VŠB - Technická univerzita Ostrava

Fakulta elektrotechniky a informatiky

DIPLOMOVÁ PRÁCE

2007 Bc. Petr Sedlář

VŠB - Technická univerzita Ostrava

Fakulta elektrotechniky a informatiky

Katedra Informatiky a výpočetní techniky

Spojovací pole sériových linek na bázi FPGA

2007 Bc. Petr Sedlář

Zadání diplomové práce

Téma: Spojovací pole sériových linek na bázi FPGA

Diplomant: Bc. Petr Sedlář

Vedoucí: Ing. David Seidl

Oponent: Ing. Petr Grygárek, Ph.D.

Akad. rok: 2006/2007

Obor: 2612T025 Informatika a výpočetní technika

Zásady pro vypracování:

Cílem práce je reimplementovat spojovací pole sériových linek ASSSK vytvořené v rámci DP Davida Seidla (FMMI, katedra automatizace a počítačové techniky v metalurgii, 2004) s použitím programovatelných logických polí. Návrh bude zredukován o podporu ethernetu, navíc bude implementována možnost nezávislého stanovení hodinové frekvence pro jednotlivé porty.

1. Seznamte se s HW a SW původního návrhu ASSSK.
2. Vyberte vhodné obvody pro reimplementaci, zvažte možnost rozšíření množství spojovaných prvků.
3. Navrhněte obvodové schéma a plošný spoj s důrazem na opakovatelnost konstrukce.
4. Upravte stávající SW ASSSK s ohledem na novou koncepci HW.
5. Funkčnost celého zařízení otestujte a zdokumentujte výsledky.

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně.

Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 9.května 2007 ……..

Seznam použitých zkratek a symbolů:

ASSSK	Automatizovaný systém spojování síťových konfigurací
CPLD	Complex PLD (komplexní PLD)
DPS	Deska plošných spojů
EPROM	Přepisovatelná ROM
FPGA	on Field Programmable Gate Arrays (na místě programovatelné logické obvody)
I/O	input/output (vstupně výstupní)
LB	Logic block (logický blok
LVTTL	Low voltage TTL (nízko napěťová TTL)
PC	Personal computer (osobní počítač)
PLD	Programmable logic device (programovatelné hradlové pole)
RAM	Random access memory
ROM	Read only memory
RxD	Recive data (přijímaná data)
SRAM	Statická RAM
TTL	Tranzistor-tranzistor logic (tranzistorově tranzistorová logika)
TxC	Transit clock (synchronizační hodiny)
TxD	Transit data (vysílaná data)
VHDL	Hardware Description Language (jazyk pro popis hardware)

Abstrakt

Diplomová práce reimplementuje automatizovaný systém spojování síťových konfigurací (ASSSK), vytvořený v rámci diplomové práce Ing. Davida Seidla (FMMI, katedra automatizace a počítačové techniky v metalurgii, 2004), s použitím programovatelných logických polí (FPGA). Návrh je zredukován o podporu ethernetu. Navíc je implementována možnost nezávislého stanovení hodinové frekvence pro jednotlivé porty.

Práce obsahuje popis současného stavu využití ASSSK 1, výběr a popis obvodů pro realizaci nového zařízení. Dále je v práci obsažen návrh řešení a popis realizace nového ASSSK 2. V závěru práce je uveden způsob testování funkčnosti a prezentovány dosažené výsledky.

Klíčová slova: ASSSK, FPGA, VHDL, AT89C51ED2, sériové rozhranní, RS232, virtuální laboratoř, Actel, A3P060, ProAsic3.

Obsah

Úvod

Pojmy počítač a internet jsou v dnešní době většině lidí velmi dobře známy. Mnoho z nás si dokonce ani nedovede představit svůj život bez těchto nástrojů. Používáme je denně. Nejen proto, že nám usnadňují práci, ale také proto, že s jejich pomocí můžeme získat nepřeberné množství potřebných a užitečných informací. K údržbě tak velkého informačního prostředí je zapotřebí mnoho lidských sil, které musí mít odbornou kvalifikaci. Proto je důležité, aby se co nejefektivněji mohli zdokonalovat v konfigurování počítačových sítí.

Tato práce úzce navazuje na diplomovou práci Ing. Davida Seidla [6], který měl za cíl zrealizovat dálkově řízené spojovací pole WAN a LAN portů, které umožnilo automatizovat manuální hardwarové konfigurace laboratorních prvků počítačové sítě. Toto zařízení slouží ve virtuální laboratoři síťových technologií na katedře informatiky, FEI VŠB TU Ostrava.

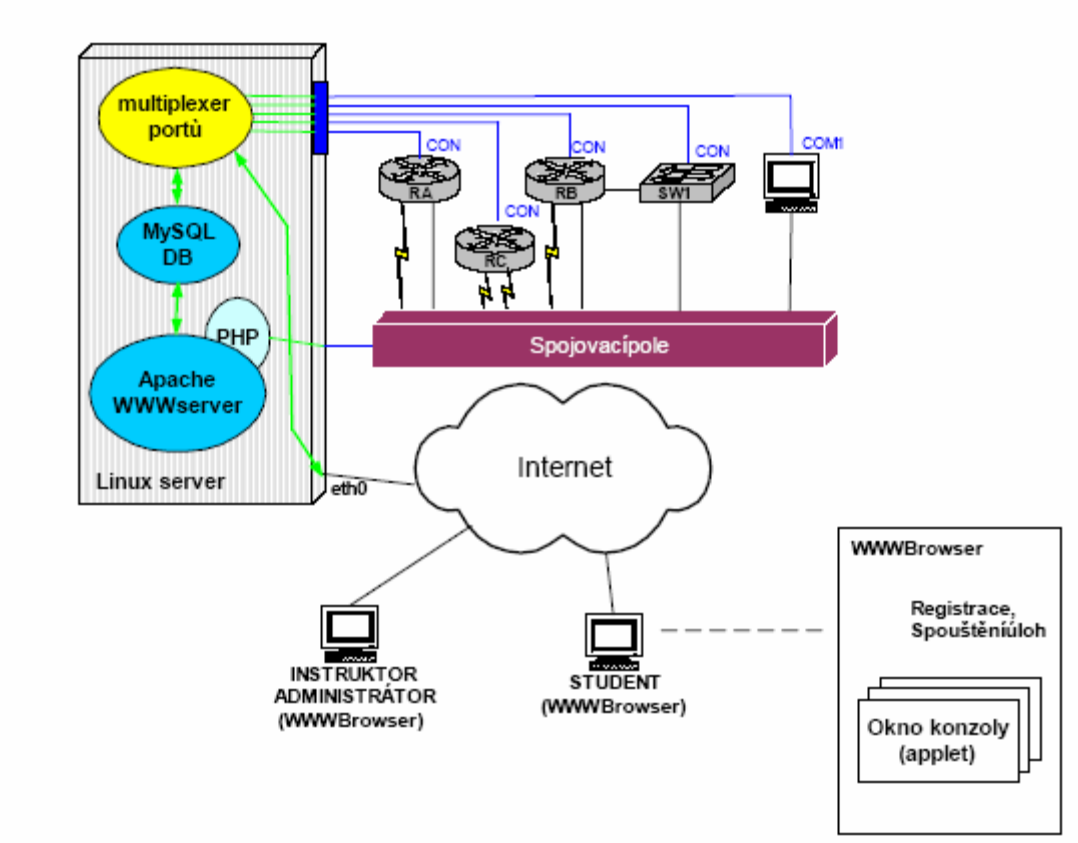
Cílem této práce je reimplementace tohoto spojovacího pole. Hlavním rozdílem od práce Ing. Davida Seidla by mělo být použití jiných technologií spojování sériových portů a to pomocí programovatelného hradlového pole. Toto pole umožní kromě spojování portů i nezávislé stanovení hodinové frekvence pro jednotlivé porty. Navrhovaný systém bude redukován o možnost spojování ethernetu. Nedílnou součástí této práce bude popis a dokumentace všech použitých technologií.

