikož zcela obchází hostitelský procesor. Nevýhodou tohoto přístupu ovšem je, že v rámci OCS musíme stále zachytávat stisky kláves pro ovládání OCS a ty do hostitelského procesoru předávat. Z toho plynou omezené možnosti ovládání menu na obrazovce, jelikož do hostitelského procesoru předáváme informace o stisku pouze některých kláves. Pokud požadujeme flexibilnější ovládání menu na obrazovce, tak nejspíš musíme do hostitelského procesoru předávat informace o stisku všech kláves. Pokud chceme navíc mít možnost ovládat menu na obrazovce i myší, tak se situace dále komplikuje.

Další nevýhodou je nutnost vyhradit další čtyři signály na desce Pipistrello pro komunikaci s kitem Minerva. Pokud plánujeme využít externí procesor ekvivalentní Motorola MC68000, tak opět narazíme na nedostatek volných signálů.

Realizujme tedy raději variantu druhou nebo třetí, kde tento problém nenastává. Dodatečně získáme i absolutní kontrolu nad vstupem v rámci hostitelského procesoru a můžeme se zcela zbavit potřeby předávat informace o ovládání menu na obrazovce z OCS do hostitelského procesoru.

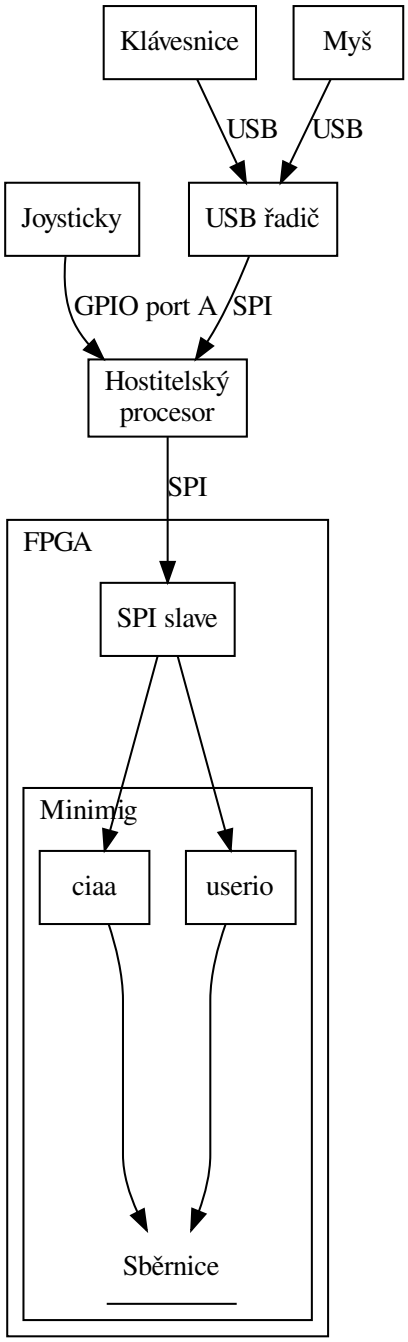
Nakonec se zaměřme ještě na komunikaci s joysticky. Jak již bylo zmíněno v kapitole 4.5.3, tak k počítači A500 je možno připojit dva joysticky, každý s šesti spínači. Tyto spínače jsou kabelem vedeny přímo do počítače, není tedy použito žádného sériového rozhraní či podobného způsobu komunikace. Pro komunikaci s oběma joysticky je tedy nutné použít dvanáct signálových vodičů. Ty můžeme v nejjednodušším případě připojit přímo k OCS nebo hostitelskému procesoru. V případě nedostatku volných signálů můžeme použít posuvný registr s paralelním vstupem (např. některou z běžně dostupných variant integrovaného obvodu 74589) pro serializaci stavu spínačů joysticků.

Pokud chceme joysticky připojit přímo k hostitelskému procesoru, abychom získali stejnou kontrolu nad vstupem jako v případě myši a klávesnice, tak můžeme využít dvanáct ze třinácti dostupných obecných vstupně-výstupních pinů procesoru vyvedených na rozšiřující konektor P1 na kitu Minerva. Jedná se o bity 6 až 11 a 24 až 29 včetně portu A a bit 28 portu E. Pro jednoduchost zvolme pouze port A.

Diagram finální architektury je na obrázku 5.2. Zbytek této části se bude zabývat zakomponováním této architektury do OCS Minimigu a detaily komunikace periferních zařízení, hostitelského procesoru a OCS.

Komunikace s klávesnicí

V kapitole 4.5.2 bylo ukázáno rozhraní klávesnice použité v počítači Amiga 500 a jak je komunikace s klávesnicí realizována v rámci Minimigu. Nyní se podíváme na způsob, jak modul `ps2keyboard` zakomponujeme do zvolené architektury.



Obrázek 5.2: Diagram zvolené komunikace vstupních periferních zařízení, hostitelského procesoru a OCS v Minimigu.

Víme, že řadič klávesnice posílá při vzniku události jednobajtový identifikátor události, který je v Minimigu dostupný na výstupním vektoru `keydat` v případě, že je aktivní signál `keystrobe`.

Jelikož budeme stisky kláves přijímat z hostitelského procesoru přes rozhraní SPI, tak je nasnadě jednoduché rozhraní, které po přijetí jednoho bajtu přes SPI vystaví tento bajt na vektoru `keydat` a aktivuje signál `keystrobe`.

Modul `ps2keyboard` ovšem zároveň ovládá LED indikátory na klávesnici, jejich stav získává ze signálů `leda` a `ledb`. Hostitelský procesor tedy potřebuje mít možnost získat i stav těchto signálů, a to nezávisle na stisku či uvolnění kláves.

