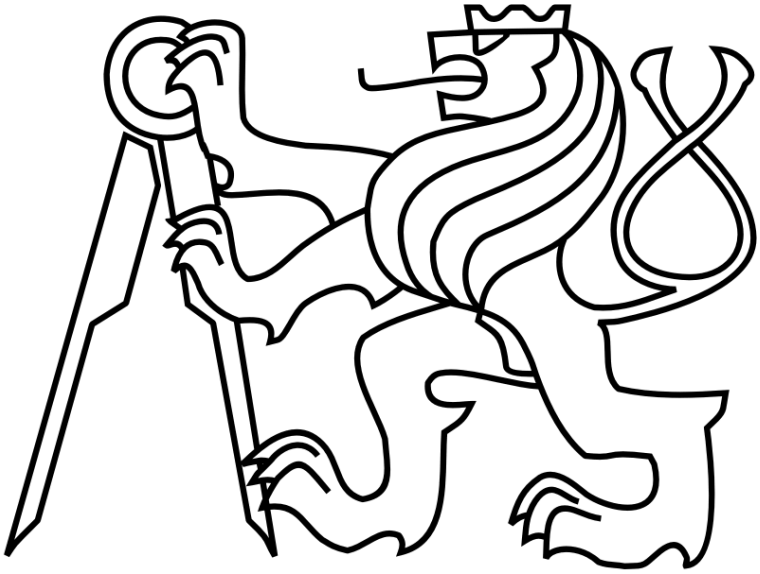
České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra měření



DIPLOMOVÁ PRÁCE

Modul zpracování obrazu se signálovým procesorem Blackfin ADSP-BF53x

Lukáš Grepl, 2008

Vedoucí práce: Ing. Jan Fischer, CSc.

České vysoké učení technické v Praze — Fakulta elektrotechnická

Katedra **měření 13138** — Školní rok 2006/2007

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student **Lukáš Grepl**

Obor **Měření a přístrojová technika**

Název tématu: **Modul zpracování obrazu se signálovým procesorem Blackfin ADSP-BF53x**

Zásady pro vypracování:

Navrhněte a posuďte možnost realizace modulu zpracování obrazu s procesorem ADSP-BF532, který by obsahoval také velkou paměť typu SDRAM a podporoval tak implementaci operačního systému uClinux.

Zhodnoťte možnosti použití operačního systému uClinux jako platformy pro vestavné zpracování obrazu v oblasti bezdotykového měření. Pro experimenty využijte např. desku Black Fin Stamp.

Navrhněte a realizujte blok digitalizace videosignálu s obvodem typu videokodek, který bude spolupracovat se signálovým procesorem Analog Devices ADSP BF532. Blok koncipujte tak, aby byl schopen spolupráce i s jinými procesory, případně i s hradlovým polem FPGA.

Navrhněte způsob spolupráce signálového procesoru s FPGA pro zvýšení rychlosti zpracování obrazu. Vyřešte spolupráci ADSP s rozhraním USB 2.0, případně s rozhraním Ethernet, pro přenos obrazu a výsledků měření na nadřazený systém. Správnost řešení modulu ověřte na základních úlohách určování polohy objektu.

Prohlášení

Prohlašuji, že jsem svou diplomovou práci vypracoval samostatně a použil pouze podklady (literaturu, projekty, SW) uvedené v přiloženém seznamu.

V Olomouci dne 18.1.2008

Lukáš Grepl

Rád bych poděkoval vedoucímu práce, Ing. Janu Fischerovi, CSc. za vedení práce a odborné konzultace. Dále bych rád poděkoval svým rodičům a své přítelkyni Daně Václavkové za podporu.

Anotace

Práce se zabývá návrhem a realizací modulu pro zpracování videosignálu se signálovým procesorem ADSP Blackfin. Modul obsahuje velkou paměť typu SDRAM, která představuje dostatečně rozsáhlý paměťový prostor pro zpracování barevného obrazu. Dále obsahuje obvody pro rychlé připojení k PC pomocí rozhraní USB 2.0 High-speed a Ethernet. Diskutovány jsou možnosti spolupráce procesoru s hradlovým polem FPGA pro zvýšení rychlosti zpracování.

Blok digitalizace videosignálu je realizován jako samostatný, univerzálně využitelný modul, připojitelný k základní desce, ale i k jiným zařízením s DSP/FPGA.

Práce se dále zabývá možnostmi použití operačního systému uClinux pro aplikace bezdotykového měření. uClinux byl úspěšně implementován na vyvinutý hardware a jeho použití bylo ověřeno na typických úlohách.

Annotation

This thesis deals with design and construction of module for video signal processing using ADSP Blackfin signal processor. Proposed module contains large SDRAM memory for color image manipulation and also enables implementation of uClinux embedded operating system. The module further contains USB 2.0 High-speed and Ethernet interfaces for high speed communication with PC-based computer. Potential of cooperation between DSP processor and FPGA is also discussed.

Video signal digitization block is designed as independent, general-purpose module, which can be attached to developed main board, as well as to other DSP- or FPGA-based applications.

This work also deals with ability of uClinux operating system to be used in contact-less measurement applications. uClinux has been successfully ported to developed hardware and its usage for implementation of typical measurement tasks has been evaluated.

Obsah

1. Úvod

1.1 Úvod do problematiky

V průmyslovém prostředí se často setkáváme s potřebou bezkontaktního měření parametrů objektů s využitím vizuální informace. Jde např. o měření rozměrů, polohy, určení lokálních vad materiálu apod. Zpracovaní se provádí na základě informace o jasu a/nebo barvě pocházející z řádkového nebo plošného snímače typu CMOS nebo CCD. Tato úloha se typicky řeší pomocí kompaktních videosenzorů, které v sobě obsahují hardware a software pro zpracování obrazu a jeho vyhodnocení, obvykle je integrovaná i samotná kamera.

Kompaktní videosenzory nabízené na trhu a používané v průmyslových aplikacích jsou do určité míry jednoúčelová zařízení, která se dají nakonfigurovat na řešení úlohy v rámci předem daných možností. Problém nastává při potřebě řešení specifických požadavků koncového zákazníka. Ke komerčním senzorům nebývá samozřejmě k dispozici vývojová dokumentace a zdrojové kódy firmware, takže úprava aplikace senzoru není v zásadě možná a je nutné zaplatit zákaznické úpravy výrobku nebo vyvinout vlastní řešení.

Moderní senzory obsahují obvykle výkonný signálový procesor a umožňují komunikaci přes rychlá rozhraní jako jsou např. USB, Ethernet a FireWire. Při požadavku na pokud možno co největší univerzálnost senzoru, který by měl být použitelný pro řešení rozličných úloh v oblasti bezdotykového měření, je vhodné zvážit využití standardního operačního systému jako základní části firmware. Hlavní výhodou takového řešení je velká podpora pro různé periferie a komunikační protokoly, a dále množství existujících programů a programových knihoven, které je možné na dané platformě přímo použít nebo pro použití relativně jednoduše upravit.

1.2 Cíle práce

Modul pro zpracování obrazu, který je předmětem této práce, by měl rozvinout předchozí implementace videosenzorů vyvinutých na naší katedře (viz např. práce [5]) v několika směrech. Bude obsahovat velkou paměť typu SDRAM, která umožní zpracování barevného videosignálu s možností lineárního uložení celého snímku (i několika snímků) do paměti. Přítomnost této paměti dále umožní implementaci operačního systému uClinux. Modul bude implementovat rychlá komunikační rozhraní USB 2.0 High-speed a Ethernet, která umožní přenos naměřených dat i samotného digitalizovaného barevného videosignálu v reálném čase do PC, popřípadě do jiného nadřazeného systému. Rozhraní pro připojení zdroje signálu je řešeno tak, aby bylo elektronicky i mechanicky kompatibilní s několika typy dříve vyvinutých senzorů. Prakticky je tak možné ve spolupráci se základní deskou použít moduly implementující různé zdroje videosignálu jako jsou řádkové snímače, integrované CMOS snímače či televizní CCD kamery.

Cílem je koncipovat modul se signálovým procesorem tak, aby umožňoval využití jako univerzální platformy pro realizaci úloh z oblasti zpracování signálů a řízení jak při praktickém použití, tak při výuce předmětů týkajících se mikroprocesorové techniky a zpracování obrazu. Vyvíjený modul je možné obecně použít jako výkonnou platformu pro tvorbu zařízení určených pro zpracování nebo generování signálů různých druhů. Příklady možných aplikací realizovatelných na této platformě jsou např. logický analyzátor, digitální osciloskop, pattern generator nebo arbitrary function generator.

Blok digitalizace je oddělen jako samostatný modul, který je možné připojit k rozhraní na základní desce, případně je možné jej využít i ve spojení s jiným systémem např. na bázi hradlového pole FPGA. Úlohou modulu je zpracovat černobílý nebo barevný videosignál z konvenčních kamer s výstupem televizního signálu podle normy PAL případně NTSC do podoby digitalizovaných dat. Modul umožní i připojení dvou nebo více kamer současně pro realizaci úloh z oblasti stereoskopického snímání obrazu.

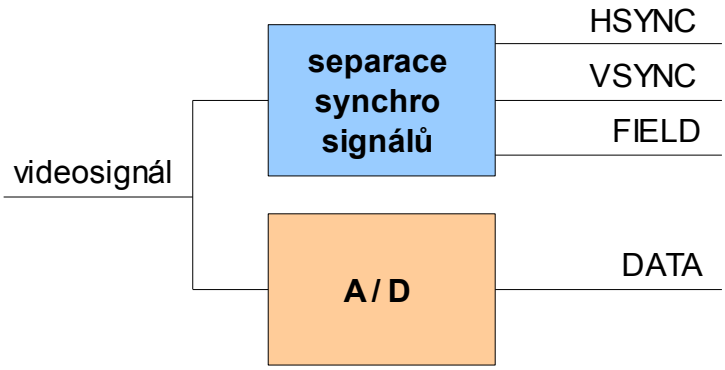
Část práce bude věnována problematice použití operačního systému uClinux. Cílem je seznámit čtenáře s možnostmi, výhodami a nevýhodami tohoto systému a shrnout kroky potřebné pro portování systému na vlastní hardware (především s ohledem na vyvíjený modul zpracování obrazu). Dále budou diskutovány možnosti využití aplikací, které jsou součástí distribuce uClinux a prostředky pro implementaci aplikací vlastních. Popis se bude věnovat především využití periferií důležitých pro použití v měřící a řídící technice a také komunikaci přes rozhraní USB a Ethernet.

Použití operačního systému může znamenat také určité výkonové omezení kvůli jeho vlastní režii, proto je vhodné provést srovnání výkonu aplikací při použití operačního systému oproti jejich nativní implementaci na základních algoritmech pro zpracování obrazu. Řešení hardwarových modulů a využití operačního sytému bude ověřeno na úloze určování polohy objektu metodou hledání těžiště.

2. Modul digitalizace videosignálu

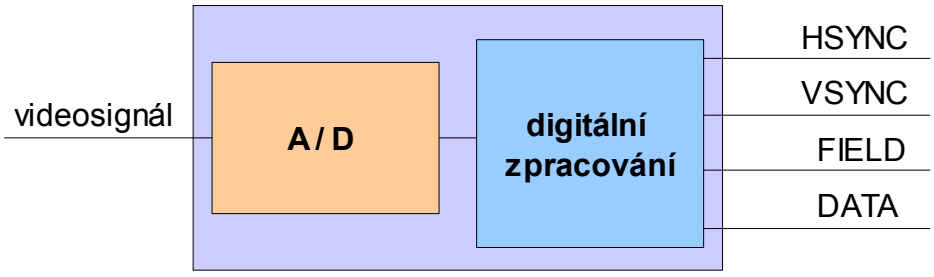
2.1 Úvod

Modul digitalizace videosignálu má zajistit převedení analogového videosignálu do digitální podoby vhodné ke zpracování signálovým procesorem. Pro vlastní digitalizaci má být dle zadání použit obvod typu videokodek (v tomto případě přesněji řečeno videodekodér). Na obr. 2.1 je pro ilustraci uvedeno ideové schéma klasické koncepce digitalizace videosignálu se samostatným separátorem synchronizační směsi a A/D převodníkem.



Obr. 2.1 Ideové schéma klasické koncepce zpracování videosignálu

Videodekodéry v sobě integrují prakticky kompletní hardware pro zpracování analogového videosignálu, přičemž těží z digitálního zpracování signálu za A/D převodníkem. Většina operací probíhá až po převedení signálu do digitální podoby. Ideové schéma zpracování signálu ve videodekodéru je znázorněno na obr. 2.2.



Obr. 2.2 Ideové schéma zpracování signálu v integrovaných videodekodérech

Digitálním zpracováním se oddělí synchronizační směs, provádí se demodulace barevné složky, je implementována smyčka fázového závěsu (DPLL), která generuje hodinový signál A/D převodníku atd.

2.1.1 Přehled obvodů pro digitalizaci videosignálu

V tab. 2.1 jsou shrnuty základní údaje o vybraných integrovaných obvodech pro digitalizaci videosignálu od různých výrobců. Tabulka obsahuje pro každý obvod jeho typ, výrobce, počet vstupních kanálů, počet bitů A/D převodníku a jeho vzorkovací frekvenci, možné typy vstupního videosignálu, formát výstupních dat, typický celkový příkon, pouzdro a orientační cenu obvodu.

Tab. 2.1 Přehled obvodů „Video decoder" různých výrobců

Typ	Výrobce	Vst. kan.	bity ADC	f_s [MHz]	Vst. signál CVBS, Y/C +	Výstupní data	P_{tot} [W]	Pouzdro	Orientační cena [USD]
ADV7180	ADI	6	10	86	YPbPr	8 & 16-bit YCbCr 4:2:2	0,25	40-LFCSP 64-LQFP	5,7
ADV7181B	ADI	6	9	54	YPbPr	8 & 16-bit YCbCr 4:2:2	0,45	64-LFCSP 64-LQFP	7,0
ADV7183B	ADI	12	10	54	YPbPr	8 & 16-bit YCbCr 4:2:2	0,50	80-LQFP	8,5
ADV7184	ADI	12	10	54	YPbPr, RGB	8 & 16-bit YCbCr 4:2:2	0,55	80-LQFP	8,5
ADV7188	ADI	12	12	54	YPbPr, RGB	8 & 16-bit YCbCr 4:2:2	0,55	80-LQFP	11,4
ADV7401	ADI	12	10	140	YPbPr, RGB	12-bit RGB DDR 24-bit YCbCr/RGB 4:4:4 8 & 16-bit YCbCr 4:2:2	0,55	100-LQFP	14,8
ADV7403	ADI	12	12	140	YPbPr, RGB	10 & 20-bit YCbCr 4:2:2 12-bit RGB DDR 24-bit YCbCr/RGB 4:4:4 8 & 16-bit YCbCr 4:2:2	0,55	100-LQFP	20,9
HMP8117	Intersil	3	10	27		8 & 16-bit YCbCr 4:2:2 15-bit (5, 5, 5) RGB 16-bit (5, 6, 5) RGB 8-bit BT.656	1,46	80-QFP 80-QFN	19,5
SAA7114	NXP	6	9	27		4:2:2, 4:1:1, 4:2:0, 4:1:0 YCbCr; 8-bit luminance; raw CVBS	0,45	156-BGA 100-LQFP	20,0
TVP5146	TI	10	10	30	YPbPr, RGB	10-bit BT.656 4:2:2 YCbCr, 20-bit 4:2:2 YCbCr	0,73	80-HTQFP	3,5
TVP5147	TI	10	10	30	YPbPr	20-bit 4:2:2 YCbCr, 10-bit BT.656 4:2:2 YCbCr, 10-bit 4:2:2 YCbCr	0,49	80-HTQFP	3,3
TVP5150A	TI	2	9	30		8-bit 4:2:2 YCbCr	0,12	32-TQFP	7,5
TVP5154	TI	4	9	27		8-bit 4:2:2 YCbCr	0,70	128-TQFP	7,0
TVP5160	TI	12	10	54	YPbPr, RGB	10-bit BT.656 4:2:2 YCbCr	0,90	128-TQFP	5,8

Poznámky k tabulce:

- Všechny obvody zpracovávají signál podle norem PAL, NTSC a SECAM, kromě HMP8117, který nemá implementován formát SECAM.
- Všechny obvody umí zpracovat vstupní signál CVBS a Y/C.
- Ceny jsou orientační ceny uváděné výrobcem pro 1000 ks.
- Výrobci: ADI – Analog Devices, Inc.; Intersil – Intersil Corporation; NXP – NXP Semiconductors (dříve Philips); TI – Texas Instruments

Obvody ADV7401 a ADV7403 pro zpracování HDTV videa jsou v současné době dostupné (včetně podrobných informací o nich) pouze pro OEM zákazníky s odběrem od 30 tisíc kusů ročně.

2.2 Rozbor

Pro realizaci modulu byl vybrán obvod ADV7180 v pouzdře LQFP-64. Důvodem pro výběr výrobce Analog Devices byla především snadná dostupnost jak vzorků, tak většího množství obvodů. Typ ADV7180 má z celé řady nejmenší spotřebu i nejnižší cenu. Varianta v pouzdře LQFP-64 má oproti alternativě LFCSP-40 větší počet vstupů signálu (6 oproti 3).

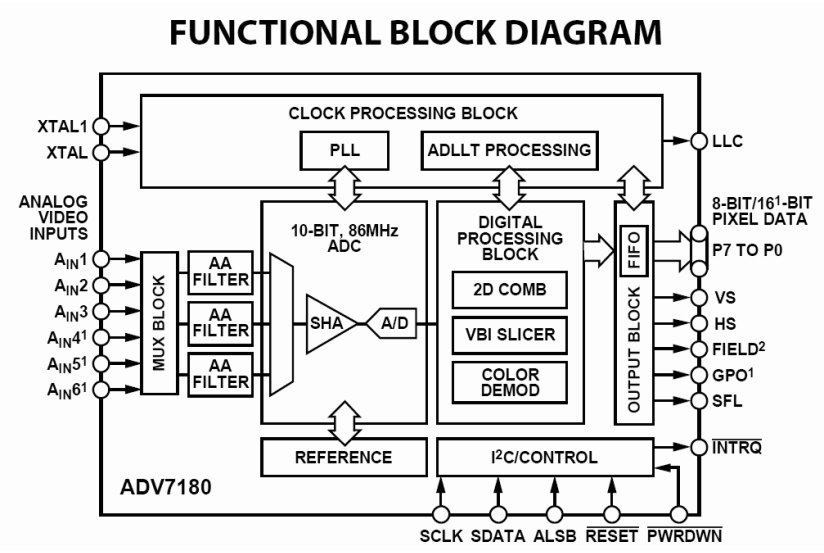
Výstupní rozhraní obvodu umožňuje jak připojení k rozhraní PPI procesoru Blackfin, tak zpracování digitalizovaného signálu hradlovými poli FPGA. Vlastnosti synchronizačních signálů a mnoho dalších parametrů obvodu je možné konfigurovat přes standardní rozhraní I2C.

2.3 Popis zapojení modulu

2.3.1 Vlastnosti obvodu ADV7180

ADV7180 obsahuje kompletní hardware potřebný pro zpracování barevného videosignálu. Obvod umožňuje multiplexovaně připojit až šest zdrojů kompozitního signálu CVBS, případně tři dvojice signálů S-Video nebo dvě dvojice signálu YPbPr. Tyto varianty se dají dále vzájemně kombinovat (např. dva vstupy CVBS a dva vstupy S-Video).

Blokové zapojení obvodu ADV7180 je uvedeno na obr. 2.3. Tento obvod v sobě integruje vstupní zesilovače, multiplexer pro přiřazení signálů k jednotlivých analogovým vstupům, antialiasingové filtry, 10-bit A/D převodník a následně obvody pro digitální oddělení synchronizace, demodulaci barevné složky a celkové dekódování videosignálu. Dokáže zpracovat signály norem PAL, SECAM a NTSC. Obvod umožňuje také zpracování doplňkových informací, obsažených ve videosignálu (teletext apod.). K dispozici jsou dále 4 volně programovatelné výstupy.



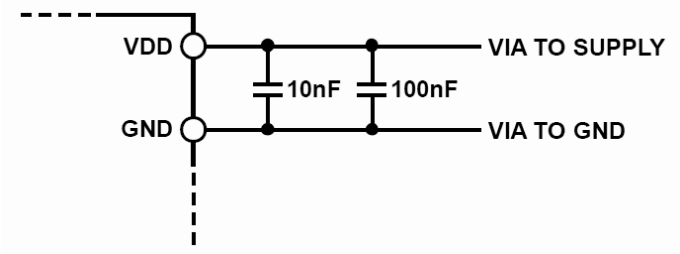
Obr. 2.3 Vnitřní blokové schéma obvodu ADV7180

Ovládání obvodu ze strany nadřazeného systému je realizováno po sběrnici I2C. Výstupní datový tok je přenášen po 8 nebo 16 bitové sběrnici, doplněné dekódovanými synchronizačními signály a výstupem pro generování přerušení.

2.3.2 Napájení obvodu

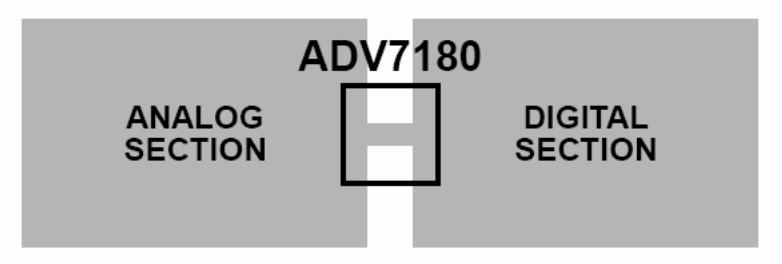
Pro svou funkci potřebuje obvod napájení 3,3 V (pro napájení I/O interface) a 1,8 V (pro napájení digitální části, analogové části a PLL). Typický celkový příkon obvodu je cca 250 mW. Výrobce doporučuje použít pro každý napájecí vstup samostatný stabilizátor, toto doporučení bylo dodrženo.

Důležité je také důkladné zablokování všech napájecích vývodů co nejblíže pouzdru obvodu (viz obr. 2.4 převzatý z datasheetu [13]). Každá dvojice napájecích vývodů je zablokována kondenzátory 10 nF a 100 nF podle doporučení uvedených v datasheetu.



Obr. 2.4 Způsob zablokování napájecích vývodů

Dalším důležitým požadavkem uvedeným v datasheetu je nízkoimpedanční rozvod analogové a digitální země, které mají být spojeny v jednom bodě pod obvodem videokodeku, viz obr. 2.5, taktéž z datasheetu [13].



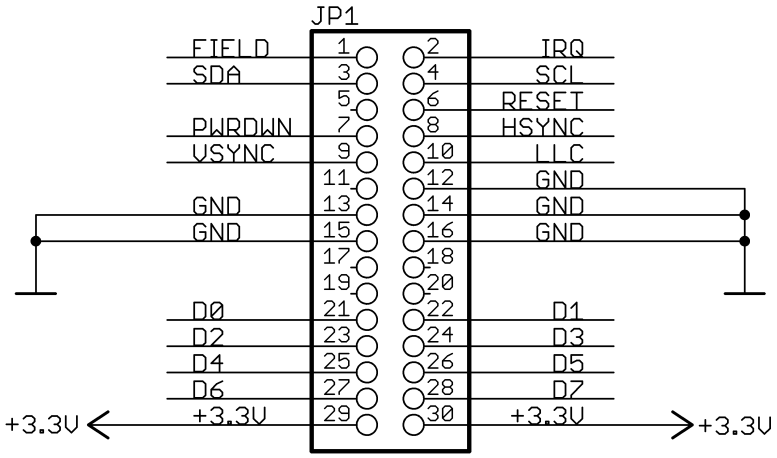
Obr. 2.5 Způsob propojení analogové a digitální země

Při návrhu plošného spoje byl kladen velký důraz na dodržení doporučení ohledně layoutu uvedená v dokumentaci výrobce – zejména způsob blokování napájení a rozvodu zemí.

2.3.3 Vstupy a výstupy modulu

Zapojení modulu vychází z výrobcem doporučeného základního zapojení. Pro připojení videosignálů jsou k dispozici 3 konektory Cinch/RCA (pro připojení kompozitního videosignálu) a jeden konektor Mini-DIN (typicky pro připojení videosignálu S-Video). Všechny podstatné signály jsou vyvedeny na konektor JP1, včetně napájení 3,3 V. Výstupy GPO jsou připojeny na LED diody, které jsou rozmístěny u vstupních konektorů.

Zapojení konektoru JP1 a mechanický formát modulu je zvolen tak, aby byl kompatibilní s modulem CMOS kamery dříve vyvinutým na naší katedře. Výstupní datová sběrnice je 8 bitová. Zapojení konektoru viz obr. 2.6 a tab. 2.2.



Obr. 2.6 Konektor JP1 – interface nadřazeného modulu

Tab. 2.2 Význam signálů konektoru JP1 modulu videokodeku

Piny	Název signálu	Směr	Popis
1	FIELD	Out	**Field synchronization** (indikace sudého/lichého půlsnímku)
2	IRQ#	Out	**Interrupt request** – je vyvolán při detekci specifických příznaků ve vstupním signálu (blíže viz dokumentace k obvodu; aktivní v log. 0)
3	SDA	In/Out	**I2C data**
4	SCL	In	**I2C clock**
6	#RESET	In	**Reset** (aktivní v log. 0, minimální šířka impulsu 5ms)
7	#PWRDWN	In	**Power down** – režim snížené spotřeby (aktivní v log. 0)
8	HSYNC	Out	**Horizontal synchronization** (řádková synchronizace)
9	VSYNC	Out	**Vertical synchronization** (snímková synchronizace)
10	LLC	Out	**Line Locked Output** (hodinový signál udávající platnost dat D0-D7 a synchronizačních signálů)
21-28	D0-D7	Out	**Data output**
29, 30	+3,3V	Power	**+3,3V power input** (proudová spotřeba cca 130-180mA)
12-16	GND	Ground	**Ground**

Signál SFL (Subcarrier Frequency Lock) je vyveden mimo konektor JP1 pouze na testovací plošku. Tento signál obsahuje barvonosnou frekvenci a synchronizační směs – tzv. „Genlock telegram“. Využívá se pro spojení s videoenkodéry Analog Devices při digitalizaci a opětovném generování televizního signálu v reálném čase.

Signál IRQ# je ve výchozím nastavení aktivní v log. 0, přičemž v klidu je ve stavu vysoké impedance (výstup typu open-drain). Nastavením konfiguračního registru INTRQ_OP_SEL[1:0] je možné změnit chování signálu IRQ# tak, aby byl výstup buzen v obou úrovních, a dále nastavit, ve které úrovni je signál aktivní.

2.4 Konfigurace obvodu

Pro konfiguraci obvodu je k dispozici obousměrná sběrnice I2C (signály SDA, SCL). Komunikační adresa obvodu nastavená úrovní na pinu ALBS je 0x40. Celkem je možné nastavovat parametry v přibližně 90 konfiguračních registrech.

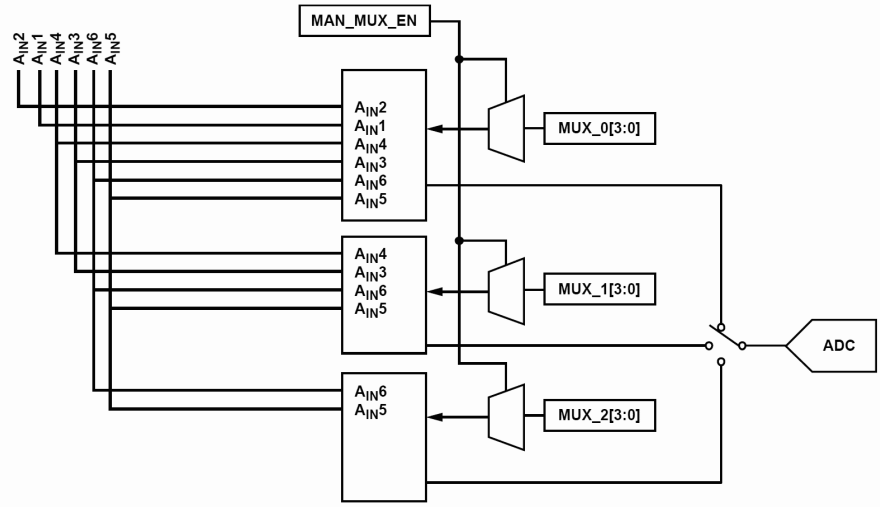
2.4.1 Nastavení frekvence krystalu

Obvod ADV7180 předpokládá ve standardním nastavení připojení krystalu s poněkud méně obvyklou frekvencí 28,63636 MHz. Alternativně je možné použít krystal 27 MHz, který je mnohem lépe dostupný. Při použití tohoto krystalu je nutné nastavit bit 6 EN28XTAL registru 0x1D na hodnotu 0.

Pokud frekvence použitého krystalu neodpovídá nastavení v uvedeném registru, není možné u barevného signálu dekódovat barevnou složku. Dále v režimu, kdy není připojen vstupní signál, frekvence signálu LLC neodpovídá nominální hodnotě 27 MHz.

2.4.2 Konfigurace vstupů

Na modulu je vyvedeno pět ze šesti vstupů, které jsou na obvodu kodeku k dispozici (vstup č. 3 není vyveden). Obvod ADV7180 má poměrně široké možnosti využití jednotlivých vstupů, které ilustruje následující diagram (převzatý z [13]).



Obr. 2.7 Vnitřní propojení vstupů videosignálu

Pro praktické použití je většinou dostačující využití 11 předdefinovaných konfigurací, které je možno zvolit v registru INSEL (adresa 0x00):

Tab. 2.3 Konfigurace vstupů pomocí registru INSEL

INSEL[3:0]	Video Format	Analog Input
0000	Composite	CVBS → $A_{IN}1$
0001	Composite	CVBS → $A_{IN}2$
0010	Composite	CVBS → $A_{IN}3$
0011	Composite	CVBS → $A_{IN}4$
0100	Composite	CVBS → $A_{IN}5$
0101	Composite	CVBS → $A_{IN}6$
0110	Y/C (S-video)	Y → $A_{IN}1$ C → $A_{IN}4$
0111	Y/C (S-video)	Y → $A_{IN}2$ C → $A_{IN}5$
1000	Y/C (S-video)	Y → $A_{IN}3$ C → $A_{IN}6$
1001	YPrPb	Y → $A_{IN}1$ Pb → $A_{IN}4$ Pr → $A_{IN}5$
1010	YPrPb	Y → $A_{IN}2$ Pb → $A_{IN}3$ Pr → $A_{IN}6$
1011 to 1111	Not used	Not used

Pokud by ve specifických případech nastavení registrem INSEL nevyhovovalo, je možné pomocí registrů MAN_MUX_EN, MUX_0, MUX_1 a MUX_2 nastavit téměř libovolné přiřazení vstupů – podrobnosti viz datasheet [13].

2.4.3 Ovládání LED

Indikační LED jsou připojeny na výstupy GPO (General Purpose Outputs). Tyto výstupy je možné ovládat přes registr GPO CONTROL, který se nachází na adrese 0x59. Pro aktivaci výstupů je nutné nastavit bit 4 GPO_Enable na 1. Spodní 4 bity registru poté odpovídají jednotlivým LED (D1-D4). LED svítí při nastavení příslušného výstupu do úrovně log. 0. Pokud je GPO_Enable vynulován, jsou výstupy ve stavu vysoké impedance.

2.4.4 Nastavení polohy a šířky synchronizačních impulsů

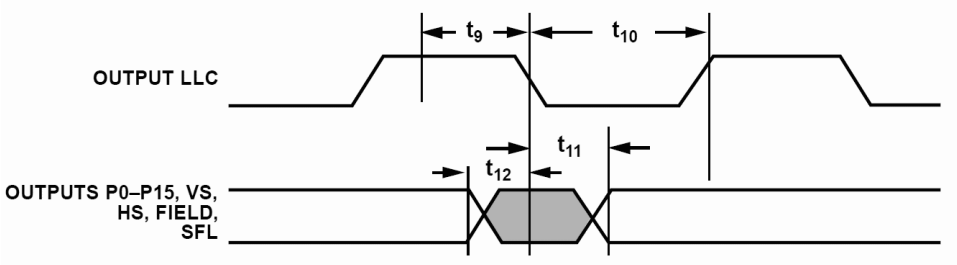
Obvod umožňuje nastavit polohu a šířku synchronizačních impulsů HSYNC a VSYNC, což může být výhodné zejména pro zpracování digitalizovaného signálu jednoduššími obvody, jako je jsou hradlová pole CPLD ve spolupráci s méně výkonnými procesory. Vhodným nastavením tvaru synchronizačních impulsů je možné přímo těmito signály specifikovat zájmovou oblast snímku, kterou je možné následně jednoduše vyklíčovat, čímž se zjednoduší zpracování a omezí potřebná šířka pásma pro případný přenos snímku po komunikačním rozhraní.

Pro nastavení polohy signálu HSYNC slouží registry HSB[10:0] (HSYNC beginning), HSE[10:0] (HSYNC end) a dále bit PHS, který nastavuje polaritu signálu. Poloha impulsu je nastavována po jednotlivých cyklech signálu LLC a impuls může být umístěn kdekoliv v řádku. Výchozí hodnoty jsou HSB = 2, HSE = 0, PHS = 0 (signál je aktivní v log. 1). Signál HSYNC je tedy ve výchozím nastavení v úrovni log. 0 první dva pixely řádku a zbytek periody je v úrovni log. 1.

Nastavení parametrů signálu VSYNC má více možností, podrobnosti viz datasheet obvodu [13], kapitola „VS and FIELD Configuration“, str. 44.

2.5 Výstup digitalizovaného videosignálu

Digitalizovaný videosignál je vyveden na paralelní výstupní rozhraní, obsahující data D0-D7, synchronizační signály VSYNC, HSYNC a FIELD. Všechny tyto signály jsou synchronní s hodinovým signálem LLC, viz obr. 2.8 převzatý z datasheetu [13]. Polaritu signálu LLC lze v případě potřeby změnit v konfiguračních registrech obvodu, při implicitním nastavení jsou signály platné při jeho náběžné hraně.



Obr. 2.8 Časovací diagram výstupních signálů ADV7180

Typická hodnota frekvence signálu LLC je 27 MHz, přesná aktuální hodnota ovšem závisí na vstupním signálu, protože signál LLC je na něj fázově zavěšen. Jeden televizní řádek je digitalizován do 720 pixelů, celkový počet řádek je 625 resp. 525, z toho aktivních je 576 resp. 480, podle normy vstupního signálu PAL resp. NTSC.

V případě, že na vstupu není přítomen signál nebo se obvod nedokáže na připojený signál zachytit, je interně generován signál LLC o frekvenci 27 MHz včetně synchronizačních impulsů HS a VS. Výstupní data obvodu tvoří jednolitý obraz implicitně modré barvy (barvu je možné změnit v konfiguračních registrech).

Výstupní datový tok má formát YCbCr 4:2:2 kompatibilní se standardem ITU-R BT.656 [21]. BT.656 je mezinárodní standard pro přenos digitalizovaného videosignálu, který definuje sériové a paralelní rozhraní pro přenos signálu včetně synchronizačních impulsů zakódovaných do datového toku.

Synchronizační signály VSYNC, HSYNC a FIELD jsou k dispozici pro připojení k obvodům, které neumí zpracovat signál podle normy ITU-R BT.656.

2.5.1 Formát datového toku

Ve výstupním datovém toku se střídají hodnoty jasové složky (luminance, hodnota Y) a barvových složek (rozdílová složka modré barvy C_B, rozdílová složka červené barvy C_R). Datová slova, která obsahují samé jedničky (0xFF) nebo samé nuly (0x00) jsou rezervovaná pro synchronizační účely. Pro signálové hodnoty zbývá tedy 254 z 256 možných hodnot. Pořadí jasových a barvových složek je

C_B, Y, C_R, Y, C_B, Y, C_R, …,

přičemž skupina C_B, Y, C_R odpovídá jednomu obrazovému bodu, dále následuje samostatná hodnota jasu Y dalšího bodu a celá situace se opakuje.

Hodnoty barvových složek u samostatných jasových hodnot musí být interpolovány ze sousedních dat. V praxi je možné použít např. jednoduché lineární interpolace z přilehlých hodnot, což vzhledem k omezené šířce pásma barvonosného signálu dostačuje.

2.5.1.1 Barevný prostor YCbCr, přepočet na RGB

Barevný prostor YCbCr je definován v normě ITU-R BT.601 [22] (dříve CCIR 601). Při popisu přepočtu vycházím především z dokumentu [26].

Jasová složka Y (luminance) má podle [26] definován nominální rozsah 16-235 (černá-bílá), barvové rozdílové složky C_B a C_R mají nominální rozsah 16-240, přičemž hodnota 128 odpovídá nule. V reálných signálech se mohou ojediněle vyskytnout i vzorky mimo uvedené rozsahy – dochází k tomu vlivem zpracování, zaokrouhlovacích chyb a šumu.

Barevný prostor YCbCr je odvozen lineární transformací z RGB dat po gamma-korekci, které budou dále označeny jako R'G'B'. Vztahy pro přepočet mezi YCbCr a R'G'B' jsou:

$$Y = \frac{77}{256}\cdot R' + \frac{150}{256}\cdot G' + \frac{29}{256}\cdot B'$$
$$C_B = -\frac{44}{256}\cdot R' - \frac{87}{256}\cdot G' + \frac{131}{256}\cdot B' + 128$$
$$C_R = \frac{131}{256}\cdot R' - \frac{110}{256}\cdot G' - \frac{21}{256}\cdot B' + 128$$

$$R' = Y + 1{,}371\cdot(C_R - 128)$$
$$G' = Y - 0{,}698\cdot(C_R - 128) - 0{,}336(C_B - 128)$$
$$B' = Y + 1{,}732\cdot(C_B - 128)$$

V případě, že potřebujeme mít data R'G'B' v rozsahu 0-255 (což je běžné např. pro zobrazení a další zpracování na PC, konverzi do obrazových formátů apod.), je třeba použít následující vzorce:

$$Y = 0{,}257\cdot R' + 0{,}504\cdot G' + 0{,}098\cdot B'$$
$$C_B = -0{,}148\cdot R' - 0{,}291\cdot G' + 0{,}439\cdot B' + 128$$
$$C_R = 0{,}439\cdot R' - 0{,}368\cdot G' - 0{,}071\cdot B' + 128$$

$$R' = 1{,}164\cdot(Y - 16) + 1{,}596\cdot(C_R - 128)$$
$$G' = 1{,}164\cdot(Y - 16) - 0{,}813\cdot(C_R - 128) - 0{,}392(C_B - 128)$$
$$B' = 1{,}164\cdot(Y - 16) + 2{,}017\cdot(C_B - 128)$$

Výsledné hodnoty R'G'B' je třeba omezit na rozsah 0-255, kvůli možným výchylkám hodnot YCbCr mimo nominální rozsah.

2.5.1.2 Odstranění gamma korekce

Při zpracování digitalizovaného signálu je podle situace vhodné až nutné odstranit gamma korekci hodnot R'G'B'. Pro gamma korekci se používá exponentu 2,2 podle normy NTSC a 2,8 podle normy PAL. Nicméně v dnešní době je běžné používat i u normy PAL exponent 2,2 (uvedeno v [26]).