Zařízení bude sloužit, stejně jako jeho předchůdce, ve virtuální laboratoři, která si klade za cíl zkvalitnit výuku počítačových sítí, a to především její praktickou část.

1 Popis současného využití ASSSK 1

Laboratoř výuky počítačových sítí, která je vybavena množstvím síťového hardwaru, je v období noci, víkendů, zkouškového období a prázdnin nevyužita. Proto se začala budovat virtuální laboratoř (VirtLab), jejímž cílem je zpřístupnit studentům tuto techniku právě v těchto nevyužívaných dobách.

Filozofie VirtLabu je taková, že student či skupina studentů si může přes webové rozhraní, které bylo vyvíjeno v rámci diplomové práce (Němec,P.: Virtuální síťová laboratoř [5]), zarezervovat určitou dobu a určitou topologii, na které chtějí řešit svůj problém. V tuto dobu se připojí přes internet k serveru virtuální laboratoře a mohou si odzkoušet své teoretické znalosti.



Obr. 1: Schéma zapojení virtuálni laboratoře

Zdroj: http://www.cs.vsb.cz/vl-wiki/index.php

Automatizovaný systém spojování síťových konfigurací (ASSSK 1) slouží k tomu, aby administrátor studentům mohl zpřístupnit jejich vybranou konfiguraci na dálku bez nutnosti být fyzicky v laboratoři. Zmenší se i možnost chyb na topologii, protože konfigurace již bude předem odzkoušena.

Celý proces tedy probíhá tak, že si studenti zarezervují dobu a topologii s předstihem a poté administrátor (instruktor) pouze zadá sekvenci příkazů a vytvoří tím potřebnou topologii.

Současný ASSSK 1 umožňuje propojování sériových (RS232) a ethernetových (10BaseT) linek. Nejedná se o zařízení typu switche, ale o „ústřednu“ spojující porty, které mají spolu komunikovat. Celý systém funguje na první vrstvě referenčního modelu ISO/OSI, tedy nezabývá se daty, které jsou přes něj přenášeny. Kopíruje pouze napěťové úrovně z daného vstupu na daný výstup.

Tento systém již běží ve virtuální laboratoři. Na návrh Ing. Petra Grygárka, Ph.D. je vhodné přepracovat ASSSK 1, neboť na něj jsou kladeny jiné nároky. Následník ASSSK 2 bude mít stejnou funkci, budou jen použity novější technologie, které byly v předchozích letech cenově nedostupné.

2 Výběr spínaných rozhraní

V předchozí verzi ASSSK 1 se požadovalo spínání jak sériových linek, tak ethernetového rozhraní. Během používání systému se však ukázalo, že použití ASSSK pro spojování rozhraní Ethernet není efektivní. ASSSK je efektivnější využít pouze na propojování sériových rozhraní, které nejsme schopni propojovat jiným standardním zařízením. Dále dovoluje spojovat pouze Ethernet 10Mbps z důvodu frekvenčních omezení použitých součástek. Z tohoto důvodu bylo rozhodnuto spojovat Ethernet porty laboratorních síťových prvků s použitím standardního přepínače s vhodnou konfigurací virtuálních sítí (VLAN) tak, že dvojice portů síťových prvků, které mají být propojeny, budou zařazeny do společné VLAN. K tomuto účelu se používají starší přepínače Cisco Catalyst 1900, které jsou momentálně vyřazovány ze síťové infrastruktury školy, avšak díky podpoře VLAN a dobré konfigurovatelnosti jsou pro náš účel zcela dostačující a to i přesto, že disponují pouze porty 10Mbps. Omezení rychlosti linek Ethernet na 10Mbps ve většině výukových konfigurací není na závadu. Pro situace, kde je skutečně třeba spínat linky Ethernet 100Mbps nebo 1000Mbps, je možné pro spojování použít jiný přepínač, např. řady Cisco 2900 nebo i levnější model jiného výrobce.[1]

Nový ASSSK 2 bude proto spínat jen sériové rozhraní RS232. Toto rozhraní je hojně používané v radioamatérských aplikacích. Díky tomu je dobře zdokumentováno a podpůrné obvody a konektory jsou snadno dostupné.

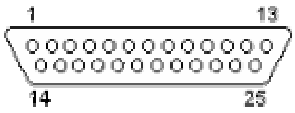
Pro správné pochopení funkčnosti spínaní je důležité popsat fyzickou vrstvu tohoto rozhraní, což obsahuje následující podkapitola.

2.1 Fyzická vrstva rozhraní RS232

Data jsou přenášena pomocí osmi vodičů. Napěťové signály nabývají svých hodnot vůči společnému potenciálu. Logická jednička je reprezentována záporným napětím od -3V do -12V a logická nula rozmezím hodnot napětí 3V až 12V. Hlavní dva datové signály jsou TxD (transit data) a RxD (Recive data). Další signály jsou pomocné a slouží k řízení poloduplexní

[1] Zdroj: Grygárek, P.: Zkušenosti z nasazení virtuální laboratoře počítačových sítí a další směry jejího rozvoje, Sborník semináře Technologie pro e-vzdělávání, FEL ČVUT Praha, katedra počítačů, 2006. ISBN 80-01-03512-3

komunikace. V plně duplexních komunikačních zařízení ztrácejí řídící signály svůj původní význam. Používají se pouze k určení, zda je či není připojen sériový kabel nebo je využívá přímo softwarové vybavení. Jedná se o signály RTS (Request To Send), CTS (Clear To Send), DSR (Data Set Ready), DTR (Data Terminal Redy). Dalším velmi důležitým signál je TxC (Transit Clock), což je hodinový signál, který musí být generován vždy jedním z dvojice propojovaných síťových prvků. Síťový prvek jež generuje tyto hodinové signály nese označení DCE a prvek, který se těmito hodinovými signály řídí, se označuje DTE. Jeden z používaných konektorů je znázorněn na *Obr. 2*. a popis jednotlivých pinů je v tabulce 1.