Z tohoto důvodu rozdělme řízení klávesnice na dvě oddělená podřízená SPI rozhraní, jedno pro příjem událostí klávesnice a druhé pro odesílání stavu LED indikátorů na klávesnici. Protokoly pro komunikaci s těmito zařízeními mohou být zcela triviální:

- Zařízení pro příjem událostí přijímá jeden bajt označující událost.
- Zařízení pro odesílání stavu indikátorů odesílá jeden bajt, kde spodní dva bity určují stav indikátorů dle signálů `leda` a `ledb`.

Komunikace s myší

V 4.5.1 jsme se zabývali tím, jak byla myš počítače Amiga 500 k počítači připojena, jak byla ze strany softwaru přítomna a jak je situace s myší řešena v rámci projektu Minimig. Jako jednoduchý návrh nahrazení PS/2 myši zde bylo zmíněno nahrazení hodnot signálů s počítadly pozice a stavem tlačítek (tj. vektory `xcount`, `ycount` a signály `_mleft`, `_mright` a `_mthird` modulu `ps2mouse`) vlastními signály, jejichž hodnotu by bylo možné měnit pomocí SPI rozhraní z hostitelského počítače.

Tento přístup se zdá zcela validní a vskutku je zcela použitelný pro tlačítka myši. Situace s pozicí myši je ovšem jiná. Amiga 500 totiž disponuje jedním registrem, který tento jednoduchý pohled na věc mírně komplikuje. Jedná se o registr nazvaný `JOYTEST`, který umožňuje zápis do horních šesti bitů horizontální i vertikální pozice obou myší zároveň. Hardware Reference Manual pro počítače Amiga se o tomto registru zmiňuje velice stroze [21].

Funkce tohoto registru není v původním Minimigu nijak emulována, ovšem verze Minimigu od Jakuba Bednarskiho tento nedostatek napravuje. V modulu `userio` najdeme následující přiřazení:

```
//JB: some trainers writes to JOYTEST register to reset current
//mouse counter
assign test_load = regaddress[8:1]==JOYTEST[8:1] ? 1 : 0;
assign test_data = datain[15:0];
```

V modulu `ps2mouse` jsou pak signály `test_load` a `test_data` použity pro nahrazení horních šesti bitů registrů `xcount` a `ycount`.

Zvolené řešení pro komunikaci s myší by tedy nemělo zcela nahrazovat registry `xcount` a `ycount` vlastními absolutními hodnotami a mělo by je zachovat, aby bylo možné je měnit zápisem do registru `JOYTEST`. Řešením je tudíž posílat z hostitelského procesoru do OCS relativní změnu vertikální i horizontální pozice myši. OCS pak může tyto hodnoty při jejich přijetí jednorázově přičíst k hodnotám registrů `xcount` a `ycount`.

Zaveďme tedy tříbajtovou zprávu posílanou přes rozhraní SPI z hostitelského počítače ve formě dle tabulky 5.1.

Číslo bajtu	Bit	Význam
0	0 - 7	Horizontální posun myši
1	0 - 7	Vertikální pohyb myši
2	0	Stav levého tlačítka myši
	1	Stav pravého tlačítka myši
	2	Stav prostředního tlačítka myši
	3 - 7	Bez významu

Tabulka 5.1: Popis významu jednotlivých bajtů protokolu pro předávání informací o myši.

Komunikace s joysticky

Jak již bylo zmíněno v úvodu této části, tak joysticky připojíme přímo k portu A hostitelského procesoru. Dle [1] jsou joysticky v zásadě tvořené pouze několika spínači, které spínají jednotlivé datové signály k nulovému napěťovému potenciálu. Z toho plyne, že v hostitelském procesoru stačí monitorovat stav portu A a v případě změny informovat OCS o dané změně.

Jelikož se stav joysticky skládá pouze z šestice bitů, tak bude protokol pro tuto komunikaci opět poměrně triviální. Aby byla velikost zasílané zprávy co nejmenší, tak zvolme protokol, kde je přenášen pouze jeden bajt. V tomto bajtu bude celý stav jednoho joysticku a jednobitový identifikátor joysticku pro odlišení mezi dvěma různými joysticky. Při příjmu této zprávy podřízeným SPI zařízením v FPGA dojde k aktualizaci hodnot pro joystick identifikovaný ve zprávě.

Význam jednotlivých bitů ve zprávě je popsán v tabulce 5.2.

Bit	Význam
0	Stav spínače pohybu vpravo
1	Stav spínače pohybu vlevo
2	Stav spínače pohybu dolů
3	Stav spínače pohybu nahoru
4	Primární akční tlačítko
5	Sekundární akční tlačítko
6	Identifikátor joysticku
7	Bez významu

Tabulka 5.2: Popis významu jednotlivých bitů protokolu pro předávání informací o joystic-ích.

Nyní se ještě vraťme k elektrickému zapojení. Ačkoliv typické zapojení joysticků používá pouze spínače vůči nulovému napěťovému potenciálu, tak je jistě možné, že jiné modely joysticků (např. joysticky s tzv. turbo módem, který uživateli napomáhá s rychlými opakovanými stisky akčních tlačítek) obsahují dodatečnou řídící elektroniku, která může být napájena z pinu 7 konektorů pro joysticky, který poskytuje pětivoltové napájení.

V případě, že by takový joystick silně budil svoje výstupy, by se mohlo oněch pět voltů objevit i na výstupech tohoto joysticku a tedy i na vstupech hostitelského procesoru. Procesory řady K60 ovšem pracují s napětím až 5,5 V (viz [20, s. 12]), tuto situaci tudíž není nutné nijak dodatečně řešit.