Pro přepočet gamma-korigovaných hodnot R'G'B' na lineární RGB se použijí následující vztahy (opět podle [26]):

$$R = 255\cdot\left(\left(\frac{R'}{255} + 0{,}099\right)\cdot\frac{1}{1{,}099}\right)^{2{,}2}$$
$$G = 255\cdot\left(\left(\frac{G'}{255} + 0{,}099\right)\cdot\frac{1}{1{,}099}\right)^{2{,}2}$$
$$B = 255\cdot\left(\left(\frac{B'}{255} + 0{,}099\right)\cdot\frac{1}{1{,}099}\right)^{2{,}2}$$

Pro exponent 2,8 u systému PAL se použijí vztahy:

$$R = 255 \cdot \left(\frac{R'}{255} \right)^{2,8}$$
$$G = 255 \cdot \left(\frac{G'}{255} \right)^{2,8}$$
$$B = 255 \cdot \left(\frac{B'}{255} \right)^{2,8}$$

Implementaci uvedených vztahů je vhodné provést pomocí předem vypočítaných vyhledávacích tabulek – přímý výpočet mocniny v reálném čase by byl velmi problematický.

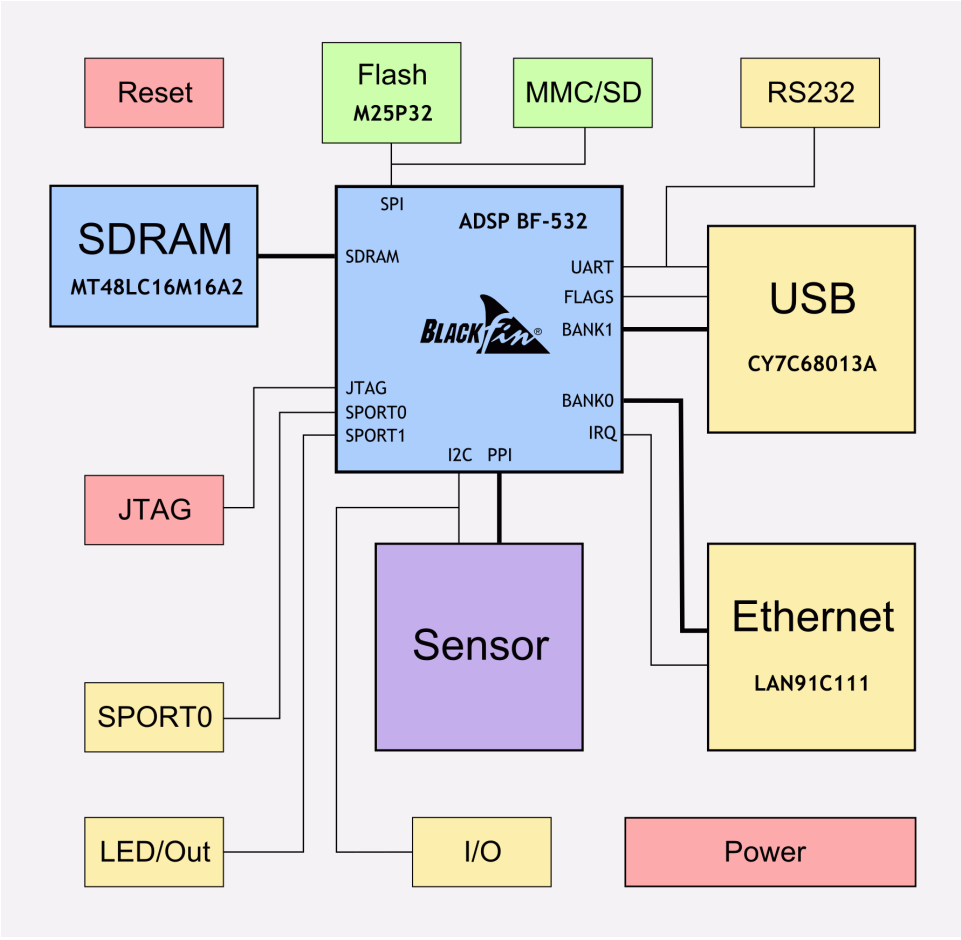
3. Modul pro zpracování obrazu

3.1 Úvod

Zapojení tohoto modulu vzniklo z potřeby univerzální platformy pro zpracování obrazu obsahující signálový procesor řady ADSP Blackfin, na které by bylo dostupné rychlé komunikační rozhraní (Ethernet, USB 2.0 High-speed) a dostatečně velká paměť SDRAM. Modul by měl umožnit komfortní zpracování digitalizovaného barevného obrazu a realizaci rozličných úloh z oblasti bezdotykového měření.

Přítomnost paměti SDRAM dále umožní vyzkoušet a zhodnotit možnosti použití operačního systému uClinux pro úlohy bezdotykového měření.

Na obr. 3.1 je uvedeno blokové schéma znázorňující klíčové části modulu.



Obr. 3.1 Blokové schéma modulu pro zpracování obrazu

Jádrem zapojení je procesor ADSP Blackfin BF-532 společně s pamětí SDRAM o velikosti 32 MB. Operační systém (případně samostatná jednoúčelová aplikace) je uložen v paměti flash o velikosti 4 MB (IC6, M25P32). Dalším paměťovým zařízením je karta MMC (případně kompatibilní SD), na

které může být uložena část operačního systému (kořenový souborový systém) a mohou na ni být ukládána nasnímaná data.

Blok „Sensor“ reprezentuje připojený externí obrazový senzor, což může být např. CMOS nebo CCD plošná či řádková kamera, modul digitalizace videosignálu (viz kapitola 2) apod. Je možné připojit také samostatný A/D (nebo i D/A) převodník pro digitalizaci nebo generování obecných signálů. Dále je možné využít toto rozhraní i jako rychlý digitální vstup nebo výstup.

Komunikaci s dalšími systémy zajišťuje několik digitálních rozhraní:

- USB 2.0 High-speed (obvod Cypress CY7C68013A)
- Ethernet 10/100 Mbps (obvod SMSC LAN91C111)
- RS232, bez handshake (obvod ST3232)
- SPORT (vyvedeny všechny signály SPORT0)
- 8× digitální I/O (obvod PCF8574 na sběrnici I2C)
- 4× digitální výstup (obvod 74HC595 připojený na SPORT1)

Pro indikaci vnitřních stavů je možné využít čtyři LED (připojené přes posuvný registr na SPORT1). Dále jsou na desce svítivé diody indikující přítomnost napájecího napětí, přístup na MMC/SD kartu. Další LED je připojena k obvodu rozhraní USB pro indikaci jeho činnosti.

Obvod pro generování resetu ovládá procesor Blackfin, rozhraní USB, rozhraní Ethernet a signál je vyveden také na konektor senzoru.

Pro ladění je k dispozici rozhraní JTAG.

Napájení celého modulu zajišťuje spínaný zdroj, který stabilizuje základní napájecí napětí 3,3 V, dále se lineárním stabilizátorem vytváří napětí 1,2 V pro jádro procesoru. Rozsah vstupního napětí je 5 až 36 V, maximální odběr ze zdroje je cca 700mA.

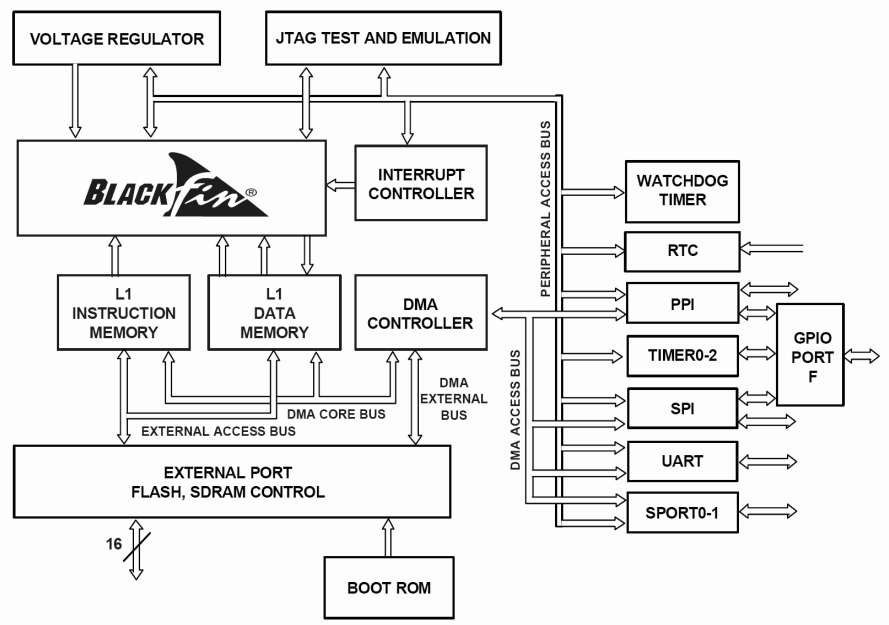
Zapojení bylo realizováno na 4-vrstvé desce plošných spojů.

3.2 Popis zapojení modulu

3.2.1 CPU

3.2.1.1 ADSP-BF532

Srdcem modulu pro zpracování obrazu je signálový procesor Analog Devices Blackfin ADSP-BF532BST400 v pouzdru TQFP-176. Procesor obsahuje dvě 16 bitové jednotky MAC (Multiply-and-accumulate), dvě 40 bitové aritmeticko-logické jednotky (ALU), čtyři 8 bitové jednotky video-ALU a 40 bitový posuvný registr (barrel shifter). Vnitřní blokové schéma obvodu je uvedeno na obr. 3.2 převzatém z datasheetu [1].



Obr. 3.2 Vnitřní blokové schéma procesoru Blackfin

Hodinové signály pro jádro (CCLK) a pro periferie (SCLK) jsou generovány interním obvodem s fázovým závěsem, kterým je možné násobit frekvenci krystalu koeficientem 0,5 až 64 (CCLK). Poměr frekvencí SCLK a CCLK může být 1:1 až 1:15.

Rozsah napájecího napětí jádra procesoru je 0,85-1,30 V a toto napětí může být regulováno spínaným stabilizátorem přítomným na čipu. Napájecí napětí periferních obvodů může být 1,8, 2,5 nebo 3,3 V.

Procesor může pracovat s hodinovou frekvencí jádra CCLK až 400 MHz (při napětí jádra $V_{dd,int} \geq$ 1.14 V) a s maximální frekvencí pro taktování periferií SCLK 133 MHz (při napětí jádra $V_{dd,int} \geq 1.14$ V a při napájecím napětí $V_{dd,ext} \geq 2.5$ V) – tyto údaje platí pro použité pouzdro TQFP-176.

V obvodu se nachází paměť RAM o celkové velikosti 84 KB, která je rozdělena na několik bloků:

- Instruction SRAM/Cache 16 KB
- Instruction SRAM 32 KB
- Data SRAM/Cache 32 KB
- Scratchpad 4 KB

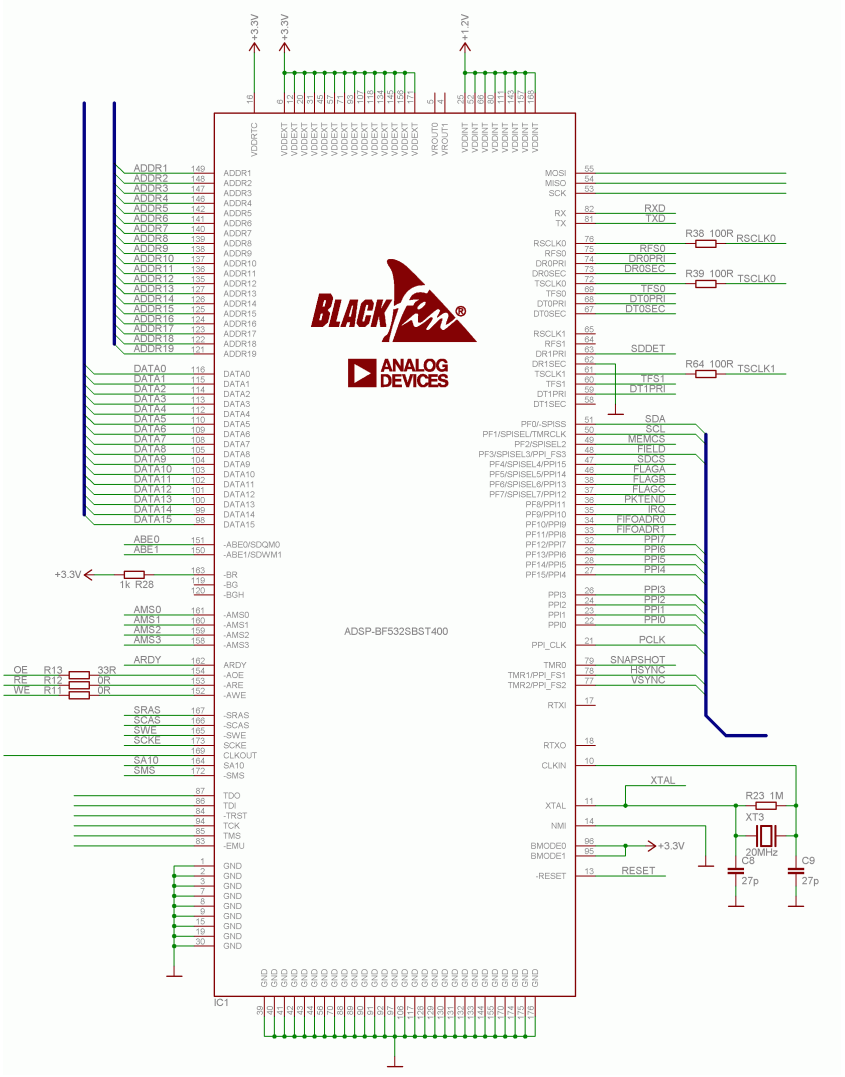
Dále procesor obsahuje řadič externí paměti EBIU, který poskytuje podporu pro přímé připojení pamětí SDRAM, SRAM, flash a dalších zařízení připojitelných ke sběrnici. V obvodu je integrována řada periferií:

- 16 GPIO pinů (PF), alternovaných s rozhraním PPI
- dva synchronní sériové porty SPORT
- asynchronní sériový port UART s podporou IrDA

- rozhraní SPI
- čtyři kanály DMA pro přenos paměť-paměť
- osm kanálu DMA pro přenos dat mezi pamětí a periferiemi
- tři 32 bitové časovače
- hodiny reálného času RTC se zálohováním
- JTAG interface pro ladění

Bootování programu je možné ze sériové paměti SPI flash nebo z paměti flash připojené ke sběrnici.

3.2.1.2 Popis zapojení CPU



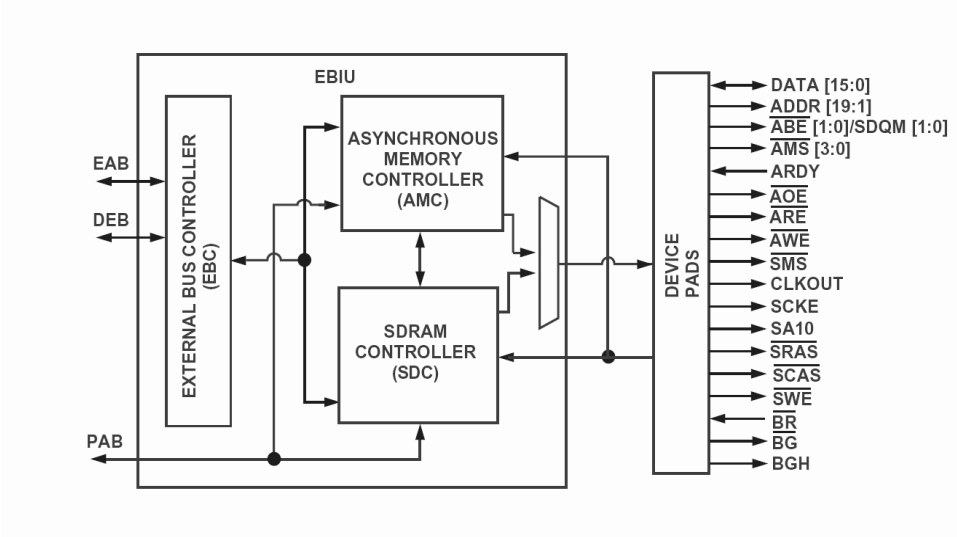
Obr. 3.3 Zapojení procesoru ADSP-BF532 (IC1)

3.2.2 EBIU

Jednotka vnější sběrnice (External Bus Interface Unit) umožňuje přímé připojení vnějších synchronních pamětí SDRAM (PC100 a PC133 SDRAM), asynchronních pamětí SRAM, flash apod. a dalších periferií připojitelných ke sběrnici.

V tomto zapojení je rozhraní EBIU použito pro připojení paměti SDRAM a pro připojení obvodů realizujících rozhraní USB a Ethernet.

Blokové schéma a signály rozhraní EBIU jsou znázorněny na obr. 3.4 (převzato z manuálu [2]).



Obr. 3.4 Blokové schéma rozhraní EBIU a signály tohoto rozhraní

3.2.3 SDRAM

Paměť SDRAM je připojena k rozhraní EBIU. Principy připojení paměti SDRAM jsou popsány v manuálu [2], kapitola 17 „External Bus Interface Unit“.

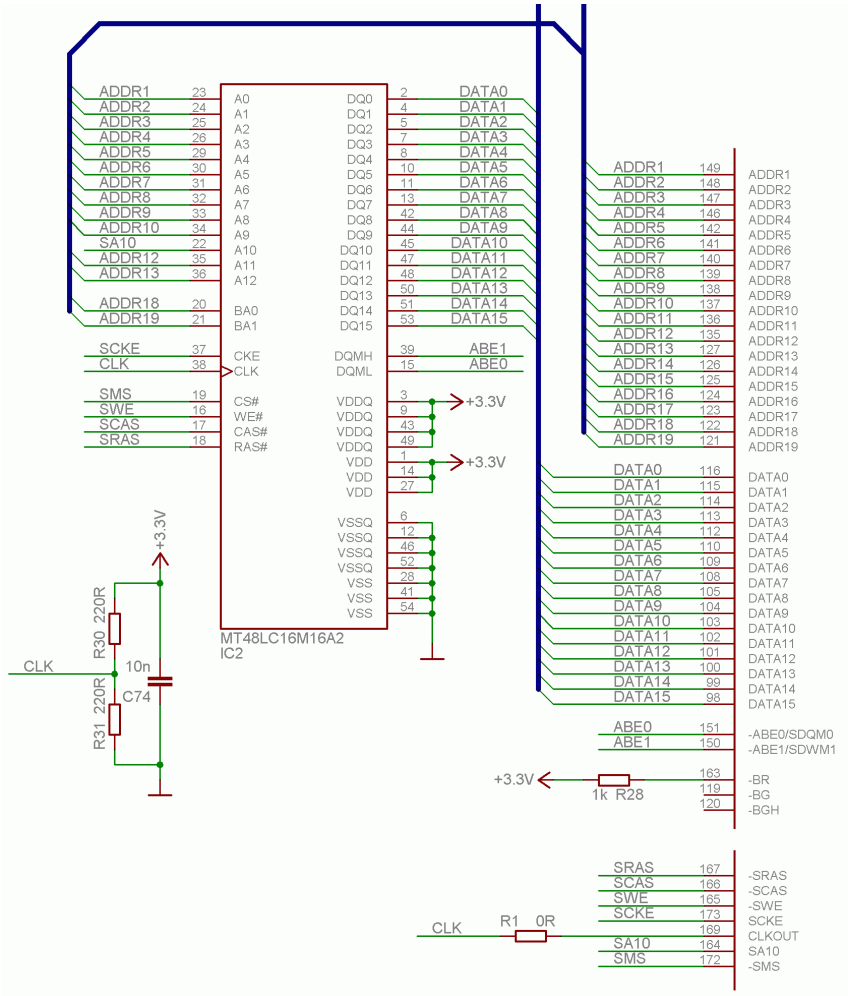
Použitá paměť SDRAM je typu Micron MT48LC16M16A2P-7E. Jde o 32 MB paměť s organizací 4 M × 16 bit × 4 banky. Maximální hodinový kmitočet je 133 MHz při CAS latency 2 cykly, případně 143 MHz při CL 3.

Pro připojení paměti SDRAM jsou využity adresové signály ADDR[1..10, 12..13, 18..19], SA10 a SDQM[0..1] (ABE0, ABE1), datové DATA[0..15], řídící SCKE, SMS, SWE, SCAS, SRAS a hodinový signál CLKOUT.

Adresovací vývod A10 není připojen na ADDR11, což by se na první pohled mohlo zdát logické. Vývod A10 totiž slouží totiž mimo jiné pro spuštění příkazu „Auto-Refresh“. Aby bylo možné zadávat paměti SDRAM řídící příkazy, zatímco probíhá komunikace s asynchronním zařízením připojeným na stejnou sběrnici, je vyveden zvláštní signál SA10, který je určen pro připojení na vývod A10 paměti SDRAM.

Vstupní signály SDQM[0..1] (ABE0, ABE1) slouží pro výběr horní nebo dolní poloviny datového slova v případě, že je nutné zapsat do paměti pouze jeden byte. Při čtení jsou aktivní vždy oba tyto signály.

Propojení procesoru a paměti SDRAM je znázorněno na obr. 3.5:



Obr. 3.5 Připojení paměti SDRAM

Pro impedanční zakončení hodinového signálu CLK bylo použito Theveninovo zapojení pro impedanci 110 Ω (rezistory R30 a R31). Blokovací kondenzátor snižuje impedanci mezi napájením a zemí v daném místě. Alternativně je možné použít sériové přizpůsobení na straně budiče: rezistory R30 a R31 vynechat a osadit vhodnou hodnotu na pozici R1.

Stejný způsob terminace je použitý také pro signály asynchronního čtení a zápisu RE a WE a pro signál IFCLK.

3.2.4 Připojení externího modulu (rozhraní PPI)

Interface externího modulu (konektor X4) umožňuje připojení různých druhů senzorů, kodeků a převodníků. Mechanická podoba a zapojení konektoru je realizováno tak, aby bylo kompatibilní

s CMOS senzory dříve vyvinutými na naší katedře. K interface je možné připojit modul pro digitalizaci videosignálu, popsaný v kapitole 2.

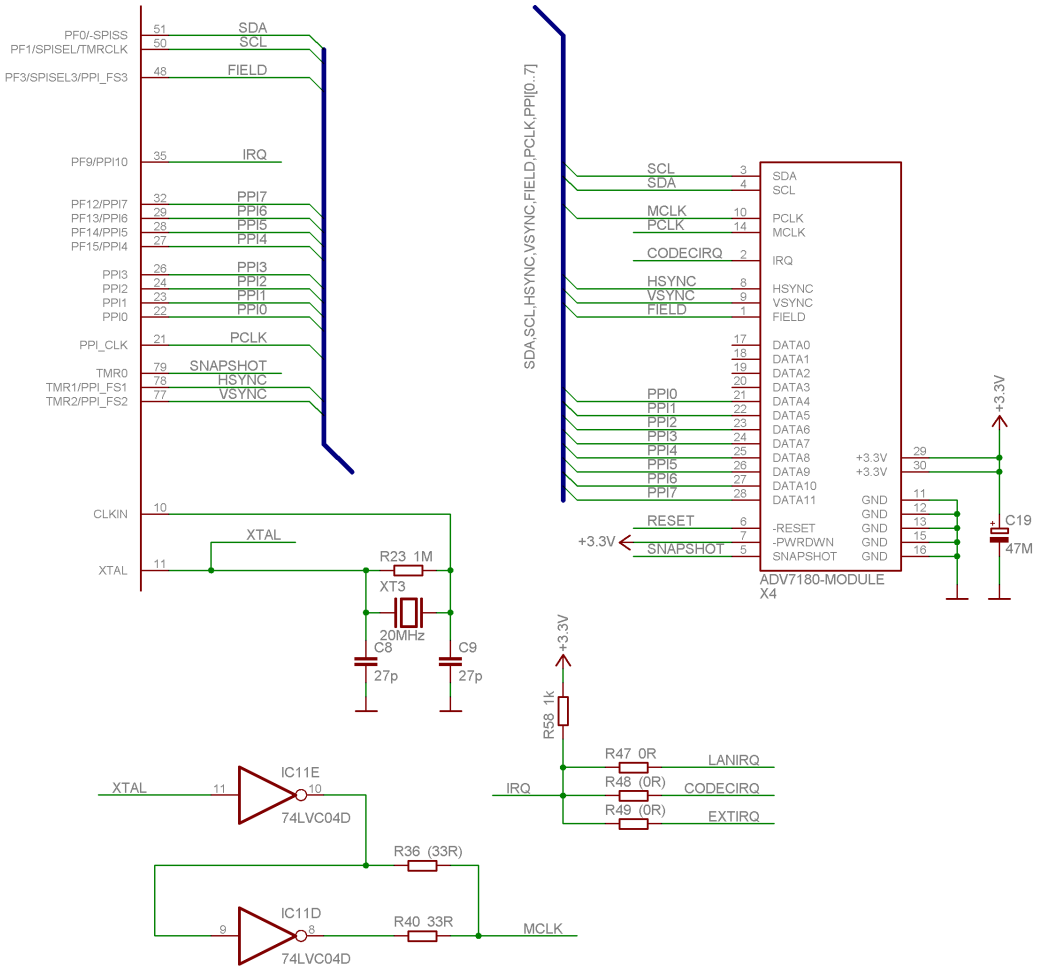
Procesor Blackfin obsahuje rozhraní Parallel Peripheral Interface (PPI), které realizuje rychlý vstup nebo výstup streamovaných dat za pomocí DMA. Šířka přenášeného datového slova je 8–16 bitů. PPI má dedikovaný hodinový vstup, který určuje vzorkování všech signálů a který dokáže zpracovat frekvenci až do SCLK/2. Pro synchronizaci jsou určeny až tři synchronizační vstupy/výstupy. PPI může přijímat nebo odesílat data, přičemž synchronizace může být v obou případech generována interně nebo externě.

Externí modul se připojí na konektor X4, na který jsou vyvedeny následující signály:

- napájení 3,3 V
- 8 bitový kanál PPI
- všechny tři synchronizační signály PPI (HSYNC, VSYNC a FIELD)
- vstupní hodinový signál PPI (PCLK)
- sběrnice I2C pro konfiguraci senzoru/kodeku (SDA, SCL)
- výstupní hodinový signál 20 MHz (MCLK)
- RESET z obvodu resetu
- vstup přerušení IRQ (může být připojen na PF9)
- start pořízení snímku SNAPSHOT, připojený na časovač TMR0

Schéma zapojení interface externího modulu je uvedeno na obr. 3.6. Jedná se o téměř přímočaré propojení mezi konektorem X4 a procesorem. Za zmínku stojí signál CODECIRQ, který je možné připojit na vývod PF9 procesoru IC1 osazením propojky R48. Blíže viz kapitola 3.2.7.

Uvedené zapojení neobsahuje pull-up rezistory pro sběrnici I2C, na které bylo v návrhu zapomenuto. Na každý ze signálů SDA a SCL je třeba zapojit rezistor 2,2 kΩ proti napětí 3,3 V.

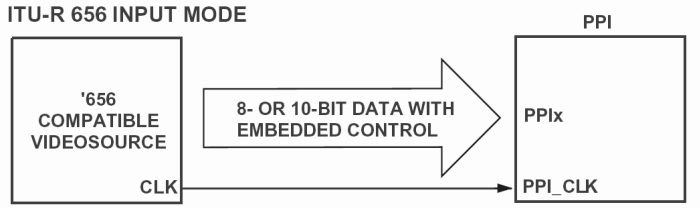


Obr. 3.6 Připojení externího modulu

3.2.4.1 Připojení zdroje signálu podle normy ITU-R 656

Pokud je datový výstup zdroje signálu kompatibilní s normou ITU-R 656, stačí pro připojení k PPI pouze datový tok a hodinový signál bez dalších synchronizačních signálů (viz obr. 3.7, převzatý z manuálu [2]). Synchronizace je zahrnuta v datovém toku a rozhraní PPI dokáže synchronizační impulsy z datového toku automaticky dekódovat a zpracovat.

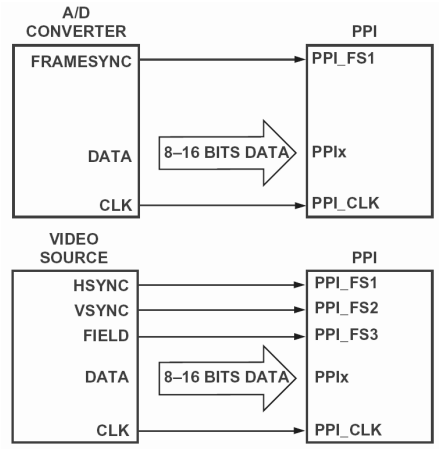
Tento režim se použije typicky pro připojení videodekodérů, např. modulu popsaného v kapitole 2.



Obr. 3.7 Principiální připojení kodeku s výstupem podle normy ITU-R 656

3.2.4.2 Připojení obecného zdroje signálu

Pokud je zdrojem signálu např. CMOS senzor, A/D převodník doplněný obvody pro dekódování synchronizace apod., potom je nutné kromě vlastního datového toku připojit také synchronizační impulsy (PPI_FS1 až PPI_FS3), viz obr. 3.8 (převzatý z manuálu [2]).



Obr. 3.8 Principiální připojení zdroje signálu se se synchronizací

3.2.4.3 Generování signálů pomocí PPI

Přestože v rámci této práce je rozhraní PPI uvažováno především jako vstupní (pro připojení různých druhů obrazových senzorů), je možné ho využít také pro generování obrazového signálu (ve spojení s externím D/A převodníkem nebo kodekem) nebo pro generování obecných signálů (pattern generator, arbitrary function generator).

3.2.5 USB

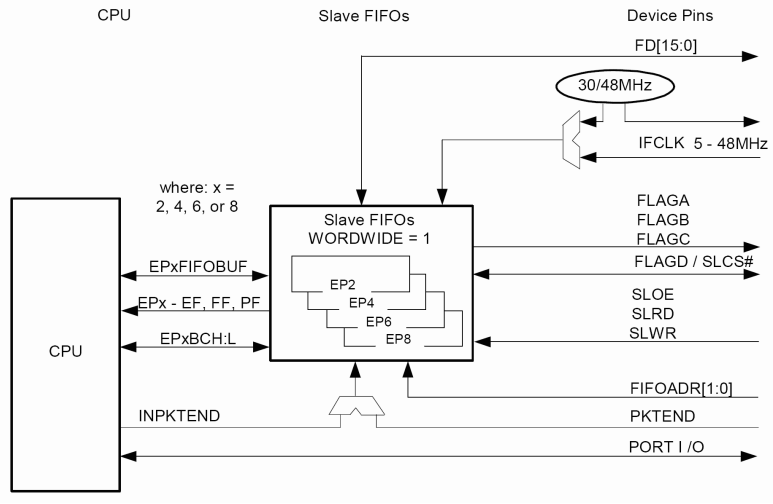
Rozhraní USB je realizováno obvodem Cypress CY7C68013A-100AXC (IC4; pouzdro TQFP-100). Tento obvod byl vybrán zejména z důvodů dobré dostupnosti a předchozích zkušeností s jeho použitím. Obvod obsahuje interface USB 2.0 High-speed (480 Mbit/s) a dále procesor standardu 8051 s rozšířeními.

3.2.5.1 Obvod Cypress CY7C68013A

Napájení obvodu je 3,3 V, vstupy jsou 5 V tolerantní. Jádro x51 může pracovat na frekvenci 48, 24 nebo 12 MHz, přičemž základní instrukční cyklus trvá 4 cykly. Základní hodinový signál je ve všech případech generovaný krystalem o frekvenci 24 MHz. K dispozici jsou tři časovače, dvě sériová rozhraní UART a řadič sběrnice I2C. Mezi další rozšíření patří dva registry DPTR a paměť RAM o velikosti 16 KB, která může být nahrána přes rozhraní USB nebo z externí I2C paměti EEPROM. Další rozšíření jsou specifická pro práci s USB – konfigurační registry, práce s deskriptory, rozšířený přerušovací systém, datové buffery s ukazateli s automatickou inkrementací a další.

Pro komunikaci přes USB je možno využít celkem čtyři konfigurovatelné endpointy. Každý

z nich je možné provozovat v režimech Bulk, Interrupt nebo Isochronous. Z hlediska externího připojení se obvod chová jako FIFO s asynchronním nebo synchronním rozhraním se šířkou datové sběrnice 8 nebo 16 bitů. Vnitřně mohou být data ukládána do dvou- až čtyřúrovňové vyrovnávací paměti o velikosti bloků 512 nebo 1024 Byte. K dispozici je navíc ještě jeden programovatelný Bulk/Interrupt endpoint s pevnou velikostí bloku 64 Byte. Blokové schéma rozhraní FIFO je uvedeno na obr. 3.9 (převzato z manuálu [17]).



Obr. 3.9 Rozhraní FIFO (Cypress USB)

Obvod umožňuje také vytváření speciálních rozhraní (např. ATA pro připojení pevných disků IDE) pomocí bloku GPIF (General Programmable Interface). Tento nabízí poměrně široké možnosti komunikace s dalším zařízením pomocí řídících (CTL – Control) a stavových (RDY – Ready) signálů, včetně generování adresy o šířce 8 bitů. Komunikaci řeší vnitřní stavový automat na základě definování průběhů řídících signálů (pomocí tzv. „waveform descriptors“) v reakci na stavové signály a vnitřní stavy obvodu. Není předpokládáno, že by se rozhraní GPIF využilo v tomto modulu, nicméně bylo by možné zvážit využití GPIF pro synchronní komunikaci s nadřazeným procesorem Blackfin.

3.2.5.2 Připojení rozhraní USB k procesoru Blackfin

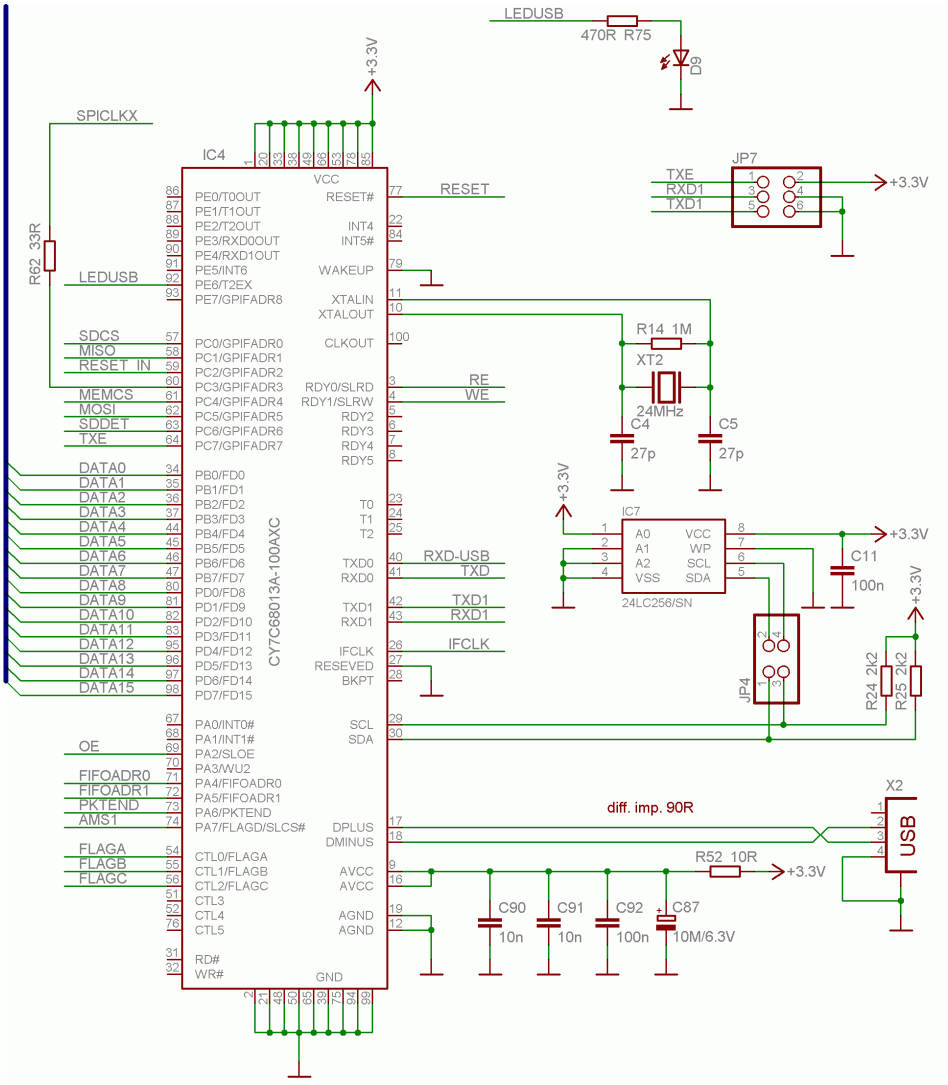
Schéma bloku rozhraní USB je uvedeno na obr. 3.10.

Brána FIFO obvodu IC4 je připojena na asynchronní banku 1 procesoru Blackfin (IC1; výběrový signál AMS1). Šířka datové sběrnice je 16 bitů, adresová sběrnice není k IC4 připojena. Řídící signály jsou WE, RE, OE a AMS1. Použitý režim komunikace s FIFO je uvažován jako asynchronní. Problémem asynchronní komunikace s tímto obvodem je ovšem omezená šířka pásma (teoretické maximum je cca 15,9 MB/s). Pro případné experimenty s využitím synchronního režimu FIFO je možné alternativně osazením R37 a R70 připojit signál CLKOUT (SCLK) obvodu IC1 na synchronní hodinový signál IFCLK obvodu IC4. V tomto případě je ovšem nutné adekvátně snížit frekvenci SCLK tak, aby byla v mezích specifikací obvodu IC4.

Dále jsou k IC1 na rozhraní PF připojeny signály FLAGA, FLAGB a FLAGC, což jsou programovatelné stavové signály, které mohou typicky signalizovat zaplněnost vstupního a výstupního FIFO bufferu. Pro okamžité odeslání paketu, který je menší než velikost bufferu slouží signál PKTEND. Signály FIFOADR0 a FIFOADR1 slouží pro volbu endpointu a jsou taktéž připojeny na piny PF.

Obvod IC4 má k dispozici dva sériové kanály UART. Kanál 1 je vyveden na konektor JP7 společně s jedním I/O pinem, který může sloužit např. pro ovládání vysílání sběrnice RS-485. Kanál 0 je možné využít pro „tunelování“ sériového rozhraní přes USB do PC – zapojení viz kapitola 3.2.8.

K obvodu je připojena přes rozhraní I2C paměť EEPROM (IC7). Paměť je oddělena jumpery vloženými do konektoru JP7. Je tak možné ji odpojit a např. naprogramovat externím programátorem.



Obr. 3.10 Blok rozhraní USB

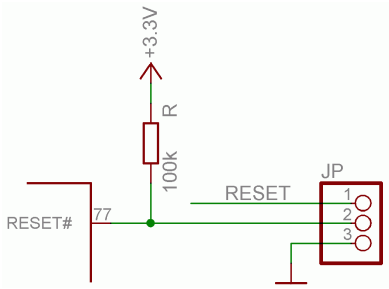
Na vývod PE6 je připojena LED pro indikaci vnitřních stavů běžícího programu.

Signál RESET je generovaný obvodem resetu IC12.

3.2.5.3 Možnost programování SPI Flash, přístup k MMC/SD kartě

Při návrhu zapojení byla zvažována možnost přístupu k programové paměti procesoru Blackfin, SPI flash IC6. Důvodem je možnost programování paměti bez nutnosti použít externí programátor. Za tímto účelem byla na PORT C připojena celá SPI sběrnice, včetně signálů pro výběr paměti flash a pro výběr a detekci přítomnosti MMC/SD karty. Dále je zde možnost ovládání signálu RESET, protože za normálního chodu procesoru IC1 není možné sběrnici SPI externě používat. Hodinový signál SCK (označený jako SPICLKX) je kvůli impedančnímu přizpůsobení oddělen rezistorem R62 v blízkosti obvodu IC4.