Obr. 2: Konektor rozhraní RS232

Pin	Název	Směr toku dat	Popis
1	GND	------	Ground
2	TXD	→	Transmit Data
3	RXD	←	Receive Data
4	RTS	→	Ready To Send
5	CTS	←	Clear To Send
6	DSR	←	Data Set Ready
7	GND	------	Ground
8	DCD	←	Data Carrier Detect
9	n/c		Not connected
10	n/c		Not connected
11	n/c		Not connected
12	n/c		Not connected
13	n/c		Not connected
14	n/c		Not connected
15	TxC	→	Transit Clock
16	RXD	←	Receive Data
17	RxC	←	Receive Clock
18	LTST		Link Test
19	n/c		Not connected
20	DTR	→	Data Terminal Ready
21	n/c		Not connected
22	n/c		Not connected
23	n/c		Not connected
24	TxCE	→	Transit Clock
25	n/c		Not connected

Tabulka 1: Piny konektoru CAN25

Zdroj:Diplomová práce Ing. Davida Seidla [6]

Chceme-li propojit dvojici síťových prvků pomocí rozhraní RS232, je nutné propojit signály dle Tabulka 2.

Zařízení typu DTE	Tok dat	Zařízení typu DCE
TxD	→	RxD
RxD	←	TxD
DTR	→	DCE
DCE	←	DTR
CTS	←	RTS
RTS	→	CTS
RxC	←	TxC
TxC	→	RxC

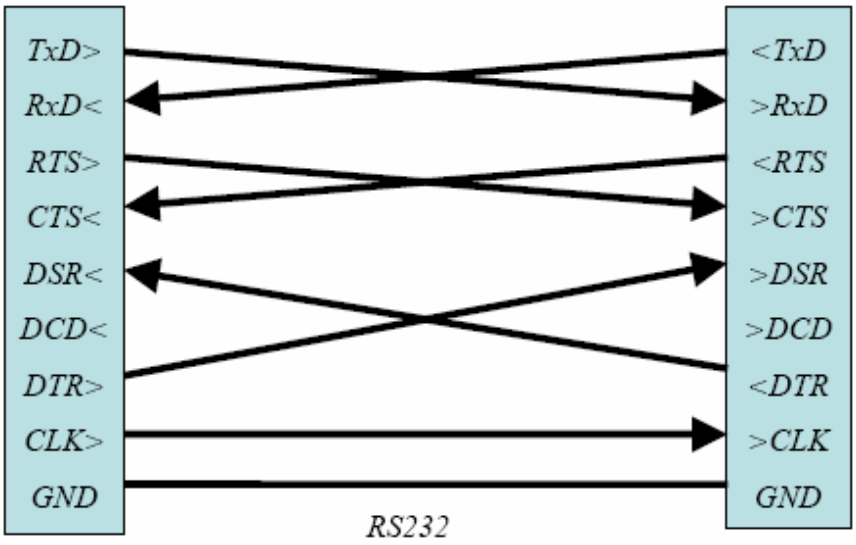
Tabulka 2: Popis propojení dvou konektorů rozhraní RS232

Zdroj: Diplomová práce Ing. Davida Seidla [6]

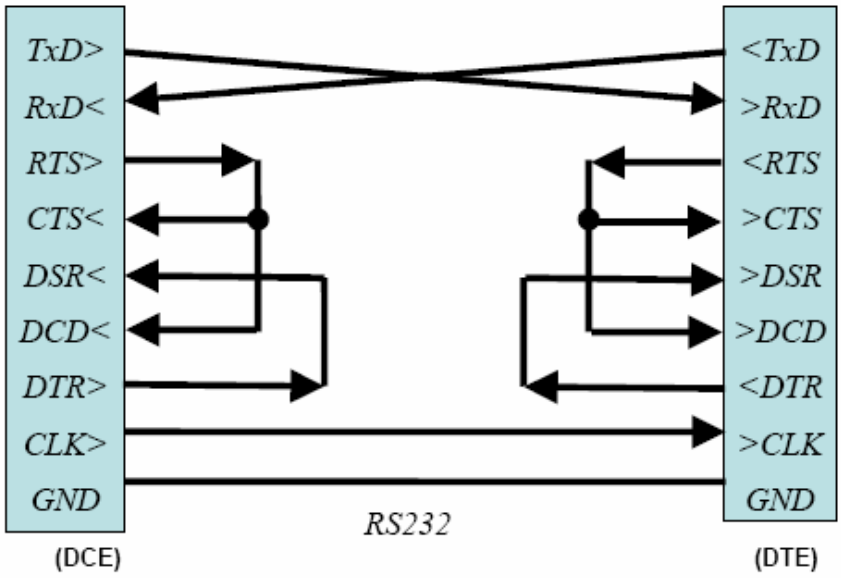
2.2 Úprava signálu pro rozhraní RS232

Při standardním zapojení rozhraní RS232 se musí propojit 7 signálů (2 datové, 4 pomocné a hodinový signál), což by celou konstrukci komplikovalo a prodražilo. Propojení dvou zařízení je naznačeno na Obr. 3.

U předchozího ASSSK1 se použilo zapojení do takzvaného „Null Modemu“. Toto zapojeni se osvědčilo, a proto byl problém s velkým počtem spínaných signálů vyřešen tímto způsobem. Hodinový signál bude generovat samotné spínací pole. Toto zapojení je znázorněnona Obr. 4.



Obr. 3: Standardní zapojení konektorů rozhraní RS232



Obr. 4: Zapojení konektorů rozhraní RS232 do tzv. Null Modemu

3 Výběr a popis obvodů pro realizaci

3.1 Řídící procesor

Řízení celého zařízení není náročné na rychlost zpracovávání informací narozdíl od obvodu pro spínání. Pro tento účel byl vybrán stejný mikroprocesor Atmel AT89c51ED2 jako u předchozí verze ASSSK 1, protože splňuje všechny požadavky.