Grafický výstup

Grafický výstup Minimigu je ve formě použitelné pro běžné VGA rozhraní, ovšem pro získání analogových signálu pro ono rozhraní je použit jednoduchý DA převodník složený z rezistorů, který je pochopitelné umístěn mimo FPGA.

Barevné výstupy Minimigu jsou tedy čistě digitální, což je ideální pro rozhraní DVI či HDMI. Je ovšem nutné dodržet minimální hodinovou frekvenci příchozích pixelů — tu stanovuje [10] jako 25 MHz. Pro dosažení této frekvence obsahuje Minimig tzv. scandoubler implementovaný v modulu `Amber`. `Amber` je komponenta, která duplikuje každý řádek výstupního obrazu a poskytuje tak vyšší vertikální rozlišení, což zvyšuje frekvenci příchozích pixelů, v případě Minimigu na 28,375 16 MHz, což je pro pro účely DVI či HDMI rozhraní dostatečné.

Zdvojování řádků pomocí komponenty `Amber` je nutné zapnout vnějším řídícím signálem `_15khz`.

Samotné rozhraní DVI či HDMI pak bude řízeno mimo Minimig na základě jeho grafického výstupu.

5.1.4 Hodinové signály

Hlavní zdroj hodinového signálu v projektu Minimig je krystalový oscilátor o frekvenci 4,433 619 MHz. Od tohoto hodinového signálu jsou pak odvozeny tři další, které jsou již použity přímo pro běh Minimigu:

- `clk` – Nejrozšířenější hodinový signál o frekvenci 7,093 79 MHz, použitý pro běh hlavního procesoru Freescale MC68SEC000 a většiny OCS.
- `qclk` či `clk90` – Hodinový signál o stejné frekvenci jako `clk`, ovšem oproti němu je o 90° posunutý. Tento signál je použit pouze pro časování mostů pro připojení hlavního procesoru a operační paměti. `clk28m` – Hodinový signál o frekvenci čtyřikrát vyšší, než signál `clk`, tj. 28,375 16 MHz. Tento signál je použit pro výstupní scandoubler v modulu `Amber` a pro vykreslování spritů v modulu `Agnus`.

Signál `clk28m` je generován pomocí PLL z hodinového signálu z výše zmíněného krystalového oscilátoru v modulu `clock_generator`. Hodinové signály `clk` a `qclk` jsou pak generovány z hodinového signálu `clk28m` pomocí sekvenční logiky.

Frekvence těchto hodinových signálů je nutné co nejlépe dodržet, aby bylo splněno časování grafického výstupu, jemuž je běh celého systému podřízen. Pozměňme tedy rozhraní Minimigu tak, že jako hlavní hodinový signál bude sloužit právě `clk28m`. Pro generování ostatních hodinových signálů ponechme sekvenční logiku v modulu `clock_generator`.

Změna tedy spočívá v odstranění použité PLL a vyvedení signálu `clk28m` na rozhraní entity Minimigu ve formě vstupu. O generování tohoto hodinového signálu se pak postarají komponenty mimo Minimig.

5.1.5 Shrnutí

Tato kapitola pojednávala o změnách v Minimigu, které je nutno provést pro dosažení stanovených cílů emulátoru. Změny jsou značné především v komunikaci s myší a klávesnicí, kde je nutné nahradit existující komunikaci přes PS/2 za odlišnou metodu komunikace s hostitelským procesorem přes rozhraní SPI. Tento pak bude zařizovat komunikaci s periferiemi přes rozhraní USB.

Drobných změn se dočkalo i rozhraní pro připojení paměti, kde je nutné pozměnit adresování, a rozhraní pro připojení procesoru, kde je nutné nahradit jeden V/V datový signál za dva oddělené, jeden vstupní a jeden výstupní.

Dodatečně byl přesunut hodinový signál `clk28m` mimo Minimig, aby byla resetovací logika systému, která je založená i na běhu PLL generující tento hodinový signál, umístěna mimo Minimig.

5.2 Komponenty mimo Minimig

V předchozí kapitole byly zmíněny změny v jádře Minimigu, které slouží k dosažení celkových cílů implementovaného emulátoru. Některé tyto změny ovšem potřebují dodatečnou komponenty mimo Minimig, aby byl emulátor jako celek funkční a integrovatelný s okolím. Těmito komponentami se bude zabývat právě tato kapitola.

5.2.1 SPI rozhraní

Minimig disponuje pro komunikaci mezi OCS a hostitelským procesorem jedním SPI rozhraním se třemi adresovacími signály pro výběr podřízeného zařízení. Jedním z těchto podřízených zařízení je modul `Paula`, který se, mimo jiné, stará o emulaci disketové mechaniky. Druhým podřízeným zařízením je modul `userio`, který má na starosti řízení menu na obrazovce. Třetí výběrový signál není využit a je ponechán pro budoucí rozšíření.