V realizovaném zapojení je přítomna logická chyba – signál RESET je společný pro procesor Blackfin (IC1) i Cypress (IC4). Proto v okamžiku, kdy obvod Cypress aktivuje reset, dojde k resetu i jeho samotného. Zapojení je nutné upravit tak, aby byl signál pro obvod IC4 oddělen jumperem, tak aby bylo možné uvést do resetu všechny ostatní obvody kromě IC4. Upravené zapojení je uvedeno na obr. 3.11.



Obr. 3.11 Upravené zapojení resetu IC4

3.2.5.4 Problém s komunikací v High-speed režimu

Při testech komunikace se objevil problém s funkcí USB 2.0 High-speed režimu. Enumerace proběhla pouze v režimu Full-speed, přičemž operační systém vypsal chybové hlášení, že zařízení by mohlo pracovat v High-speed režimu, nicméně nepracuje.

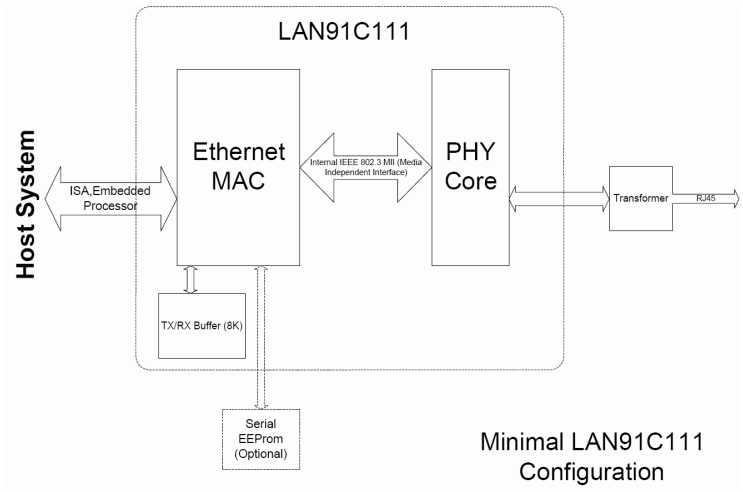
Ačkoliv nebyla prokázána přesná příčina tohoto problému, je pravděpodobné, že chyba v komunikaci je způsobena impedančním nepřizpůsobením vedení mezi obvodem IC4 a konektorem USB X2. Parametry tohoto vedení byly navrženy s ohledem na určitou tloušťku substrátu (která byla uvedena v technických podmínkách výrobce DPS), ovšem vyrobený vzorek měl tloušťku jednotlivých vrstev substrátu odlišnou od předpokládané. Důsledkem toho je rapidní snížení impedance všech signálů vedených na DPS – jediná komplikace se ovšem objevila u rozhraní USB. V další revizi desky je nutné vedení s definovanou impedancí po konzultaci s výrobcem DPS přepracovat.

3.2.6 Ethernet

Rozhraní Ethernet je realizováno obvodem SMSC LAN91C111-NU (IC3) v pouzdru TQFP-128. Rozteč pinů pouzdra je 0,4 mm.

3.2.6.1 Obvod SMSC LAN91C111

Jedná se o řadič 10/100 Mbps Ethernetu, který v sobě integruje linkovou vrstvu MAC i fyzickou vrstvu PHY. Alternativně je možné využít standardní rozhraní MII (Media Independent Interface) pro připojení externího obvodu fyzické vrstvy. Externě je potřeba pouze transformátor integrovaný s konektorem RJ-45 a několik málo diskrétních součástek. Napájení obvodu je 3,3 V, vstupy jsou 5 V tolerantní. Hodinový signál je generovaný krystalem o frekvenci 25 MHz.



Obr. 3.12 Vnitřní koncepce obvodu SMSC LAN91C111

Základní bloky vnitřního zapojení obvodu jsou znázorněny na obr. 3.12 (převzato z datasheetu [15]). Odesílané a přijímané pakety jsou ukládány do interní vyrovnávací paměti FIFO o celkové velikosti 8 KB. Připojení k nadřazenému systému může být realizováno po 8, 16 nebo 32 bitové sběrnici. Konfigurace obvodu (bázová adresa, MAC adresa a další nastavení) může být uložena v sériové paměti EEPROM.

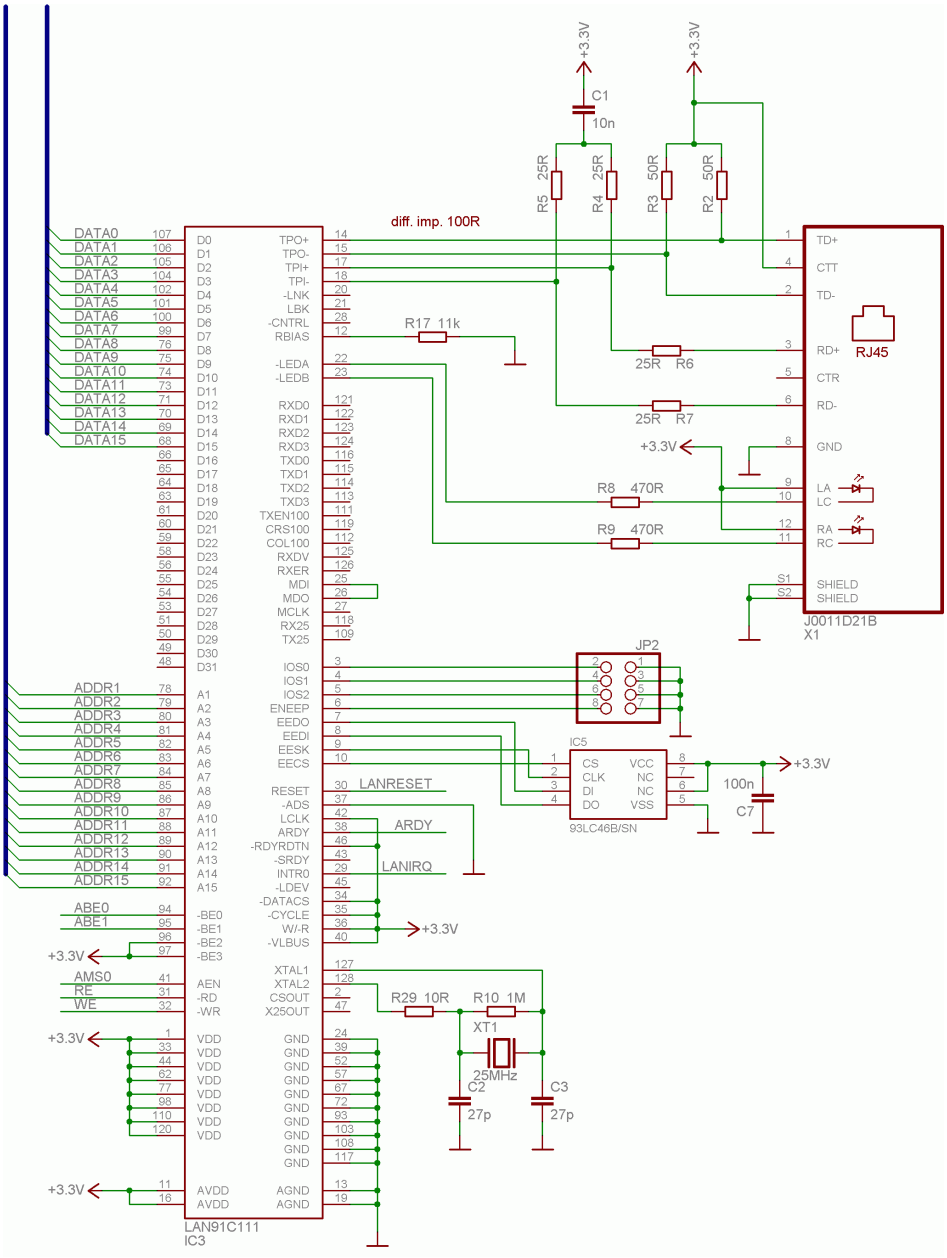
3.2.6.2 Připojení k procesoru Blackfin

Schéma zapojení bloku Ethernet je uvedeno na obr. 3.13.

Rozhraní Ethernetu (IC3) je připojeno k asynchronní bance 0 procesoru IC1. Šířka datové sběrnice je 16 bitů, šířka adresové sběrnice je 15 bitů, ovšem nejmenší adresovatelnou jednotkou je WORD. Proto je adresa doplněná o signály ABE0 a ABE1, kterými je možné zvolit zápis pouze do jedné poloviny slova. Řídící signály čtení a zápisu jsou RE, WE a AMS0. Dále je k procesoru přiveden signál ARDY, který se ovšem ve standardní konfiguraci nepoužívá. Pokud by jeho použití bylo nutné, stačí povolit zpracování signálu v konfiguračních registrech EBIU. Konektor JP2 je určen pro vložení zkratovacích propojek (jumperů) pro nastavení některých vlastností obvodu. Bližší podrobnosti viz datasheet [15].

Dále je k procesoru připojen signál přerušení LANIRQ. Tento signál je aktivní v úrovni log. 1. Bližší údaje o připojení signálu k procesoru IC1 jsou uvedeny v kapitole 3.2.7.

Signál RESET je generován obvodem resetu IC12 a dále invertován hradlem IC11B.



Obr. 3.13 Blok rozhraní Ethernet

3.2.7 Signál IRQ

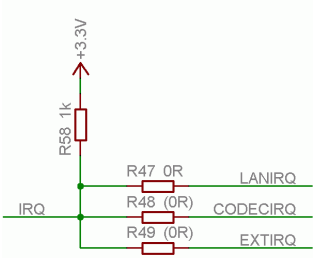
Signál IRQ (vývod PF9 procesoru IC1) slouží ke generování přerušení. Z důvodu nedostatku PF pinů na procesoru bylo nutné přistoupit ke kompromisnímu připojení několika signálů pomocí rezistorů R47, R48 a R49 - zapojení viz obr. 3.14. Tímto způsobem jsou sloučeny signály LANIRQ od Ethernetu, CODECIRQ od externího modulu digitalizace a EXTIRQ od I2C expandéru IC14.

Signál LANIRQ je aktivní v log. 1 a v klidu je trvale v log.0. Vlastnosti signálu CODECIRQ závisí na připojeném modulu. U modulu s ADV7180 je možné nastavit výstup aktivní jen v log. 0 (open-drain; pokud je signál neaktivní, výstup je ve stavu vysoké impedance) a výstup v obou

úrovních, přičemž aktivní může být v log. 0 nebo log. 1 (viz. kapitola 2.3.3). Signál EXTIRQ je aktivní v log. 0 (open-drain).

Běžně se osadí jen jedna z propojek R47, R48 a R49. Standardně je vývod PF9 připojený na signál přerušení z rozhraní Ethernetu (LANIRQ, rezistor R47).

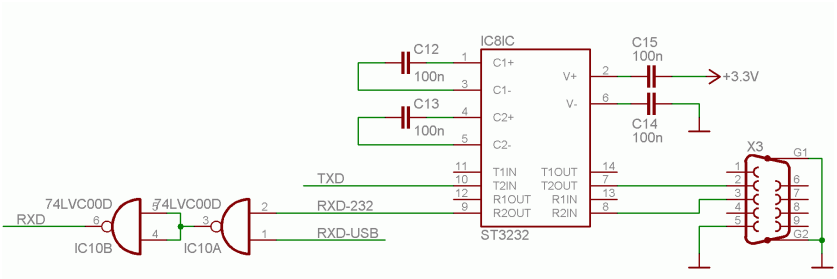
Je ale také možné např. připojit signály CODECIRQ a EXTIRQ tím způsobem, že se neosadí pull-up R58, na pozici R49 se osadí propojka a na pozici R48 se osadí rezistor 1 kΩ. Zároveň je samozřejmě nutné nakonfigurovat externí modul tak, aby budil signál CODECIRQ v obou úrovních, aktivní v log. 0.



Obr. 3.14 Sloučení signálů přerušení

3.2.8 UART (USB, RS-232)

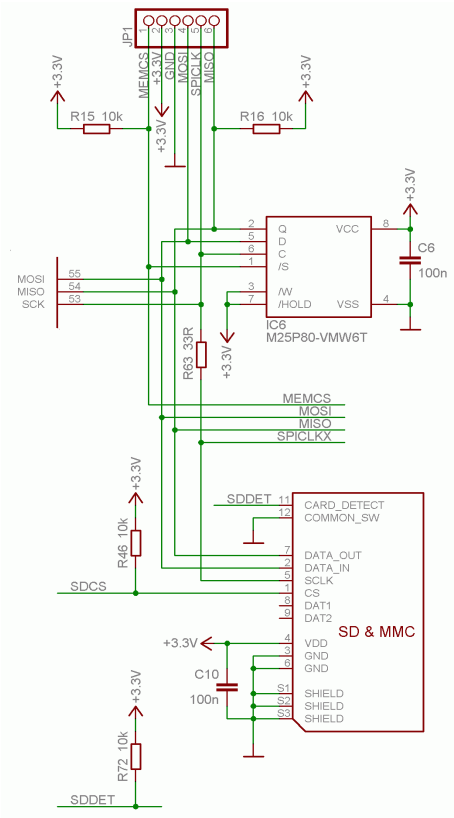
Rozhraní UART procesoru IC1 je převedeno převodníkem úrovní IC8 na RS-232 a vyvedeno na konektor X3. Zároveň je rozhraní UART IC1 přivedeno na UART 0 IC4. Příjem z obou rozhraní je sloučen hradly IC10A, IC10B. Schéma zapojení bloku UART je na obr. 3.15.



Obr. 3.15 Rozhraní UART (RS-232)

3.2.9 SPI, paměť flash, MMC/SD

Rozhraní SPI procesoru IC1 se využívá pro dvě základní funkce: připojení paměti flash, ve které je uložen kód programu, a dále připojení MMC/SD karty, která může být využívána k rozličným účelům. Sběrnice SPI je tvořena signály MISO, MOSI, SPICLK a výběrovými signály MEMCS a SDCS. Připojení k procesoru IC1 viz obr. 3.16.



Obr. 3.16 Rozhraní SPI, flash paměť, MMC/SD karta

Použitá paměť flash je M25P32 výrobce ST o kapacitě 4 MB, přičemž je možné použít kteroukoliv jinou z řady M25Pxx. Footprint na desce je typu SO8W, ovšem podle doporučení výrobce je možné osadit na tento footprint i paměti v pouzdru QFN. Paměť je možné zapisovat po stránkách o velikosti až 256 B, mazání je možné po stránkách o velikosti 64 KB. Maximální frekvence hodinového signálu je u paměti M25P32 až 75 MHz. Výběrový signál pro paměť flash je signál MEMCS připojený na vývod PF2 procesoru. Celé rozhraní SPI i signál MEMCS jsou vyvedeny na konektor JP1 pro připojení externího programátoru.

Karta MMC resp. SD pracuje v režimu SPI. Oba typy karet se vyrábějí v maximální kapacitě 4 GB. Maximální hodinový kmitočet je 20 MHz u konvenčních MMC karet a 25 MHz u karet SD. Výběrový signál pro kartu MMC/SD je SDCS připojený na vývod PF4. Hodinový signál pro MMC/SD kartu je oddělen rezistorem R63, který realizuje impedanční přizpůsobení. Vzhledem k nedostatku volných I/O pinů na IC1 je signál pro detekci vložení karty (SDDET) připojen na vývod DR1PRI (SPORT 1).

Dále je kompletní sběrnice SPI včetně výběrových signálů a signálu pro detekci karty připojena na obvod USB rozhraní IC4. Je tak možné obě paměti programovat z tohoto obvodu.

3.2.10 SPORT

Procesor BF532 má k dispozici dvě dvoukanálová synchronní sériová rozhraní (SPORT 0,

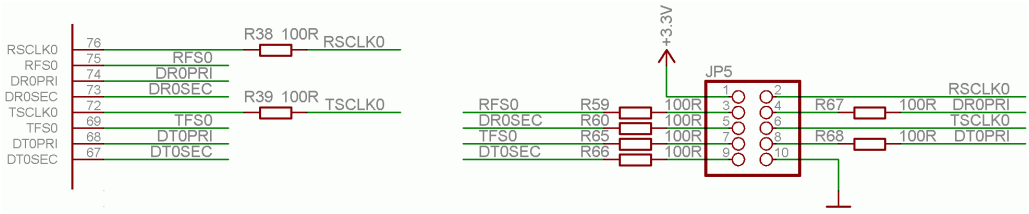
SPORT 1), která umožňují připojit různé druhy periferií, jako jsou především A/D a D/A převodníky a kodeky. Rozhraní mohou být také využita např. pro meziprocesorovou komunikaci ve víceprocesorovém systému, pro komunikaci s obvody FPGA a CPLD nebo pro připojení sériových registrů pro rozšíření možností I/O.

Každý SPORT má dvě skupiny pinů – pro příjem a vysílání. V každé skupině jsou dva datové signály (primární a sekundární), hodinový signál (může být generován interně nebo externě) a pin pro rámcovou synchronizaci (frame sync; tento signál může být také generovaný interně nebo externě). Příjem i vysílání jsou nezávislé a komunikace tedy může být plně duplexní.

SPORT je možné také využít pro emulaci rozhraní UART.

3.2.10.1 SPORT 0

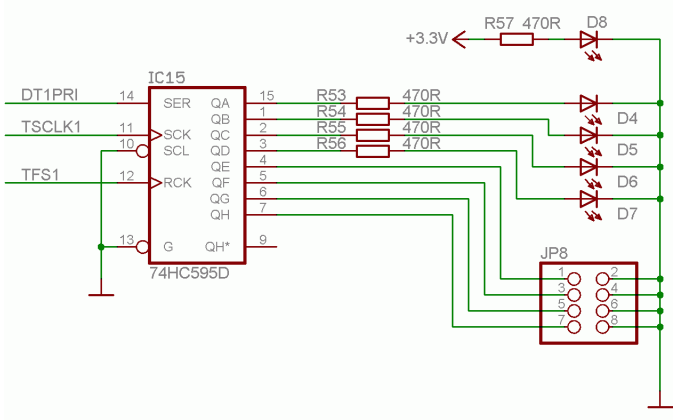
Všechny signály rozhraní SPORT 0 jsou vyvedeny na konektor JP5. Impedanční přizpůsobení hodinových signálů (RSCLK0, TSCLK0) je realizováno rezistory R38 a R39, které jsou umístěny blízko procesoru IC1. Ostatní signály jsou odděleny ochrannými rezistory.



Obr. 3.17 Rozhraní SPORT0

3.2.10.2 SPORT 1 (LED a výstupy)

Rozhraní SPORT 1 je využito pro připojení sériového registru IC15. Na jeho výstupy jsou připojeny čtyři LED pro indikaci libovolných stavů a zbývající čtyři signály jsou vyvedeny na konektor JP8. Schéma zapojení je uvedeno na obr. 3.18. V realizovaném zapojení je vývod SCL omylem připojen na zem, správně má být trvale připojen na napájecí napětí 3,3 V.

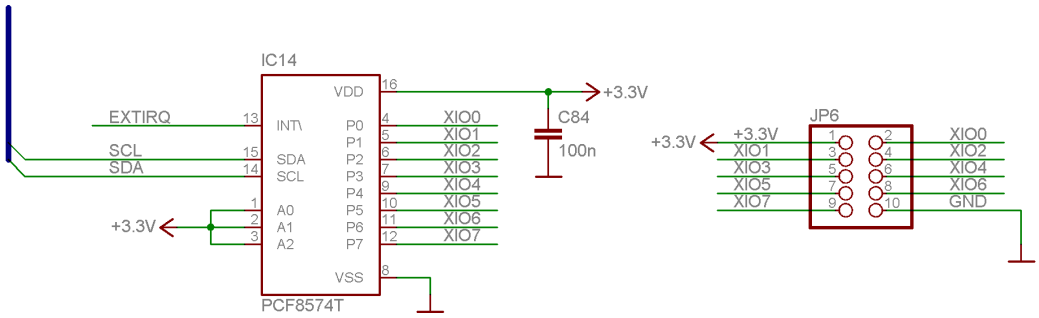


Obr. 3.18 Indikační LED a digitální výstupy

3.2.11 I2C (externí I/O)

Sběrnice I2C je připojena kromě rozhraní externího modulu také na expandér IC14. Tento obvod typu PCF8574 resp. PCF8574A slouží pro vytvoření univerzálního 8 bitového vstupně-výstupního rozhraní. Jednotlivé I/O piny obvodu IC14 jsou typu open-drain s integrovaným pull-upem (o typické velikosti 100µA), tedy obdobně jako u procesorů řady x51. Zapojení expandéru je na obr. 3.19.

Obvod má k dispozici také výstup přerušení typu open-drain. Výstup je aktivní v log. 0 po každé změně na vstupu obvodu. Po přečtení stavu po sběrnici I2C se výstup přerušení deaktivuje.



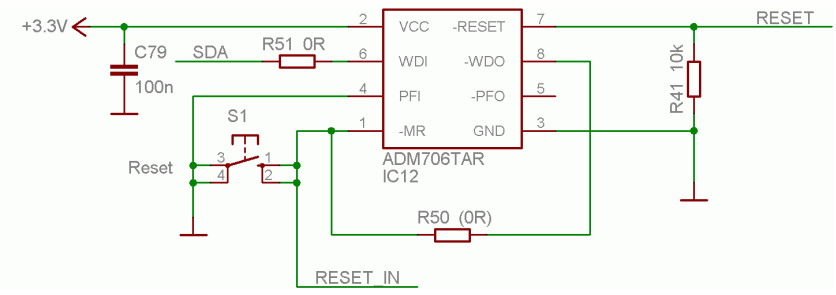
Obr. 3.19 I/O přes I2C Expandér

Komunikační adresa je 0x4E při použití PCF8574 resp. 0x7E při použití PCF8574A (kromě komunikační adresy jsou obvody identické).

3.2.12 Reset

Obvod pro generování resetu (IC12) je typu ADM706TAR. Výstupní signál RESET je aktivní v log. 0 a je připojen na obvody IC1 (Blackfin), IC3 (Ethernet; přes invertor IC11B), IC4 (USB) a dále je vyveden na konektor externího modulu X4. Vstup obvodu je připojen na mikrospínač S1, dále k obvodu IC4 (signál RESET_IN) a na konektor JP10, obojí kvůli možnosti zastavení hlavního procesoru při programování paměti flash.

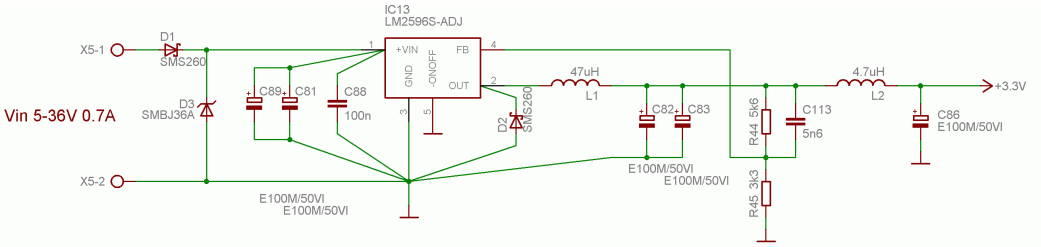
Obvod IC12 realizuje také funkci watchdogu. Vstup je možné připojit na signál SDA osazením rezistoru R51, výstup watchdogu aktivujeme osazením rezistoru R50. Další funkce obvodu, hlídání podpětí, nebyla v zapojení využita.



Obr. 3.20 Obvod generování signálu reset

3.2.13 Napájení

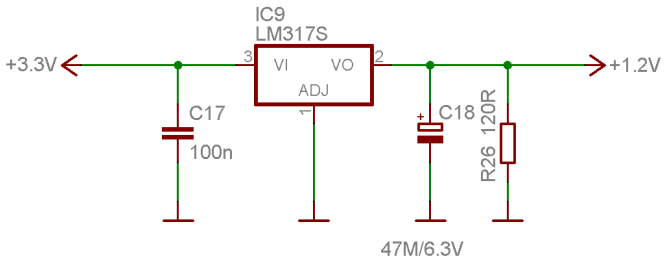
Napájecí zdroj desky je realizován spínaným stabilizátorem LM2596-ADJ (IC13). Zapojení stabilizátoru vychází z katalogového zapojení uvedeného v katalogovém listu obvodu, hodnoty součástek byly optimalizovány konfiguračním programem od výrobce obvodu. Obvod dodává hlavní napájecí napětí 3,3 V. Schéma zapojení je uvedeno na obr. 3.21.



Obr. 3.21 Spínaný zdroj napájení 3,3 V

Výstupní napětí je definováno děličem R44 a R45. Kondenzátor C113 zvyšuje stabilitu zapojení. Výstupní napětí je dále filtrováno LC členem tvořeným L2 a C86 pro zmenšení zvlnění. Vstup je chráněn diodou D1 proti přepólování vstupního napětí a dále transilem D3 proti přepětí. Obvod je dimenzován na rozsah vstupního napětí 5-36 V. Odběr proudu z výstupu 3,3 V je až 800 mA. Vstupní proud je až cca 700 mA při minimálním vstupním napětí. Typický celkový odběr desky je cca 3 W.

Dále je na desce přítomen stabilizátor LM317 (IC9), který stabilizuje napětí 1,2 V pro jádro procesoru IC1. Předpokládaný odběr je cca 200 mA.



Obr. 3.22 Lineární zdroj napětí 1,2 V pro jádro procesoru

3.2.14 Chyby a možná vylepšení

Při návrhu zapojení vzniklo několik chyb, z nichž některé již byly zmíněny v popisu zapojení. Zde jsou rekapitulovány všechny problémy, které byly dosud zjištěny:

- na sběrnici I2C procesoru IC1 chybí rezistory pull-up o hodnotě 2,2 kΩ
- vývod SCL obvodu IC15 je omylem připojen na zem, má být připojen na 3,3 V
- vývod RESET obvodu IC4 je třeba oddělit jumperem (viz kapitola 3.2.5.3)
- parametry vedení s definovanou impedancí neodpovídají použité tloušťce substrátu

Dále by bylo vhodné udělat v případné další verzi desky některé úpravy. Jedná se zejména o tato možná vylepšení a modifikace:

- doplnit krystal RTC a zálohování napájení $V_{dd,RTC}$
- zvážit možnost vyvedení jednoho nebo několika výběrových signálů společně s již vyvedenou sběrnicí SPI – bylo by tak možné připojit k této sběrnici externě některé periferie, jako jsou např. A/D a D/A převodníky apod.
- upravit zapojení, které slučuje signály přerušení z různých zdrojů tak, aby bylo možné využívat více zdrojů přerušení současně, případně pohodlně konfigurovat výběr zdroje přerušení
- doplnit pull-up rezistor na signál PKTEND a pull-down rezistory na signály FIFOADR0 a FIFOADR1, aby na nich byla za všech okolností definovaná úroveň

3.3 Konfigurace procesoru a periferií

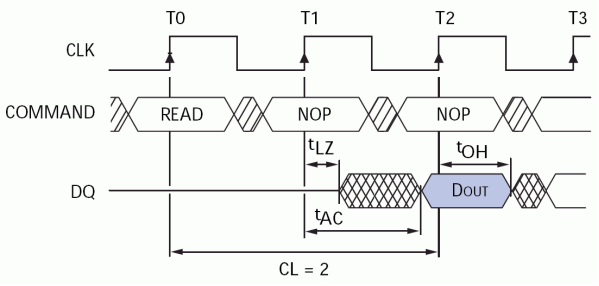
Procesor Blackfin je třeba nakonfigurovat tak, aby mohl spolupracovat se základními připojenými periferiemi. Klíčová je zejména paměť SDRAM, dále bude popsáno nastavení potřebné pro optimální komunikaci s rozhraním Ethernetu a USB.

V dalším textu předpokládáme nastavení samotného procesoru na maximální výkon. To pro použitý typ procesoru odpovídá frekvenci jádra (CCLK) 400 MHz a frekvenci taktování systémových periferií (SCLK) 133 MHz.

3.3.1 Konfigurace SDRAM

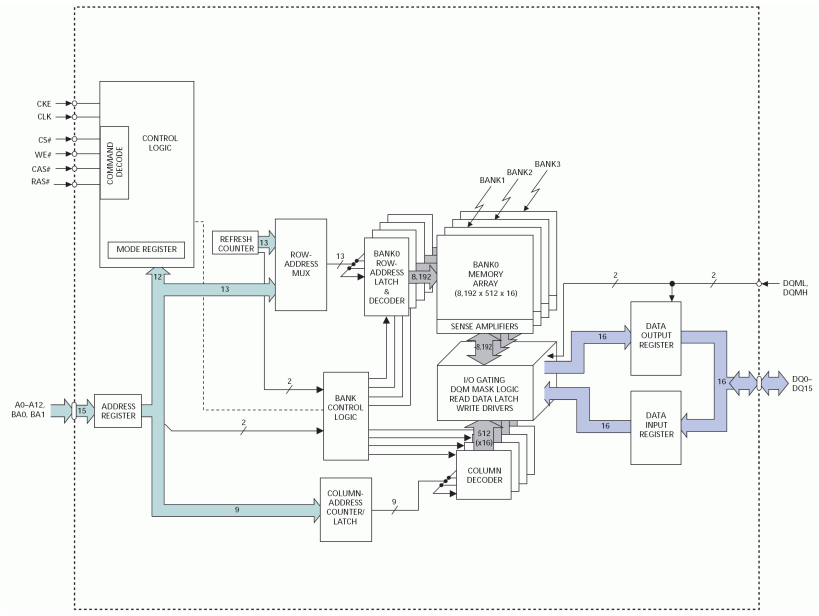
Použitá paměť SDRAM je má kapacitu 32 MB při organizaci 4 M × 16 bit × 4 banky. Šířka datové sběrnice je tedy 16 bitů. Její rozhraní odpovídá standardům PC-100 a PC-133. S těmito standardy pamětí SDRAM umí rozhraní EBIU procesoru Blackfin spolupracovat.

Maximální frekvence hodinového signálu, se kterým může paměť pracovat je 133 MHz při CAS latency 2 cykly a 143 MHz při CAS latency 3 cykly. CAS latency (CL; viz obr. 3.23 převzatý z datasheetu [14]) je počet cyklů mezi přijetím příkazu ke čtení a výstupem platných dat a je to jeden ze základních parametrů pamětí SDRAM.



Obr. 3.23 CAS Latency

Vnitřní organizace obvodu je znázorněna na blokovém schématu na obr. 3.24 (převzato opět z datasheetu [14]).



Obr. 3.24 Vnitřní blokové schéma paměti MT48LC16M16A2

Rozhraní EBIU, přes které je paměť SDRAM připojena, je konfigurováno několika registry, jejichž parametry je nutné pro správnou funkci nastavit. Pro konfiguraci připojení paměti SDRAM jsou určeny tyto registry:

- Global Control Register (EBIU_SDGCTL)
- Bank Control Register (EBIU_SDBCTL)
- Refresh Rate Control Register (EBIU_SDRRC)

Dále je k dispozici „Control Status Register“ (EBIU_SDSTAT), který obsahuje stavové bity informující o činnosti.

Základní parametry časování, které se nastavují v registru EBIU_SDGCTL, jsou shrnuty v tab. 3.1. V tabulce jsou také uvedeny minimální hodnoty, které udává výrobce v datasheetu. Nastavení všech registrů je udáno pro frekvenci SCLK 133 MHz.

Tab. 3.1 Základní parametry časování paměti SDRAM a nastavení rozhraní EBIU pro SCLK 133 MHz

hodnota	rozsah nastavení [SCLK cycles]	min. hodnota z DS [ns]	min. cyklů (přesně) [SCLK cycles]	nastavení EBIU [SCLK cycles]
CL	2, 3		2	**2**
t_{RAS}	1-15	*37*	4,92	**5**
t_{RP}	1-7	*15*	2	**2**
t_{RCD}	1-7	*15*	2	**2**
t_{WR}	1-3	*8*	1,06	**2**

Registr EBIU_SDBCTL obsahuje nastavení velikosti paměti (možné hodnoty jsou 16, 32, 64 a 128 MB) a dále šířku adresy sloupce („column address width“, CAW; 8, 9, 10 nebo 11 bitů). Pro použitou paměť je velikost 32 MB a CAW 9 bitů.

Registr EBIU_SDRRC slouží pro nastavení periody pro obnovování obsahu paměti (refresh). Pro výpočet hodnoty RDIV (nachází se v uvedeném registru) je v manuálu [2] uveden vzorec:

$$RDIV = \frac{f_{SCLK} \cdot t_{REF}}{NRA} - (t_{RAS} + t_{RP})$$

kde:

- f_{SCLK} je frekvence signálu SCLK, tedy v našem případě 133 MHz
- t_{REF} je maximální perioda obnovování – pro použitou paměť je to 64 ms
- NRA („Number of Row Addresses“) – počet řádků paměti: 8192
- tRAS, tRP jsou výše uvedené parametry časování (v cyklech SCLK): hodnoty 5 a 2

Registr EBIU_SDRRC je tedy pro daný typ paměti a SCLK 133 MHz nutné nastavit na hodnotu:

$$RDIV = \frac{133\,\text{MHz} \cdot 64\,000\,\mu\text{s}}{8192} - (5+2) = 1032$$

Následující úsek zdrojového kódu ukazuje konkrétní způsob inicializace registrů EBIU pro použitou paměť SDRAM:

```
void sdram_init(void)
{
    *pEBIU_SDBCTL = EBE | EBSZ_32 | EBCAW_9;
    *pEBIU_SDRRC = 1032;
    *pEBIU_SDGCTL = SCTLE | CL_2 | PASR_ALL |
        TRAS_5 | TRP_2 | TRCD_2 | TWR_2 | PSS;

    *((unsigned char *)0) = 0;       // do initialization
}
```

Poslední řádek (zápis do paměti) je přítomen proto, že inicializace je ve skutečnosti provedena až při prvním přístupu do příslušného paměťového prostoru. Tento řádek tedy spustí proces inicializace.

3.3.1.1 Ověření funkce SDRAM

S uvedeným nastavením byla otestována práce s pamětí. Paměť pracovala bezchybně, nebyl identifikován žádný problém s integritou dat. Změřená rychlost sekvenčního zápisu byla cca 240 MB/s, což poměrně odpovídá limitní hodnotě 266 MB/s. Rychlost sekvenčního čtení (jednoduché čtení ve smyčce) byla ovšem pouze cca 53 MB/s.

Pro testování rychlosti přístupu do paměti byl použitý následující zdrojový kód:

```
void test_sdram(void)
{
    int i;
    volatile unsigned int * r = 0;

    for (i = 0; i < SDRAM_SIZE / sizeof(int); i++)
        *r++;     // *r++ = j; při testování zápisu
}
```

Smyčka je kompilátorem přeložena jako tzv. „hardware loop“, která má nulový overhead, takže zpomalení nutně vzniká při přístupu do paměti. Tato překvapivě nízká rychlost je ovšem za daných okolností v pořádku, vzhledem k tomu, že nebyla použita vyrovnávací paměť (cache). Pro čtení každého slova (v tomto případě o šířce 32 bitů) totiž řadič paměti znovu inicializuje čtení z určené adresy, i když jde o sekvenční operaci. Zahájení čtení včetně přečtení prvního 16 bitového slova trvá 9 cyklů SCLK ($t_{RAS} + t_{RCD} + CL$), další slovo je již přečteno sekvenčně v nejbližším cyklu. Pro přečtení 32 bitového slova je tedy potřeba celkem 10 cyklů SCLK, což přesně odpovídá změřené hodnotě 53 MB/s.

V případě, že by čtení probíhalo po menších datových slovech, bude bilance ještě horší – např. pro přečtení dvou 16 bitových slov je potřeba celkem 18 cyklů SCLK (každé slovo je přečteno samostatně v 9 cyklech).

Možná řešení tohoto problému jsou v principu dvě:

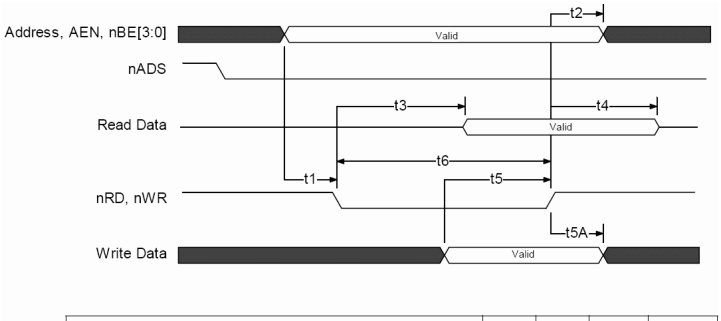
- nakonfigurovat použití vyrovnávací paměti (cache)
- použít pro přenos dat Memory-to-Memory DMA

V obou případech je vlastní čtení z paměti SDRAM prováděno v sekvenčním režimu s minimálním overheadem, a je tak možné se při vhodném použití těsně přiblížit k limitní hranici 266 MB/s.

3.3.2 Konfigurace přístupu k řadiči Ethernetu

Pro maximální využití dostupné šířky pásma při práci s periférierni připojenými k asynchronním bankám je třeba nastavit hodnoty přístupových časů v registrech EBIU na co nejmenší hodnoty, zároveň je ale nutné dodržet minimální časy uvedené v katalogovém listu příslušných obvodů.

Časování asynchronního čtení a zápisu pro obvod SMSC LAN91C111 je znázorněno na obr. 3.25 níže (převzato z datasheetu [15]).



	PARAMETER	MIN	TYP	MAX	UNITS
t1	A1-A15, AEN, nBE[3:0] Valid to nRD, nWR Active	2			ns
t2	A1-A15, AEN, nBE[3:0] Hold After nRD, nWR Inactive (Assuming nADS Tied Low)	5			ns
t3	nRD Low to Valid Data			15	ns
t4	nRD High to Data Invalid	2		15	ns
t5	Data Setup to nWR Inactive	10			ns
t5A	Data Hold After nWR Inactive	5			ns
t6	nRD Strobe Width	15			ns

Obr. 3.25 Časování asynchronního čtení a zápisu LAN91C111

Pro konfiguraci asynchronních bank řadiče EBIU slouží tyto tři registry:

- Asynchronous Memory Global Control register (EBIU_AMGCTL)
- Asynchronous Memory Bank Control 0 register (EBIU_AMBCTL0)
- Asynchronous Memory Bank Control 1 register (EBIU_AMBCTL1)

Registr EBIU_AMGCTL slouží pro povolení práce s jednotlivými bankami, zbývající dva nastavují parametry pro jednotlivé banky: EBIU_AMBCTL0 pro banky 0 a 1, EBIU_AMBCTL1 pro banky 2 a 3. Výchozí nastavení časování je konzervativní – všechny přístupové časy jsou nastaveny na maximum.

Obvod je připojen k asynchronní bance 0 (výběrový signál AMS0). Předpokládáme použití frekvence SCLK 133 MHz, jeden cyklus SCLK je tedy 7,5 ns. Hodnoty přístupových časů pro banku 0 jsou uvedeny v tabulce 3.2. Výsledné konfigurační slovo pro banku 0 je **0x139A**.

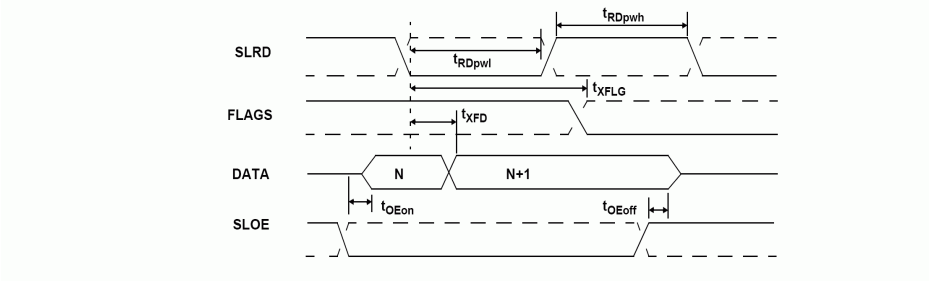
Tab. 3.2 Nastavení přístupových časů pro banku 0 (řadič Ethernetu)

parametr EBIU	rozsah nastavení [SCLK cycles]	nastavení EBIU [SCLK cycles]
read access time	1-15	**3**
write access time	1-15	**1**
setup time	1-4	**1**
hold time	0-3	**2**
bank transition time	1-4	**2**

Trvání čtecího cyklu je 6 cyklů, což odpovídá maximální přenosové rychlosti cca 44 MB/s. Trvání zápisového cyklu je 4 cykly, což odpovídá rychlosti cca 66 MB/s. Šířka pásma je tedy dostatečná pro dosažení maximální možné rychlosti přenosu přes Ethernetové rozhraní i při větším zatížení paměťové sběrnice.