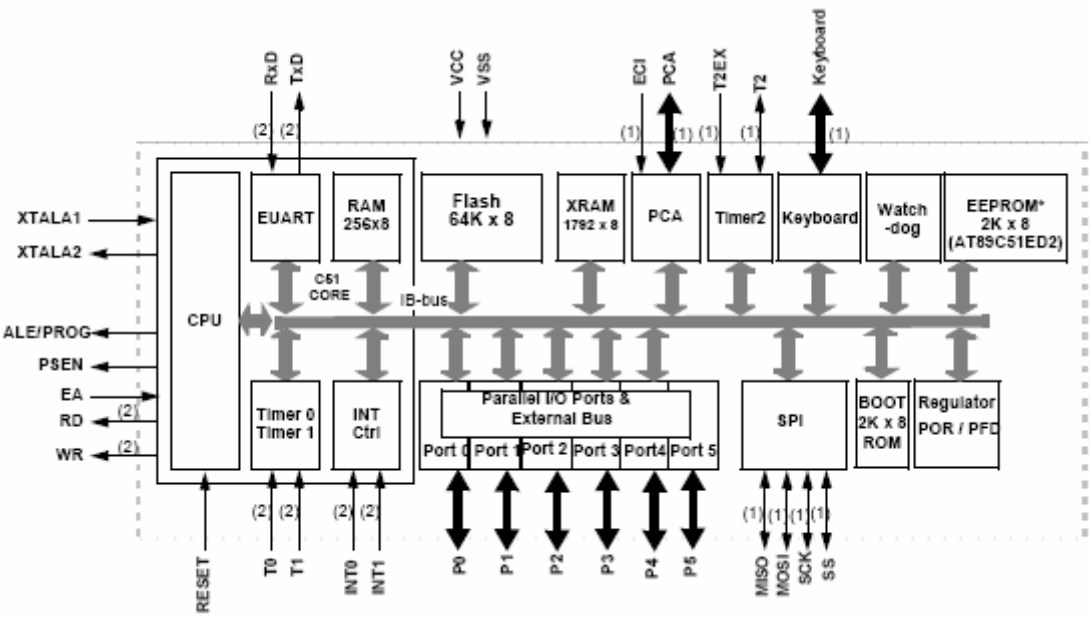
Základní vlastnosti osmibitového procesoru Atmel AT89c51ED2:

- Čítače: 3x16bit
- EEPROM: 2048 B
- Flash: 64 kB
- Maximální kmitočet krystalu: 60 MHz
- Počet vstupních/výstupních pinů: 32
- Pouzdro: DIL40
- RAM: 256B

Jednou z důležitých vlastností je 64kB paměti Flash, využívající se jako paměť programu. Předchozí ASSSK z této paměti používá 40kB, proto bude její velikost dostatečná i pro naši aplikaci.

Pro celou aplikaci je významná ještě další vlastnost zvoleného mikroprocesoru. A to způsob nahrávání řídicího programu. Mikroprocesor umožňuje programování hned třemi způsoby. První klasický způsob je umístění mikroprocesoru do programátoru a zavedení programu. Druhý způsob je programování pomocí SPI rozhraní. Jde o synchronní sériové rozhraní využívající čtyři signály mikroprocesoru. Poslední způsob programování je specifický pro mikroprocesor AT89c51ED2. Jestliže po restartu mikroprocesoru přivedeme na vývod PSEN na nulový potenciál(GND), nezačne se program vykonávat od adresy 0x00 v paměti FLASH, ale ze speciální paměti ROM, ve které je od výrobce nahrána aplikace BOOT Loader. Tento program očekává data po sériovém asynchronním rozhraní a tato data začne zapisovat do paměti FLASH. Tímto způsobem lze velice snadno tento mikroprocesor naprogramovat a to přímo v zařízení, bez nutnosti vyjímat mikroprocesor. Při návrhu plošného spoje je ale nutné umožnit pohodlné připojení signálu PSEN mikroprocesoru na nulový potenciál.

Blokové schéma mikroprocesoru AT89c51ED2 je vyobrazeno na Obr. 5.



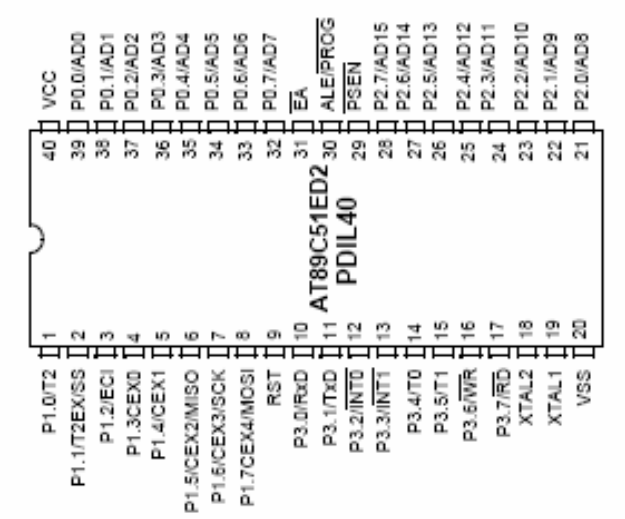
Obr. 5: Blokové schéma mikroprocesoru AT89c51ED2

Zdroj:katalogový list obvodu AT89c51ED2

3.1.1.1 Popis jednotlivých bloků mikroprocesoru

- **CPU** - modul centrální procesorové jednotky
- **Timer0, Timer1** - modul časovačů /čítačů
- **INT** - modul externího přerušení
- **I/O Ports** - vstupně výstupní paralelní porty
- **SPI** - modul pro sériové SPI programování
- **BOOT Rom** - paměť pro aplikaci Boot Loader
- **Regulátor POR/PFC**
- **EEPROM** - paměť EEPROM
- **WatchDog** - modul hlídající běh procesoru
- **KeyBoard** - modul obsluhy externí klávesnice
- **Timer2** - modul časovače/čítače
- **PCA** - modul programově řiditelného čítačového pole
- **XRAM** - rozšířená paměť RAM
- **FLASH** - paměť programu
- **RAM** - základní paměť RAM
- **EUTAR** - modul asynchronního sériového rozhraní

Z blokového schéma je zřejmé, že mikroprocesor má šest vstupně výstupních bran a to P0 až P5. Verze mikroprocesoru v pouzdru DILL40 má vyvedeny pouze čtyři brány, P4 a P5 vzhledem k nedostatku pinů v pouzdře vyvedeny nejsou. Některé piny přísluší jak vstupně výstupní bráně tak nějaké periferii. Popis jednotlivých pinů je znázorněn na Obr. 6.



Obr. 6 Popis pinů mikroprocesoru AT89c51ED2 [6]

Zdroj:katalogový list obvodu AT89c51ED2

3.2 Obvod pro spínání

Pro realizaci spínání byl u ASSSK 1 použit obvod MT8816. Tento obvod je určen ke spínání analogových signálů. Spínací analogové pole obsahuje spínací matici 8x16, tedy pro spínání 16-ti portů (32 signálů) bylo zapotřebí použití dvou těchto obvodů. Při požadavku na větší počet spínaných portů by už muselo být použito čtyř obvodů, což byl jeden z důvodů přepracovat tento návrh s pomocí programovatelného hradlového pole.