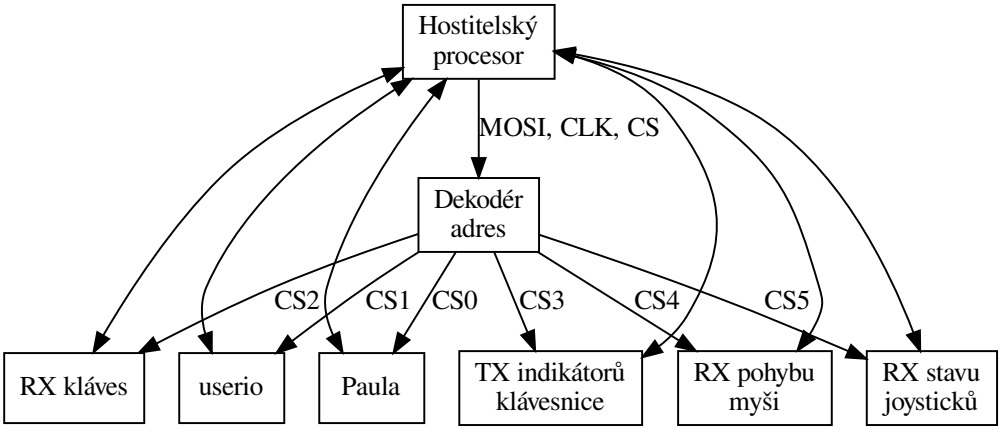
Tento přístup bohužel není reálně přenositelný do situace, kdy mezi hostitelským procesorem a FPGA s OCS existuje několik dalších komunikačních kanálů. Pokud tedy připočteme k oněm dvěma použitým zařízením v Minimigu další čtyři zařízení popsaná v části 5.1.3, tak dojdeme k celkovému počtu šesti podřazených zařízení. Adresovat takové množství podřazených zařízení z hostitelského procesoru, tedy MCU rodiny Kinetis K60 na kitu Minerva, se ukazuje jako náročné a to hned z několika důvodů:

- Na kitu Pipistrello po připojení externího procesoru MC68SEC000 zůstane pouze sedm volných signálů. To je dostatečné pouze pro tři adresovací signály.
- MCU MK60DN512VMD10 na kitu Minerva disponuje třemi SPI rozhraními [20], z nichž pouze jedno (zvané SPI1) je na kitu Minerva dostupné na rozšiřujícím konektoru J1. Rozhraní SPI1 poskytuje pouze dva adresní signály [19, s. 155], přičemž pouz jeden z nich, `SPI_PCS1`, je dostupný na konektoru P1. Signál `SPI_PCS0` je využit pro komunikaci s řadičem USB FTDI Vinculum-II.
- Použití obecných vstupně-výstupních pinů na MCU pro další adresování je taktéž nepraktické, protože po připojení joysticků způsobem popsaným v části 5.1.3 zůstane na rozšiřujícím konektoru volný pouze bit 28 portu E.

Poměrně jednoduchým způsobem řešení této situace je přenášení adresy cílového zařízení v rámci SPI komunikace samotné. V takovém případě předřadíme před veškerá cílová zařízení dekodér adres, jehož smyslem bude zachycení stanoveného počtu počátečních bitů transakce a na základě jejich hodnoty následná aktivace příslušného cílového zařízení. Koncepčně je takový systém znázorněn na obrázku 5.3.

Jako počet bitů zachycených dekodérem stanovme osm, aby nebylo nutné v hostitelském procesoru během transakce měnit velikost přenášené jednotky. Těchto osm bitů je následně možné použít přímo pro řízení jednotlivých adresových signálů uvnitř FPGA a adresovat

tak až osm zařízení. Alternativně můžeme všech osm bitů považovat za osmibitový číselný identifikátor cílového zařízení a adresovat tak až 256 zařízení. To ovšem není nutné.



Obrázek 5.3: Adresace cílových zařízení v rámci komunikace mezi hostitelským procesorem a FPGA s OCS.

Pokud bude hostitelský procesor dodržovat tento protokol a bude na začátku každé transakce zasílat adresní bajt, tak bude pro cílová zařízení tato metoda adresování zcela transparentní. Díky tomuto mohou být využity komponenty provádějící SPI komunikaci, které již v Minimigu existují, a to bez nutnosti provádět v nich nějaké změny.

Díky této metodě adresování je také možné realizovat situace podobné komunikaci s klávesnicí popsané v části 5.1.3, kde je komunikace rozdělena na dvě zařízení. Běžným způsobem řešení takových situací by bylo např. přidání bajtu s příkazem do samotného komunikačního protokolu daného zařízení.

5.2.2 Grafický výstup

Kit Pipistrello disponuje HDMI konektorem, který je připojen přímo k FPGA a pro jeho řízení tedy není k dispozici žádný dodatečný hardware, jako je tomu například na kitu Minerva.

Pro jednoduchost použijeme tento výstup dle specifikace rozhraní DVI, tj. [10]. Pro dosažení tohoto cíle lze využít modulu `DVI` poskytnutého uživatelem magnus na fórech kitu Pipistrello[1].

Použití tohoto modulu je triviální, v zásadě stačí dodat pouze barevnou informaci a synchronizační signály, které získáme přímo z VGA rozhraní Minimigu, a hodinové signály pro běh. Modul se pak postará o korektní kódování těchto dat a samotný přenos přes rozhraní DVI.

Jedinou komplikací je signál `vde` (tzv. video data enable), jehož aktivní stav určuje, že barevné informace na vstupu modulu reprezentují pixely, které je nutné prezentovat uživateli na obrazovce, a tudíž se nejedná o pixely mimo viditelnou oblast obrazu. Rozhraní DVI

[1] Viz http://saanlima.com/forum/viewtopic.php?f=12&t=1249.

využívá tohoto signálu pro přenos speciálních rezervovaných kódů na modrém barevném kanále, které v sobě kódují vertikální a horizontální synchronizaci [10].