3.3.3 Konfigurace přístupu k USB FIFO

Obvod Cypress CY7C68013A (IC4) je připojen k bance 1 (výběrový signál AMS1). Komunikace probíhá v asynchronním režimu obvodu IC4. Časování čtení a zápisu pro obvod je znázorněno na obr. 3.26 a obr. 3.27 (převzato z datasheetu [16]).

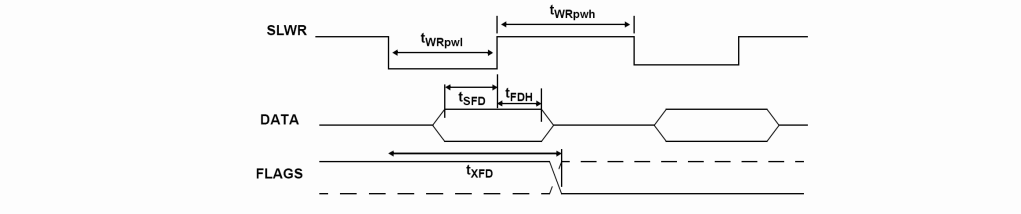


Parameter	Description	Min.	Max.	Unit
t_{RDpwl}	SLRD Pulse Width LOW	50		ns
t_{RDpwh}	SLRD Pulse Width HIGH	50		ns
t_{XFLG}	SLRD to FLAGS Output Propagation Delay		70	ns
t_{XFD}	SLRD to FIFO Data Output Propagation Delay		15	ns
t_{OEon}	SLOE Turn-on to FIFO Data Valid		10.5	ns
t_{OEoff}	SLOE Turn-off to FIFO Data Hold		10.5	ns

Obr. 3.26 Časování asynchronního čtení FIFO obvodu CY7C68013A

Jak je uvedeno na obr. 3.27, doba po kterou má být signál SLWR v neaktivním stavu je 70 ns, což je po zaokrouhlení 10 cyklů SCLK (133 MHz). Rozhraní EBIU ovšem umožňuje nastavit setup time na max. 4 cykly a hold time na max. 3 cykly, což je dohromady 7 cyklů, tedy pouze cca 52 ns.

V dané situaci si můžeme vypomoci tím, že po zápisu do FIFO provedeme vzápětí zápis do jiné banky paměti, která v tomto zapojení není nijak využita (banky 2 a 3 jsou volné a je možné je použít pro tento účel). V konfiguraci dané banky nastavíme např. setup, write i hold time na 1 cyklus, čímž získáme zbývající 3 potřebné cykly pro pokrytí doby t_{WRpwh} o délce 70 ns. Dobu t_{WRpwl} o délce 50 ns je již možné pokrýt bežným nastavením write acess time. Při tomto způsobu komunikace ovšem není možné ze zřejmých důvodů použít přenos DMA.



Parameter	Description	Min.	Max.	Unit
t_{WRpwl}	SLWR Pulse LOW	50		ns
t_{WRpwh}	SLWR Pulse HIGH	70		ns
t_{SFD}	SLWR to FIFO DATA Set-up Time	10		ns
t_{FDH}	FIFO DATA to SLWR Hold Time	10		ns
t_{XFD}	SLWR to FLAGS Output Propagation Delay		70	ns

Obr. 3.27 Časování asynchronního zápisu do FIFO obvodu CY7C67013A

Další možností, jak se vyrovnat s tímto problémem je snížení frekvence SCLK na 100 MHz nebo menší, aby bylo možné dodržet správné časování pouhým nastavením registrů EBIU. Snížení frekvence SCLK ovšem vede k celkovému snížení výkonu systému (zejména kvůli omezení rychlosti komunikace s pamětí SDRAM).

Výsledné hodnoty přístupových časů pro banku 1 jsou uvedeny v tabulce 3.3. Konfigurační slovo pro banku 1 je **0xA6FA**. Konfigurační slovo pro banku 2 (pro „dummy" zápis) je **0x1F52**.

Tab. 3.3 Nastavení přístupových časů pro banku 1 (řadič USB)

parametr EBIU	rozsah nastavení [SCLK cycles]	nastavení EBIU [SCLK cycles]
read access time	1-15	**6**
write access time	1-15	**10**
setup time	1-4	**4**
hold time	0-3	**3**
bank transition time	1-4	**2**

3.4 Spolupráce s hradlovým polem FPGA

Kromě signálových procesorů se pro zpracování obrazu a digitálních signálů obecně využívají také hradlová pole FPGA. V případě systému, který již signálový procesor obsahuje, je možné hradlové pole využít pro zvýšení celkového výkonu. Je ovšem nutné vyřešit problém vhodného připojení k procesoru.

Do hradlového pole je možné velmi efektivně implementovat některé operace, které by jinak samotný procesor zbytečně zatížily. Zejména je možné využít masivně paralelního nebo naopak velmi rychlého a jednoduchého (on-the-fly) zpracování. Jedná se např. o tyto dílčí operace:

- prahování
- jasové transformace
- konvoluční filtrace, hledání hran
- výpočet DCT
- gamma korekce
- přepočet barevných modelů
- … a další

3.4.1 Připojení FPGA k procesoru Blackfin

Pro spolupráci procesoru Blackfin s hradlovým polem FPGA je možné využít několik způsobů připojení. Rozhraní, která by bylo možné potenciálně použít ke komunikaci s FPGA jsou zejména:

- SPI – plně duplexní, maximální rychlost 4,1 MB/s (33,3 Mbit/s) duplexně
- SPORT – plně duplexní, maximální rychlost 16,3 MB/s (2× 66,6 Mbit/s) duplexně

- PPI – pouze jednosměrné, maximální rychlost 133 MB/s
- EBIU, asynchronní rozhraní – obousměrné, maximální rychlost limitně 133 MB/s

Přenosové rychlosti platí pro frekvenci SCLK 133 MHz. Maximální frekvence hodinového signálu je SCLK/4 pro SPI a SCLK/2 pro SPORT, PPI a EBIU.

Jak je vidět z uvedeného přehledu, největší šířka pásma pro přenos dat mezi procesorem a FPGA je k dispozici na rozhraních PPI a EBIU. Sběrnice SPI je z uvedených možností nejpomalejší a je vhodné ji použít (spíše než pro přenos signálu) jako sekundární komunikační kanál určený pro předávání konfiguračních parametrů.

3.4.1.1 Komunikace přes rozhraní PPI

Rozhraní PPI bývá obvykle v úlohách zpracování obrazu použito pro připojení kamery nebo kodeku, proto jej není možné využít jako dedikované rozhraní pro komunikaci s hradlovým polem. Je ovšem možné zařadit FPGA do datové cesty mezi zdroj digitalizovaného signálu a rozhraní PPI. Přenosová rychlost může být na vstupu a výstupu FPGA stejná (jako např. v úlohách prahování, jasových transformací apod.), případně může být výstupní přenosová rychlost odlišná od vstupní (např. při přepočtu barevného modelu YCbCr na RGB). V případě potřeby předzpracování digitalizovaného videosignálu v reálném čase je tato topologie pravděpodobně nejlepší variantou.

3.4.1.2 Komunikace přes paměťovou sběrnici EBIU

Další z možností je využití rozhraní EBIU, které je určeno především pro připojení pamětí. Toto rozhraní není duplexní a nemůže být použito jako dedikované pro připojení FPGA. Na rozhraní je (v typickém zapojení) připojena paměť SDRAM nebo SRAM a přenos dat mezi procesorem a pamětí proto výrazně snižuje šířku pásma použitelnou pro komunikaci s FPGA až na zlomek maximální rychlosti (dle vytížení sběrnice),

3.4.1.3 Komunikace přes synchronní sériové porty SPORT

Rozhraní SPORT obsahuje dva duplexní kanály (primární a sekundární) a umožňuje tak dosáhnout přenosové rychlosti až 16,3 MB/s při současném přenosu po obou kanálech s hodinovou frekvencí 66,6 MHz (SCLK/2). Procesor Blackfin BF-532 obsahuje dvě rozhraní SPORT, takže je možné dosáhnout celkové přenosové rychlosti až 32,6 MB/s při paralelním přenosu po obou rozhraních (tedy celkem po čtyřech datových linkách v jednom směru).

Pokud je tato přenosová rychlost pro danou úlohu dostatečná, může být toto rozhraní použito buď pro připojení FPGA ve funkci „koprocesoru“ (hradlové pole bude dostávat data od procesoru a současně nebo později vracet data zpracovaná), nebo může být pole zařazeno do datové cesty a zpracovávat signál v reálném čase. Přenosová rychlost na výstupu FPGA může být v případě zpracování on-the-fly opět stejná nebo odlišná od rychlosti na vstupu.

Hradlové pole může v takovém případě sloužit zároveň jako převodník mezi paralelním

rozhraním zdroje digitalizovaného signálu a sériovým rozhraním procesoru. Taková koncepce může být výhodná zejména při potřebě zpracovávat signál současně z několika zdrojů, případně při současné digitalizaci a generování videosignálu – v takovém případě může být např. zdroj signálu připojen přes FPGA na SPORT a výstupní kodek může být připojen přímo na rozhraní PPI.

Pro optimální tok dat při paralelně-sériovém převodu je vhodné implementovat v FPGA smyčku fázového závěsu DPLL, která bude násobit vstupní hodinový signál paralelního rozhraní a jejímž výstupem bude hodinový signál pro rozhraní SPORT o N× vyšší frekvenci (např. pro 16 bitů vstupních dat při využití primárního i sekundárního kanálu SPORT bude N rovno 8).

4. Operační systém uClinux

4.1 Úvod

Cílem této kapitoly je rámcově seznámit čtenáře s možnostmi, výhodami a omezeními operačního systému (dále OS) uClinux v aplikacích pro zpracování obrazu a popsat postup portování tohoto systému na vlastní hardware.

4.1.1 Platforma uClinux

Operační systém uClinux je varianta OS GNU/Linux určená pro embedded systémy s procesory, které nemají podporu MMU (Memory Management Unit). GNU/Linux i uClinux jsou svobodné operační systémy unixového typu s otevřenými zdrojovými kódy, vyvíjené pod licencí GPL. uClinux je portovaný na celou řadu platforem, mezi které patří i Analog Devices Blackfin.

Základní částí operačního systému Linux (a stejně tak uClinux) je jádro (kernel), jehož elementární úkoly jsou:

- spouštění uživatelských aplikací ve víceúlohovém a vícevláknovém prostředí (preemptivní multitasking a multithreading)
- vzájemná komunikace procesů pomocí signálů a rour (pipes)
- správa paměti
- správa periferních zařízení

OS uClinux poskytuje širokou podporu pro různé periferie, protokoly a knihovny funkcí, jako jsou např.:

- síťová rozhraní
- pevné disky, paměti flash a paměťové karty (MMC/SD, CF)
- různé síťové a diskové souborové systémy
- vstupní a výstupní zařízení pro interakci s uživatelem (displeje, klávesnice, myši)
- video dekodéry a enkodéry
- obvody pro práci se zvukem
- rozhraní USB, SPI, I2C a Dallas 1-wire
- podpora pro kryptografii (šifrování, autentizace, hašovací funkce)
- knihovny pro komprimaci a dekomprimaci obrazu, zvuku a videa
- servery HTTP, FTP, Telnet a další
- databáze
- a další …

Přehled možností, vlastností a praktického použití operačního systému uClinux na platformě Blackfin je uveden ve vynikajícím článku [28].

4.1.2 Specifika platformy uClinux

uClinux má ve srovnání se systémem Linux několik specifických vlastností, které souvisí s použitím na omezených embedded platformách. Ve srovnání se systémem Linux má varianta uClinux následující omezení:

- není k dispozici virtuální paměťový prostor ani ochrana bloků paměti
- chybí systémová funkce *fork()* pro rozvětvení procesu
- velikost stacku je pevná a nemění se podle potřeby, může dojít k přetečení

Vzhledem k tomu, že uClinux pracuje na rozdíl od Linuxu bez MMU, není možné zajistit ochranu paměťového prostoru na úrovni jednotlivých aplikací. Pokud obsahuje určitá aplikace chybu, která způsobí zápis mimo vlastní alokovaný prostor, může tak způsobit pád ostatních procesů v systému. Na platformě Blackfin jsou nicméně některé základní funkce ochrany paměti k dispozici. Ačkoliv Blackfin sice neobsahuje plnohodnotnou jednotku MMU, je přesto možné detekovat a zabránit např. dereferenci nulových ukazatelů (první 4 KB paměti jsou vyhrazeny pro tento účel) a zápisu do paměťově mapovaných konfiguračních registrů MMR.

Systémová funkce *fork()* v Linuxu slouží k vytvoření nového procesu, který je kopií procesu, který *fork()* volá. Od této chvíle pracují oba procesy paralelně. Tato funkce není v systému uClinux dostupná. Jako náhrada je k dispozici funkce *vfork()*, která se chová obdobně jako *fork()*, ovšem je zde několik rozdílů – největší z nich je, že namísto paralelního spuštění se čeká na ukončení procesu. U stávájících aplikací pro Linux, které je třeba portovat na uClinux je tedy podle situace nutné buď použít *vfork()*, nebo aplikaci přepsat tak, aby používala vlákna (POSIX threads), které uClinux podporuje.

V případě běžného Linuxu je možné díky virtuálnímu paměťovému prostoru stack aplikací dynamicky zvětšovat. V systému uClinux toto není kvůli absenci MMR možné, proto je nutné definovat pevnou velikost stacku již při kompilaci a při jeho přetečení aplikace havaruje. Bližší podrobnosti jsou uvedeny v podkapitole 4.9.4.

Bližší podrobnosti k tématu specifických vlastností uClinuxu ve srovnání s Linuxem lze nalézt např. v článku [29].

4.1.3 uClinux jako platforma pro aplikace zpracování obrazu

Pro aplikace zpracování obrazu může být použití operačního systém uClinux výhodné z několika důvodů:

- snadná implementace síťové komunikace pro posílání obrazu a dat
- HTTP server s podporou CGI pro nastavování parametrů
- využití paměťových médií, případně síťových souborových systémů
- možné použití skriptovacích jazyků pro vyšší úroveň logiky zpracování
- možnost poměrně snadného portování stávajících aplikací pro systém Linux
- využití vláken při programování

Při rozhodování o případném použití systému uClinux jako platformy pro určitou aplikaci je třeba zvážit, nakolik v dané aplikaci využijeme přidanou hodnotu danou operačním systémem a na druhou stranu nakolik vadí v dané aplikaci snížený výkon způsobený režií operačního systému. Pro vytvoření malé jednoúčelové aplikace může být použití operačního systému zbytečně komplikované. Nicméně pokud je už jednou OS na daném hardware zprovozněn, může být obtížnost vytvoření samostatně bežícího programu proti vytvoření programu využívajícího operační systém srovnatelná. Pro složitější aplikace bude pravděpodobně využití operačního systému výhodné i pokud se využijí jen některé z mnoha možností OS.

Pro zkompilování OS uClinux se používá GNU Toolchain s kompilátorem GCC. Pokud se rozhodneme nepoužít operační systém, je otázkou jaký kompilátor zvolit – tedy konkrétně do jaké míry je výhodnější využít kompilátor, který je součástí prostředí VDSP, ve srovnání s kompilátorem GCC, který je součástí GNU Toolchain.

Podle neověřených zdrojů (z internetových diskuzí na webu projektu uClinux pro Blackfin [18]) by měl být kompilátor GCC z hlediska rychlosti srovnatelný s kompilátorem VDSP s rychlostními rozdíly v řádu 10 % na obě strany podle druhu kompilované aplikace. Obecně se dá říci, že pro aplikace zpracování signálů bude rychlostní výhoda spíše na straně VDSP, pro obecný kód bude rychlejší spíše kód produkovaný kompilátorem GCC. Kompilátor GCC také údajně nabízí vyšší kompatibilitu se standardem ANSI C, protože preferuje striktní dodržení standardu před rychlostí kódu, oproti VDSP, jehož základní prioritou je rychlost výsledného kódu a některé aspekty standardu bere spíše jako doporučení.

Nezanedbatelným aspektem rozhodování je samozřejmě také cena – zatímco prostředí VDSP stojí řádově desítky tisíc korun, GNU Toolchain i OS uClinux je k dispozici zdarma.

4.1.4 Typografické konvence

Vzhledem k tomu, že v této kapitole bude uvedena celá řada výpisů zdrojových kódů, příkazů operačního systému apod., je nutné pro zvýšení přehlednosti a srozumitelnosti uvést některé konvence vyznačování různých druhů textu.

- příkaz pro operační systém na platformě PC, běžný uživatel (tučně, prompt $)

```
$ příkaz pro PC v prostředí běžného uživatele
```

- příkaz pro operační systém na platformě PC, uživatel „root“ (tučně, prompt #)

```
# příkaz pro PC v prostředí uživatele root
```

- příkaz pro operační systém na cílové platformě (tučně, prompt >)

```
root:~> příkaz na cílové platformě
```

- příkaz pro bootloader U-Boot

```
myboard> příkaz na bootloader U-Boot
```

- zalomení dlouhého řádku; výpis pro uživatele (běžný řez písma)

```
root:~> velmi dlouhý příkaz na cílové platformě, který se zalomí
     do následujícího řádku
výpis pro uživatele z tohoto příkazu, který je také natolik dlouhý,
     že se zalomí do druhého řádku
```

- zdůraznění aktuální pracovní cesty (~ je domovský adresář uživatele)

```
 ~/uClinux-dist.R1.1-RC3/linux-2.6.x $ make menuconfig
```

- název souboru, adresáře nebo příkazu: **/cesta_k_souboru/název_souboru**
- název funkce: *funkce()*
- část zdrojového kódu s vyznačením změněné (nebo jinak důležité) části

```
choice
        prompt "System type"
...
config BFIN532_MYBOARD
```

4.2 Základy práce s operačním systémem GNU/Linux

V této kapitole budou uvedeny některé základní příkazy operačního systému Linux (většina z nich jsou standardní unixové příkazy) a několik základních pokynů pro práci v příkazové řádce (tzv. Command Line Interface – CLI). Kapitola je určena především těm, kteří mají se systémem GNU/Linux jen minimální zkušenosti. Vhledem k omezenému prostoru se jedná pouze o velmi stručný přehled bez nároku na úplnost a podrobnost popisu.

Základní příkazy jsou zde uvedeny formou tabulky (tab. 4.1), ve které je uveden název příkazu, význam (stručný popis) a jeden nebo několik málo příkladů použití. Významy příkazů jsou uvedeny v angličtině, protože název většiny příkazů je tvořen zkratkou a je tak možné si název snadněji zapamatovat.

Tab. 4.1 Přehled příkazů pro systémy unixového typu

příkaz	význam	příklad
soubory a adresáře		
ls	list files in directory	`ls`
	list files – detailed (long) view	`ls -l`
cd	change directory	`cd directory`
	to parent directory	`cd ..`
cp	copy	`cp file1 file2`
mv	move	`mv file1 file2`
rm	remove	`rm file`
	remove recursive	`rm -r -f directory`
mkdir	make directory	`mkdir directory`
pwd	print working directory	`pwd`
ln	create link – symbolic link	`ln -s file link_to_file`
tar	„tape archiver" – extract files	`tar -xvjf archive.tar.bz2`
	„tape archiver" – compress files	`tar -cvjf archive.tar.bz2 directory`
diff	find differences	`diff original_file modified_file`
	diff – recursive, quick (info only)	`diff -r -q original_dir modified_dir`
df	disk free – show free disk space	`df`
du	disk usage – show size of work. dir.	`du -s -h`
práce s textem		
echo	echo text	`echo abcdef >> /etc/abc`
grep	global reg-exp print – find pattern	`grep ^#define main.h`
cat	concatenate	`cat file1 file2 > outfile`
	print file	`cat file`
less	scrollable view of file	`less /etc/profile`
	scrollable view of directory list	`ls | less`
tail	show lines from end of file	`dmesg | tail`
head	show lines from begin of file	`head -n 20 filename`
nano	Nano's ANOther editor	`nano /etc/fstab`
vyhledávání		
grep -r	grep, recurse directories	`grep -r what_to_search where`
	find files containg main() function	`grep -r \\Wmain\(.*\) .`
locate	locate file or directory by name	`locate filename`
find	show filenames, recursively	`find | grep /include/`
vzdálené soubory		
wget	„WWW get" - download HTTP file	`wget http://www.site.com/file.txt`
	download FTP file	`wget ftp://ftp.site.com/pub/file.zip`
scp	secure copy to remote host	`scp file user@hostname:dir/file`
	secure copy from remote to local	`scp user@hostname:dir/file file`
uživatelské účty a práva		
useradd	create user	`useradd -m blackfin`
su	switch to superuser (root)	`su`
	login superuser	`su -`
	login user blackfin	`su - blackfin`
exit	exit shell	`exit`
chmod	change file mode (permissions)	`chmod u+wx file`
chown	change owner/group	`chown -hR root:root directory`

procesy		
ps	show processes	`ps`
	show all users' processes	`ps aux | grep root`
kill	kill process	`kill -9 1234`
killall	kill processes by name - interactive	`killall -i sshd`
top	show live table of processes	`top`
time	measure time	`time make`
připojení zařízení (disků apod.)		
mount	mount device	`mount /dev/hda1 /mnt/disk_c`
umount	umount device	`umount /mnt/disk_c`
ovladače		
lsmod	list modules	`lsmod`
modprobe	add kernel module	`modprobe mmc_spi`
	remove kernel module	`modprobe -r mmc_spi`
dmesg	display kernel messages	`dmesg`
konfigurace sítě		
ifconfig	network interface configuration	`ifconfig eth0 192.168.1.1`
	network interface shutdown	`ifconfig eth0 down`
route	print routing table	`route`
	add default gateway	`route add default gw 10.0.2.2`
ostatní		
poweroff	shutdown system	`poweroff`
make	compile project	`make`
	clean project	`make clean`
man	show manual page	`man emerge`
xargs	execute with args – count lines (.c files)	`find | grep \\.c$ | xargs cat`
gentoo portage		
emerge	install package	`emerge bfin-toolchain`
	search by name	`emerge --search ftp`
	search by description	`emerge --searchdesc Blackfin`
	update (pretend)	`emerge --pretend --update system`

Naprostá většina příkazů vypíše při spuštění s parametrem *--help* nápovědu se seznamem možných parametrů předávaných na příkazové řádce. Dále je možné vypsat podrobnou nápovědu (manuálovou stránku) příkazem **man**. Pro ukončení prohlížení manuálové stránky je třeba stisknout klávesu Q (totéž platí pro uzavření prohlížeče **less**).

V shellu (**bash**) je možné použít několik funkcí, které velmi usnadní práci:

- šipky nahoru/dolů umožňují pohybovat se v seznamu dříve zadaných příkazů
- po stisku Ctrl-R je možné v seznamu zadaných příkazů vyhledávat, předchozí výskyty je možné zobrazit opakovaným stiskem Ctrl-R
- po stisku tabulátoru shell doplní napsaný název souboru nebo adresáře, případně nabídne seznam možných alternativ doplnění; tímto způsobem je možné velmi rychle zadávat i dlouhé a komplikované názvy

Pro přesměrování výstupu jednoho programu do druhého je třeba použít operátor „|“,

k přesměrování výstupu do souboru slouží operátor „>“ (případně „>>“ pro přidání na konec souboru, v případě že soubor již existuje). Příklady jsou uvedeny v tab. 4.1.

Doplněním „&“ na konec příkazu se program spustí na pozadí, např.

```
# emerge --deep --update world > messages.txt &
```

Pozastavení programu můžeme provést stiskem Ctrl-Z po spuštění příkazu. Dále můžeme zadat příkaz **fg** pro pokračování v běhu na popředí, případně **bg** pro běh na pozadí (stejně jako při spuštění s „&“).

Pokud chceme spustit několik příkazů po sobě, můžeme je spojit operátorem „&&“:

```
$ make mkproper && make config && make
```

Pokud některý z příkazů v řetězci vrátí chybový status, dojde k ukončení a následující příkazy se již nespustí.

4.3 GNU Toolchain

Základním vývojovým prostředkem pro platformu uClinux je „GNU Toolchain for the Blackfin Processor”, což je balík aplikací potřebných pro kompilaci, linkování, ladění apod. Základní části GNU Toolchain jsou:

- **GNU Binutils** – základní části jsou GNU Linker (**ld**) a GNU assembler (**as**)
- **GNU Compiler Collection** (**gcc**) – kolekce kompilátorů pro jazyky C, C++, Objective-C, Fortran a Ada; na plaformu Blackfin jsou portovány pouze kompilátory jazyků C a C++
- **GNU Debugger** (**gdb**) – standardní knihovna pro ladění, umožňuje lokální i vzdálené ladění aplikací
- další utility jako **elf2flt**, **genext2fs**, **cramfs**, **ldrviewer**
- knihovny **libdsp**, **newlib**, **libgloss, uClibc** a další

Pro vývoj aplikací je dále možné použít integrovaná vývojová prostředí (IDE) jako např. Eclipse, Insight, Kdevelop a další, případně editory jako např. Kate, emacs, vim, PSPad, nano, mcedit a další. Volba vývojového prostředí či editoru závisí plně na uživateli, nicméně např. pro zmíněná vývojová prostředí existují zásuvné moduly (plug-in) zefektivňující práci.

Základním zdrojem informací i místem pro stažení vývojových nástrojů a zdrojových kódů je web [18].

4.4 Instalace vývojového prostředí

V této kapitole popíšeme, jaké kroky je potřeba provést pro zprovoznění kompletního vývojového prostředí pro platformu uClinux. Jako hostitelská platforma bude uvažován operační

systém Microsoft Windows XP.

Popsaný postup není v žádném případě jedinou alternativou, naopak možnosti jsou velmi široké a v některých případech bude na alternativy upozorněno. Nicméně popis bude vedený především se snahou o reprodukovatelnost.

4.4.1 GNU Blackfin Toolchain na platformě Linux

Abychom mohli provozovat vývojové nástroje pracující v prostředí Linux na platformě Windows, je třeba použít některý z virtualizačních či emulačních nástrojů. Následující popis předpokládá použití emulátoru QEMU. V případě instalace na nativní systém, resp. při použití jiného virtualizačního/emulačního nástroje je třeba ignorovat, resp. přiměřeně upravit kroky, které se bezprostředně týkají emulátoru QEMU.

Podrobnou dokumentaci pro QEMU je možné nalézt na webu [20], případně v HTML dokumentech uložených v adresáři, kde je emulátor nainstalován.

4.4.1.1 Emulátor QEMU

Domovská stránka projektu QEMU je [20]. Na ní je možné nalézt odkaz pro stažení balíčku pro systém Windows **qemu-0.9.0-windows.zip**. Tento soubor stáhneme a rozbalíme do libovolného adresáře (např. **C:\qemu**). Dále stáhneme akcelerátor emulace **Kqemu-1.3.0pre11-install.exe**, který nainstalujeme běžným způsobem. Akcelerátor není nezbytně nutné instalovat, nicméně výrazně zrychluje chod emulátoru. Uvedené verze byly k dispozici v době psaní této práce.

Po úspěšné instalaci QEMU je třeba vytvořit image, na který se později nainstaluje emulovaný systém. Je třeba zvolit název (např. **hda.qcow**), formát a velikost. Jako formát doporučuji nativní formáty qcow nebo qcow2, neboť na rozdíl od prostého raw formátu se jejich velikost zvětšuje až postupně podle potřeby (raw formát, zabírá od začátku plnou alokovanou velikost). QEMU obsahuje ve verzi 0.9.0 chybu při zpracování novějšího formátu qcow2, takže použijeme starší formát qcow (tato chyba by měla být již v novějších verzích opravena). Velikost image by měla být alespoň 5 až 10 GB. V případě že se bude pracovat se zdrojovými kódy přes SVN (viz kapitola 4.4.1.10), doporučuji raději ještě více.

Vytvoříme tedy image s formátem qcow např. o velikosti 20 GB:

```
C:\qemu> qemu-img create -f qcow hda.qcow 20G
Formating 'hda.qcow', fmt=qcow, size=20971520 kB
```

Tím je vytvořený prázdný image (soubor má v tuto chvíli velikost cca 80 KB), do kterého můžeme nainstalovat jakoukoliv distribuci Linuxu nebo principiálně libovolný operační systém (emulátor QEMU není v tomto směru nijak omezen).

4.4.1.2 Instalace OS GNU/Linux

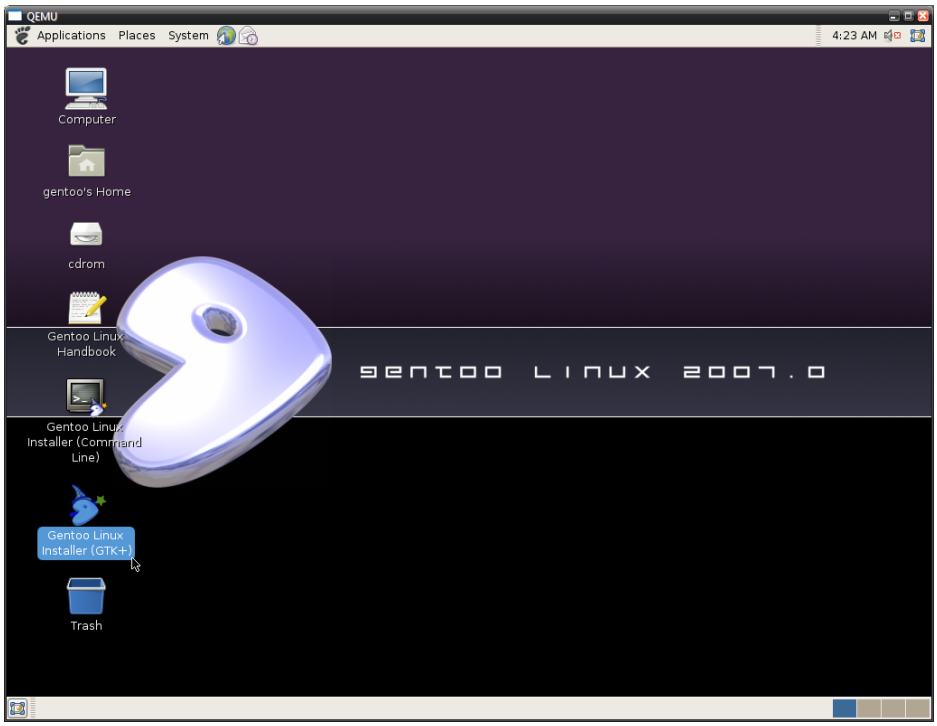
Pro účely práce s vývojovými nástroji pro uClinux můžeme použít v zásadě libovolnou distribuci. Další popis bude platit pro distribuci Gentoo, která byla zvolena především s ohledem na autorovy osobní preference a zkušenosti s touto distribucí. V případě použití jiné distribuce je potřeba brát níže uvedené kroky jako inspiraci a přiměřeně je upravit.

Z webu Gentoo Linux [33] (resp. z některého ze zrcadel, jejichž adresy na něm nalezneme) stáhneme aktuální Live CD (v době psaní to byla verze 2007.0). Poté spustíme emulátor QEMU tak, aby nabootoval z tohoto image. Parametr *-m 256* specifikuje velikost paměti RAM emulovaného počítače v MB – pokud je k dispozici dostatek operační paměti, můžeme specifikovat větší hodnotu.

```
C:\qemu> qemu -L . -m 256 -hda hda.qcow
      -cdrom livecd-i686-installer-2007.0.iso -localtime -boot d
```

Vyčerpávající popis instalačního procesu je obsažen v Gentoo Handbooks na webu [33]. Konkrétně budeme postupovat podle „Gentoo 2007.0 Networkless Handbook“, instalace Gentoo Reference Platform (GRP), v rámci které jsou instalovány předkompilované balíčky z instalačního CD. Na tomto místě zdůrazníme pouze základní kroky.

Po nabootování instalačního CD by se mělo automaticky spustit grafické prostředí, ve kterém zvolíme ikonku „Gentoo Linux Installer (GTK+)“, viz obr. 4.1.



Obr. 4.1 Grafické prostředí Live CD Gentoo Linux

V instalátoru vybereme režim „Networkless“. V dalším okně zvolíme „Recommended layout“ (doporučené rozdělení oddílů na disku). Na potvrzující dotaz odpovíme kladně. Další okna

přeskočíme a začnou se kopírovat soubory. Tato část instalace trvá přibližně 45 minut.

Nastavení **make.conf** přeskočíme, na další obrazovce zvolíme heslo superuživatele „root“. Dále vybereme časovou zónu (Europe/Prague). V dalším kroku se nainstaluje jádro systému (kernel). V nastavení sítě vybereme Interface „eth0“ a zvolíme „Save“ (nastavení sítě se bude získávat z DHCP serveru, který je emulovaný v QEMU). Přeskočíme oznámení o instalovaných komponentách system logger (**syslog-ng**) a cron daemon (**vixie-cron**). „Extra kernel parameters“ není třeba nastavovat. V dalším okně je možné vytvořit uživatelské účty, nicméně tuto možnost můžeme přeskočit (pracovní uživatelský účet vytvoříme později).

Na stránce „Extra Packages“ zaškrtneme balíček **slocate** a případně další podle uvážení (užitečné utility jsou např. **nmap**, **screen** a **netcat**). Podle množství vybraných balíčků může trvat tato část instalace jednotky až desítky minut.

Na další stránce zvolíme, aby se služba **sshd** spouštěla automaticky po startu a v dalších nastaveních zvolíme Clock: local. Tím je instalace dokončena – ukončíme instalátor tlačítkem Exit. Vypneme systém v menu System/Shutdown…, tlačítko Shutdown a počkáme na automatické zavření emulátoru QEMU. Image disku má v tuto chvíli velikost přibližně 2 GB (záleží na množství zvolených alternativních balíčků).

Nyní můžeme spustit nainstalovaný operační systém.

```
C:\qemu> qemu -L . -m 256 -hda hda.qcow -localtime -boot c
      -redir tcp:22::22 -redir udp:69::69
```

Parametr *-redir tcp:22::22* zajišťuje „tunelování“ portu 22/TCP (protokol SSH – Secure Shell) tak, aby byl přístupný pro hostitelský OS (Windows). Obdobně *-redir udp:69::69* zpřístupňuje port 69/UDP pro protokol TFTP.

Po startu systému zadáme uživatelské jménu „root“ a heslo, které jsme si zvolili. Pro instalaci dalších komponent předpokládám připojení k internetu.

Nejprve aktualizujeme strom informací o balíčcích a samotný systém Portage na nejnovější verzi. V závislosti na rychlosti počítače a připojení k internetu, může tento krok trvat až cca hodinu.

```
# emerge --sync
# emerge portage
```

Nyní je potřeba ještě zaktualizovat některé konfigurační soubory. Spustíme

```
# etc-update
```

a zadáme „-3“ (bez uvozovek), čímž budou potřebné soubory automaticky aktualizovány.

4.4.1.3 Vytvoření běžného uživatelského účtu

Nyní vytvoříme uživatelský účet, pod kterým budeme běžně pracovat. Zvolíme uživatelské

jméno, např. *blackfin* (můžeme vybrat jakékoliv, v tom případě je nutné odpovídajícím způsobem změnit následující příkazy). Uživateli nastavíme pomocí parametru *-G* členství ve skupině *wheel*, aby bylo možné v případě potřeby příkazem **su** přepnout na uživatele *root*.

```
# useradd -m -G wheel blackfin
```

Dále nastavíme heslo:

```
# passwd blackfin
New UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

4.4.1.4 Nastavení X server forwarding

V linuxovém vývojovém prostředí je možné spouštět i grafické aplikace, které se případně mohou zobrazovat přímo jako okna na ploše Windows. Pro zprovoznění této funkcionality je třeba nainstalovat do systému Windows X Server (např. server Xming, ke stažení na webu [31]) a v linuxovém systému povolit tzv. X-forwarding.

Pro povolení X-forwardingu pro SSH daemon je třeba editovat soubor **/etc/ssh/sshd_config**:

```
# nano /etc/ssh/sshd_config
```

Nalezneme příslušný řádek a upravíme jej na podobu

```
X11Forwarding yes
```

Uložíme soubor a následně restartujeme SSH server příkazem

```
# /etc/init.d/sshd restart
```

V případě, že chceme používat grafickou konfiguraci uClinuxu, je třeba doinstalovat knihovny pro grafické rozhraní GTK+ a knihovny Tcl/Tk. Textová konfigurace má stejné možnosti, může se však zdát méně uživatelsky přívětivá (záleží na subjektivním pohledu).