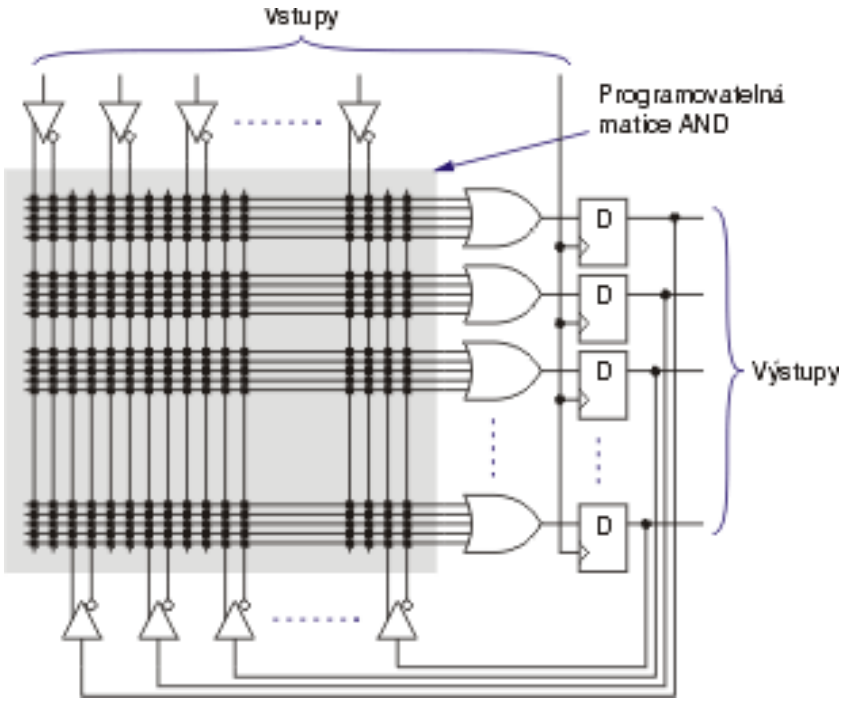
Programovatelné součástky a zejména hradlová pole jsou velmi důležité prvky dnešní elektroniky. Díky nim je možné vytvořit si vlastní zákaznický integrovaný obvod přizpůsobený konkrétní aplikaci s minimálními náklady. Využití programovatelných logických obvodů je často mnohem výhodnější a efektivnější.

Všechny číslicové programovatelné součástky se souhrnně označují PLD, což znamená Programmable Logic Device. Číslicové programovatelné součástky je možné podle vnitřní

struktury rozdělit do tří skupin. První skupina se označuje klasické PLD, druhá komplexní PLD a do třetí skupiny patří obvody typu FPGA.

3.2.1 Klasické PLD

Obvody této kategorie jsou charakteristické vnitřní strukturou podle Obr. 7.

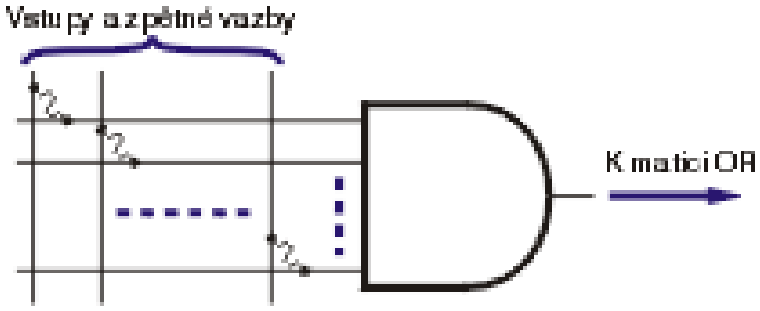


Obr. 7: Struktura obvodu PLD

Zdroj: http://sweb.cz/fpga/

Každá vodorovná čára v programovatelné matici AND představuje vždy jedno součinové hradlo. Na vstupy každého hradla lze připojit "libovolnou" kombinaci vstupních signálů, zpětných vazeb a jejich negací. Počet vstupů každého součinového hradla je však omezen. Zapojení jednoho součinového hradla je znázorněno na Obr. 8.

Vlnovky na tomto obrázku představují programovatelné spínače. Jejich realizace závisí na výrobní technologii obvodu. Například u bipolárních obvodů se jednalo o jakousi pojistku, která se při programování obvodu "přepálila" proudovým impulsem. V technologii CMOS jsou spínače realizovány stejnými principy jako u pamětí PROM, EPROM nebo EEPROM. Složitější obvody z kategorie FPGA mají často spínače řízeny statickou pamětí RAM.



Obr. 8: Součinové hradlo v obvodu PLD

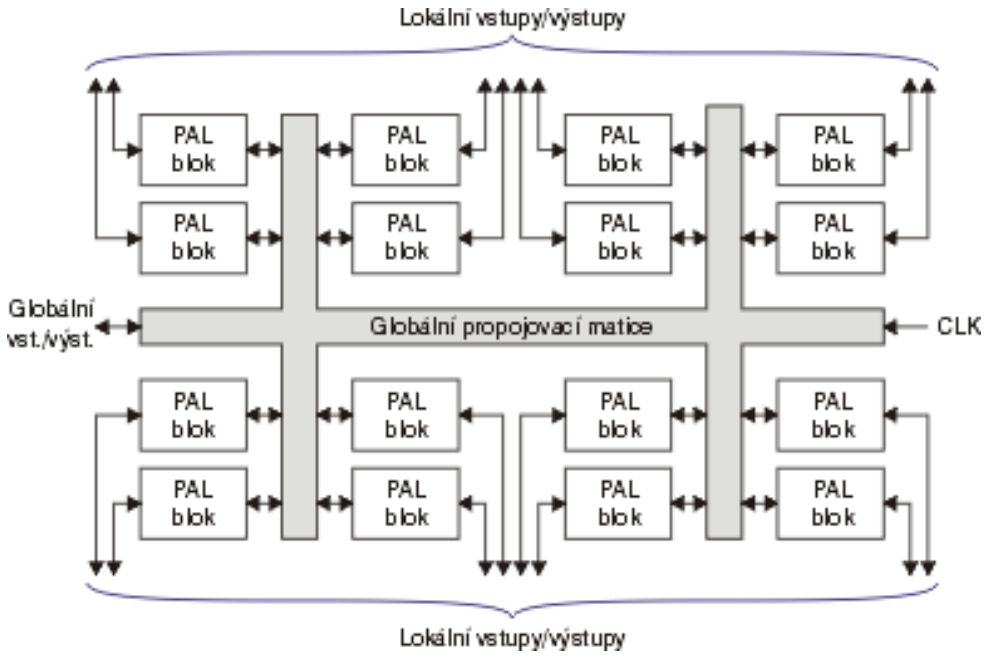
Zdroj: http://sweb.cz/fpga/

Do kategorie klasických PLD je možné zařadit obvody následujících typů:

- Obvody typu PAL (Programmable Array Logic) mají strukturu podle výše uvedených obrázků. Některé starší typy neměly například výstupní registry, takže byly vhodné spíše pro kombinační logiku. Zástupci této kategorie jsou obvody PAL, GAL a PALCE.
- Obvody typu PLA (Programmable Logic Array) mají obecnější strukturu než PAL na horním obrázku. Obsahují totiž programovatelnou nejenom matici logických součinů, ale i následující matici logických součtů.