Informace použitelné pro odvození signálu `vde` jsou k dispozici v rámci Minimigu, ovšem po zapojení zdvojování řádků v modulu `Amber` dojde k jejich ztrátě. Signál `vde` tedy musíme odvodit z vertikální a horizontální synchronizace. Hlavní myšlenka je následující:

- Každý snímek je uvozen poklesem signálu vertikální synchronizace. Pokud zachytíme tento okamžik, tak byl nalezen začátek snímku a můžeme začít počítat řádky snímku, nastavíme tedy čítač řádků na nulu.
- Každý řádek snímku je uvozen poklesem signálu horizontální synchronizace. Tuto událost použijeme pro inkrementaci čítače řádků. Zároveň ovšem potřebujeme počítat jednotlivé pixely na řádku, pokles horizontální synchronizace tedy použijeme i pro nastavení čítače pixelů na nulu.
- Výstupní pixely Minimigu jsou při zapnutém zdvojování řádků v modulu `Amber` vystavovány na obrazové výstupy na základě hodinového signálu `clk28m`, tj. 28,375 16 MHz (viz kapitola 5.1.3). Při každém taktu hodinového signálu `clk28m` tedy zvýšíme čítač pixelů.
- Díky počítadlům řádků a pixelů známe přesnou pozici na obrazovce a jsme tedy schopni určit, zda se nacházíme ve viditelné oblasti obrazu.

Takto získaný signál `vde` tedy přivedeme do modulu `DVI`. Výstup tohoto modulu pak vyvedeme na HDMI konektor.

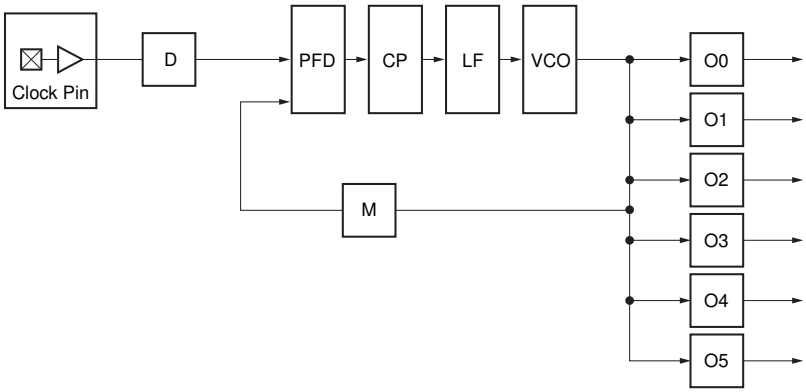
5.2.3 Hodinové signály

Na kitu Pipistrello je jako zdroj hlavního hodinového signálu použit krystal o frekvenci 50 MHz. Pro dosažení požadovaného hodinového signálu popsaného v kapitole 5.1.4 je tedy nutné použít PLL s vhodně zvolenými koeficienty násobení a dělení. PLL má následující relevantní vlastnosti [32, s. 104][31, s. 57]:

- Parametr `DIVCLK_DIVIDE` – Dělení vstupní frekvence od 1 do 52.
- Parametr `CLKFBOUT_MULT` – Násobení frekvence vnitřního oscilátoru od 1 do 64.
- Parametr `CLKOUT[0:5]_DIVIDE` – Dělení frekvence jednotlivých výstupu od 1 do 128.
- Frekvence vstupního hodinového signálu musí být po vydělení dle parametru `DIVCLK_DIVIDE` v rozsahu 19 MHz až 540 MHz.
- Frekvence oscilátoru PLL musí být v rozsahu 400 MHz až 1080 MHz.

FPGA XC6SLX45, dostupné na kitu Pipistrello disponuje čtyřmi PLL jednotkami [30, s. 2]. Musíme ovšem počítat s potenciálním využitím jedné z nich pro DVI rozhraní popsané v kapitole 5.2.2. Nejprve ovšem zvažme řešení za použití jedné PLL jednotky. Pouze v případě, že by nejlepší nalezené řešení používající pouze jednu jednotku PLL bylo nedostatečné, použijeme jednotek více.

Zjednodušené schéma PLL je na obrázku 5.4. Konfigurovatelné prvky této architektury jsou následující:



Obrázek 5.4: Schéma PLL v zařízeních Xilinx Spartan-6.

- Dělička vstupního hodinového signálu D konfigurovatelná parametrem `DIVCLK_DIVIDE`.
- Dělička zpětnovazebního signálu M konfigurovatelná parametrem `CLKFBOUT_MULT`. Právě dělením frekvence zpětné vazby dochází ke zvýšení operační frekvence oscilátoru PLL vůči vstupnímu hodinovému signálu.
- Děličky jednotlivých výstupů $O0$ až $O5$ konfigurovatelné parametry `CLKOUT0_DIVIDE` až `CLKOUT5_DIVIDE`.

Právě výběrem parametrů pro tyto prvky se nyní budeme zabývat.

Pro vygenerování použitelných konfigurací PLL byl použit jednoduchý program, jehož zdrojový soubor je na přiloženém CD, viz A. Konfigurace, jejichž absolutní chyba oproti požadované frekvenci je menší než 0,5 %, jsou v tabulce 5.3. Sloupec f_O zde vyjadřuje dosaženou výstupní frekvenci, sloupce f_D a f_{VCO} pak frekvenci vstupního signálu po dělení a frekvenci oscilátoru.