```
# emerge gtk+ libglade
# emerge tk
```

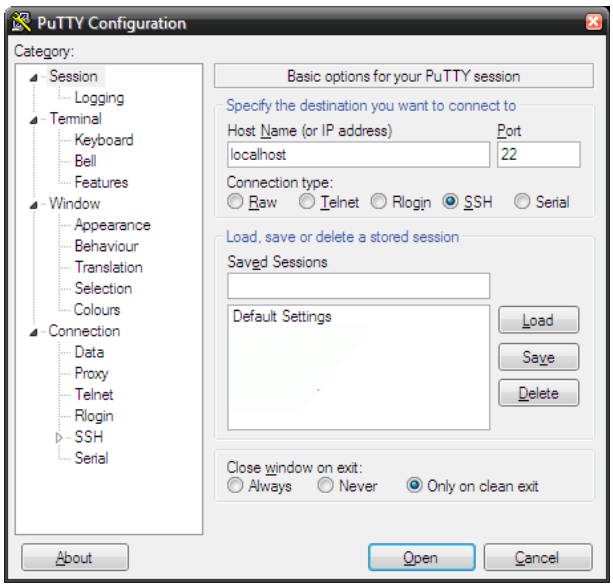
4.4.1.5 Terminál (SSH, sériový port)

Práce se systémem Linux pouze přes konzoli v okně emulátoru QEMU není příliš pohodlná a má také některá principiální omezení. Pro běžnou práci je lepší připojit se k emulovanému Linuxu přes protokol SSH (Secure Shell).

Pro tento účel použijeme klient PuTTY, který je ke stažení na domovské stránce [32]. PuTTY umožňuje připojit se ke vzdálenému systému protokoly SSH a Telnet. Dále umožňuje připojení přes

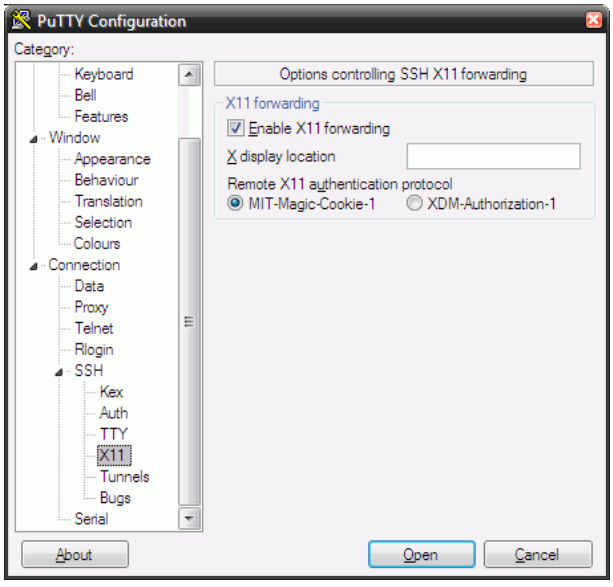
sériový port, což využijeme později pro připojení k desce s uClinuxem. Stáhneme soubor **putty.exe**, který není třeba nijak instalovat, stačí jej jen spustit.

Po spuštění zadáme „localhost“ do kolonky „Host Name“ (viz obr. 4.2) a tlačítkem Open se připojíme k SSH serveru běžícímu v emulátoru.



Obr. 4.2 Základní okno klienta PuTTY

V SSH klientovi PuTTY můžeme dále povolit X-forwarding v nastavení Connection/SSH/X11:

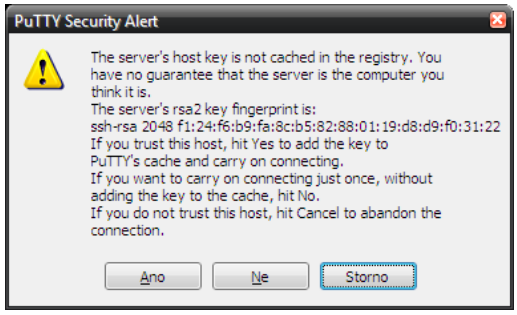


Obr. 4.3 Povolení X-forwardingu v PuTTY

V dalších částech konfigurace je možné nastavit mj. uživatelské jméno pro přihlášení, počáteční velikost okna, velikost paměti pro uchování řádků mimo obrazovku a mnoho dalších parametrů. Kompletní nastavení je následně možné uložit.

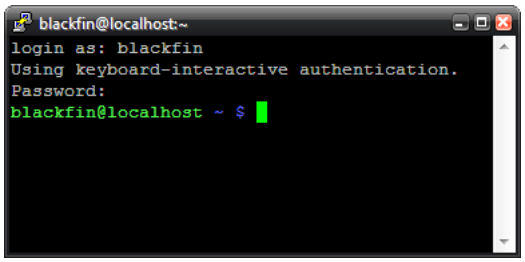
Při prvním připojení (nebo v případě, že se změní klíč serveru) se objeví dialogové okno

(obr. 4.4), vyžadující potvrzení přidání otisku klíče (fingerprint) do databáze PuTTY. Odpovíme kladně.



Obr. 4.4 Upozornění na neznámý fingerprint klíče serveru

Zadáme uživatelské jméno (pokud už nebylo zadáno v nastavení) a heslo (viz obr. 4.5).



Obr. 4.5 Přihlášení přes protokol SSH v klientovi PuTTY

Můžeme také vyzkoušet činnost grafických aplikací, např. spuštěním utility **xclock** (obr. 4.6).

```
$ xclock
```

Obr. 4.6 xclock (vyzkoušení funkce X Serveru)

4.4.1.6 Instalace Subversion (SVN)

Subversion je systém pro správu verzí, který je používán v projektu uClinux ke spravování zdrojových kódu. Pokud chceme mít možnost pracovat s nejaktuálnější revizí zdrojových kódů (tzv. „Live Sources“), je třeba nainstalovat klienta SVN:

```
# emerge subversion
```

4.4.1.7 Instalace TFTP serveru

Server TFTP je nutný pro nahrání obrazu zkompilovaného systému uClinux do cílové platformy přes síť – ve spolupráci s bootloaderem U-Boot – případně pro pozdější aktualizaci samotného bootloaderu. TFTP server může běžet jak pod operačním systémem Linux, tak v prostředí Windows.

Pod systémem Windows můžeme použít např. Tftpd32, který v sobě kromě TFTP serveru integruje i servery DHCP, SNTP a Syslog a dále TFTP klienta. Tftpd32 je možné stáhnout z webu [36].

Pod Linuxem můžeme nainstalovat např. server **atftp** následujícím příkazem

```
# emerge atftp
```

Dále vytvoříme adresář **/tftpboot**, který bude sloužit jako kořenový adresář serveru, a nastavíme mu patřičná práva:

```
# mkdir /tftpboot
# chmod 777 /tftpboot
# chown nobody:nobody /tftpboot
```

Poté přidáme do **/etc/conf.d/local.start** spuštění TFTP serveru a restartujeme příslušnou službu:

```
# echo /usr/sbin/atftpd -v --daemon /tftpboot >>
     /etc/conf.d/local.start
# /etc/init.d/local restart
```

Při použití TFTP serveru pod emulátorem QEMU je nutné přidat emulátoru při spuštění parametr „*-redir udp:69::69*“ pro přesměrování komunikace se serverem.

4.4.1.8 Instalace správce souborů Midnight Commander (MC)

Pro pohodlnější práci je vhodné nainstalovat souborový manažer, např. obvykle používaný Midnight Commander (mc). Tento program umožňuje mj. pohodlně procházet adresářovou strukturu, vytvářet adresáře, editovat, kopírovat a přesouvat soubory, připojovat se k FTP a SCP serverům a mnoho dalších funkcí.

```
# echo app-misc/mc slang -unicode >> /etc/portage/package.use
# emerge mc
```

Spuštění provedeme jednoduše příkazem **mc**:

```
$ mc
```

Textový editor, který je součástí Midnight Commanderu, je kromě použití v rámci prostředí souborového manažeru možné spustit i přímo, což může být užitečné např. pro editaci různých konfiguračních souborů apod. Příklad použití:

```
$ mcedit /etc/fstab
```

4.4.1.9 GNU Toolchain

Dále nainstalujeme GNU Toolchain, klíčovou komponentu pro vývoj aplikací v rámci operačního systému uClinux. V distribuci Gentoo je připraven balíček **(sys-devel/bfin-toolchain**), který nainstaluje poslední oficiální verzi Blackfin GNU Toolchain:

```
# emerge bfin-toolchain
```

Po instalaci je potřeba nastavit cesty pro vývojové nástroje. To provedeme vytvořením souboru **/etc/env.d/80bfin-toolchain**, do kterého vložíme nastavení proměnné „PATH“ spuštěním následujícího příkazu (vše na jeden řádek).

```
# echo PATH=\"/opt/uClinux/bfin-elf/bin:/opt/uClinux/bfin-
    uclinux/bin:/opt/uClinux/bfin-linux-uclibc/bin\" >
    /etc/env.d/80bfin-toolchain
```

Poté spustíme aktualizaci proměnných prostředí

```
# env-update && source /etc/profile
```

aby se změny projevily, a ověříme, že jsou nástroje správně nainstalovány a že jsou přístupné ve vyhledávací cestě (PATH):

```
# bfin-elf-gcc --version
# bfin-uclinux-gcc --version
# bfin-linux-uclibc-gcc --version
```

4.4.1.10 Získání zdrojových kódů

Zdrojové kódy uClinux, U-Boot a případně také GNU Toolchain je možné získat v zásadě dvěma způsoby:

a) stáhneme z webu [18] jeden z pravidelně vydávaných distribučních balíčků
b) použijeme systém Subversion pro stažení nejaktuálnějších zdrojových kódů

Ad a) Stáhneme archívy **u-boot-1.1.5-bf1-061210.tar.bz2** (bootloader U-Boot; verze z projektu BlackfinOne, upravená pro BF-532) a **uClinux-dist-2007R1.1-RC3.tar.bz2** (distribuce uClinux). Uvedené verze jsou nejaktuálnější verze dostupné v době psaní této práce. Pro stažení a rozbalení archívů použijeme např. následující příkazy:

```
$ wget http://blackfin.uclinux.org/gf/download/frsrelease/
    330/2208/u-boot-1.1.5-bf1-061210.tar.bz2
$ wget http://blackfin.uclinux.org/gf/download/frsrelease/
    350/3340/uClinux-dist-2007R1.1-RC3.tar.bz2
$ tar -xvjf u-boot-1.1.5-bf1-061210.tar.bz2
$ tar -xvjf uClinux-dist-2007R1.1-RC3.tar.bz2
```

Ad b) Je nutné mít nainstalovaný Subversion (kapitola 4.4.1.6). Provedeme tzv. checkout – inicializujeme pracovní kopii zdrojových kódů (working copy) a stáhneme aktuální revizi zdrojových kódů:

```
$ mkdir svn
$ cd svn
$ svn checkout svn://blackfin.uclinux.org/u-boot/trunk/u-boot-1.1.6
      u-boot
$ svn checkout svn://blackfin.uclinux.org/uclinux-dist/trunk
      uclinux-dist
```

Checkout je třeba provést pouze jednou. Pokud dojde k přerušení (distribuce uClinux obsahuje přes 90 tisíc souborů o celkové velikosti téměř 1 GB; checkout může trvat na pomalejší lince i několik hodin), provede se navázání spuštěním příkazu svn update v příslušném adresáři – např. pro adresář **uclinux-dist**:

```
$ cd uclinux-dist
$ svn update
```

Stejným způsobem (příkazem **svn update**) je možné kdykoliv aktualizovat zdrojové kódy na nejaktuálnější existující revizi, přičemž případné změny, které jsme ve zdrojových kódech provedli, zůstanou zachovány. Možnosti systému Subversion jsou velmi široké, detailní popis lze nalézt v manuálu [34].

4.4.1.11 Aktualizace vyhledávací databáze

Pokud chceme využívat vyhledávání v souborovém systému pomocí příkazu **locate**, je nutné průběžně aktualizovat databázi souborů. Aktualizace se provádí automaticky jednou denně, v případě potřeby ji můžeme vyvolat manuálně příkazem

```
# updatedb
```

4.4.1.12 Sdílení souborů mezi platformami

Z linuxového prostředí je možné připojit sdílené disky systému Windows pro pohodlné sdílení souborů mezi oběma platformami. Nejprve je potřeba doinstalovat balíček **mount-cifs**:

```
# emerge mount-cifs
```

Poté už stačí jen připojit sdílenou složku do libovolného (existujícího) adresáře:

```
# mount -t cifs -o user=username //server/folder /mnt/mountpoint
```

Parametry *username*, *server*, *folder*, *mountpoint* je třeba nahradit příslušným uživatelským jménem, heslem, jménem serveru, názvem sdílené složky a názvem adresáře, do kterého chceme sdílenou složku připojit. Většinou se pro připojování dalších souborových systémů používá adresář /

mnt. Pokud se připojujeme k lokálnímu počítači, na kterém provozujeme linuxový systém v emulátoru QEMU, použijeme pro *server* adresu 10.0.2.2. Konkrétní příklad připojení implicitně sdílené složky **C$** pod uživatelským jménem *Administrator*:

```
# mount -t cifs -o user=Administrator //10.0.2.2/c\$ /mnt/smb
Password:
```

Další možnosti pro sdílení souborů jsou např. protokoly SCP nebo FTP/SFTP. Pro systém Windows je možné použít např. klient WinSCP, který je ke stažení na webu [35].

4.4.1.13 „Vypnutí“ systému

Po ukončení práce je vhodné systém Linux regulérním způsobem ukončit. Pro tento účel můžeme např. spustit příkaz **powerdown** (emulátor QEMU se poté sám zavře).

```
# powerdown
```

4.4.1.14 Soubory v DVD příloze

Na přiloženém DVD je k dispozici image pro QEMU, na kterém jsou nainstalovány výše uvedené aplikace a zdrojové kódy, včetně úprav, které jsou popsány níže. Dále jsou na DVD umístěny patche obsahující změny v distribuci uClinux a bootloaderu U-Boot (viz kapitola 4.6.1.7), archivy s jejich originální verzí a zkompilované obrazy pro přímé nahrání do desky.

4.4.2 GNU Blackfin Toolchain na platformě Windows

Toolchain na platformě Windows je možné použít pouze pro kompilaci jednotlivých aplikací pro systém uClinux, případně samostatných programů (bez operačního systému). Vzhledem k omezením platformy Windows – jako jsou především nerozlišování velkých a malých písmen v názvech souborů a neexistence symbolických odkazů na soubory – není možné kompilovat na této platformě větší projekty, jako je kernel uClinux nebo U-Boot. V této práci se nebudeme nativní práci na platformě Windows blíže věnovat.

Pokud chceme nainstalovat Toolchain pro Windows, stáhneme soubor **blackfin-toolchain-win32-SVN.exe** z webu [18] a standardním způsobem nainstalujeme. Jedná o instalační balíček zkompilovaný z aktuální (SVN) verze zdrojových kódů.

4.5 U-Boot

U-Boot (oficiální název „Das U-Boot“) je projekt bootloaderu s otevřenými zdrojovými kódy, který je portovaný na celou řadu platforem – mj. PowerPC, ARM, MIPS, x86 a Blackfin. Umožňuje bootovat z celé řady pamětí flash typu NOR a NAND, sériových pamětí flash a dalších. U-Boot má podporu pro síť a umožňuje načítat bootované aplikace přes protokol TFTP. Další možností je načítání aplikací přes sériový port. Obsah paměti flash, včetně sebe sama, je možné aktualizovat, takže po

počátečním naprogramování externím programátorem je možné její obsah aktualizovat pouze bootloaderem.

U-Boot je preferovaný bootloader pro načítání jádra operačního systému uClinux.

4.5.1 Nastavení a kompilace bootloaderu U-Boot

V této kapitole bude popsán postup portování bootloaderu U-Boot na hardware popsaný v kapitole 3. Popis vychází především z informací uvedených v [18], přičemž je upraven a doplněn o podrobnosti týkající se přímo této práce.

Portování bootloaderu na vlastní hardware spočívá ve vytvoření vlastní konfigurace bootloaderu a nastavení různých parametrů jako je např. frekvence jádra a sběrnice procesoru, typ a velikost paměti flash, parametry paměťi SDRAM, použitý řadič Ethernetu, jeho připojení a nastavení, volba vnitřních funkcí, které budou zakompilovány a další nastavení.

4.5.1.1 Úpravy zdrojových kódů

Jako základ vezmeme zdrojové kódy bootloaderu v úpravě pro projekt BlackfinOne (viz web [19]), protože má připravenou podporu pro BF532 a taktéž používá SPI flash. Úpravy by proto měly být jednodušší, než pokud bychom vycházeli z jiné verze.

Zdrojové kódy **u-boot-1.1.5-bf1** stáhneme z webu [19] (viz kapitola 4.4.1.10).

V adresáři **board** vytvoříme pro novou desku nový adresář (nazvaný např. **myboard**) kopií z **board/bf1**. Soubory **bf1.c** a **bf1.h** přejmenujeme na **myboard.c** a **myboard.h**. Nyní je potřeba zkontrolovat a upravit nastavení v jednotlivých souborech.

Soubor **board/myboard/config.mk** by měl obsahovat následující nastavení:

```
TEXTBASE = 0x01FE0000
PLATFORM_CPPFLAGS += -I$(TOPDIR) -Werror
```

která platí pro 32MB SDRAM a U-Boot o velikosti 128 KB. Soubor **spi.c** (tamtéž) obsahuje driver pro SPI flash řady M25Pxx. V něm je potřeba změnit počet sektorů na 64 (pro použitou paměť M25P32):

```
#define NUM_SECTORS     64      /* number of sectors */
```

Další úpravy se týkají souboru **myboard.c**. Funkce *swap_to()* zajišťuje speciální akci při přepínání bank paměti pro různé periferie. Tato funkcionalita není potřeba, takže obsah funkce vymažeme:

```
void swap_to(int device_id)
{
}
```

Dále upravíme a doplníme funkci *misc_init_r()* pro inicializaci desky, přidáme funkci pro

indikaci průběhu bootovacího procesu pomocí LED *show_boot_progress()* a další pomocné funkce:

```
void sport_write(unsigned char data)
{
    *pSPORT1_TCLKDIV =
        (CONFIG_SCLK_HZ / CONFIG_SPORT1_TCLK_FREQ - 1) / 2;
    *pSPORT1_TCR2 = (CONFIG_SIPO_REG_BITS-1) & SLEN;
    *pSPORT1_TFSDIV = CONFIG_SIPO_REG_BITS-1;
    *pSPORT1_TCR1 = TCKFE | LTFS | ITFS | ITCLK | TSPEN;
    *pSPORT1_TX = data; *pSPORT1_TX = data;
    while (!(*pSPORT1_STAT & TXHRE))
        ;
    *pSPORT1_TCR1 = 0;
}

#if defined(CONFIG_MISC_INIT_R)
// miscellaneous platform dependent initialisations
int misc_init_r (void) {
    *pFIO_FLAG_D = PF8;
    *pFIO_DIR = PF8 | PF10 | PF11;
    sport_write(0);

    return 1;
}
#endif

void show_boot_progress(int status)
{
      if (status >= 0x0f)
         status = 0;
      if (status < 0)
         status = 0x0f;
      sport_write(status);
}
```

V souboru **myboard.h** není třeba dělat žádné funkční změny.

Do hlavního souboru **Makefile** doplníme target pro novou desku

```
#########################################################################
## Blackfin
#########################################################################
...
myboard_config    :     unconfig
      @$(MKCONFIG) $(@:_config=) blackfin bf533 myboard
```

a následně spustíme příkazy

```
$ make mrproper
$ make myboard_config
```

které odstraní případné předchozí nastavení konfigurace a vytvoří příslušné symbolické linky a nastaví tak kompilaci U-Boot pro desku myboard.

4.5.1.2 Hlavní konfigurační soubor

Dále je třeba zkopírovat soubor **include/configs/bf1.h** do **include/configs/myboard.h** a provést

v tomto souboru změny, tak aby nastavení odpovídalo prostředkům dostupným na daném hardware a požadované funkci bootloaderu. Změn je velké množství, zde jsou uvedeny nejdůležitější z nich:

- nastavení velikosti paměti SPI flash

```
#define CONFIG_SPI_PAGE_WRITE_SIZE   0x400000
```

- konfigurace frekvence připojeného krystalu, parametry PLL modulu pro generování frekvencí CCLK, SCLK (nastavená frekvence jádra CCLK je 400 MHz, frekvence SCLK je 133 MHz) a dělící poměr pro generování hodinového kmitočtu SPI

```
#define CONFIG_CLKIN_HZ        20000000
#define CONFIG_CLKIN_HALF      0
#define CONFIG_PLL_BYPASS      0
#define CONFIG_VCO_MULT        20
#define CONFIG_CCLK_DIV        1
#define CONFIG_SCLK_DIV        3
#define CONFIG_SPI_BAUD        4
```

- nastavení parametrů kernelu pro načítání kořenového souborového systému

```
#define CONFIG_BOOTARGS "root=/dev/mtdblock0 rw"
```

- volba rychlosti sériového portu

```
#define CONFIG_BAUDRATE        57600
```

- nastavení použitého driveru Ethernetu a jeho bázové adresy

```
#define CONFIG_DRIVER_SMC91111      1
#define CONFIG_SMC91111_BASE        0x20000300
```

- zrušení podpory pro paralelní paměť flash

```
#define CFG_NO_FLASH
```

- zapnutí podpory pro zobrazování průběhu bootovacího procesu

```
#define CONFIG_SHOW_BOOT_PROGRESS
```

- volba seznamu příkazů U-Boot, pro které bude zkompilována podpora; zejména je nutné vyřadit příkazy, které souvisí s hardwarem, který není na desce přítomný

```
#define CONFIG_COMMANDS \
     ((CONFIG_CMD_DFL & ~CFG_CMD_FLASH &   | \
     ~CFG_CMD_IMLS & ~CFG_CMD_SNTP) | \
     CFG_CMD_PING       | \
     CFG_CMD_DHCP       | \
     CFG_CMD_EEPROM     | \
     CFG_CMD_ELF        | \
     CFG_CMD_CACHE      | \
     CFG_CMD_JFFS2)
```

4.5.1.3 Ostatní úpravy, kompilace

Dále je nutné provést několik úprav v dalších souborech, aby bylo možné projekt s daným nastavením zkompilovat. Problém souvisí s odstraněním podpory pro (paralelní) paměti flash. Konkrétně se jedná o soubory **fs/cramfs/cramfs.c**, **fs/jffs2/jffs2_1pass.c** a **lib_blackfin/board.c**. Dále byla v souboru **drivers/smc91111.c** provedena úprava související s automatickým nastavováním MAC adresy (viz kapitola 4.5.4.2). Bližší popis úprav je nad rámec této práce, konkrétní změny je možné zjistit porovnáním originální a modifikované verze.

Po provedení všech úprav spustíme kompilaci a sestavení bootloaderu

```
$ make clean
$ make
```

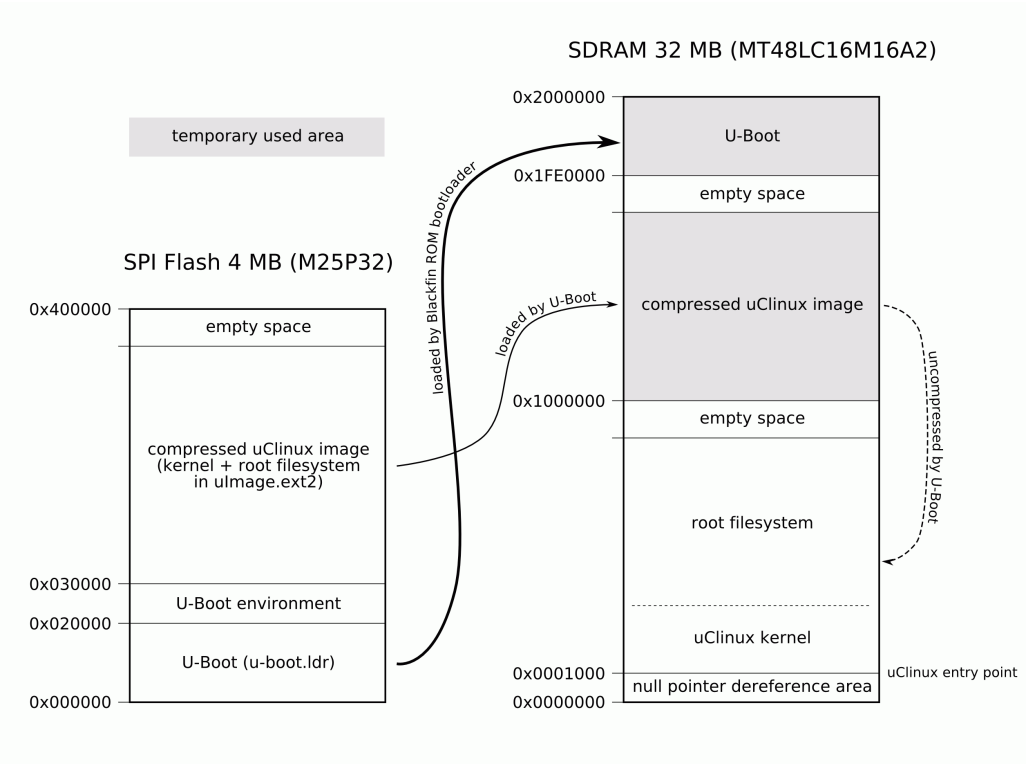
Při kompilaci vzniknou výsledné soubory pro nahrání do flash (**u-boot.ldr**) a image pro přímé spuštění, např. při aktualizaci bootloaderu (**u-boot.bin**).

Soubor **u-boot.ldr** nahrajeme pomocí běžného programátoru (např. ASIX Presto) do SPI flash M25P32.

4.5.2 Postup bootování

Na obr. 4.7 je znázorněna mapa paměti flash a paměti SDRAM v okamžiku bootování. Bootovací proces je následující:

- bootloader umístěný v ROM procesoru Blackfin načte z flash kód U-Boot do SDRAM na adresu 0x1FE0000 (těsně pod vrchol paměti) a spustí jej
- bootloader U-Boot přečte z paměti flash svoje proměnné prostředí (blok „U-Boot environment“) a spustí příkazy uvedené v proměnné *bootcmd* (dále předpokládáme bootování z SPI flash)
- komprimovaný obraz jádra a kořenového souborového systému je nahrán do SDRAM od poloviny paměti (na adresu 0x1000000)
- dále je zkontrolováno CRC a obraz je dekomprimován od adresy 0x1000
- nakonec je proveden skok na adresu 0x1000 a tím je spuštěn kernel uClinuxu



Obr. 4.7 Schéma bootování systému uClinux

Po spuštění jádra uClinuxu není již obsah oblastí vyznačených na obr. 4.7 šedým podbarvením potřeba a tento paměťový prostor je volně využíván operačním systémem.

4.5.3 Uživatelské rozhraní U-Boot

Veškeré funkce bootloaderu U-Boot je možné ovládat přes sériové rozhraní UART.

Připojíme se k desce sériovým kabelem a spustíme terminál (např. již dříve představený terminál PuTTY). Nastavíme příslušný COM port a rychlost 57600 Bd, případně jinou podle nastavení (viz kapitola 4.5.1.2). Spustíme desku – odblokujeme reset – a v terminálu by se měl objevit výpis bootloaderu:

```
U-Boot 1.1.5 (Nov 27 2007 - 23:06:06)

CPU:   ADSP BF532 Rev.: 0.5
Board: BF532 Video processing development board
       Lukas Grepl <L.Grepl@sh.cvut.cz>
Clock: VCO: 400 MHz, Core: 400 MHz, System: 133 MHz
SDRAM: 32 MB
In:    serial
Out:   serial
Err:   serial
Net:   SMC91111 at 0x20000300
starting from spi flash
Hit any key to stop autoboot:  0

EEPROM @0x0 read: addr 0x01000000  off 0x30000  count 0x110000

done
## Booting image at 01000000 ...
Bad Magic Number
myboard>
```

Paměť flash samozřejmě v tuto chvíli neobsahuje image uClinuxu, čehož důsledkem je chybové hlášení „Bad Magic Number“.

Nyní je na místě ověřit, že jsou proměnné prostředí U-Boot nastaveny adekvátně pro dané použití. Proměnné vypíšeme následujícím příkazem:

```
myboard> printenv
```

Zejména je nutné ověřit nastavení sítě - proměnné *ipaddr* (vlastní IP adresa), *serverip* (adresa TFTP serveru), *gatewayip* (adresa výchozí brány sítě – v lokální síti není podstatná) a *netmask* (maska sítě):

```
ipaddr=192.168.0.2
serverip=192.168.0.1
gatewayip=192.168.0.1
netmask=255.255.255.0
```

Uvedené hodnoty jsou pouze ilustrativní, je nutné použít takové hodnoty, které odpovídají nastavení sítě, do které je zařízení připojeno. Pokud některé hodnota neodpovídá, můžeme ji změnit následujícím příkazem:

```
myboard> set variable value
```

kde v*ariable* je název proměnné, kterou chceme nastavit, *value* je nová hodnota proměnné. Hodnoty parametru, které obsahují středník, musí být uvedeny v jednoduchých uvozovkách, např.:

```
myboard> set tftpboot 'tftp 0x1000000 uImage.ext2;bootm'
```

Pokud chceme, aby se změny projevily i po restartu U-Bootu, je nutné uložit je do flash:

```
myboard> saveenv
```

4.5.3.1 Aktualizace bootloaderu

V paměti flash je uložen samotný bootloader, obraz jádra a kořenový souborový systém. Pokud chceme aktualizovat obsah paměti flash, nejprve stáhneme nový obsah do SDRAM (např. přes protokol TFTP). Pro vlastní zápis do SPI flash se používá příkaz:

```
myboard> eeprom write addr offset count
```

kde *addr* je adresa v paměťovém prostoru procesoru, *offset* je adresa v paměti flash a *count* je velikost zapisovaných dat. Pokud tedy chceme např. aktualizovat U-Boot a novou verzi stahujeme přes TFTP, použijeme následující příkazy:

```
myboard> tftp 0x1000000 u-boot.ldr
myboard> eeprom write 0x1000000 0x0 $(filesize)
```

přičemž *$(filesize)* je proměnná, která obsahuje velikost naposledy staženého souboru. Nakonec je potřeba provést reset (mikrospínačem na desce; příkaz reset nestačí protože nedojde k novému natažení kódu z flash do SDRAM). Pro běžné praktické použití bylo vytvořeno makro, které obsahuje výše uvedené příkazy – vyvoláme jej příkazem

```
myboard> run update
```

Novou verzi bootloaderu je také možné otestovat bez zapisování do paměti flash. V tomto případě je nutné použít soubor **u-boot.bin**, který obsahuje přímo spustitelný kód. Spuštění nové verze provedeme pomocí příkazů:

```
myboard> tftp 0x1000000 u-boot.bin
myboard> go 0x1000000
```

4.5.4 Poznámky

4.5.4.1 Uložení proměnných prostředí U-Boot

```
Warning - bad CRC, using default environment
```

Pokud se objeví toto varování, zpravidla se jedná jen o to, že image U-Boot byl právě nahrán do paměti flash a část paměti, ve které má být uloženo nastavení proměnných prostředí (environment variables), je v danou chvíli prázdná. Proto se použije implicitní nastavení definované při překladu. Pro uložení nastavení proměnných prostředí je potřeba spustit příkaz

```
myboard> saveenv
```

kterým se uloží aktuální nastavení do nonvolatilní paměti. Při příštím restartu se již použije uložené nastavení a varování by se nemělo znovu objevit.

4.5.4.2 Inicializace síťového rozhraní U-Boot

Síťové rozhraní není po startu aktivováno automaticky, ale až ve chvíli prvního přístupu, např. příkazem ping. V tu chvíli se nastaví MAC adresa, která je buď pevně nastavena v konfiguračních souborech U-Boot (výchozí hodnota je 02:80:AD:20:31:B8) nebo alternativně může být v konfiguraci zvoleno načítání z EEPROM řadiče LAN.

Při inicializaci uClinuxu se předpokládá, že MAC adresa síťového rozhraní je již v okamžiku spuštění jádra správně nastavena, proto je nutné přimět U-Boot, aby v rámci bootování tuto adresu nastavil. Nejjednodušší možnost je spustit zmíněný příkaz ping, např. definováním proměnných prostředí U-Boot v následujícím tvaru:

```
bootcmd=run ping; run spiboot
ping=ping $(serverip)
```

Další možnou variantou je upravit zdrojové kódy U-Boot tak, aby bylo nastavení MAC adresy provedeno při startu systému automaticky. Výhodou tohoto řešení je eliminace zpoždění u příkazu ping, kdy se čeká na odpověď.

4.6 uClinux Kernel

V této kapitole popíšeme postup portování systému uClinux na hardware vytvořený v rámci této práce (viz kapitola 3). Popis vychází především z informací uvedených v [18], přičemž je upraven a doplněn o podrobnosti týkající se přímo této práce. Dále budou popsány některé obecné možnosti tohoto systému, základní informace o tvorbě aplikací a způsob práce s periferiemi ve vlastních aplikacích.

Stažení souborů jádra je popsáno v kapitole 4.4.1.10. V dalším popisu bude předpokládáno použití zdrojových souborů z uClinux distribuce ve verzi 2007R1.1-RC3, která obsahuje kernel ve verzi 2.6.19. Všechny soubory jádra uClinux se nacházejí v adresáři **uClinux-dist.R1.1-RC3/linux-2.6.x**.

4.6.1 Nastavení

Nastavení jádra je nutné přizpůsobit konkrétnímu hardware, který je dispozici (portovat jádro). Pro definici nové konfigurace jádra pro danou desku je potřeba provést zásahy do souborů **Kconfig** a **Makefile** v podadresáři **arch/blackfin/mach-bf533/** – použitý procesor ADSP-BF532 patří v rámci uClinuxu pod architekturu mach-bf533. Dále vytvoříme soubor **myboard.c** tamtéž (název souboru můžeme nahradit jiným vhodným názvem desky). Následně můžeme jádro nakonfigurovat.

4.6.1.1 Založení definice desky (Kconfig, Makefile)

Nejprve je potřeba formálně definovat novou desku (board). Aby se tato nová deska objevila v konfiguračním menu, je nutné přidat příslušnou definici do souboru **Kconfig**. Otevřeme tedy soubor **arch/blackfin/mach-bf533/board/Kconfig** a upravíme příslušně sekci „System type“:

```
choice
        prompt "System type"
...
config BFIN532_MYBOARD
        bool "BF532-MYBOARD"
        depends on (BF532)
        help
          ADSP-BF532 Blackfin video processing development board.
...
endchoice
```

Doplníme typ paměti SDRAM:

```
config MEM_MT48LC16M16A2TG_75
     bool
     depends on (BFIN533_EZKIT || BFIN561_EZKIT \
           || BFIN533_BLUETECHNIX_CM || BFIN537_BLUETECHNIX_CM \
           || BFIN532_MYBOARD)
     default y
```

Dále můžeme doplnit defaulty u některých voleb, jako např.:

```
config CLKIN_HZ
     int "Crystal Frequency in Hz"
     default "11059200" if BFIN533_STAMP
     default "27000000" if BFIN533_EZKIT
     default "25000000" if BFIN537_STAMP
     default "30000000" if BFIN561_EZKIT
     default "24576000" if PNAV10
     default "20000000" if BFIN532_MYBOARD
     help
       The frequency of CLKIN crystal oscillator on the board in Hz.
```

Podrobnější informace o syntaxi souborů **Kconfig** nalezneme v souboru **Documentation/kbuild/kconfig-language.txt**.

Dále je potřeba upravit soubor **arch/blackfin/mach-bf533/boards/Makefile**. Tento soubor specifikuje, které soubory budou kompilovány pro sestavení jádra a jaká nastavení budou pro kompilaci použita pro danou architekturu a desku. Přidáme do něj odkaz na objektový soubor této desky:

```
obj-$(CONFIG_BFIN532_MYBOARD) += myboard.o
```

Na základě toho se při kompilaci pro desku BF532-MYBOARD zahrne soubor **myboard.c** z adresáře **arch/blackfin/mach-bf533/boards/**.

4.6.1.2 Konfigurace prostředků desky (myboard.c)

Nyní je třeba definovat nastavení specifická pro daný hardware. K dispozici je soubor **arch/blackfin/mach-bf533/boards/generic_board.c** s generickým nastavením, který můžeme použít jako šablonu nastavení nové desky. Výhodnější je ale založit konfiguraci na existujícím souboru např. pro desku BF533 Stamp nebo BlackfinOne. Zkopírujeme tedy soubor **stamp.c** (ze stejného adresáře) do **myboard.c** a provedeme v něm změny:

- nastavíme adresový rozsah řadiče Ethernetu a pin, na který je připojeno IRQ

```
static struct resource smc91x_resources[] = {
      {
            .name = "smc91x-regs",
            .start = 0x20000300,
            .end = 0x20000300 + 16,
            .flags = IORESOURCE_MEM,
      },{
            .start = IRQ_PF9,
            .end = IRQ_PF9,
            .flags = IORESOURCE_IRQ | IORESOURCE_IRQ_HIGHLEVEL,
      },
};
```

- změníme mapu sekcí v paměti flash

```
static struct mtd_partition bfin_spi_flash_partitions[] = {
      {
            .name = "bootloader",
            .size = 0x00030000,
            .offset = 0,
            .mask_flags = MTD_CAP_ROM
      },{
            .name = "kernel",
            .size = 0x3d0000,
            .offset = 0x30000
      }
};
```

- nastavíme typ použité paměti flash

```
static struct flash_platform_data bfin_spi_flash_data = {
      .name = "m25p80",
      .parts = bfin_spi_flash_partitions,
      .nr_parts = ARRAY_SIZE(bfin_spi_flash_partitions),
      .type = "m25p32"
};
```

- provedeme nastavení SPI řadiče pro přístup k MMC kartě

```
#if defined(CONFIG_SPI_MMC) || defined(CONFIG_SPI_MMC_MODULE)
static struct bfin5xx_spi_chip spi_mmc_chip_info = {
      .ctl_reg = 0x1c00,
      .enable_dma = 1,
      .bits_per_word = 8,
};
#endif
```

- nastavíme parametry SPI MMC driveru

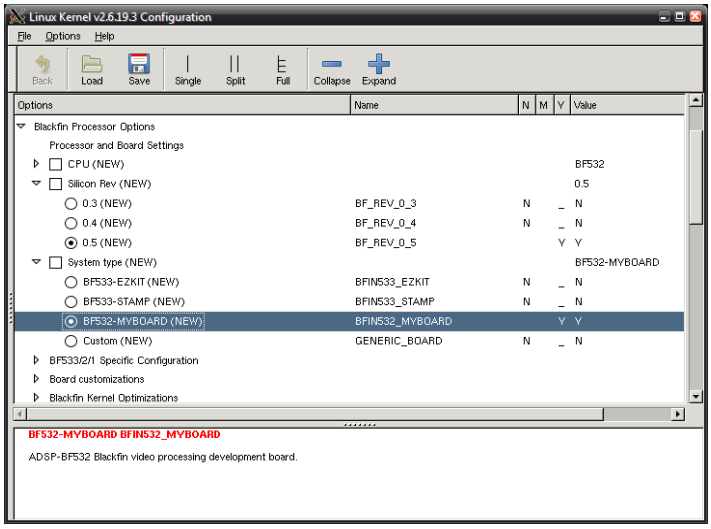
```
static struct spi_board_info bfin_spi_board_info[] __initdata = {
...
#if defined(CONFIG_SPI_MMC) || defined(CONFIG_SPI_MMC_MODULE)
      {
            .modalias = "spi_mmc_dummy",
            .max_speed_hz = 20000000,
            .bus_num = 1,
            .chip_select = 0,
            .platform_data = NULL,
            .controller_data = &spi_mmc_chip_info,
      },
      {
            .modalias = "spi_mmc",
            .max_speed_hz = 20000000,
            .bus_num = 1,
            .chip_select = CONFIG_SPI_MMC_CS_CHAN,
            .platform_data = NULL,
            .controller_data = &spi_mmc_chip_info,
      },
#endif
};
```

4.6.1.3 Konfigurace jádra

Dále můžeme v tuto chvíli nastavit parametry jádra (můžeme to udělat i kdykoliv později). V adresáři **linux-2.6.x** spustíme textovou konfiguraci

```
$ make menuconfig
```

případně grafickou konfiguraci pomocí **make gconfig** resp. **make xconfig** (je třeba mít nainstalované knihovny GTK+ resp. Tcl/Tk a povolený X-forwarding, viz kapitola 4.4.1.4). Možnosti nastavení jsou v obou variantách ekvivalentní, nicméně použití grafického konfigurátoru může být pohodlnější. Podoba grafického konfigurátoru (**make gconfig**) je zobrazena na obr. 4.8.



Obr. 4.8 Grafická konfigurace (Gtk) jádra uClinux

V konfiguraci kernelu je možné nastavit mj.:

- typ použitého procesoru a jeho revizi
- přiřazení vektorů přerušení
- frekvenci krystalu
- velikost paměti SDRAM
- parametry UART pro konzoli
- frekvenci vnitřního časovače jádra
- parametry EBIU pro jednotlivé banky paměti
- podporu pro síťové protokoly a rozhraní
- podporu pro paměti flash a paměťové karty
- parametry rozhraní I2C (použité piny, rychlost)
- podporu rozhraní SPI a Dallas 1-wire
- drivery pro souborové systémy a kódové stránky pro různé jazyky
- … a mnoho dalších nastavení (celkem asi 600 různých voleb)

Popis nastavení jádra pro podporu vybraných periferií a protokolů je uveden v kapitole 4.6.4.

4.6.1.4 Konfigurace aplikací v kořenovém souborovém systému

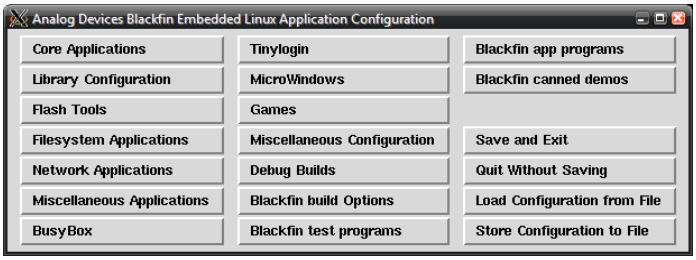
Kromě samotného jádra je nutné nakonfigurovat také zbytek systému uClinux. Jedná se především o to, jaké aplikace budou zahrnuty v kořenovém souborovém systému a o nastavení některých jejich parametrů. Pro tuto konfiguraci zadáme příkaz:

```
~/uClinux-dist.R1.1-RC3 $ make config_menuconfig
```

případně

```
~/uClinux-dist.R1.1-RC3 $ make config_xconfig
```

V této části konfigurace je možné nastavit především které ze základních unixových příkazů budou připojeny, a to buď jako samostatné programy, případně jako součást balíku BusyBox. Dále jsou zde nastavení pro zahrnutí některých ukázkových programů apod. Podoba základní obrazovky grafického konfigurátoru je na obr. 4.9.



Obr. 4.9 Grafický konfigurátor (Qt) aplikací uClinuxu

4.6.1.5 Vytvoření „vendor“ záznamu

Dále je nutné vytvořit ve **vendors/** pro příslušného „dodavatele“ (vendor) a cílový systém (target board) nový adresář. Tento obsahuje výchozí konfiguraci jádra, aplikací i výchozí soubor pro konfiguraci nejvyšší úrovně (viz dále).

Vyjdeme z konfigurace pro desku BF533-STAMP od Analog Devices. Zkopírujeme tedy adresář **vendors/AnalogDevices/BF533-STAMP/** do nového adresáře **vendors/*Vendor*/*Target*/**, kde ***Vendor*** je jméno „dodavatele“ a **Target** je jméno cílového systému, tedy např. **vendors/LukasGrepl/BF532-MYBOARD/**.

4.6.1.6 Konfigurace distribuce uClinux

V kapitolách 4.6.1.3 a 4.6.1.4 byla popsána konfigurace jádra a konfigurace integrovaných aplikací. Dále je k dispozici nejvyšší úroveň konfigurace, kde je možné vybrat Vendor a Target (viz přechozí kapitola), dále nastavit výchozí konfiguraci (pro zvolený Target), vyvolat konfiguraci jádra a aplikací a uložit aktuální konfiguraci jako výchozí.

Spuštění textové konfigurace:

```
~/uClinux-dist.R1.1-RC3 $ make menuconfig
```

Alternativně můžeme spustit grafický konfigurátor (Qt) příkazem **make xconfig**.

Volby top-level konfigurace:

```
Vendor/Product Selection  --->
  --- Select the Vendor you wish to target
  (LukasGrepl) Vendor
  --- Select the Product you wish to target
  (BF532-MYBOARD) LukasGrepl Products

Kernel/Library/Defaults Selection  --->
  --- Kernel is linux-2.6.x
  (None) Libc Version
  [ ] Default all settings (lose changes)
  [ ] Customize Kernel Settings (NEW)
  [ ] Customize Vendor/User Settings
  [ ] Update Default Vendor Settings
```

4.6.1.7 Zjištění modifikací, vytvoření a aplikace patche

Využitím příkazu **diff** je možné jednoduše porovnáním upravené verze s originální zjistit všechny modifikace, které byly provedeny v různých souborech distribuce. Takto vytvořený seznam změn (tzv. patch) je možné naopak aplikovat na originální distribuci a tím do ní zanést provedené změny.

Pro vytvoření patche zahrnujícího změny provedené v distribuci uClinux nejprve smažeme všechny dočasné soubory příkazem

```
~/uClinux-dist.R1.1-RC3 $ make mrproper
```

a následně vytvoříme samotný patch:

```
~ $ diff -urN orig/uClinux-dist.R1.1-RC3/ uClinux-dist.R1.1-RC3/
     > uClinux-dist-R1.1-RC3-myboard.patch
```

(předpokládáme, že v podadresáři **orig** je nezměněná originální verze).

Pro aplikaci patche na originální distribuci se použije příkaz **patch**:

```
~/uClinux-dist.R1.1-RC3 $ patch -p1 -f -i
     ../uClinux-dist.R1.1-RC3-myboard.patch
```

Analogický postup platí pro vytvoření a aplikaci patche pro bootloader U-Boot.

4.6.2 Kompilace

Nejprve je třeba provést kompilaci závislostí (dependencies) a následně vlastní kompilaci. Ne vždy je třeba při změnách konfigurace závislosti rekompilovat, nicméně nejjednodušší je spustit tuto akci vždy (pokud není v danou chvíli kompilace závislostí potřeba, tento krok je automaticky vynechán). Následně je provedena hlavní část kompilace a je vytvořen obraz pro nahrání do paměti flash. Spustíme tedy obě části kompilace:

```
$ make dep && make
```

První sestavení celého obrazu trvá přibližně hodinu, inkrementální rekompilace při menších změnách už pouze několik minut. Při úspěšném sestavení je vypsáno hlášení o výsledném obrazu podobné následujícímu:

```
Image Name:   uClinux Kernel and ext2
Created:      Fri Jan 11 04:10:11 2008
Image Type:   Blackfin Linux Kernel Image (gzip compressed)
Data Size:    3584137 Bytes = 3500.13 kB = 3.42 MB
Load Address: 0x00001000
Entry Point:  0x00001000
```

Zkontrolujeme velikost komprimovaného obrazu (Data Size), která nesmí překročit 3,81 MB (přesně 3 997 696 B), což je prostor dostupný ve 4 MB paměti M25P32. Řešením problémů s překročením velikosti se zabývá kapitola 4.9.1.

Zkompilované obrazy najdeme v adresáři **images**. Pro nahrání do paměti flash použijeme **uImage.ext2**, který obsahuje jádro a kořenový souborový systém (root filesystem). Ten nahrajeme do flash pomocí bootloaderu U-Boot.

4.6.3 Nahrání sestaveného obrazu (image) do flash

Výsledný image nahrajeme do desky pomocí U-Boot a protokolu TFTP. Po nahrání je možné jej přímo spustit (aniž by byl zapsán do flash) a po vyzkoušení jej případně zapsat do SPI flash, ze které bude při příštím startu systému automaticky načten.

Nejprve načteme image přes protokol TFTP do paměti SDRAM na adresu 0x1000000:

```
myboard> tftp 0x1000000 uImage.ext2
```

Nyní máme dvě možnosti:

- spustit uClinux pro vyzkoušení

```
myboard> bootm
```

- aktualizovat image uložený ve flash

```
myboard> eeprom write 0x1000000 0x30000 $(filesize)
```

Pro zjednodušení běžné práce jsou pro výše uvedené příkazy vytvořená makra **tftpboot** a **updatekernel**. Pro provedení aktualizace jádra ve flash stačí proto jednoduše zadat:

```
myboard> run updatekernel
```

4.6.4 Nastavení parametrů jádra

V této kapitole budou popsána některá základní nastavení jádra a dále nastavení potřebná pro zahrnutí podpory určitých zařízení a protokolů:

- typ procesoru, revize, frekvence krystalu, velikost SDRAM, UART konzole

```
Blackfin Processor Options  --->
  --- Processor and Board Settings
  CPU (BF532)  --->
  Silicon Rev (0.5)  --->
  System type (BF532-MYBOARD)  --->
  Board customizations  --->
    --- Board Setup
    (20000000) Crystal Frequency in Hz
    (32) SDRAM Memory Size in Mbytes
    (9) SDRAM Memory Address Width
    (0x1000) Kernel load address for booting
    --- Console UART Setup
    Baud Rate (57600)  --->
    Parity (No Parity)  --->
    Stop Bits (1)  --->
```

- konfigurace asynchronních bank paměti

```
Blackfin Processor Options  --->
  --- Asynchonous Memory Configuration
  EBIU_AMBCTL Global Control  --->
    Enable Asynchonous Memory Banks (Enable Bank 0 & 1 & 2)  --->
  EBIU_AMBCTL Control  --->
    (0x139A) Bank 0
    (0xA6FA) Bank 1
    (0xFFC2) Bank 2
    (0xFFC2) Bank 3
```

- řadič Ethernetu SMSC LAN91C111

```
Network device support  --->
  [*] Network device support
        Ethernet (10 or 100Mbit)  --->
          [*] Ethernet (10 or 100Mbit)
          <*> SMC 91C9x/91C1xxx support
```

- PPI pro připojení kamery; druhý uvedený driver je univerzálnější a je nutné ho použít v případě obecnějšího využití rozhraní PPI

```
Device Drivers  --->
  Character devices  --->
    [*] Blackfin BF5xx PPI Camera frame capture driver
    [ ] Blackfin BF5xx PPI Driver
```

- GPIO Programmable Flags

```
Device Drivers  --->
  Character devices  --->
    [*] Blackfin BF53x Programmable Flags Driver
```

- I2C

```
Device Drivers  --->
  I2C support  --->
    <*> I2C support
    <*> I2C device interface
        I2C Hardware Bus support  --->
          <*> Blackfin GPIO based I2C interface
                Blackfin I2C SDA/SCL Selection  --->
                  (0) SDA GPIO pin number
                  (1) SCL GPIO pin number
           (40)  Cycle Delay in uSec
```

- SPORT

```
Device Drivers  --->
  Character devices   --->
    [*] Blackfin BF53x SPORT support
```

- driver pro SPI a MMC/SD kartu

```
Block layer  --->
 [*] Enable the block layer
     IO Schedulers  --->
       <*> Anticipatory I/O scheduler
           Default I/O scheduler (Anticipatory)  --->
Device Drivers --->
  SPI support --->
    [*] SPI support
    <*> SPI controller driver for ADI Blackfin5xx
  MMC/SD Card support --->
    <M> MMC/SD for SPI support (EXPERIMENTAL)
    (4)   SPI chip select signal for MMC/SD card
```

- podpora souborových systémů FAT a NTFS

```
File systems  --->
  DOS/FAT/NT Filesystems  --->
    <M> MSDOS fs support
    <M> VFAT (Windows-95) fs support
    (437) Default codepage for FAT
    (iso8859-1) Default iocharset for FAT
    <M> NTFS file system support
    [ ]   NTFS debugging support
    [*]   NTFS write support
  Native Language Support  --->
    (iso8859-1) Default NLS Option
    <M>   NLS ISO 8859-1  (Latin 1; Western European Languages)
```

- SMB, CIFS

```
File systems  --->
  Network File Systems  --->
    <M> SMB file system support (to mount Windows shares etc.)
    <M> CIFS support (advanced network filesystem for Samba
```

4.7 Práce se systémem uClinux, integrované aplikace

4.7.1 Přístup na MMC/SD kartu

Na desce je přítomný slot, do kterého lze vložit MMC, případně SD kartu, se kterou je poté možné transparentně pracovat v rámci systému uClinux. Pro přístup na kartu slouží driver **spi_mmc**. Základní operace, které můžeme provádět, jsou tyto:

- inicializace driveru, detekce a inicializace karty

```
root:~> modprobe spi_mmc
```

- ukončení činnosti driveru (před vložením nové karty)

```
root:~> modprobe -r spi_mmc
```

- výpis ladících informace o kartě

```
root:~> cat /proc/spi_mmc
```

- připojení souborového systému na kartě

```
root:~> mount /dev/mmc1 /mnt/mmc
```

- odpojení souborového systému

```
root:~> umount /dev/mmc1
```

Funkce rozhraní MMC/SD byla otestována na celkem třech kartách:

- MMC Noname 32MB „Made in Korea“

```
root:~> modprobe spi_mmc
root:~> New MMC/SD card found: 30 MB | CS channel on PF4

root:~> cat /proc/spi_mmc
...
         Mean read throughput:   830 kB/s
         Mean write throughput:  812 kB/s
```

- SD Transcend 80x

```
root:~> modprobe spi_mmc
root:~> New MMC/SD card found: 976 MB | CS channel on PF4

root:~> cat /proc/spi_mmc
...
Performance for last 100 transfers
         Mean read throughput:   702 kB/s
         Mean write throughput:  724 kB/s
```

- SD Panasonic 16MB – tato karta v systému nepracovala

```
root:~> modprobe spi_mmc
root:~> Could not read this MMC/SD card CSD/CID registers.
```

S jednou z testovaných karet (32MB Noname MMC) byl zaznamenán problém, kdy za přítomnosti karty docházelo při programování SPI flash ke porušení hodnoty některých byte zapisovaných do paměti. K problému docházelo jak při zápisu bootloaderem U-Boot, tak při programování externím programátorem (ASIX Presto). S jinou kartou (či pokud byl slot prázdný) se popisovaná chyba se neprojevila. Čtení pracovalo korektně ve všech případech.

4.7.2 Síťová konfigurace

Nastavení sítě je možné provést několika způsoby:

- použije se nastavení z bootloaderu U-Boot předané přes parametry jádra
- IP adresa a ostatní nastavení se získá z DHCP serveru pomocí klienta **dhcpcd**:

```
root:~> dhcpcd &
```

- nastavení se provede manuálně

```
root:~> ifconfig eth0 192.168.0.1
root:~> ifconfig eth0 up
```

DHCP klienta případně manuální nastavení síťových parametrů je možné spustit automaticky umístěním uvedených příkazů do skriptu **/etc/rc**.

4.7.3 Přístup na sdílené síťové prostředky přes SMB/CIFS

Systém uClinux obsahuje knihovnu SAMBA pro přístup k prostředkům v sítích MS Windows, případně i pro sdílení v těchto sítích. Pro připojení sdílené složky můžeme použít následující příkazy:

```
root:~> mkdir /mnt/mountpoint
root:~> mount -t smbfs -o
     user=username,pass=password //server/folder /mnt/mountpoint
```

Parametry *username*, *heslo*, *server*, *folder*, *mountpoint* je třeba nahradit příslušným uživatelským jménem, heslem, jménem serveru, názvem sdílené složky a názvem adresáře, do kterého chceme sdílenou složku připojit.

V případě, že je při připojení vypsáno chybové hlášení

```
smbfs: mount_data version 1919251317 is not supported
mount: mounting //notebook/shared on /mnt/smb failed
```

znamená to, že síťový prostředek, ke kterému se připojujeme, již nepodporuje starší protokol SMB, takže musíme použít novější protokol CIFS:

```
root:~> mount -t cifs -o
     user=username,pass=password //server/folder /mnt/mountpoint
```

Závažným problémem je, že tento příkaz vyvolá Kernel panic:

```
Data access misaligned address violation
 - Attempted misaligned data memory or data cache access.

...
Call Trace:
[<000016d0>] _name_to_dev_t+0x110/0x250
[<000b77e4>] _kernel_recvmsg+0x0/0x34
[<012cfae4>] _cifs_mount+0x2c7c/0x3078 [cifs]
[<012cfb04>] _cifs_mount+0x2c9c/0x3078 [cifs]
[<00001996>] _kernel_thread_helper+0x6/0xc

Kernel panic - not syncing: Kernel exception
```

Toto je projev chyby v implementaci CIFS. Dle informací nalezených na internetu by v aktuální (SVN) verzi zdrojových kódů měla být tato chyba již opravena.

4.7.4 Telnet server

K přístupu ke konzoli běžícího systému uClinux je možné kromě přímého připojení přes sériový port využít také připojení přes protokol Telnet. Z prostředí Linuxu je možné jednoduše spustit

```
$ telnet 192.168.0.2
```

(případně jiná adresa, podle aktuálního nastavení), uživatelské jméno a heslo není vyžadováno.

Pro připojení z prostředí Windows použijeme např. klient PuTTY.

4.7.5 FTP server

Ve standardní konfiguraci uClinuxu je k dispozici FTP server, který sdílí celý souborový systém. Připojení pomocí FTP serveru můžeme využít k nahrávání uživatelských aplikací, úpravě konfiguračních souborů apod. Přihlásit se k němu můžeme pod uživatelem „root“ s heslem „uClinux“.

Pod systémem Linux můžeme pro připojení využít např. Midnight Commander, kde v menu Left/Right zvolíme „FTP link...“ a následně zadáme IP adresu nebo hostname. Dále je možné standardním způsobem připojit uživatelské jméno, případně i heslo ve tvaru

```
root@192.168.0.2 případně
root:uClinux@192.168.0.2
```

Ve Windows můžeme k připojení využít běžné souborové manažery s příslušnou podporou pro FTP, jako je např. Total Commander nebo Altap Salamander, případně specializované FTP klienty.

4.7.6 HTTP server Boa

Boa je jednoduchý HTTP server určený zejména pro embedded zařízení a jeho vývoj je od začátku zaměřený na rychlost při běhu na systémech s omezeným množstvím prostředků. Na rozdíl

od běžných „velkých“ HTTP serverů, jako je např. Apache, pracuje Boa vnitřně s jediným procesem a nevytváří žádná další vlákna, ani se nedělí na další procesy. Všechny relace vnitřně obsluhuje sekvenčně a funkci *fork()* pro rozvětvení procesu (přesněji řečeno její variantu *vfork*) používá pouze pro spouštění CGI skriptů, které musí být spuštěny jako samostatné procesy. Protože proces spuštěný přes *vfork()* nepracuje paralelně s původním, je nutné, aby spouštěné CGI skripty měly přiměřeně krátkou dobu zpracování, jinak se tím omezí odezva HTTP serveru na další požadavky.

4.7.6.1 Konfigurace HTTP serveru Boa

Vlastní konfigurace serveru se nachází v souboru **/etc/boa.conf**. Do adresáře **/home/httpd** je možno umístit HTML stránky, které bude server poskytovat klientům, implicitně načítaný soubor je **index.html**. CGI skripty je třeba umístit do **/home/httpd/cgi-bin**.

Spuštění Boa na pozadí z příkazové řádky:

```
$ boa -f /etc/boa.conf &
```

Pro automatické spuštění po startu systému je třeba přidat výše uvedený příkaz do souboru **/etc/rc**, což je skript, který je automaticky spuštěn při startu systému uClinux. Pokud je třeba upravit tento soubor v distribuci uClinux (pro zakompilování do image), nachází se v adresáři **vendors/…/…/**.

Pokud chceme přidat některé soubory do kořenového souborového systému, je třeba umístit tyto soubory do adresáře **vendors/…/…/** a modifikovat **vendors/…/…/Makefile** – přidat do něj odkazy na tyto soubory, jako např.:

```
romfs:
...
      $(ROMFSINST) /home/httpd/index.html
      $(ROMFSINST) /home/httpd/tux-wink.gif
      $(ROMFSINST) /home/httpd/tuxsit_small.gif
```

4.8 Vývoj aplikací

Způsob vývoje aplikací pod operačním systémem je v některých aspektech odlišný od vývoje aplikací samostatně běžících bez OS. Největší rozdíl je ve způsobu přístupu k periferiím, protože přímý přístup k nim je plně v režii operačního systému. V této kapitole budou popsány základní kroky potřebné pro kompilaci uživatelských aplikací pro systém uClinux a způsob práce se základními periferiemi.

4.8.1 Kompilace aplikace pro systém uClinux

Kompilaci jednoduchých aplikací demonstrujeme na klasickém příkladu. Vytvoříme nový soubor **helloworld.c**:

```
#include <stdio.h>

int main(int argc, char ** argv)
{
      printf("Hello, World!\n");
      return 0;
}
```

a spustíme kompilaci do formátu FLAT:

```
$ bfin-uclinux-gcc -Wl,-elf2flt helloworld.c -o helloworld
```

Při úspěšné kompilaci vznikne nový soubor **helloworld**, který nahrajeme (např. přes FTP) do souborového systému běžícího systému uClinux a spustíme.

V praxi je výhodné si vytvořit tzv. **Makefile**, ve kterém je definováno, které soubory se do projektu kompilují a jaké jsou mezi nimi závislosti. Příklad triviálního souboru **Makefile**:

```
helloworld: helloworld.c
	bfin-uclinux-gcc -Wl,-elf2flt helloworld.c -o helloworld
```

Žlutou barvou zvýrazněný začátek řádku *musí* být tabulátor (nikoliv tedy například několik mezer apod.). Následuje příklad o něco složitějšího souboru **Makefile**, ve kterém je více cílů kompilace (target) a kde jsou použity proměnné, mj. pro snadnější zápis cest k hlavičkovým souborům a knihovnám funkcí:

```
UCLDIST = /home/blackfin/uClinux-dist.R1.1-RC3

INCLUDES =\
 -I$(UCLDIST)/lib/libpng/\
 -I$(UCLDIST)/lib/zlib/\
 -I$(UCLDIST)/linux-2.6.x/drivers/char/

LIBS = -lm -lpng -lz -L$(UCLDIST)/lib/libpng/\
 -L$(UCLDIST)/staging/usr/lib/

CC = bfin-uclinux-gcc
FLAGS = -Wl,-elf2flt -O2 -ffast-math -mfast-fp

.PHONY: all       # cílům all a clean neodpovídá přímo žádný soubor
.PHONY: clean

all: ppiudp adv7180reg

clean:
	rm -f *.o ppiudp adv7180

ppiudp: ppi.c i2c.c udp.c
	$(CC) $(FLAGS) ppi.c i2c.c udp.c $(INCLUDES) $(LIBS) -o ppiudp

adv7180reg: i2cset.c i2c.c
	$(CC) $(FLAGS) i2cset.c i2c.c $(INCLUDES) $(LIBS) -o adv7180
```

Spuštění kompilace prvního (a tedy implicitního) cíle **all** se poté provede jednoduše spuštěním samotného programu **make**:

```
$ make
```

Pokud chceme konkrétně specifikovat jeden nebo více cílů, zapíšeme je jako parametry:

```
$ make clean ppiudp
```

4.8.2 Knihovna pro zpracování signálů

Pod systém uClinux je portována knihovna „DSP library“, která je původně součásti vývojového prostředí VisualDSP++. Tato knihovna obsahuje kolekci funkcí běžně používaných v oblasti zpracování signálů, jako jsou základní matematické operace nad oborem komplexních čísel, FIR a IIR filtry, interpolační filtry, jednorozměrné a dvourozměrné FFT a konvoluce. Všechny tyto funkce jsou dostupné z jazyka C, přičemž parametry jsou většinou ve formátu fraction, optimalizovaném pro vyjádření čísel v oblasti DSP.

Ilustrační příklad použití funkce pro konvoluci:

```
#include <filter.h>

void test_fr16_conv(void)
{
    fract16 cin1[4] = {-0.1, 0, 0.1, 0.2};
    fract16 cin2[3] = {-0.2, 0, 0.2};
    fract16 cout[6];  // (4 + 3) - 1

    convolve_fr16(cin1, 4, cin2, 3, cout);
}
```

Knihovna se nachází v podadresáři **lib/libbfdsp** a pro použití knihovny je třeba spustit kompilátor s následujícími parametry:

```
$ bfin-uclinux-gcc -Wl,-elf2flt bfdsp_test.c
     -I/home/blackfin/uClinux-dist.R1.1-RC3/lib/libbfdsp/include
     -L/home/blackfin/uClinux-dist.R1.1-RC3/lib/libbfdsp/
     -lm -lbfdsp -o bfdsp_test
```

4.8.3 Přístup k periferiím

Práce s periferiemi pod operačním systémem uClinux je značně odlišná od situace, kdy operační systém nepoužíváme a periferie máme plně pod svou kontrolou. Vzhledem k obecné koncepci systému uClinux je nutné ke všem periferiím přistupovat přes driver, který je součástí systému, případně je samozřejmě možné si ve specifických případech napsat driver vlastní. Na platformě Blackfin je dále aktivní ochrana paměťového prostoru registrů MMR, takže není (až na vyjímky) možné přistupovat k periferiím přímo. V následujících podkapitolách budou popsána práce

se základními periferiemi pod operačním systémem uClinux na aplikační úrovni.

Každá periferie je obvykle v OS reprezentována určitým zařízením, které je přístupné jako soubor nacházející se zpravidla v adresáři **/dev**. Na systémové úrovni je každé zařízení určeno dvěma čísly – *major* a *minor* – kterými je zařízení jednoznačně identifikováno.

Např. pro periferie SPORT je *major* = 237 a *minor* = 0-1 (podle pořadí). Můžeme se o tom přesvědčit výpisem adresáře **/dev**:

```
root:~> ls -l /dev/sport*
crw-rw----    1 root   root   237,   0 Mar 29  2007 /dev/sport0
crw-rw----    1 root   root   237,   1 Mar 29  2007 /dev/sport1
```

Nové soubory reprezentující zařízení je možné vytvářet příkazem **mknod** (bližší podrobnosti viz např. jeho manuálové stránky).

Čtení a zápis z/do zařízení probíhá obvykle přes standardní příkazy pro čtení a zápis souboru, nastavování různých parametrů (a v některých případech i samotná komunikace) se provádí pomocí systémového volání *ioctl()* (I/O Control).

4.8.3.1 I/O piny (Programmable Flags)

Přístup k I/O pinům (Programmable Flags PF0-PF15) je realizován přes zařízení **/dev/pf*x***, kde *x* je číslo PF 0-15 (tedy např. **/dev/pf0** pro PF0).

Pro nastavení parametrů se používají následující volání IOCTL, která odpovídají jednotlivým registrům PF (bližší popis viz datasheet [2]):

- SET_FIO_DIR – Peripheral Flag Direction Register
- SET_FIO_POLAR – Flag Source Polarity Register
- SET_FIO_EDGE – Flag Source Sensitivity Register
- SET_FIO_BOTH – Flag Set on BOTH Edges Register
- SET_FIO_INEN – Flag Input Enable Register

Jednoduchý příklad ilustrující zápis na pin:

```
void pf_example(void)
{
    int pin_level = '0';
    int fd = open("/dev/pf0", O_RDWR);
    ioctl(pfzero, SET_FIO_DIR, 1);
    write(fd, &pin_level, 2);
    close(fd);
}
```

Vlastní změna úrovně na I/O pinu funkcí *write()* trvá přibližně 3 μs, což je způsobeno režií při průchodu požadavku na zápis několika vnitřními úrovněmi operačního systému.

Čtení úrovně na I/O pinu je realizováno obdobně:

```
int read_pf(int fd)
{
    int x;
    read(fd, &x, 2);
    return (x & 0xff) == '1';
}
```

Ve chvíli, kdy je I/O pin aplikací uvolněn (funkcí *close()* ve výše uvedeném příkladě), systém automaticky nastaví všechny jeho registry na hodnotu 0. Znamená to, že pokud na pin zapíšeme hodnotu a ihned jej uvolníme (jako v uvedeném příkladu), výsledkem bude jen úzký impuls na výstupu a poté se pin vrátí do stavu vysoké impedance. Tomuto chování se nedá jednoduše zabránit, pouze úpravou příslušného driveru.

Pro účely ladění je možné vypsat informace o stavu registrů příslušejících jednotlivým I/O pinům:

```
root:~> cat /proc/driver/pflags
PIN     :DATA DIR INEN EDGE BOTH POLAR MASKA MASKB
(1/0)   :H/L  0/I E/D  E/L  B/S   L/H   S/C   S/C
PF0     : 1....0....1....0....0....0.....0.....0
PF1     : 1....0....1....0....0....0.....0.....0
PF2     : 0....0....0....0....0....0.....0.....0
PF3     : 0....0....0....0....0....0.....0.....0
PF4     : 0....0....0....0....0....0.....0.....0
PF5     : 0....0....0....0....0....0.....0.....0
PF6     : 0....0....0....0....0....0.....0.....0
PF7     : 0....0....0....0....0....0.....0.....0
PF8     : 0....1....0....0....0....0.....0.....0
PF9     : 0....0....1....0....0....0.....1.....0
PF10    : 0....0....0....0....0....0.....0.....0
PF11    : 0....0....0....0....0....0.....0.....0
PF12    : 0....0....0....0....0....0.....0.....0
PF13    : 0....0....0....0....0....0.....0.....0
PF14    : 0....0....0....0....0....0.....0.....0
PF15    : 0....0....0....0....0....0.....0.....0
```

4.8.3.2 PPI

Pro práci s rozhraním PPI jsou k dispozici dva možné drivery, **bfin_ppi** („Blackfin BF5xx PPI Driver“) a **bfin_ppifcd** („Blackfin BF5xx PPI Camera frame capture driver“). První z nich je obecný driver, který poskytuje aplikacím prakticky všechny základní funkce rozhraní PPI, druhý z nich je, jak už název napovídá, zaměřený především na grabování obrazu. Použijeme tedy driver **bfin_ppifcd** (nastavení konfigurace jádra viz kapitola 4.6.4).

Rozhraní PPI je v systému reprezentováno zařízením **/dev/ppi0**. Práce s ním je poměrně jednoduchá. Základní volání IOCTL pro nastavení parametrů jsou:

- CMD_PPI_SET_PIXELS_PER_LINE – počet pixelů (přesněji bytů) na řádek
- CMD_PPI_SET_LINES_PER_FRAME – počet řádků ve snímku
- CMD_PPI_SET_PPICONTROL_REG – nastavení řídícího registru PPI

Parametr předávaný při IOCTL volání CMD_PPI_SET_PPICONTROL_REG je hodnota pro nastavení registru PPI_CONTROL. Bližší podrobnosti o možnostech nastavení viz datasheet [2].

Následuje jednoduchý příklad přečtení snímku pomocí rozhraní PPI:

```
void ppi_read_image(int width, int height, char * buffer)
{
    int fd = open("/dev/ppi0", O_RDONLY, 0);
    ioctl(fd, CMD_PPI_SET_PIXELS_PER_LINE, 2 * widht);
    ioctl(fd, CMD_PPI_SET_LINES_PER_FRAME, height);
    ioctl(fd, CMD_PPI_SET_PPICONTROL_REG,
        FLD_SEL | PACK_EN | DLEN_8);
    read(fd, buffer, 2 * width * height);
    close(fd);
}
```

4.8.3.3 I2C

Sběrnice I2C je přístupná přes zařízení **/dev/i2c-0**. Pro komunikaci po tomto rozhraní jsou klíčové dvě struktury:

- *i2c_msg*, která reprezentuje jednu zprávu – čtení nebo zápis
- *i2c_rdwr_ioctl_data*, která reprezentuje frontu zpráv ke zpracování

Každá zpráva (typ *i2c_msg*) obsahuje adresu cílového zařízení na sběrnici (pouze vyšších 7 bitů, tedy rozsah 0-127), volbu režimu čtení/zápis, délku zprávy a pointer na buffer. Vlastní čtení/zápis je proveden systémovým voláním *ioctl()*.

Následuje jednoduchý příklad použití I2C pro zápis registru obvodu ADV7180:

```
#define I2C_DEVID          (0x40 >> 1)     // I2C address 0x40

void adv7180_write_reg(unsigned char reg, unsigned char value)
{
    char msg_data[32];
    struct i2c_msg msg = {
        .addr = I2C_DEVID,     // addr
        .flags = 0,            // flags: 0 write, 1 read
        .len = 2,              // message length
        .buf = msg_data // message pointer
      };
    struct i2c_rdwr_ioctl_data rdwr = {&msg, 1}; // one message

    int fd = open("/dev/i2c-0", O_RDWR);
    msg_data[0] = reg;
    msg_data[1] = value;
    ioctl(fd, I2C_RDWR, &rdwr);
    close(fd);
}
```

a dále příklad realizace čtení registru téhož obvodu. Zároveň je v příkladu patrné zřetězení zpráv do fronty:

```
void adv7180_read_reg(unsigned char reg, unsigned char * value)
{
    struct i2c_msg msgs[2] = {
        {I2C_DEVID, 0, 1, &reg},     // write 1 byte - reg
        {I2C_DEVID, 1, 1, value}     // read 1 byte - value
    };
    struct i2c_rdwr_ioctl_data rdwr = {msgs, 2}; // two messages

    int fd = open("/dev/i2c-0", O_RDWR);
    ioctl(fd, I2C_RDWR, &rdwr);
    close(fd);
}
```

4.8.3.4 SPORT

Rozhraní SPORT jsou reprezentována zařízeními **/dev/sport0** a **/dev/sport1**. Konfigurace rozhraní se provádí pomocí IOCTL SPORT_IOC_CONFIG. Jako parametr se předává struktura *sport_config*, která obsahuje všechny potřebné parametry.

Příklad použití rozhraní SPORT1 pro nastavení LED a výstupů přepojených na SPORT přes sériový registr (viz kapitola 3.2.10.2):

```
void sport1_out(unsigned char x)
{
      int fd, data[] = {x, x, x};
      struct sport_config config = {0};

      fd = open("/dev/sport1", O_RDWR, 0);

      config.act_low = 1;
      config.late_fsync = 1;
      config.tckfe = 1;
      config.int_clk = 1;
      config.word_len = 32;
      config.serial_clk = 32000000;
      config.fsync_clk = config.serial_clk / config.word_len;
      config.dma_enabled = 1;

      ioctl(fd, SPORT_IOC_CONFIG, &config);
      write(fd, &data, sizeof(data));
      close(fd);
}
```

V příkladu je na první pohled zbytečně použitý režim DMA, i když je vlastně potřeba přenést jediné datové slovo. Driver ale pravděpodobně obsahuje chybu, protože při vypnutí režimu DMA (*config.dma_enabled* = 0) dojde při zápisu na SPORT k systémovému selhání. Proto je nutné použít režim DMA. Dále bylo zjistěno, že pro korektní funkci je nutné odeslat minimálně tři datová slova, což je ve výše uvedeném zdrojovém kódu zohledněno.

4.8.3.5 USB

Obvod Cypress CY7C68013A, který na desce realizuje interface pro sběrnici USB, je připojen 16 bitovou datovou sběrnicí k asynchronní bance 1 procesoru Blackfin a dále k jeho I/O pinům (PF). Čtení a zápis dat do/z rozhraní USB je tedy realizováno jako operace nad datovým slovem umístěným na absolutní adrese 0x20100000 v lineárním paměťovém prostoru procesoru. I/O piny slouží pro výběr endpointu, kontrolu zaplněnosti FIFO a případné okamžité odeslání paketu.

Způsob práce s I/O piny PF byl popsán v kapitole 4.8.3.1. Pro přístup k datům použijeme prosté čtení a zápis absolutně definovaného paměťového místa:

```
unsigned short * const usb   = (unsigned short *) 0x20100000;
unsigned short * const dummy = (unsigned short *) 0x20200000;

void write_usb(unsigned short i)
{
    *usb = i;
    *dummy = i;
}

unsigned short read_usb(void)
{
    return *usb;
}
```

Tato metoda není úplně korektní s ohledem na použití pod operačním systémem uClinux, nicméně umožňuje nám jednoduchý přístup na rozhraní USB, aniž bychom museli psát specializovaný ovladač pro toto zařízení. Pro použití této jednoduché metody musíme zajistit, aby v danou chvíli přistupovala na USB jen jediná aplikace. Ve složitějších případe je vhodné implementovat zmíněný ovladač (driver).

Důvod použití druhého zápisu do banky 2 je vysvětlen v kapitole 3.3.3. Rychlost přenosu nebyla otestována z důvodů problému s enumerací v High-speed režimu USB (viz kapitola 3.2.5.4). Testy prováděné ve Full-speed režimu nejsou pro předpokládané použití (přenos videosignálu v reálném čase) relevantní.

4.8.3.6 Síť

Pro realizaci síťové komunikace protokoly TCP a UDP je k dispozici obvyklé rozhraní pro práci se sockety odpovídající standardu POSIX.

Pro zahájení komunikace nejprve převedeme jméno serveru na IP adresu a vyplníme strukturu *sockaddr_in*, která obsahuje tuto adresu a číslo portu:

```
sockaddr_in remote_ip_and_port(char * dest, int port)
{
    struct hostent * h = gethostbyname(dest);
    sockaddr_in remote;

    remote.sin_family = h->h_addrtype;
    memcpy((char *)&remote.sin_addr.s_addr,
        h->h_addr_list[0], h->h_length);
    remote.sin_port = htons(port);

    return remote;
}
```

Dále vytvoříme TCP spojení následujícím způsobem:

```
int tcp_sckt;

int tcp_connect(char * dest, int port)
{
    sockaddr_in remote = remote_ip_and_port(dest, port);

    tcp_sckt = socket(PF_INET, SOCK_STREAM, 0);
    connect(tcp_sckt, (struct sockaddr *)&remote,
        sizeof(remote));

    return tcp_sckt;
}
```

Analogicky pro UDP:

```
struct sockaddr_in client, remote;
int udp_sckt;

void udp_init(char * dest, int port)
{
    remote = remote_ip_and_port(dest, port);

    udp_sckt = socket(PF_INET, SOCK_DGRAM, 0);

    client.sin_family = AF_INET;
    client.sin_addr.s_addr = htonl(INADDR_ANY);
    client.sin_port = htons(0);
    bind(sckt, (struct sockaddr *)&client, sizeof(client));
}
```

Pro odeslání dat protokolem TCP slouží funkcí *send(),* pro UDP funkce *sendto()*:

```
void tcp_send(char * data, int size)
{
    send(tcp_socket, data, size);
}

void udp_send(char * data, int size)
{
    sendto(udp_sckt, data, size, 0,
        (struct sockaddr *)&remoteServAddr, sizeof(remoteServAddr));
}
```

Ověření parametrů síťového rozhraní bylo provedeno dvěma různými testy rychlosti přenosu dat směrem do PC. Pro vyzkoušení maximální dosažitelné přenosové rychlosti bylo provedeno odesílání dat ze zařízení **/dev/zero** pomocí standardní utility **netcat**. Tímto způsobem bylo dosaženo rychlosti 8,6 MB/s při přenosu protokolem UDP a 6,1 MB/s při přenosu protokolem TCP.

Dále byla implementována aplikace, která snímá digitalizovaný obraz z kamery přes rozhraní PPI (pro příjem dat používá zařízení **/dev/ppi0** s driverem **ppifcd**) a přenáší jej do PC. Kód pro odesílání dat síťovými protokoly je přímo integrovaný v aplikaci. Tato situace odpovídá reálnému použití při přenosu nekomprimovaného obrazu a tímto způsobem byly dosaženy rychlosti 6,3 MB/s (UDP) a 3,8 MB/s (TCP).

4.9 Řešení problémů

4.9.1 Problémy s velikostí kořenového souborového systému

Chybové hlášení při sestavování obrazu jádra:

```
bfin-uclinux-genext2fs: couldn't allocate a block (no free space)
```

což znamená, že souborový systém (rootfs) se nevejde do alokovaného prostoru. Je možné upravit **vendors/…/…/Makefile** a zvětšit velikost rootfs/romfs (…), pokud máme dostatek prostoru na cílovém médiu (…) . Pokud ne, je potřeba zmenšit velikost souborů. Nejprve vypíšeme několik (20) největších souborů v **.romfs**, seřazených podle velikosti:

```
~/uClinux-dist.R1.1-RC3 $ find -P ./romfs/ -type f -printf "%s
     %p\n" | sort -nr | head -n 20
```

a dále podle podle vlastního uvážení odstraníme velké a zbytečné soubory (např. sqlite apod.) tím, že odstraníme příslušné volby z konfigurace jádra a smažeme přebytečné soubory z **romfs**. Je možné smazat i kompletní obsah tohoto adresáře, případně spustit **make clean**, ale kompilace pak bude trvat samozřejmě déle (soubory, které byly potřeba, musí být znovu vytvořeny).

4.9.2 Problém s připojením kořenového souborového systému

Při bootování jádra se objeví chybové hlášení:

```
Memory map:
     výpis neobsahuje rootfs
...
No filesystem could mount root, tried:  ext2 romfs
Kernel panic - not syncing: VFS: Unable to mount root fs on unknown-
block(31,0)
```

Pro správnou funkci je třeba povolit v konfiguraci kernelu podporu pro „Generic uClinux ROM/RAM filesystem“:

```
Device Drivers  --->
  Memory Technology Devices (MTD)  --->
    <*> Memory Technology Device (MTD) support
    [*]   MTD partitioning support
          Mapping drivers for chip access  --->
            <*> Generic uClinux RAM/ROM filesystem support
```

4.9.3 Problém s načtením kořenového souborového systému

Při bootování jádra se objeví následující nebo obdobné chybové hlášení:

```
EXT2-fs error (device mtdblock0): ext2_check_page: bad entry in
directory #5158: rec_len is smaller than minimal - offset=0,
inode=0, rec_len=0, name_len=0
```

Problém je způsoben bootloaderem U-Boot, ve kterém je nastavené omezení na maximální velikost dekomprimovaného obrazu – implicitně 8 MB. Při překročení této velikosti U-Boot nevypíše žádné chybové hlášení, ovšem výsledný obraz je chybný. Je třeba doplnit do konfiguračního souboru **include/configs/myboard.h** definici

```
#define CFG_BOOTM_LEN   0x1000000
```

Hodnota musí být větší než je velikost dekomprimovaného obrazu jádra – 16 MB je hodnota, která by měla být ve většině případů dostačující.

4.9.4 Řešení problémů přetečení stacku

Jak již bylo zmíněno v úvodu, z důvodu absence MMU má stack aplikací v systému uClinux vždy pevnou velikost stanovenou při kompilaci.

Kompilátor gcc umožňuje na platformě Blackfin/uClinux zapnout kontrolu, zda nedošlo při běhu aplikace k přetečení stacku. Pokud není tato možnost aktivována, může při přetečení stacku dojít k přepsání dat nebo dokonce kódu aplikace, což může vést k nedefinovanému chování aplikace a v horším případě i celého systému.

Pokud chceme tuto kontrolu zapnout, přidáme parameter *--fstack-limit-symbol=_stack_start* do příkazové řádky při spouštění kompilátoru. Do generovaného kódu bude pak přidán speciální kód, který testuje, zda nebyla překročena velikost stacku. Pokud ano, aplikace je ukončena s příslušným chybovým hlášením. Ve finální verzi aplikace můžeme potom tuto kontrolu opět vynechat pro zvýšení výkonu a zmenšení velikosti kódu a nebo naopak ponechat pro zvýšení robustnosti.

V případě, že zjistíme, že standardní hodnota velikosti stacku (4 KB) není dostačující, můžeme ji změnit. První možnost je provést to již v čase překladu doplněním parametru pro konverzní utilitu **elf2flt** (změna na velikost 8 KB):

```
$ bfin-uclinux-gcc -Wl,-elf2flt="-s 8192"
     helloworld.c -o helloworld
```

Pokud chceme zjistit nastavení velikosti stacku a další informace, použijeme utilitu **flthdr**, která umožňuje zobrazit a změnit nastavení hlavičky FLAT formátu:

```
$ bfin-uclinux-flthdr filename
```

Změna velikosti stacku (na 8 KB):

```
$ bfin-uclinux-flthdr -s 8192 filename
```

4.10 Zpracování obrazu na platformě uClinux

Pro porovnání výkonu v oblasti zpracování obrazových signálů mezi aplikacemi běžícími pod OS uClinux a nativními aplikacemi bez operačního systému byly na platformu uClinux portovány zdrojové kódy několika funkcí, realizujících některé základní operace z této oblasti. Implementace těchto kódů byla provedena v rámci diplomové práce [5], odkud byly kódy převzaty. Teoretický rozbor činnosti testovaných funkcí je popsán tamtéž.

Pro účel porovnání výkonu byla napsána testovací utilita, která vytvoří náhodný obraz zadané velikosti a spouští ve smyčce zvolenou funkci po zadaný počet iterací. Měří se celkový čas běhu, z čehož je vypočítána doba trvání jedné iterace, uvedená v tabulce. Jako referenční platforma pro srovnání je uvažováno použití vývojového prostředí Analog Devices VisualDSP++ s integrovaným kompilátorem.

Testovány byly následující funkce:

- *Histogram()* výpočet histogramu
- *CenterOfGravity()* výpočet těžiště obrazu
- *Edge_Sobel()* hledání hran pomocí Sobelova konvolučního operátoru
- *Blur()* odstranění šumu pomocí konvoluční filtrace
- *ImTreshold()* prahování obrazu

Naměřené výsledky jsou shrnuty v tab. 4.2. Bylo testováno 1000 iterací každé funkce nad náhodně generovaným snímkem o velikosti 664×504 pixelů.

Tab. 4.2 Srovnání doby zpracování jedné iterace algoritmů na různých platformách

funkce	uClinux [ms]	VDSP / SDRAM, SCLK 133MHz [ms]	VDSP / SRAM, SCLK 100 MHz [ms]
Histogram	11	15	20
CenterOfGravity	20	23	25
Edge_Sobel	70	33	40
Blur	34	32	35
ImTreshold	14	17	13

Z tabulky je patrné že u většiny z testovaných algoritmů byl běh pod systémem uClinux nejrychlejší. Tento na první pohled překvapivý závěr lze vysvětlit tím, že uClinux využívá standardně vyrovnávací paměť cache při přístupu do SDRAM, zatímco v případě nativního běhu cache použita nebyla. Běh na systému s SRAM (videosenzor vyvinutý v práci [5]) byl naopak ve většině případů nejpomalejší, což lze vysvětlit výrazně menší přenosovou rychlostí u paměti SRAM proti SDRAM.

Nečekaný je výsledek běhu funkce *Edge_Sobel* pod systémem uClinux. Délka zpracování je přibližně dvakrát větší, než při nativním běhu na obou platformách. Tuto anomálii se nepodařilo kvalifikovaně vysvětlit, nicméně příčina bude pravděpodobně v pomalé práci s pamětí z důvodu neefektivního využití paměti cache.

5. Závěr

Cílem této diplomové práce byla realizace modulu pro zpracování obrazu, který by byl vhodný pro komfortní práci s barevným videosignálem a obsahoval rychlá komunikační rozhraní pro přenos výsledků měření a obrazu do nadřazeného systému. Měl by také umožňovat implementaci operačního systému uClinux.

Modul zpracování obrazu byl navržen a úspěšně realizován. Obsahuje digitální signálový procesor ADSP Blackfin BF-532 a paměť SDRAM o kapacitě 32 MB, která umožňuje pohodlnou manipulaci s digitalizovaným obrazem. To je velká výhoda ve srovnání s vývojovou deskou a videosenzorem s procesorem Blackfin, vyvinutými na katedře měření FEL ČVUT, které obsahovaly pouze paměť SRAM o maximální kapacitě 1 MB. Celková složitost zapojení si vynutila použití 4-vrstvé desky plošných spojů.

Pro rychlou komunikaci s nadřazeným systémem je k dispozici rozhraní USB s obvodem Cypress EZ-USB. Tento obvod podporuje režim USB 2.0 High-speed a měl by při připojení na sběrnici procesoru Blackfin umožňovat přenos obrazových dat do PC rychlostí až 15 MB/s. Při připojení k PC se ovšem vyskytl problém s enumerací v režimu High-speed, proto bohužel nebylo možné rychlý přenos vyzkoušet. Dále je na desce modulu implementován řadič 100 Mbit/s Ethernetu. Přes toto rozhraní byl prakticky ověřen přenos dat rychlostí 8,6 MB/s protokolem UDP a 6,1 MB/s protokolem TCP.

Digitalizace videosignálu je realizována externím modulem s obvodem ADV7180, který byl taktéž navržen v rámci této práce. Ten je možné použít jak ve spojení s realizovanou základní deskou, tak jako samostatný blok, kterým umožňuje spolupráci např. s jinými procesory nebo s hradlovými poli FPGA.

Mechanický formát a datové rozhraní modulu digitalizace jsou kompatibilní s moduly CMOS snímačů používanými v laboratoři videometrie. Výhodou tohoto řešení je možnost použití těchto existujících CMOS kamer s obrazovým procesorem realizovaným v této práci a stejně tak použití stávajících modulů zpracování ve spojení s realizovaným blokem digitalizace videosignálu. Blok digitalizace byl odzkoušen na dvou základních deskách s procesorem ADSP BF-532, z nichž jedna je předmětem této práce.

Pro zvýšení rychlosti zpracování obrazu byla teoreticky diskutována možnost spolupráce procesoru Blackfin s hradlovými poli FPGA a byly navrženy možnosti realizace jejich propojení.

Významná část textu byla věnována problematice použití operačního systému uClinux, což je varianta OS GNU/Linux určená pro systémy bez podpory MMU (jednotky pro správu paměti). Byly zmíněny základní vlastnosti a možnosti tohoto systému a také odlišnosti od systému Linux, které jsou způsobeny principiálními omezeními platformy, mezi které patří zejména absence virtuálního paměťového prostoru a z toho plynoucí důsledky.

uClinux umožňuje běh vícevláknových aplikací a poskytuje širokou podporu pro různé periferie jako jsou síťová rozhraní, pevné disky a paměťové karty, videodekodéry a enkodéry či obvody pro práci ze zvukem. Využití operačního systému v aplikacích zpracování obrazu může být výhodné např. díky možnostem snadné implementace síťové komunikace pro realizaci propojení s nadřazeným systémem. Podpora přístupu na různá paměťová média včetně práce se souborovými systémy může být výhodná pro ukládání naměřených dat v offline aplikacích.

Realizovaný modul s procesorem Blackfin a pamětí SDRAM svou koncepcí umožňuje implementaci uClinuxu a tento systém byl také na desce úspěšně zprovozněn. Bylo nutné portovat na daný hardware bootloader U-Boot, který slouží pro zavádění systému a dále nakonfigurovat jádro operačního systému uClinux pro optimální využití prostředků základní desky. Úpravy, které je třeba provést pro zprovoznění systému na vlastním hardware jsou v textu popsány.

Způsob práce s periferiemi pod OS uClinux je diametrálně odlišný od jejich přímého použití v případě absence operačního systému. Čtenář je proto seznámen s používáním různých periferií v uživatelských aplikacích a dále se základy komunikace s využitím síťových protokolů UDP a TCP.

Domnívám se, že se podařilo úspěšně splnit cíle stanovené v úvodu práce i pokyny uvedené v zadání. Navržený modul pro zpracování obrazu byl realizován a zprovozněn, stejně tak jako modul pro digitalizaci obrazu. Dále se podařilo na daném hardware plně implementovat operační systém uClinux a zprovoznit pod ním mj. zpracování dat z videodekodéru a komunikaci po Ethernetu a USB. Kapitola pojednávající o operačním systému uClinux byla nad rámec zadání koncipována tak, aby mohla sloužit jako referenční manuál, obsahující maximum ze základních informací nutných pro úspěšné použití tohoto systému při realizaci různých aplikací. Použití systému uClinux jako platformy pro vývoj aplikací v oblasti bezdotykového měření považuji za perspektivní.

6. Seznam obrázků

7. Seznam použitých symbolů a zkratek

CIFS – Common Internet File System – novější implementace protokolu SMB

CGI – Common Gateway Interface – softwarové rozhraní pro předávání parametrů

CCD – Charge Coupled Device – obvod s vazbou nábojem

CMOS – Complementary Metal Oxid Semiconductor - označení technologie výroby integrovaných obvodů

CVBS – podle různých výkladů "Color, Video, Blank and Sync", "Composite Video Baseband Signal", "Composite Video Burst Signal" nebo také "Composite Video with Burst and Sync" – kompozitní videosignál

DPLL – Digital Phase Locked Loop – digitálně realizovaná smyčka fázového závěsu

DSP – Digital Signal Processing, Digital Signal Processor – digitální zpracování signálu, signálový procesor

FPGA – Field Programmable Gate Array – hradlové pole programovatelné v aplikaci

HDTV – High-definition television – standard pro přenos televizního obrazu s vysokým rozlišením

HTTP – HyperText Transfer Protocol – protokol pro přenos dokumentů po internetu (webové stránky)

I2C – IIC, Inter-Integrated Circuit – dvouvodičová datová sběrnice s nízkou rychlostí

MMU – Memory Management Unit – jednotka pro správu paměti; umožňuje mj. virtualizaci a ochranu paměťového prostoru

MMR – Memory Mapped Register – paměťově mapovaný konfigurační registr

NTSC – National Television Standards Committee (norma pro kódování analogového televizního signálu)

PAL – Phase Alternating Line (norma pro kódování analogového televizního signálu)

PLL – Phase Locked Loop – smyčka fázového závěsu

RGB – Red, Green, Blue – barevný model aditivního mísení barev; přeneseně barevný prostor

S-Video – viz Y/C

SAMBA – Opensource implementace protokolu SMB

SDRAM – Synchronnous Dynamic Random Access Memory – synchroní paměť

SECAM – Séquentiel couleur à mémoire ("Sequential Color with Memory") – norma pro kódování analogového televizního signálu

SMB – Server Message Block – protokol používaný pro sdílení prostředků v sítích realizovaných na platformě Microsoft Windows

SPI – Serial Peripherial Interface – rychlé synchronní sériové rozhraní

SVN – Subversion – systém pro správu verzí; novější alternativa k CVS

TFTP – Trivial File Transfer Protocol – velmi jednoduchý protokol pro přenos souborů po síti

USB – Universal Serial Bus – sériová sběrnice používaná pro připojení zařízení k PC na malou vzdálenost

VBI – Vertical Blanking Interval – doba zpětného běhu paprsku po konci snímku

Y/C – Luminance/Chrominance – barevný videosignál s oddělenou jasovou a barvonosnou složkou (S-Video)

YCbCr – Luma, Chroma-blue, Chroma-red – označení pro barevný prostor používaný pro digitální reprezentaci videosignálu

YPbPr – analogové vyjádření signálů v barevném prostoru YCbCr

8. Seznam použité literatury

[1] ADSP-BF531/ADSP-BF532/ADSP-BF533 Blackfin® Embedded Processor, Analog Devices, 2007

[2] ADSP-BF533 Hardware Reference, Analog Devices, 2006

[3] ADSP-BF53x/BF56x Blackfin® Processor Programming Reference, revision 1.2, Analog Devices, 2007

[4] Silicon Anomaly List ADSP-BF531/ADSP-BF532/ADSP-BF533, Analog Devices, September 2007

[5] *Pribula, O.*: Diplomová práce, ČVUT-FEL, 2007

[6] *Martínek, Š.*: Diplomová práce, ČVUT-FEL, 2005

[7] *Smith S., W.*: The Scientist and Engineer's Guide to Digital Signal Processing. Second Edition, California Technical Publishing, San Diego, 1999

[8] web firmy Analog Devices, Inc. http://www.analog.com/

[9] web firmy Intersil Corporation http://www.intersil.com/

[10] web firmy NXP Semiconductors http://www.nxp.com/

[11] web firmy Texas Instruments http://www.ti.com/

[12] web firmy Digikey http://www.digikey.com/

[13] ADV7180 10-Bit, 4× Oversampling SDTV Video Decoder, Analog Devices, 2007, http://www.analog.com/UploadedFiles/Data_Sheets/ADV7180.pdf

[14] Synchronous DRAM MT48LC16M16A2 – 4 Meg × 16 × 4 banks, Micron Technology, Inc., 2007, http://download.micron.com/pdf/datasheets/dram/sdram/256MSDRAM.pdf

[15] LAN91C111, 10/100 Ethernet Single Chip MAC + PHY, SMSC, 2005, http://www.smsc.com/main/datasheets/91c111.pdf

[16] CY7C68013A/CY7C68014A EZ-USB FX2LP™ USB Microcontroller, Cypress, 2006, http://download.cypress.com.edgesuite.net/design_resources/datasheets/contents/cy7c68013a_8.pdf

[17] EZ-USB Technical Reference Manual Version, 1.4, Cypress, 2006, http://download.cypress.com.edgesuite.net/design_resources/technical_reference_manuals/contents/ez_usb_r___technical_reference_manual__trm__14.pdf

[18] Blackfin Linux Project, http://blackfin.uclinux.org/gf/

[19] BlackfinOne BF532, http://blackfin.uclinux.org/gf/project/bf1/

[20] QEMU open source processor emulator, http://fabrice.bellard.free.fr/qemu/

[21] Recommendation ITU-R BT.656, ITU Radiocommunication Assembly, 1998

[22] Recommendation ITU-R BT.601, ITU Radiocommunication Assembly, 1998

[23] VisualDPS++ 4.5 C/C++ Compiler and Library Manual for Blackfin Processors, Revision 4.0, Analog Devices, 2006, http://www.analog.com/UploadedFiles/Associated_Docs/613293690438645_Blackfin_comp_man.pdf

[24] *Kokaly-Bannourah, M.*: SDRAM Selection Guidelines and Configuration for ADI Processors, Analog Devices, 2004, http://www.analog.com/UploadedFiles/Application_Notes/53047287221168EE210v02.pdf

[25] *Sanghai, K.*.: System Optimization Techniques for Blackfin® Processors, Analog Devices, 2007, http://www.analog.com/UploadedFiles/Application_Notes/97018720EE_324_Rev_1_07_2007.pdf

[26] *Jack, K.*: AN9717 YCbCr to RGB Considerations, Intersil, 1997, http://www.intersil.com/data/an/an9717.pdf

[27] *Krapfenbauer, H.*: Embedded Web Radio (Masterarbeit), Institut für Computertechnik, TUW, 2007, http://publik.tuwien.ac.at/files/pub-et_12836.pdf

[28] *Hennerich, M.*: uClinux on the Blackfin DSP Architecture, 2006, http://www.embedded.com/showArticle.jhtml?articleID=185300517 http://www.embedded.com/showArticle.jhtml?articleID=185301021 http://www.embedded.com/showArticle.jhtml?articleID=185300537

[29] *McCullough, D.*: uClinux for Linux Programmers, 2004, http://www.linuxjournal.com/article/7221

[30] *Vinogradov, Y.*: AN3408 Building a Sample CGI Application, 2007, http://www.freescale.com/files/32bit/doc/app_note/AN3408.pdf

[31] Xming X Server for Windows, http://sourceforge.net/projects/xming

[32] PuTTY: a free telnet/ssh client, http://www.chiark.greenend.org.uk/~sgtatham/putty/

[33] Gentoo Linux, http://www.gentoo.org/

[34] *Collins-Sussman B., Fitzpatrick, B. W. Pilato C. M.*: Version Control with Subversion (for Subversion 1.4), 2007, http://svnbook.red-bean.com/en/1.4/svn-book.pdf

[35] Free SFTP, FTP and SCP client for Windows, http://winscp.net/eng/docs/lang:cs

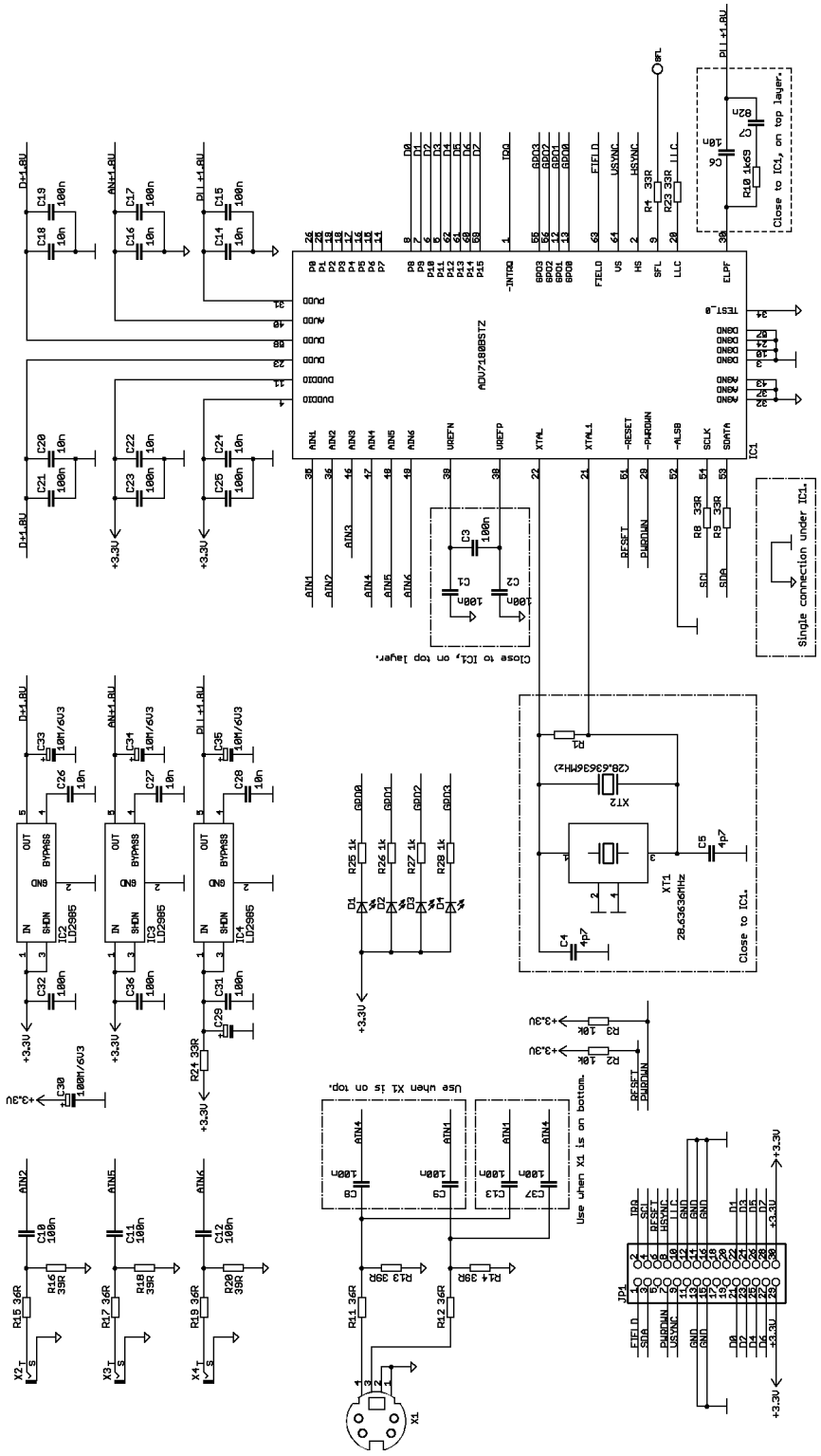
[36] Tftpd32 - DHCP, TFTP, SNTP and Syslog server, TFTP client, http://tftpd32.jounin.net/

9. Přílohy

9.1 Modul digitalizace videosignálu

Schéma zapojení, návrh DPS, fotodokumentace hotového modulu.

9.1.1 Schéma zapojení

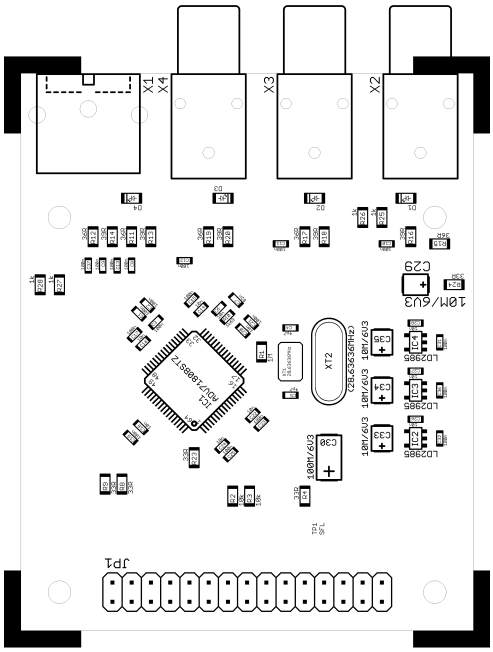


Obr. 9.1 Schéma zapojení modulu digitalizace

9.1.2 Náhled DPS

zvětšení 100%, vnitřní hrany rohových značek určují rozměr

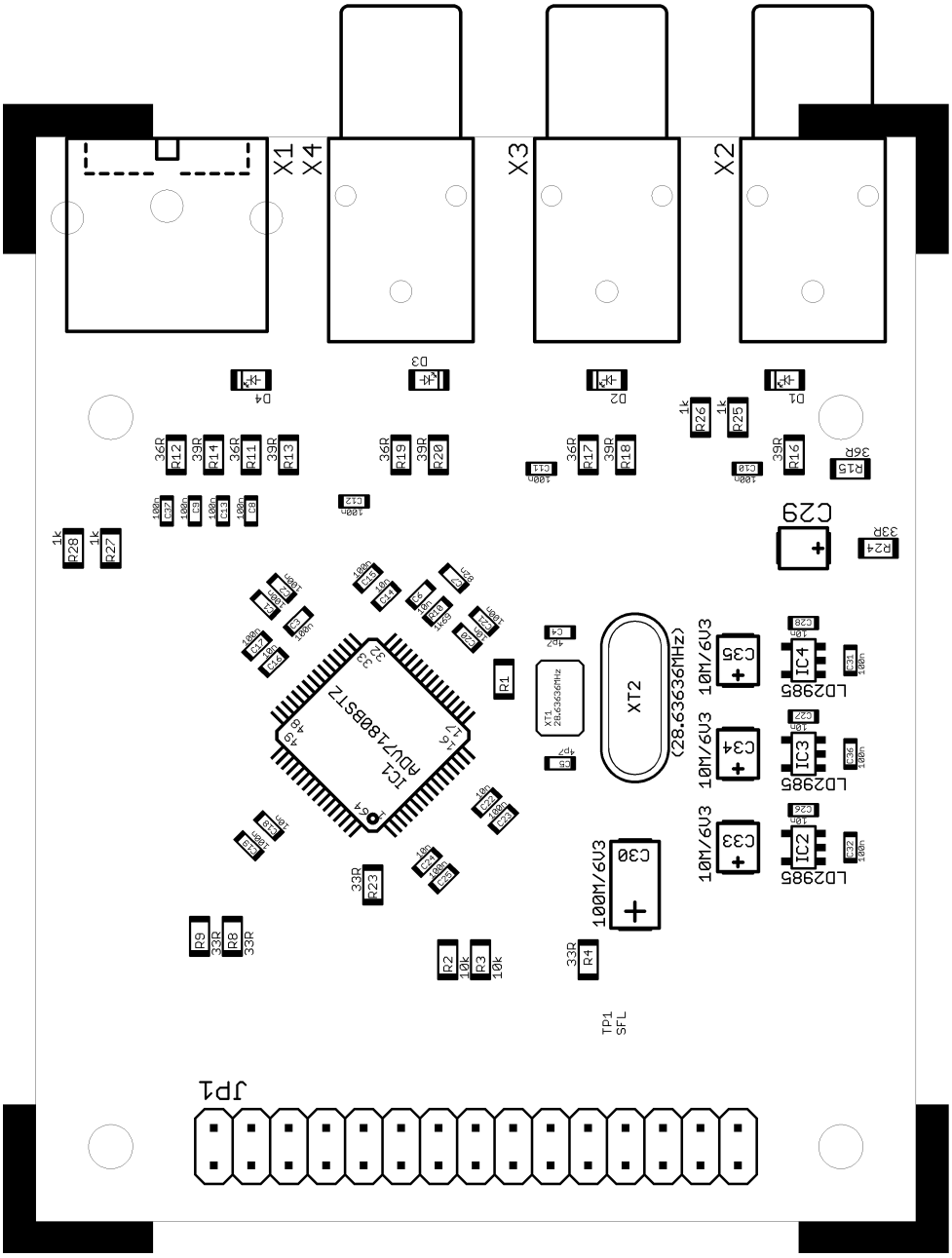
rozměr DPS: 59,7 x 73,7 mm



Obr. 9.2 Modul digitalizace – náhled DPS

9.1.3 Rozmístění součástek

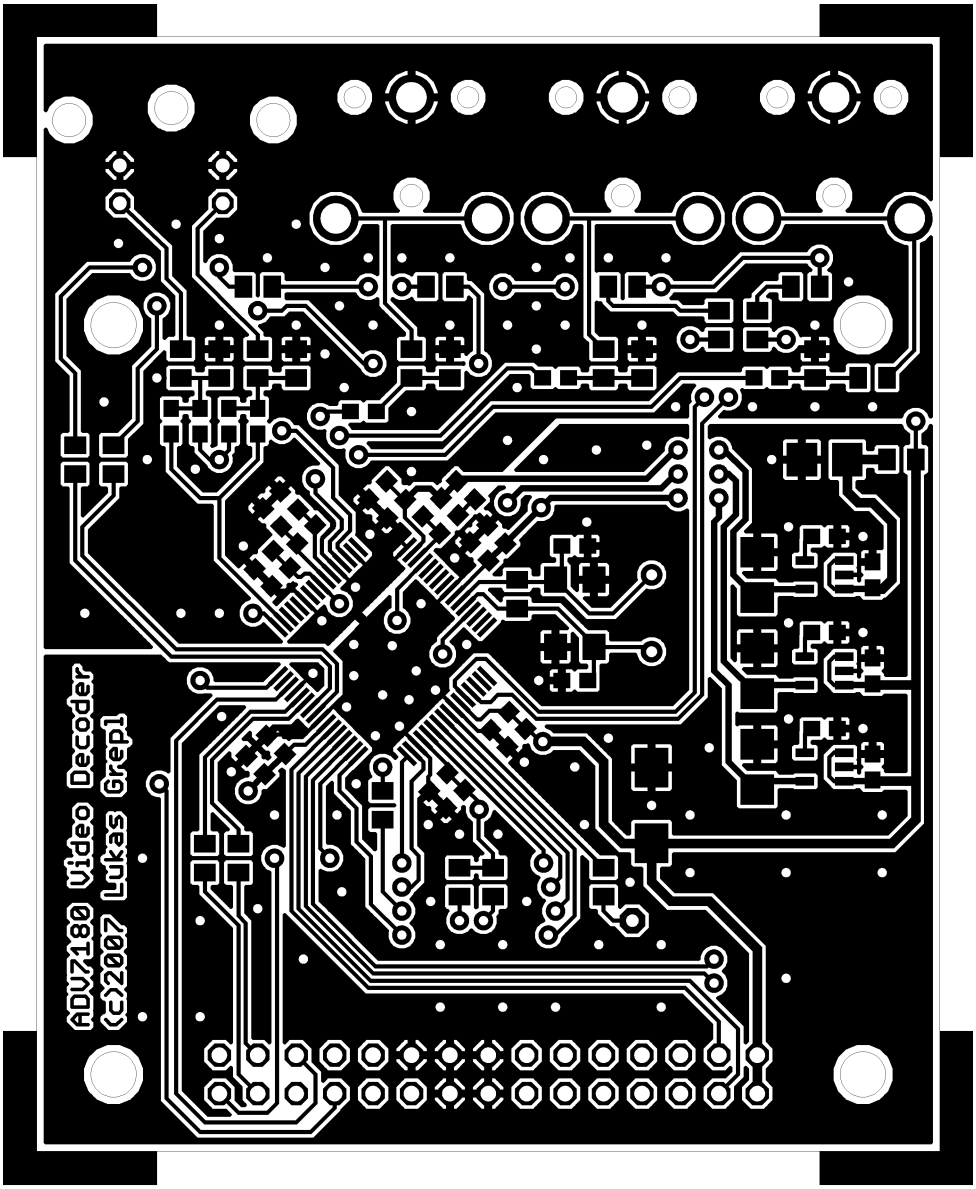
zvětšení 200%



Obr. 9.3 Modul digitalizace – rozmístění součástek

9.1.4 TOP layer

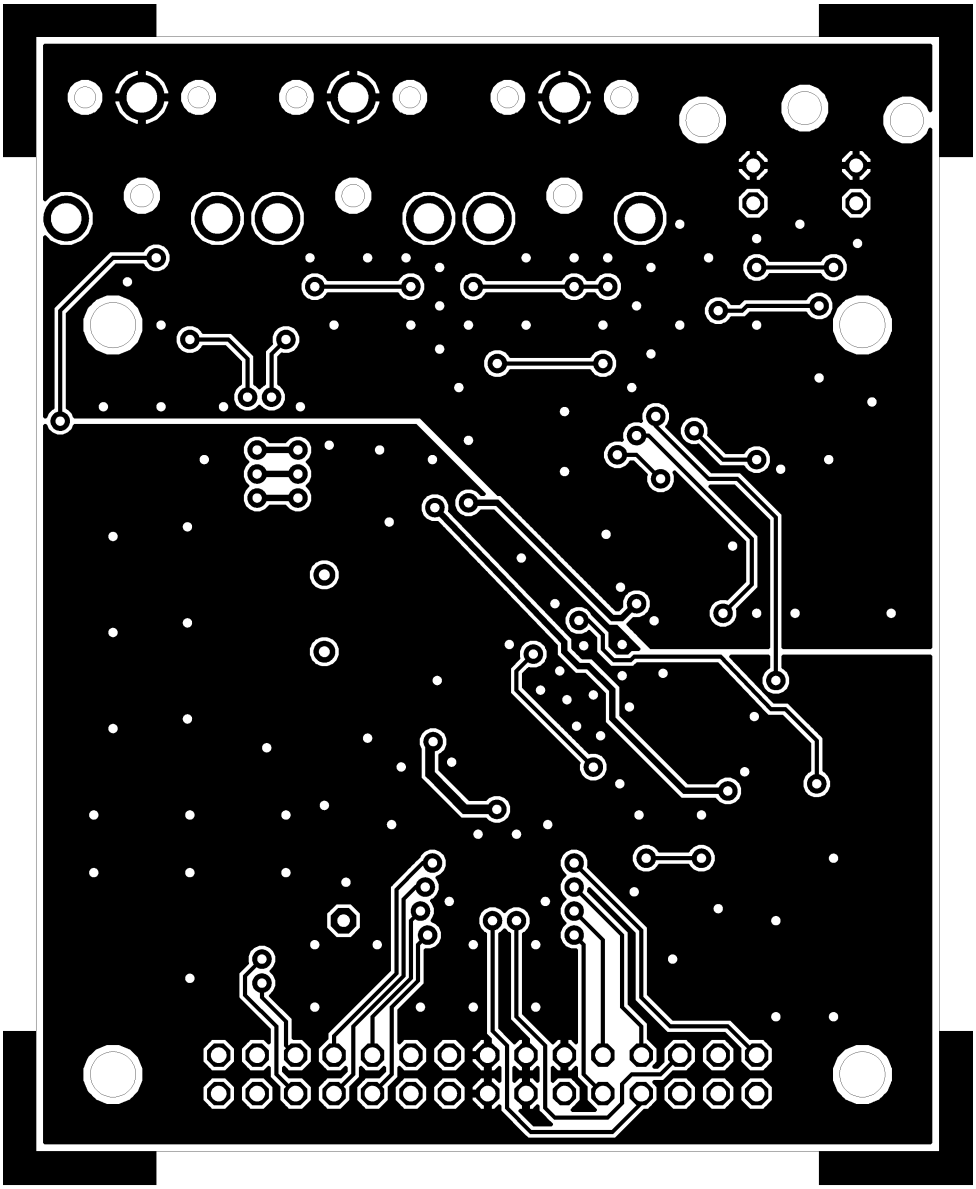
zvětšení 200%



Obr. 9.4 Modul digitalizace – TOP layer

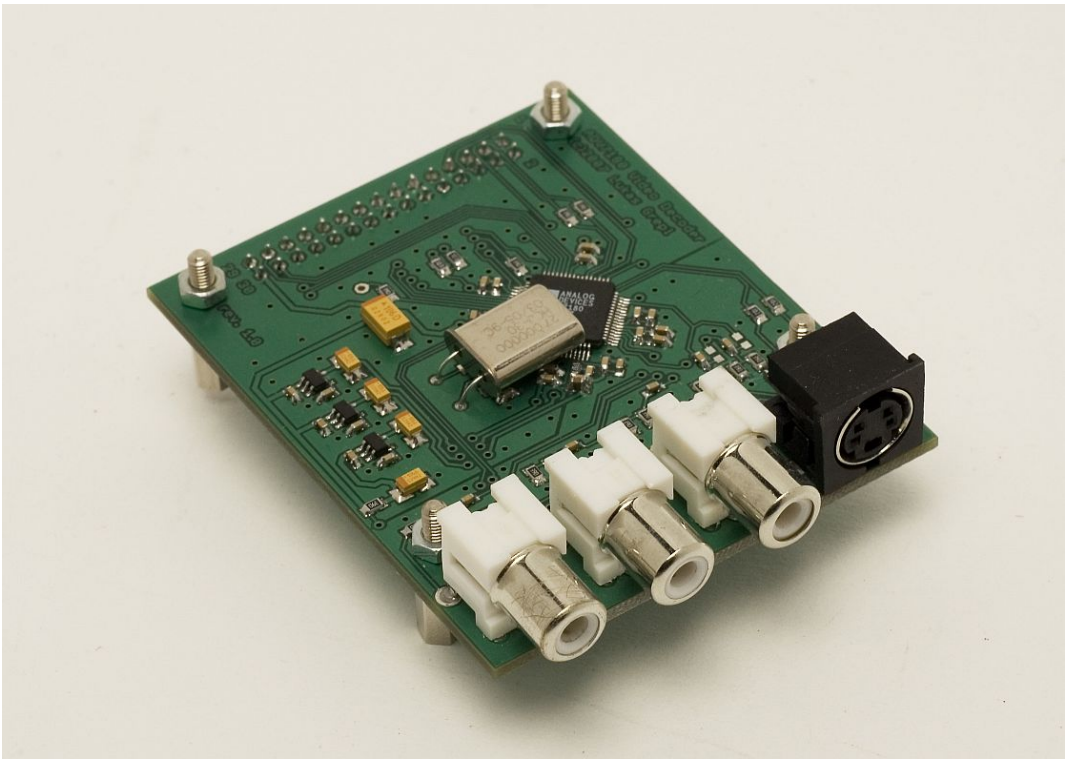
9.1.5 BOTTOM layer

zvětšení 200%

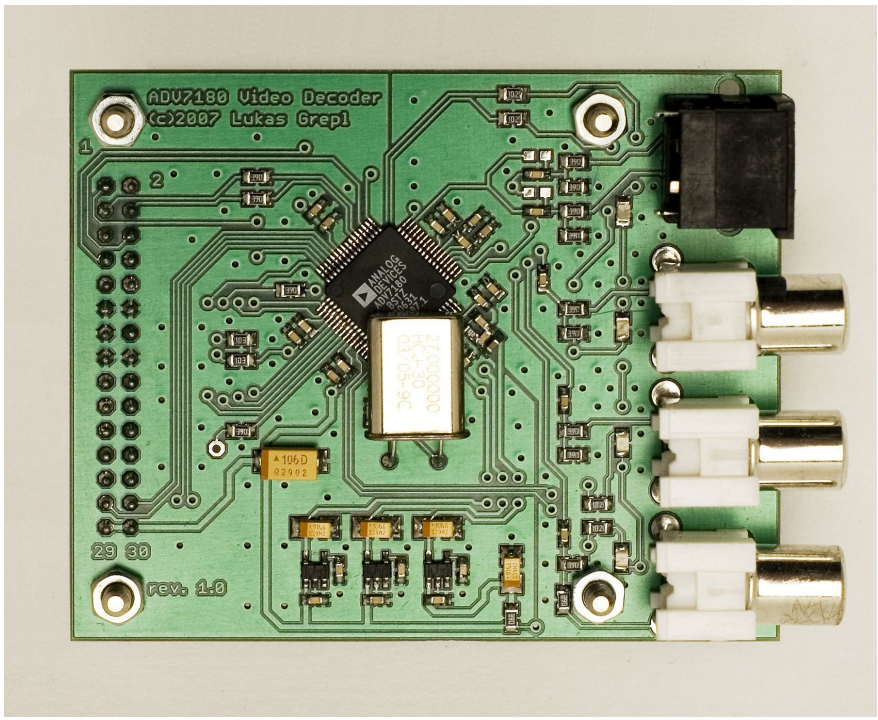


Obr. 9.5 Modul digitalizace – BOTTOM layer

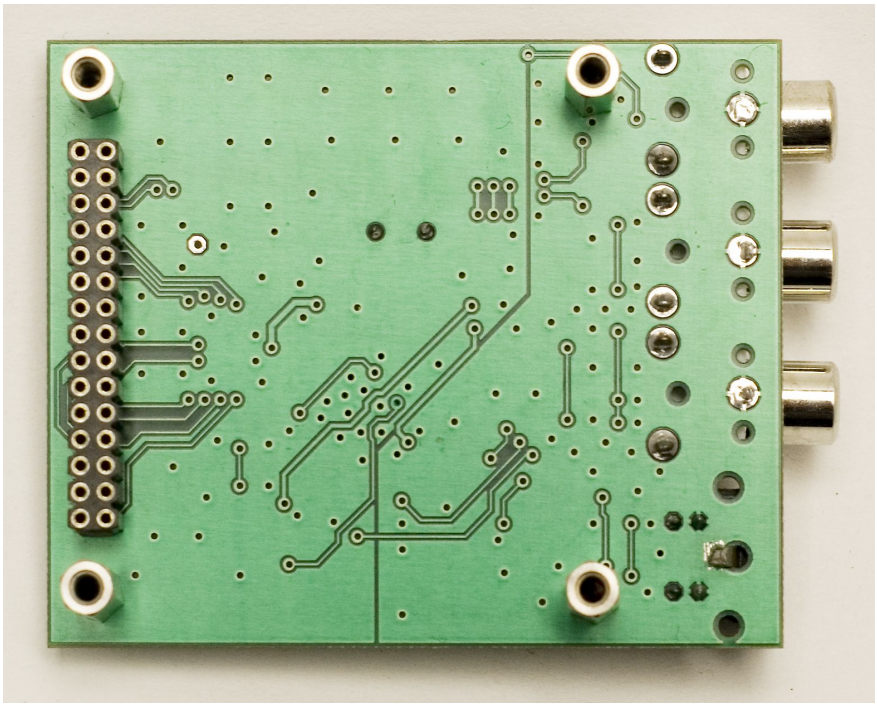
9.1.6 Fotografie hotového modulu



Obr. 9.6 Modul digitalizace - celkový pohled



Obr. 9.7 Modul digitalizace – pohled shora

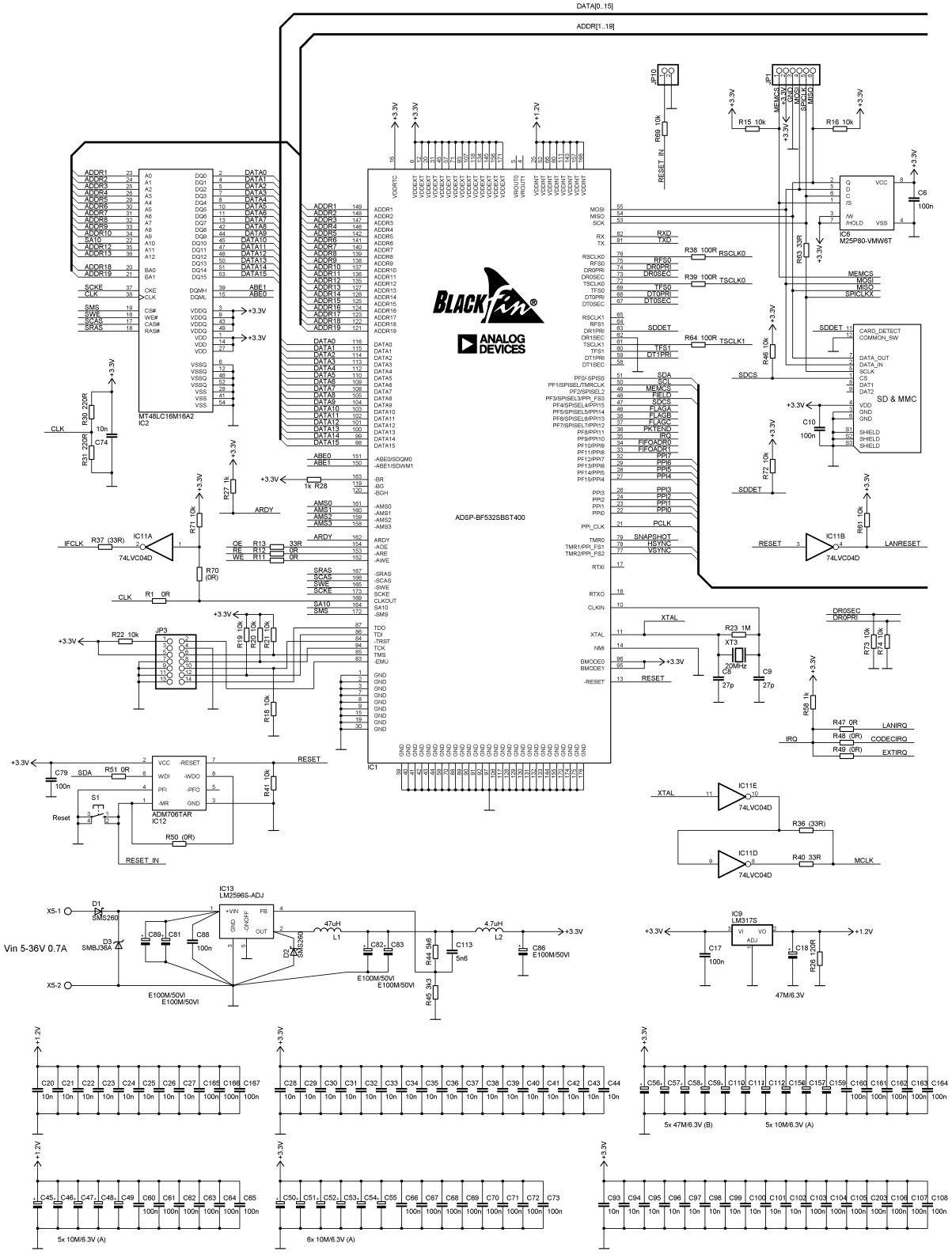


Obr. 9.8 Modul digitalizace – pohled zdola

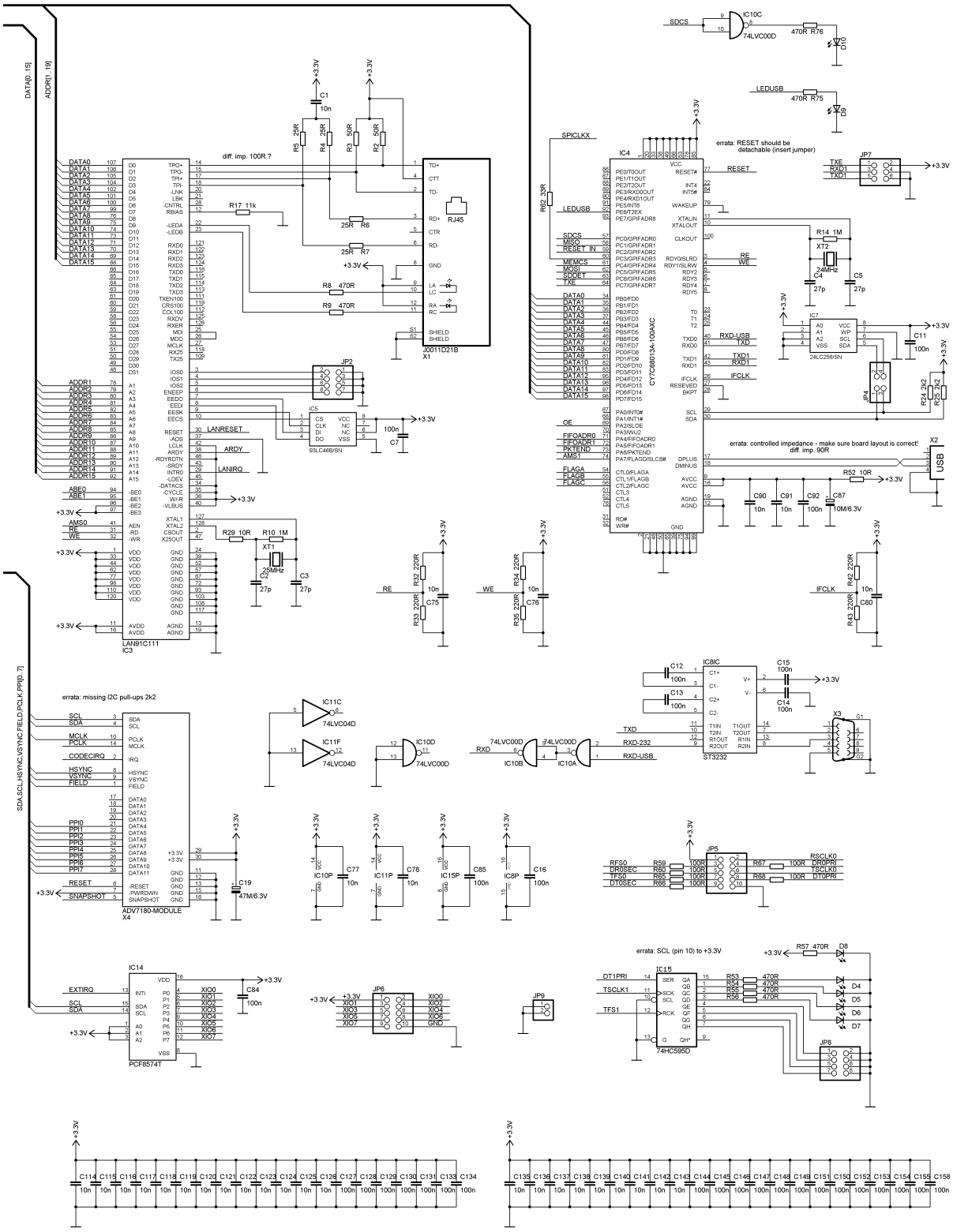
9.2 Modul pro zpracování obrazu

Schéma zapojení, návrh DPS, fotodokumentace hotového modulu.

9.2.1 Schéma zapojení



Obr. 9.9 Modul pro zpracování obrazu – schéma zapojení (procesorová část)

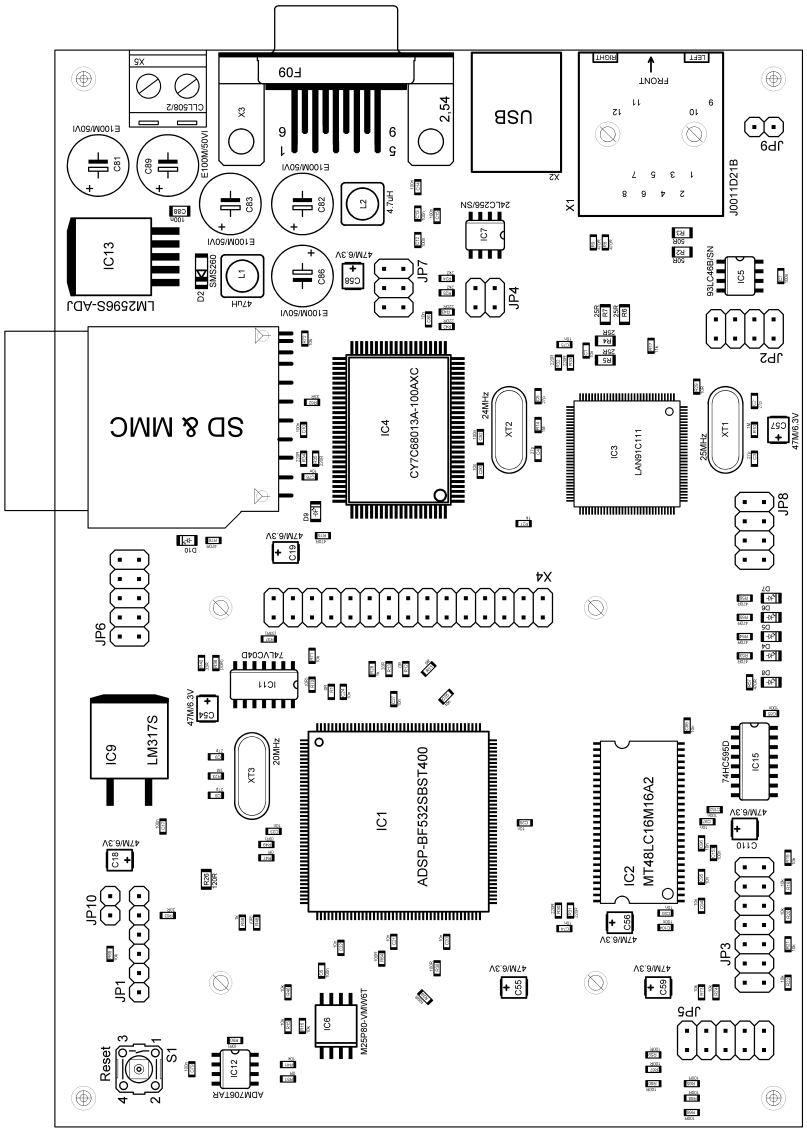


Obr. 9.10: Modul pro zpracování obrazu – schéma zapojení (periferie)

9.2.2 Náhled DPS

zvětšení 100%

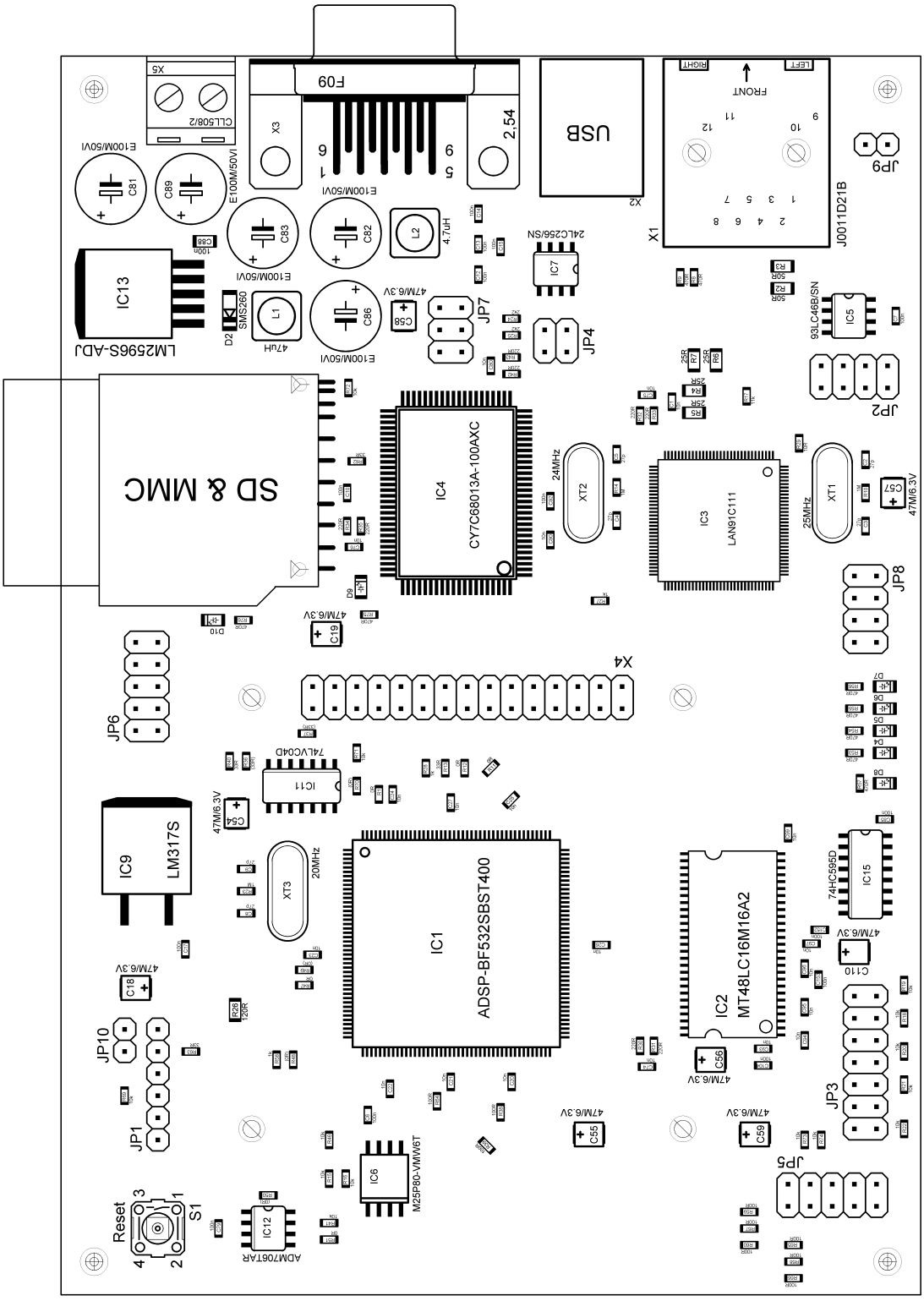
rozměr DPS: 98,6 x 142,2 mm



Obr. 9.11 Modul pro zpracování obrazu – náhled DPS

9.2.3 Rozmístění součástek TOP layer

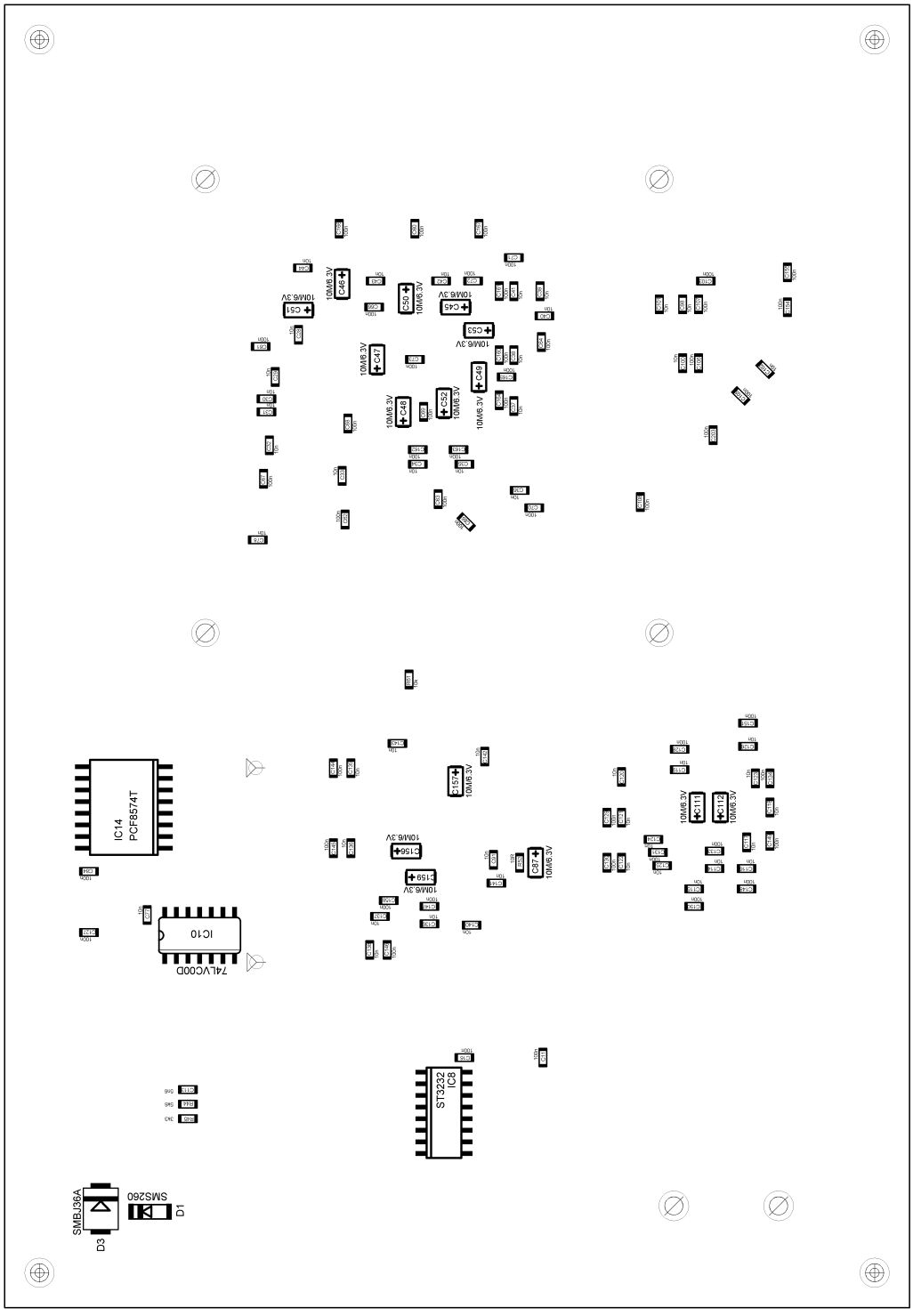
Nedefinované měřítko.



Obr. 9.12 Modul pro zpracování obrazu – rozmístění součástek TOP layer

9.2.4 Rozmístění součástek BOTTOM layer

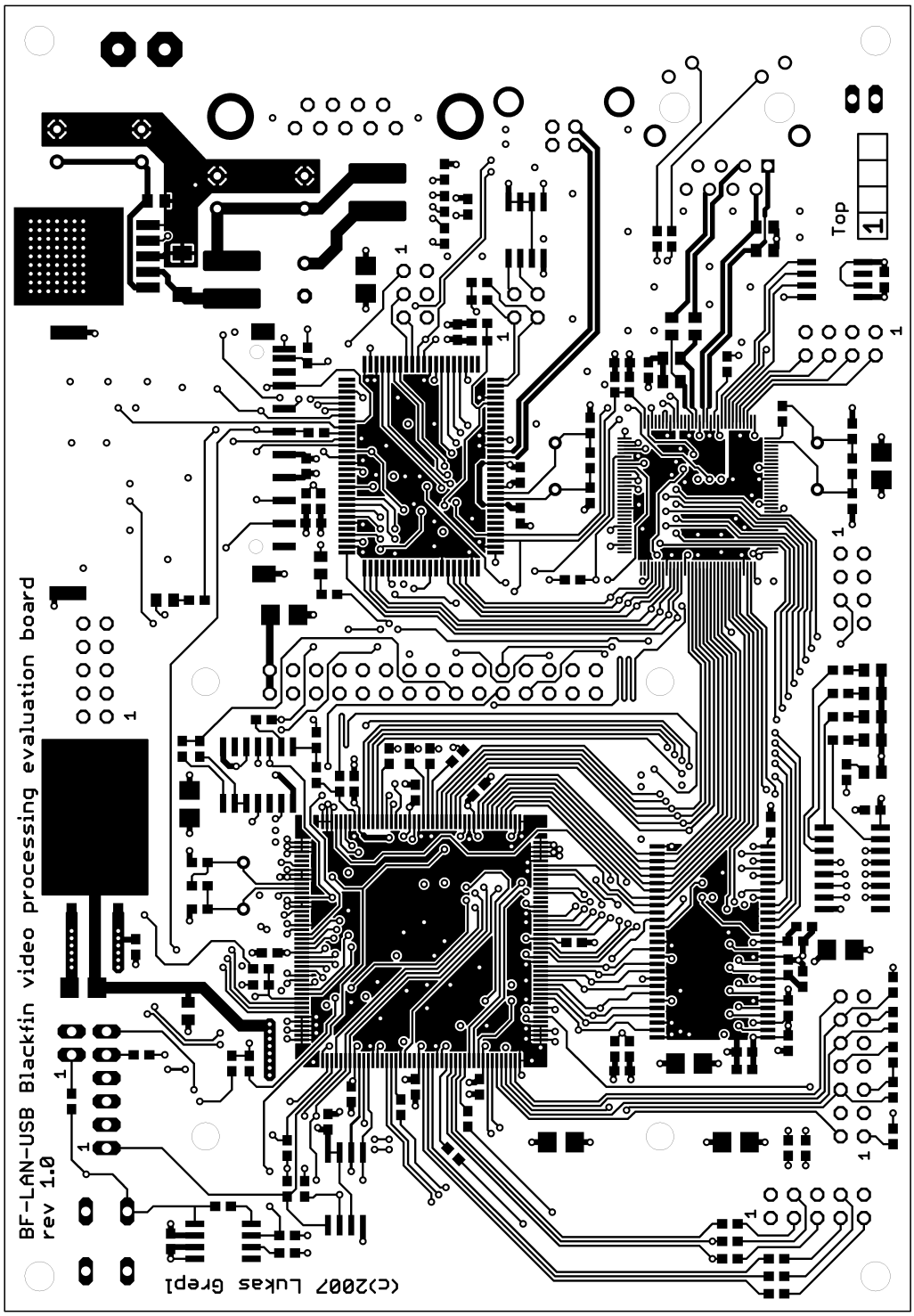
Nedefinované měřítko.



Obr. 9.13 Modul pro zpracování obrazu – rozmístění součástek BOTTOM layer

9.2.5 TOP layer

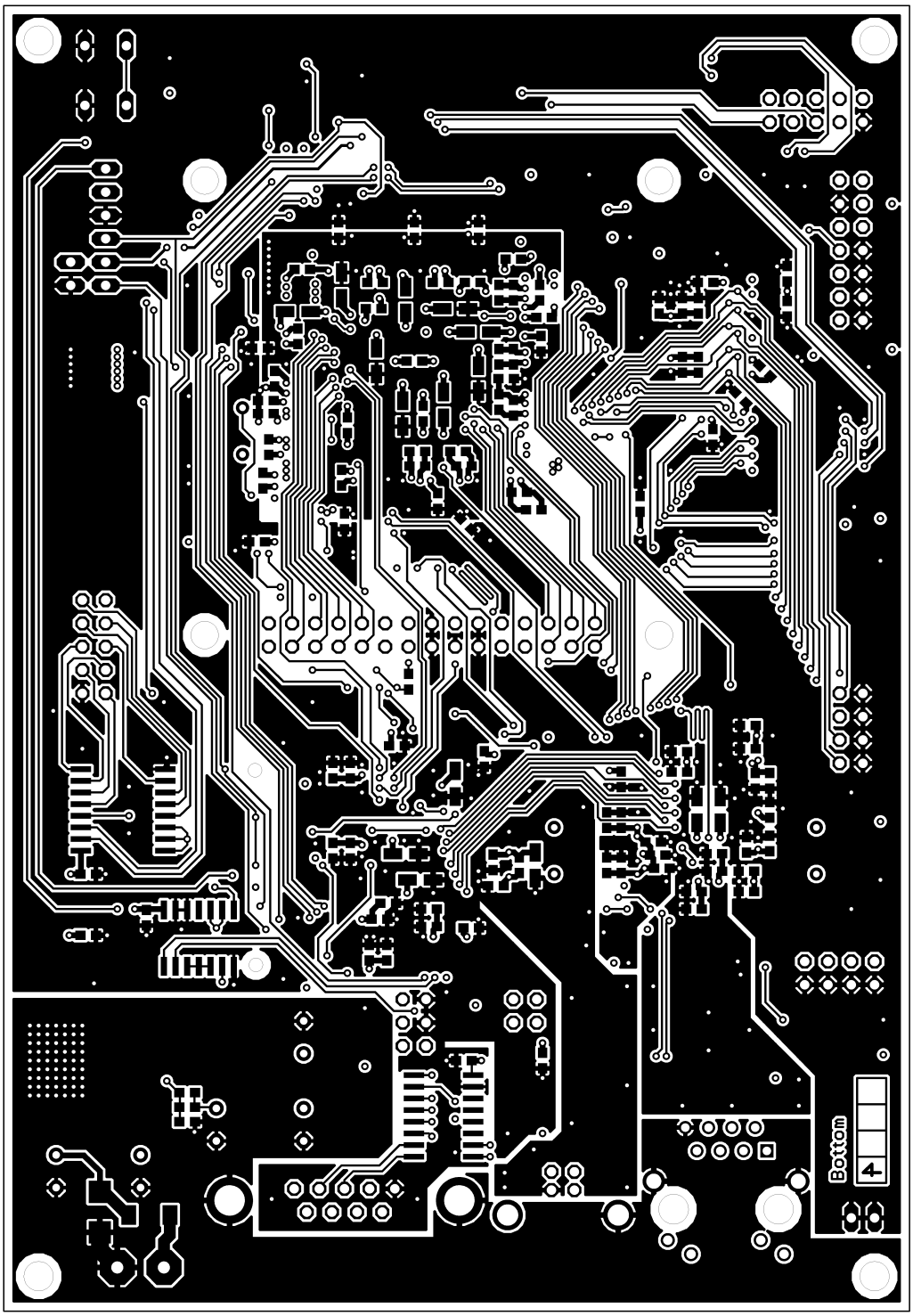
Nedefinované měřítko.



Obr. 9.14 Modul pro zpracování obrazu – TOP layer

9.2.6 BOTTOM layer

Nedefinované měřítko.

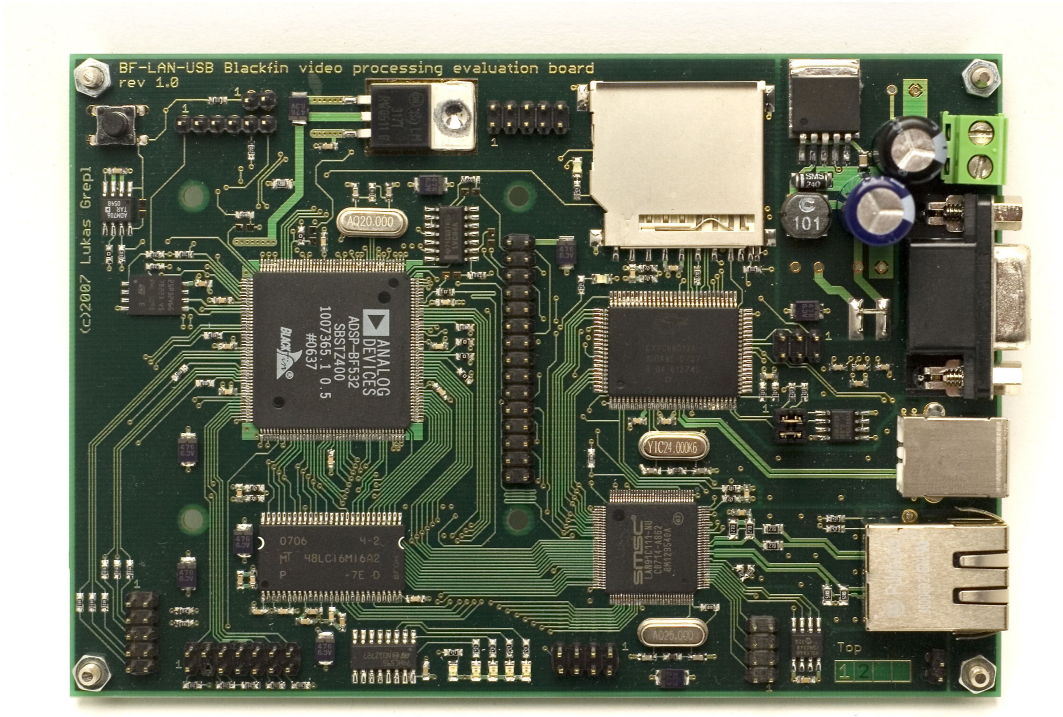


Obr. 9.15 Modul pro zpracování obrazu – BOTTOM layer

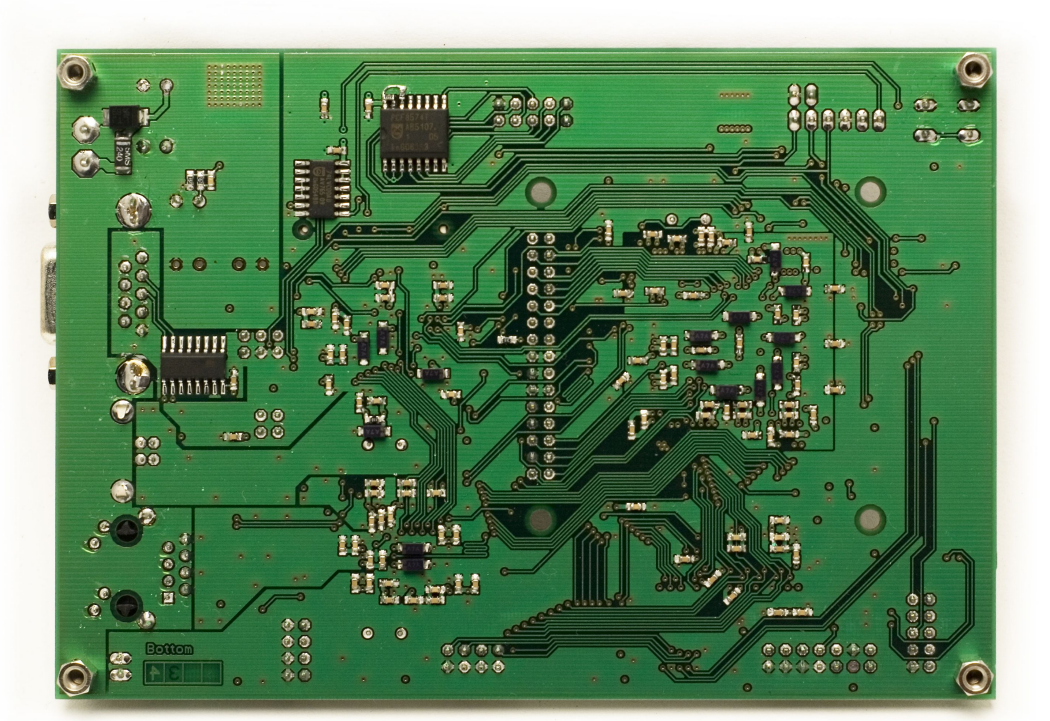
9.2.7 Fotografie hotového modulu

Obr. 9.16: Modul pro zpracování obrazu – celkový pohled

Obr. 9.17 Modul pro zpracování obrazu v sestavě s modulem digitalizace



Obr. 9.18 Modul pro zpracování obrazu – pohled shora



Obr. 9.19 Modul pro zpracování obrazu – pohled zdola

9.3 Výpis hlášení při startu U-Bootu