3.2.2 Komplexní PLD

Klasické obvody PLD mají velmi omezené prostředky, takže umožňují realizovat pouze jednodušší funkce. Proto výrobci začali sdružovat více takových obvodů na jednom čipu spolu s nutnými prostředky pro propojení. Tyto obvody se většinou označují jako CPLD (Complex Programmable Logic Device). Typická struktura obvodu CPLD je znázorněna na Obr. 9.



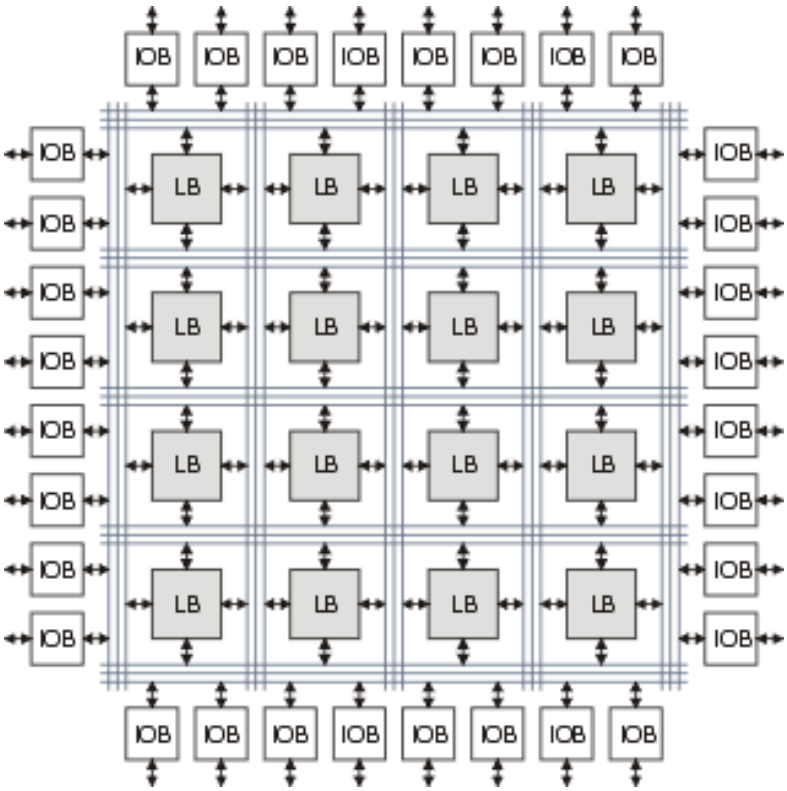
Obr. 9: Struktura obvodu CPLD

Zdroj: http://sweb.cz/fpga/

Každý výrobce CPLD používá trochu jinou interní strukturu obvodů, ale většinou vychází z tohoto schématu. CPLD od různých výrobců se obvykle liší v provedení bloků vlastní programovatelné logiky, i když většinou vychází z klasické struktury PAL.

3.2.3 Obvody FPGA

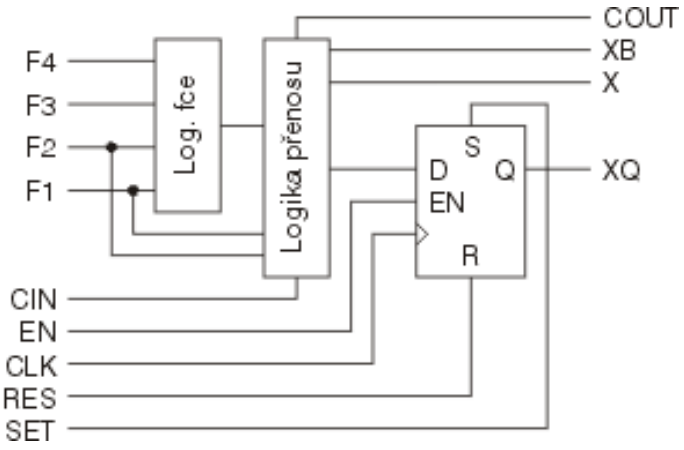
Obvody typu FPGA (Field Programmable Gate Array) mají z programovatelných obvodů nejobecnější strukturu a obsahují nejvíce logiky. Současné největší obvody FPGA obsahují až 6 milionů ekvivalentních hradel (typické dvouvstupové hradlo NAND). Typickou strukturu obvodu FPGA znázorňuje Obr. 10.



Obr. 10: Struktura obvodu FPGA

Zdroj: http://sweb.cz/fpga/

Bloky označené IOB (Input/Output Block) představují vstupně-výstupní obvody pro každý pin FPGA. Tyto bloky obvykle obsahují registr, budič, multiplexor a ochranné obvody. Bloky LB (Logic Block) představují vlastní programovatelné logické bloky. Všechny bloky mohou být různě propojeny globální propojovací maticí. Nejpoužívanější struktura konfigurovatelného logického bloku je znázorněna na Obr. 11.



Obr. 11:Struktura konfigurovatelného logického bloku

FPGA obvykle umožňuje propojit některé signály logických bloků přímo se sousedním bez nutnosti využívat globální propojovací matici. Tyto spoje mají mnohem menší zpoždění a umožňují tak realizovat například rychlé obvody šíření přenosu, což je nezbytné pro sčítačky nebo násobičky.

Kromě bloků znázorněných na předchozích obrázcích integrují výrobci do FPGA další prvky. Většina moderních FPGA obsahuje několik bloků rychlé synchronní statické paměti RAM. Velmi často obvody FPGA obsahují PLL (Phase Locked Loop) nebo DLL (Delay Locked Loop) pro obnovení charakteristik hodinového signálu, případně pro násobení nebo dělení jeho frekvence. Nejnovější hradlová pole často obsahují bloky vhodné pro vytváření složitých systémů pro číslicové zpracování signálů jako jsou například hardwarové násobičky nebo dokonce mikroprocesory.

Pro spínání byl vybrán obvod FPGA, protože nám umožní realizaci samotného hardwarové spínání a zároveň bude pracovat jako dělička hodinových signálů potřebných pro přenos dat po sériových linkách.

Konfigurace jednotlivých prvků a spínačů popsaného systému může být uložena v nejrůznějších druzích paměťových buněk. Nejtypičtější jsou systémy s konfigurací ve statické paměti (SRAM), pro jiné aplikace mohou být vhodné obvody s nepřepisovatelnými buňkami (propalovaných propojkách).

Obvody s konfigurací uloženou ve SRAM paměti je nutné znovu konfigurovat po každém připojení napájecího napětí. Potřebnou konfiguraci lze načíst buď z počítače připojeného k vývojové desce, nebo, a to je častější případ, z pevné paměti připojené k FPGA. Většinou je řešení takové, že obvod obsahuje vlastní FLASH paměť, která se při připojení napájení nahraje do SRAM paměti. Naopak systémy postavené na obvodech s propalovanými spojkami se jednou naprogramují a potom už jejich program nelze měnit. Výhodnější a také levnější je vyrábět systémy s nepřepisovatelnými buňkami ve větším množství.