Varianta	D	M	$O0$	f_D	f_{VCO}	f_O	Chyba
1	1	13	23	50.0	650.0	28.2609	-0.4028
2	2	26	23	25.0	650.0	28.2609	-0.4028
3	2	43	38	25.0	1075.0	28.2895	-0.3020
4	1	17	30	50.0	850.0	28.3333	-0.1474
5	2	17	15	25.0	425.0	28.3333	-0.1474
6	2	34	30	25.0	850.0	28.3333	-0.1474
7	1	21	37	50.0	1050.0	28.3784	0.0113
8	2	42	37	25.0	1050.0	28.3784	0.0113
9	2	25	22	25.0	625.0	28.4091	0.1196
10	2	33	29	25.0	825.0	28.4483	0.2577
11	2	41	36	25.0	1025.0	28.4722	0.3421

Tabulka 5.3: Několik možných variant konfigurace PLL pro získání požadovaného hodinového signálu o frekvenci 28,375 16 MHz.

Jako vhodné varianty se pochopitelně jeví varianty 7 a 8. Tyto varianty se mezi sebou liší pouze frekvencí poděleného vstupního signálu. Dvakrát větší dělící koeficient D je pak kompenzován dvakrát větším násobícím koeficientem M. Jelikož je frekvence poděleného

vstupního signálu varianty 8 bližší spodní hranici povolených hodnot, tak zvolíme raději variantu 7.

Takto generovaný hodinový signál pak použijeme přímo pro běh Minimigu a ostatních komponent, které vyžadují synchronizaci s hodinovým signálem Minimigu, např. grafický výstup systému.

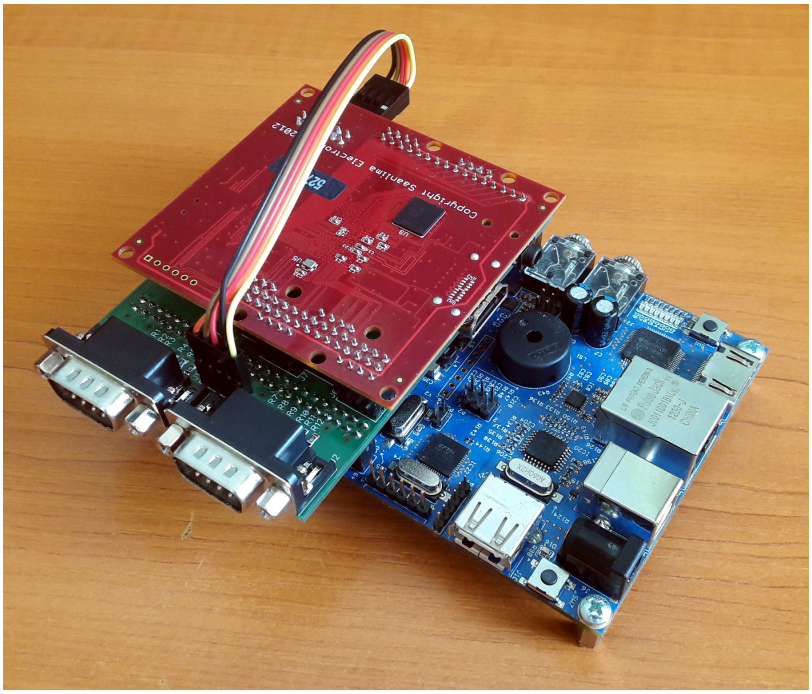
5.2.4 Propojovací deska plošných spojů

Pro propojení kitů Minerva a Pipistrello, připojení externí paměti a připojení joysticků jsem navrhl desku plošných spojů. Deska na obou stranách disponuje konektory pro zasazení do obou kitů a na horní straně i dvěma konektory typu D-sub 9 pro připojení joysticků. Deska je uzpůsobena pro osazení pamětí IS61WV102416BLL-10TLI.

Deska dodatečně umožňuje i osazení procesorem Freecale MC68SEC000. V tomto případě je ovšem nutné desku neosazovat pamětí, jelikož kvůli omezenému počtu dostupných pinů na desce Pipistrello sdílí tyto komponenty na desce většinu signálů.

Kapitola 6

Zhodnocení řešení



Obrázek 6.1: Fotografie zhotoveného řešení

Fotografie konečného řešení je na obrázku 6.1. Zde lze vidět vespod kit Minerva, nad ním propojovací desku plošných spojů a nad ní kit Pipistrello.

Důležitá rozhraní pro uživatele jsou následující:

- USB konektoru typu A na kitu Minerva pro připojení myši a klávesnice přes USB hub,
- HDMI konektor na kitu Pipistrello pro připojení zobrazovacího zařízení,
- 3,5 mm konektor pro připojení sluchátek či reproduktorů,
- konektory D-sub 9 pro připojení joysticků,
- slot pro paměťovou kartu na kitu Minerva,

- napájecí konektor či USB konektor typu B na kitu Minerva pro napájení.

Po spuštění dojde k naprogramování FPGA z flash paměti na kitu Pipistrello a následné spuštění bootloaderu Minimigu. Ten si od hostitelského procesoru na kitu Minerva vyžádá Kickstart. Ten mu je poskytnut ze souboru s názvem `KICKSTRT.ROM` z paměťové karty a o postupu této operace je uživatel informován na obrazovce. Po přenosu Kickstartu dojde k restartu OCS, které by se následně mělo spustit stejně, jako kdyby se jednalo o počítač Amiga 500. To se ovšem nestane a emulátor tedy bohužel nesplňuje svůj hlavní cíl.

Příčinou tohoto problému je nejspíše špatný reset hlavního procesoru či absence správných instrukcí na datové sběrnici pro běh procesoru. První zmíněný problém by mohl být způsoben špatnou logikou v modulu `syscontrol` v Minimigu, který se, mimo jiné, stará o řízení resetování.