```
U-Boot 1.1.5 (Jan 11 2008 - 23:54:46)

CPU:   ADSP BF532 Rev.: 0.5
Board: BF532 Video processing development board
       Lukas Grepl <L.Grepl@sh.cvut.cz>
Clock: VCO: 400 MHz, Core: 400 MHz, System: 133 Mhz
SDRAM: 32 MB
In:    serial
Out:   serial
Err:   serial
Setting MAC address to 02:80:ad:20:31:b8.
Net:   SMC91111 at 0x20000300
starting from spi flash
Hit any key to stop autoboot:  0

EEPROM @0x0 read: addr 0x01000000  off 0x30000  count 0x3d0000

done
## Booting image at 01000000 ...
   Image Name:   uClinux Kernel and ext2
   Image Type:   Blackfin Linux Kernel Image (gzip compressed)
   Data Size:    3584107 Bytes =  3.4 MB
   Load Address: 00001000
   Entry Point:  00001000
   Verifying Checksum ... OK
   Uncompressing Kernel Image ... OK
Starting Kernel at = 1000
```

9.4 Výpis hlášení při startu uClinuxu

```
Linux version 2.6.19.3-ADI-2007R1.1-svn (blackfin@localhost)
     (gcc version 4.1.1 (ADI 07R1)) #26 Tue Dec 18 02:27:17 CET 2007
Blackfin support (C) 2004-2007 Analog Devices, Inc.
Compiled for ADSP-BF532 Rev 0.5
Blackfin Linux support by http://blackfin.uclinux.org/
Processor Speed: 400 MHz core clock and 133 Mhz System Clock
Board Memory: 32MB
Kernel Managed Memory: 32MB
Memory map:
  text      = 0x00001000-0x001340cc
  init      = 0x00135000-0x00143ec0
  data      = 0x00145f9c-0x0017d964
  stack     = 0x00146000-0x00148000
  bss       = 0x0017d970-0x0018b880
  available = 0x0018b880-0x01500000
  rootfs    = 0x01500000-0x01f00000
  DMA Zone  = 0x01f00000-0x02000000
Instruction Cache Enabled
Data Cache Enabled (write-through)
Hardware Trace Enabled
Built 1 zonelists.  Total pages: 5334
Kernel command line: root=/dev/mtdblock0 rw
     ip=192.168.0.2:192.168.0.1:192.168.0.2:255.255.255.0:
     MYBOARD:eth0:off ethaddr=02:80:ad:20:31:b8
Configuring Blackfin Priority Driven Interrupts
PID hash table entries: 128 (order: 7, 512 bytes)
Dentry cache hash table entries: 4096 (order: 2, 16384 bytes)
Inode-cache hash table entries: 2048 (order: 1, 8192 bytes)
Physical pages: 1500
Memory available: 19624k/31464k RAM, (59k init code, 1228k kernel
     code, 55k data, 1024k dma)
Blackfin Scratchpad data SRAM: 4 KB
Blackfin Instruction SRAM: 32 KB
Security Framework v1.0.0 initialized
Capability LSM initialized
Mount-cache hash table entries: 512
NET: Registered protocol family 16
Blackfin GPIO Controller
Blackfin DMA Controller
stamp_init(): registering device resources
NET: Registered protocol family 2
IP route cache hash table entries: 256 (order: -2, 1024 bytes)
TCP established hash table entries: 1024 (order: 0, 4096 bytes)
TCP bind hash table entries: 512 (order: -1, 2048 bytes)
TCP: Hash tables configured (established 1024 bind 512)
TCP reno registered
NTFS driver 2.1.27 [Flags: R/O].
io scheduler noop registered
io scheduler anticipatory registered (default)
io scheduler cfq registered
pfx: pfbits driver for bf5xx
PPI: ADSP PPI Frame Capture Driver IRQ:15
Dynamic Power Management Controller Driver v0.1: major=10,
     minor = 254
: major=10, minor = 0
Serial: Blackfin serial driver
bfin-uart.1: ttyBF0 at MMIO 0xffc00400 (irq = 21) is a BFIN-UART
```

```
RAMDISK driver initialized: 16 RAM disks of 4096K size
     1024 blocksize
smc91x.c: v1.1, sep 22 2004 by Nicolas Pitre <nico@cam.org>
eth0: SMC91C11xFD (rev 2) at 20000300 IRQ 42 [nowait]
eth0: Ethernet addr: 02:80:ad:20:31:b8
uclinux[mtd]: RAM probe address=0x1500000 size=0xa00000
Creating 1 MTD partitions on "RAM":
0x00000000-0x00a00000 : "ROMfs"
uclinux[mtd]: set ROMfs:EXT2  to be root filesystem
m25p80 spi1.2: m25p32 (4096 Kbytes)
Creating 2 MTD partitions on "m25p80":
0x00000000-0x00030000 : "bootloader"
0x00030000-0x00400000 : "kernel"
i2c /dev entries driver
TCP cubic registered
NET: Registered protocol family 1
NET: Registered protocol family 17
eth0: link down
IP-Config: Complete:
      device=eth0, addr=192.168.0.2, mask=255.255.255.0,
     gw=192.168.0.2, host=MYBOARD, domain=, nis-domain=(none),
     bootserver=192.168.0.1, rootserver=192.168.0.1, rootpath=
VFS: Mounted root (ext2 filesystem).
Freeing unused kernel memory: 56k freed (0x135000 - 0x142000)
dma_alloc_init: dma_page @ 0x00140000 - 256 pages at 0x01f00000

Welcome to:
       ____ _  _
      /  __| ||_|              _  _
 _   _| |  | | _ ____  _   _ \ \/ /
| | | | |  | || |  _ \| | | | \  /
| |_| | |__| || | | | | |_| | /  \
|  ___\____|_||_|_| |_|\____|/_/\_\
|_|

For further information see:
http://www.uclinux.org/
http://blackfin.uclinux.org/



BusyBox v1.4.1 (2007-12-06 01:20:16 CET) Built-in shell (msh)
Enter 'help' for a list of built-in commands.

eth0: link up, 100Mbps, full-duplex, lpa 0xC5E1
root:~>
```

9.5 DVD příloha

- text diplomové práce ve formátu PDF
- originální zdrojové kódy uClinuxu a U-Bootu
- patche pro modifikaci
- výrobní dokumentaci
- zdrojové kódy příkladů
- katalogové listy
- použitý volně šiřitelný software