Pro naše potřeby je nejvíce vyhovující FPGA obvod s FLASH pamětí, která umožňuje programování přímo v zařízení. Tato vlastnost obvodu pro výrobu prototypu ASSSK je velmi výhodná, protože umožňuje měnit funkčnost obvodu bez nutnosti odpojení FPGA. Vybrán byl obvod od firmy ACTEL z rodiny ProASIC 3.

3.2.4 Popis obvodu ProASIC®3 Flash Family FPGA

Možný výběr obvodů z této rodiny:

ProASIC3 Devices	A3P030	A3P060	A3P125	A3P250	A3P400	A3P600	A3P1000
System Gates	30 k	60 k	125 k	250 k	400 k	600 k	1 M
VersaTiles (D-Flip-Flops)	768	1 536	3 072	6 144	9 216	13 824	24 576
RAM kbits (1,024 bits)	-	18	36	36	54	108	144
4,608 Bit Blocks	-	4	8	8	12	24	32
Secure (AES) ISP2	-	Ano	Ano	Ano	Ano	Ano	Ano
I/O Banks	2	2	2	4	4	4	4
Maximum User I/O	81	96	133	157	194	227	300
Package Pins	QN132 VQ100	QN132 VQ100 TQ144 FG144	QN132 VQ100 TQ144 PQ208 FG144	QN132 VQ100 TQ144 PQ208 FG256	PQ208 FG144 FG256 FG484	PQ208 FG144 FG256 FG484	PQ208 FG144 FG256 FG484

Tabulka 3: Přehled obvodů ProAsic3

Zdroj: Actel Datasheet

Funkce obvodu nebude náročná na počet použitých buněk, a proto bude plně dostačující typ **A3P 060**, který obsahuje 60 tisíc systémových bran. Obvod je také schopen pracovat s frekvencemi potřebnými pro sériový přenos dat v dostatečné míře pro naší aplikaci.

Tento obvod má možnost standardně používat pro vstupy a výstupy LVTTL 3,3V (v tomhle režimu zvládne i 5V, ale při dlouhodobém používaní by mohlo dojít k poškození obvodu), LVCMOS 2,5V, LVCMOS 1,8V a LVCMOS 1,5V logiku. Jelikož procesor i MAX232 pracují v 5V TTL logice použijeme LVTTL 3,3V.

Základní vlastnosti tohoto obvodu jsou:

- Počet systémových bran: 60 tisíc
- Počet I/O Bank: 2
- uživatelských I/O 91 při použití pouzdra TQ144
- Pouzdro TQ144
- Maximální vstupně výstupní frekvence při LVTTL je 200 MHz

Obvod A3P060 má jádro obsahující 1536 buněk (versa tiles), které nám umožňují propojení logické sítě. Po kompilaci programu jich je zapotřebí 1405, z toho plyne, že jádro obvodu je využito na 91%. Vzhledem k tomu, že při potřebě rozšířit využití FPGA by muselo dojít i k přepracování plošného spoje, není zapotřebí použití výkonnějšího FPGA.

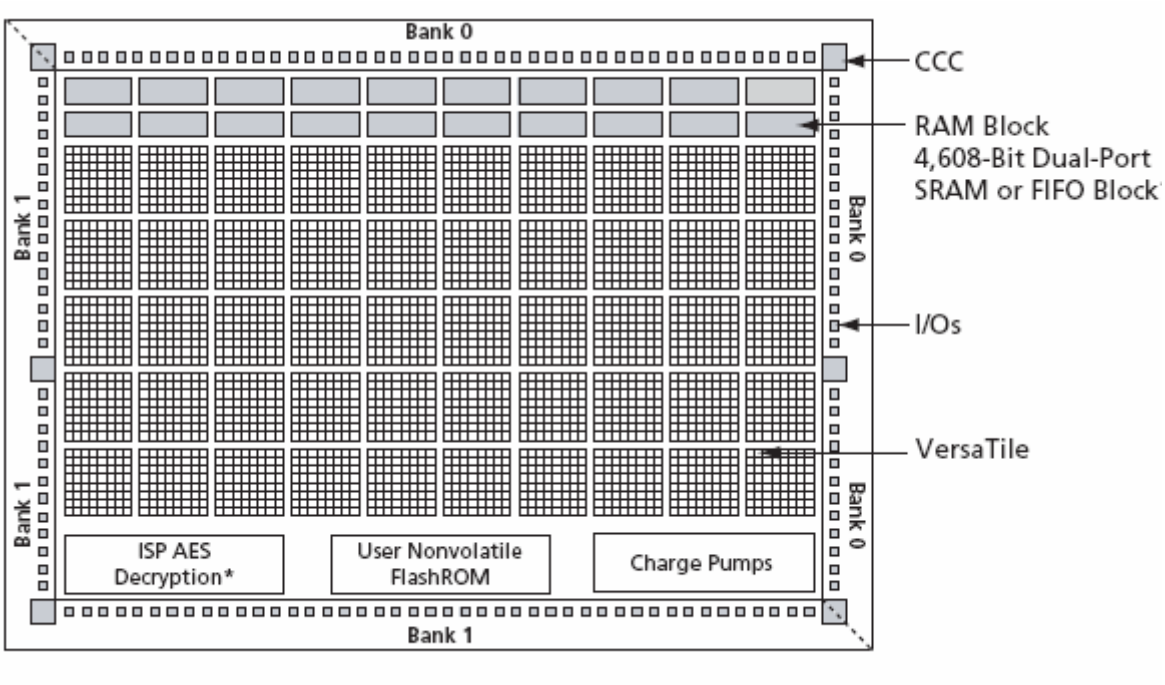
Dalším důležitým kritériem je počet potřebných vstupů/výstupů (I/O). Cílem je, aby počet spínaných RS232 rozhraní byl alespoň 20. K tomu je zapotřebí 73 (Tabulka 4) vstupně výstupních pinů. Obvod A3P060 TQ144 má počet 91 I/O, což znamená, že maximální počet spínaných portů je 26. Podrobný popis zapojení pinů obvodu FPGA je v příloze 1.

Použití pinů		**Počet**
Signály spínaných rozhraní RS232	TXD	20
	RXD	20
	CLK	20
Řídící signály z procesoru		12
Vstupní hodinový signál pro generování hodinových signálů		1
Celkem		**73**

Tabulka 4:Využití pinů v FPGA

Obvod A3P060 lze přeprogramovat přímo v zapojení. K tomuto účelu slouží konektor JTAG, proto bylo potřeba upravit návrh desky plošných spojů (DPS) tak, aby na ní byl vyveden. Konektor JTAG je určen k propojení s programátorem FlashPro3. Ten komunikuje s PC přes rozhraní USB.

Struktura obvodu A3P060 je na obrázku 12.