Druhý zmíněný problém může mít podstatně více příčin. Po spuštění Kickstartu by mělo dojít k nahrání operačního systému z diskety, což v případě emulátoru vyžaduje komunikaci s hostitelským procesorem, který by měl poskytnout obsah diskety. Tak se ovšem nestane, emulace disketové mechaniky setrvává v klidovém stavu. Můžeme se tedy domnívat, že buď vůbec nedojde ke spuštění Kickstartu a nebo Kickstart narazí na nějaký problém ještě před začátkem komunikace s disketovou mechanikou.

Nespuštění Kickstartu by mohlo být způsobeno např. špatným mapováním paměti v modulu `Gary`. Experimentálně jsem zjistil, že po resetu systému po nahrání Kickstartu obsahuje datová sběrnice pouze nulové hodnoty, což by mohlo naznačovat, že datová sběrnice procesoru není buzena žádnou komponentou a nebo je buzena komponentou, která která na sběrnici vystavuje pouze nulové hodnoty. To by mohlo nastat např. v případě, kdy při přenosu obrazu Kickstartu bootloaderem dochází k chybné operaci DMA a do paměti jsou zapsány pouze nulové hodnoty a nebo do ní není zapsáno nic a její výchozí stav je složen pouze z nulových hodnot.

Alternativně je přípustná i špatná operace externí paměti, ovšem bootloader, který paměť využívá, funguje zdánlivě bez problému.

I přes tyto problémy ukazuje navržené řešení praktické výhody a nevýhody kitů Minerva a Pipistrello pro podobný druh úloh a může poskytnout inspiraci pro budoucí vývoj v oblasti podobných počítačových systémů na těchto platformách.

Kapitola 7

Závěr

Současná situace s hardwarovou emulací počítače Amiga 500 je poměrně slabá — pro laika je k dispozici projekt MIST, což je obecný systém pro emulaci několika počítačů, herních konzolí a videoherních automatů. Projekt Minimig je pak, hlavně z důvodu jeho nedostupnosti pro širokou veřejnost, určen spíše pro pokročilé, kteří jsou schopni si jej sami sestavit.

Tato práce se zabývala možností implementace počítače Amiga 500 na platformách Minerva a Pipistrello. Kit Minerva bohužel poskytuje FPGA nedostatečné pro implementaci čipsetu počítače Amiga 500, procesoru Motorola MC68000 a další podpůrné logiky dohromady. Tento problém je částečně vyvážen přítomností moderních rozhraní jako jsou DVI a USB pro připojení periferních zařízení.

FPGA kitu Pipistrello je, na rozdíl od Minervy, dostatečně schopné. Kit Pipistrello ale nedisponuje jednoduchým způsobem připojení vstupních periferií a implementace hostitelského procesoru a neumožňuje dosáhnout dostatečně nízké latence paměti pro korektní běh systému.

Finální řešení tedy poskytuje koncepci emulátoru založeného na spojení obou kitů dohromady ve spolupráci s externí statickou pamětí — kit Pipistrello poskytuje dostatečné prostředky na FPGA, kit Minerva zastává roli hostitelského procesoru a externí paměť IS61WV102416BLL-10TLI poskytuje dostatečně nízkou latenci paměťového podsystému. Oba kity jsou propojeny dodatečnou deskou plošných spojů, která zároveň poskytuje zmíněnou paměť a konektory pro připojení joysticků.

Dané řešení jistě není ideální — obě použité vývojové platformy dohromady poskytují značné množství dalších, nevyužitých prostředků. Zlepšením této situace by bylo např. přesunutí FPGA z kitu Pipistrello na zmíněnou propojovací desku plošných spojů, a to včetně napájení a metody programování. Taková změna by měla za následek odstranění závislosti na kitu Pipistrello a tedy několika nepotřebných komponentách.

Dalším možným vylepšením by mohla být podpora USB joysticků, jelikož pozměněná architektura systému oproti projektu Minimig dává hostitelskému procesoru větší kontrolu nad vstupními zařízeními.

Literatura

[1] *Amiga Joysticks [online]*. 2008-12-11 [cit. 2017-05-11].
URL http://wiki.classicamiga.com/Amiga_Joysticks

[2] *Board13 · mist-devel/mist-board Wiki [online]*. [cit. 2016-01-10].
URL https://github.com/mist-devel/mist-board/wiki/Board13

[3] *CoreDocAmiga · mist-devel/mist-board Wiki [online]*. [cit. 2016-01-10].
URL https://github.com/mist-devel/mist-board/wiki/CoreDocAmiga

[4] *Home · mist-devel/mist-board Wiki [online]*. [cit. 2016-01-10].
URL https://github.com/mist-devel/mist-board/wiki

[5] *mist-board - Firmware and cores for the MIST FPGA board [online]*. [cit. 2016-01-10].
URL https://code.google.com/p/mist-board/

[6] *mist-devel/mist-board [online]*. [cit. 2016-01-10].
URL https://github.com/mist-devel/mist-board

[7] *mist-devel/mist-firmware [online]*. [cit. 2016-01-10].
URL https://github.com/mist-devel/mist-firmware

[8] *Contributions · mist-devel/mist-board Wiki [online]*. [cit. 2017-05-20].
URL https://github.com/mist-devel/mist-board/wiki/ContriButions

[9] Commodore: *Commodore Amiga A500/A200 Technical Reference Manual*. 1987.

[10] Digital Display Working Group: *Digital Visual Interface*. 1999-05-02, revision 1.0.

[11] Freescale Semiconductor/NXP Semiconductors: *M68000 - 8-/16-/32-Bit Microprocessors User's Manual [online]*. 1993 [cit. 2017-05-23], ninth Edition.
URL http://www.nxp.com/assets/documents/data/en/reference-manuals/MC68000UM.pdf

[12] FTDI Chip: *Vinculum-II Embedded Dual USB Host Controller IC [online]*. [cit. 2017-05-23], version - 1.7.
URL http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_Vinculum-II.pdf

[13] Fujitsu: *MB81C4256-70/-80/-10/-12 CMOS 1,048,576 Bit Fast Page Mode Dynamic RAM*. 1990-03, edition 2.2.

[14] Integrated Silicon Solution: *IS61WV102416ALL IS61WV102416BLL IS64WV102416BLL 1M × 16 High-Speed Asynchronous CMOS Static RAM with 3.3V Supply*. 2014-06-02, rev. G.

[15] Integrated Silicon Solution: *IS62/65WV102416DALL IS62/65WV102416DBLL 1M ×16 Low Voltage, Ultra Low Power & Low Power CMOS Static RAM*. 2014-12-12 [cit. 2017-05-23], rev. A.
URL http://www.issi.com/WW/pdf/61WV102416ALL.pdf

[16] ISSI: *IS43/46DR86400B, IS43/46DR16320B 512Mb (×8, ×16) DDR2 SDRAM [online]*. 2012-08-01 [cit. 2017-05-23], rev. I.
URL http://www.issi.com/WW/pdf/43-46DR86400B-16320B.pdf

[17] Micron Technology: *512Mb: x16, x32 Mobile LPDDR SDRAM [online]*. 2014-01 [cit. 2017-05-23], rev. I.
URL https://www.micron.com/~/media/documents/products/data-sheet/dram/mobile-dram/low-power-dram/lpddr/60-series/t67m_512mb_mobile_lpddr_sdram.pdf

[18] NEC Electronics: *μPD41256 262,144 × 1-Bit Dynamic NMOS RAM*.

[19] NXP Semiconductors: *K60 Sub-Family Reference Manual [online]*. 2012-06 [cit. 2017-05-23], rev. 2.
URL http://www.nxp.com/assets/documents/data/en/reference-manuals/K60P144M100SF2V2RM.pdf

[20] NXP Semiconductors: *Kinetis K6: 100Mhz Cortex-M4 256/512KB Flash (144 pin) [online]*. 2013-06 [cit. 2017-05-23], rev. 3.
URL http://cache.freescale.com/files/32bit/doc/data_sheet/K60P144M100SF2V2.pdf

[21] Peck, R.; Deyl, S.; Miner, J.; aj.: *Amiga Hardware Reference Manual*. Addison-Wesley Publishing Company, Inc., 1986, ISBN 0-201-11077-6.

[22] Siemens: *HYB 41256-10/12/-15 262,144-Bit dynamic RAM*.

[23] Texas Instruments: *TFP410 TI PanelBus^{TM} Digital Transmitter [online]*. 2014-12 [cit. 2017-05-23].
URL http://www.ti.com/lit/ds/symlink/tfp410.pdf

[24] Vašíček, Z.; Strnadel, J.: FPGA - FITkit [online]. 2009-03-18 [cit. 2016-01-10].
URL http://merlin.fit.vutbr.cz/FITkit/docs/hardware/hw_fpga.html

[25] van Weeren, D.: *MINIMIG REV 1.1 [online]*. 2007.
URL http://www.techtravels.org/wp-content/uploads/pefiles/minimig/weeren001/downloads/minimig11_schematics.pdf

[26] van Weeren, D.: Minimig homepage [online]. [cit. 2016-01-10].
URL http://www.techtravels.org/wp-content/uploads/pefiles/minimig/weeren001/home.html

[27] van Weeren, D.: Minimig - An Amiga in an FPGA [online]. [cit. 2017-05-12].
URL https://www.techtravels.org/wp-content/uploads/pefiles/minimig/weeren001/minimig.html

[28] Wiesen, B.; Kasiko, T.; Kivi, T.; aj.: Gary - Custom Chips [online].
URL http://www.bigbookofamigahardware.com/bboah/product.aspx?id=1491

[29] Xilinx: *Spartan-6 FPGA Memory Controller [online]*. 2010-08-09 [cit. 2017-05-23], v2.3.
URL https://www.xilinx.com/support/documentation/user_guides/ug388.pdf

[30] Xilinx: *Spartan-6 Family Overview [online]*. 2011-10-25 [cit. 2017-05-23], v2.0.
URL https://www.xilinx.com/support/documentation/data_sheets/ds160.pdf

[31] Xilinx: *Spartan-6 FPGA Data Sheet: DC and Switching Characteristics [online]*. 2015-01-30 [cit. 2017-05-23], v3.1.1.
URL https://www.xilinx.com/support/documentation/data_sheets/ds162.pdf

[32] Xilinx: *Spartan-6 FPGA Clocking Resources [online]*. 2015-06-19 [cit. 2017-05-23], v1.10.
URL https://www.xilinx.com/support/documentation/user_guides/ug382.pdf

Příloha A

Obsah CD

Na přiloženém CD se nachází následující složky a soubory:

- `board` – soubory se schématem a návrhem propojovací desky plošných spojů pro program DipTrace a také soubory ve formátech Gerber a Excellon pro výrobu této desky
- `fpga` – projekt pro Xilinx ISE a zdrojové soubory pro FPGA
- `host` – projekt pro Kinetis Design Studio a zdrojové soubory pro hostitelský procesor
- `vnc` – projekt pro Vinculum II IDE a zdrojové soubory pro řadič USB kitu Minerva
- `report` – složka se zdrojovými texty této zprávy
- `tools/pll.hs` – jednoduchý program pro výpočet vhodných konfigurací PLL
- `xbiber00-a500.pdf` – text této zprávy