Obr. 12: Struktura obvodu ProASIC®3 A3P060

Zdroj: Actel Datasheet

Základní prvky obvodu:

- Bank 0, Bank 1 – obsahují vstupně výstupní piny (každý Bank může pracovat na jiné napěťové úrovni)
- CCC (Clock Conditioning Circuitry) – piny, které jsou připojeny na obvody umožnují, rychle zpracovávat (upravovat) hodinové signály (např. libovolné dělení, násobení)
- Versa Tile – buňky v jádru obvodu umožňující propojení kombinační sítě

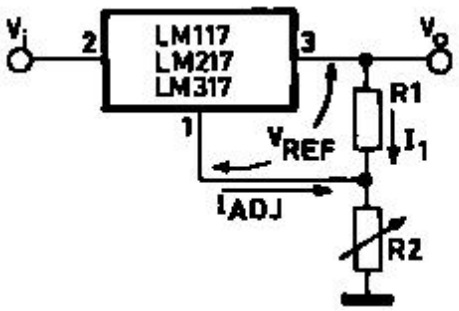
3.3 Pomocné obvody

Pro chod celého zařízení jsou potřeba i jiné obvody, které je nutné do systému zařadit. Patří mezi ně obvody pro převod logických úrovní TTL na logické úrovně RS232 a stabilizátory napětí.

Stabilizátory napětí

Pro chod zařízení je potřeba tří úrovní napětí:

- 5V pro procesor a max232, obvod 7805.
- 3,3V pro napájení I/O rozhraní FPGA, obvod LE33C.
- 1,5V pro napájení jádra FPGA. Obvod pro stabilizaci na toto napětí není běžný. Z tohoto důvodu se musí použít stabilizátor s nastavitelným výstupním napětím od 1,2V do 32V LM317L. Zapojení stabilizátoru je na Obr. 13. Hodnota výstupního napětí se počítá podle vzorce $V_0 = Vref\left(1 + \frac{R_2}{R_1}\right) + I_{adj} R_2$. Použité hodnoty odporů jsou R_1=2,2 kΩ a R_2=150 Ω.



Obr. 13: Schéma zapojení stabilizátoru LM317

Zdroj: Katalogový list obvodu LM317 [10]

3.3.1 Obvod pro převod logiky rozhraní RS232 na TTL logiku

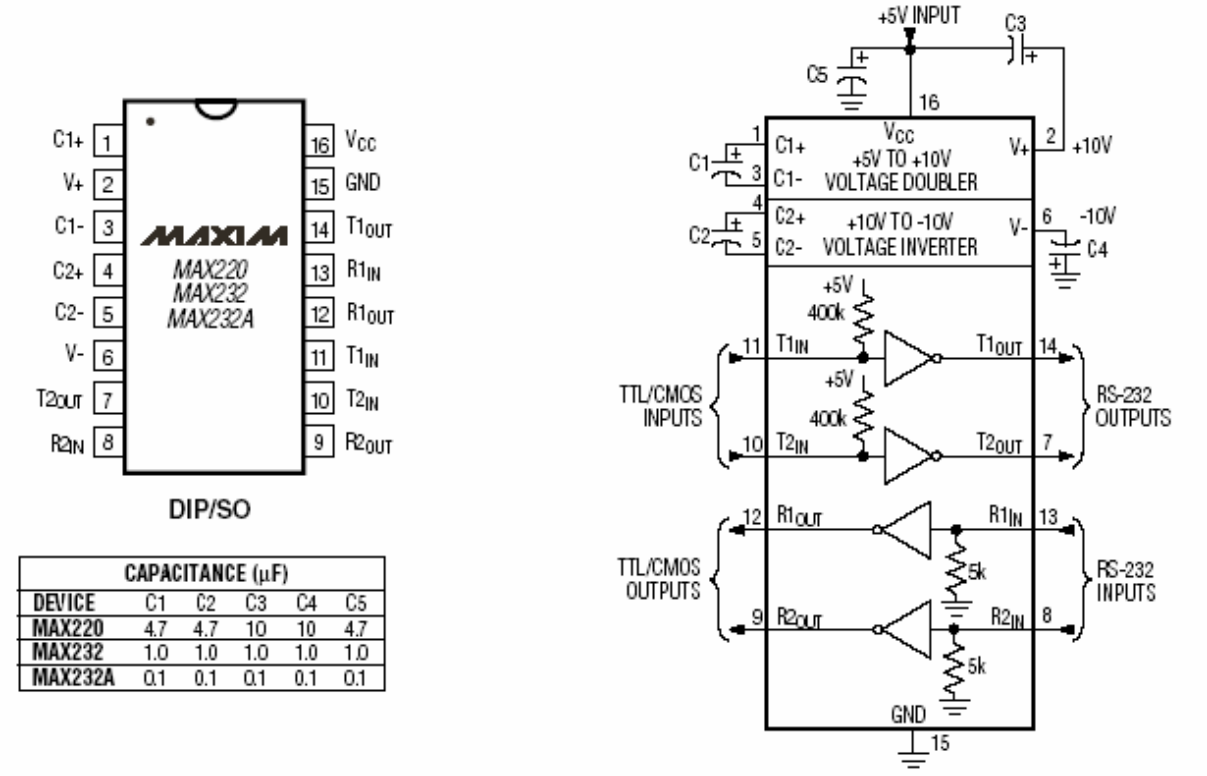
Rozhraní RS232 je specifické tím, že hodnoty logické jedničky jsou záporné od -3 do -12 V a hodnoty logické nuly kladné od +3 do +12 V. Signály o těchto napěťových hodnotách by bylo obtížné zpracovat, proto je nutné převést hodnoty na TTL logiku. Podrobný popis rozhraní RS232 je v kapitole 2.1 Fyzická vrstva rozhraní RS232.

K tomuto účelu byl vybrán obvod MAX232A od firmy Maxim. Nevýhodou tohoto obvodu je, že jeho přenosová rychlost dat je podle katalogového listu jen 200 kb/s. Při testování byl naměřen maximální přenos dat při kterém routery bezchybně přijímaly 300 kb/s. Tímto nás tyto obvody omezují v rychlejším přenosu dat. Pro naše potřeby je to zcela dostačující.

Pro dosažení větší rychlosti by bylo zapotřebí použití jiného obvodu, čímž by se zařízení prodražilo.

Obvod MAX232 obsahuje dva převodníky z TTL do RS232 a dva z RS232 do TTL. Tyto převodníky umožňují převod pouze pro jeden port, protože musíme převést z TTL na RS232 signál RXD a hodinový signál a z RS232 na TTL signál RXD.

Blokové schéma a pouzdro obvodu je na Obr. 14.



CAPACITANCE (µF)					
DEVICE	C1	C2	C3	C4	C5
MAX220	4.7	4.7	10	10	4.7
MAX232	1.0	1.0	1.0	1.0	1.0
MAX232A	0.1	0.1	0.1	0.1	0.1

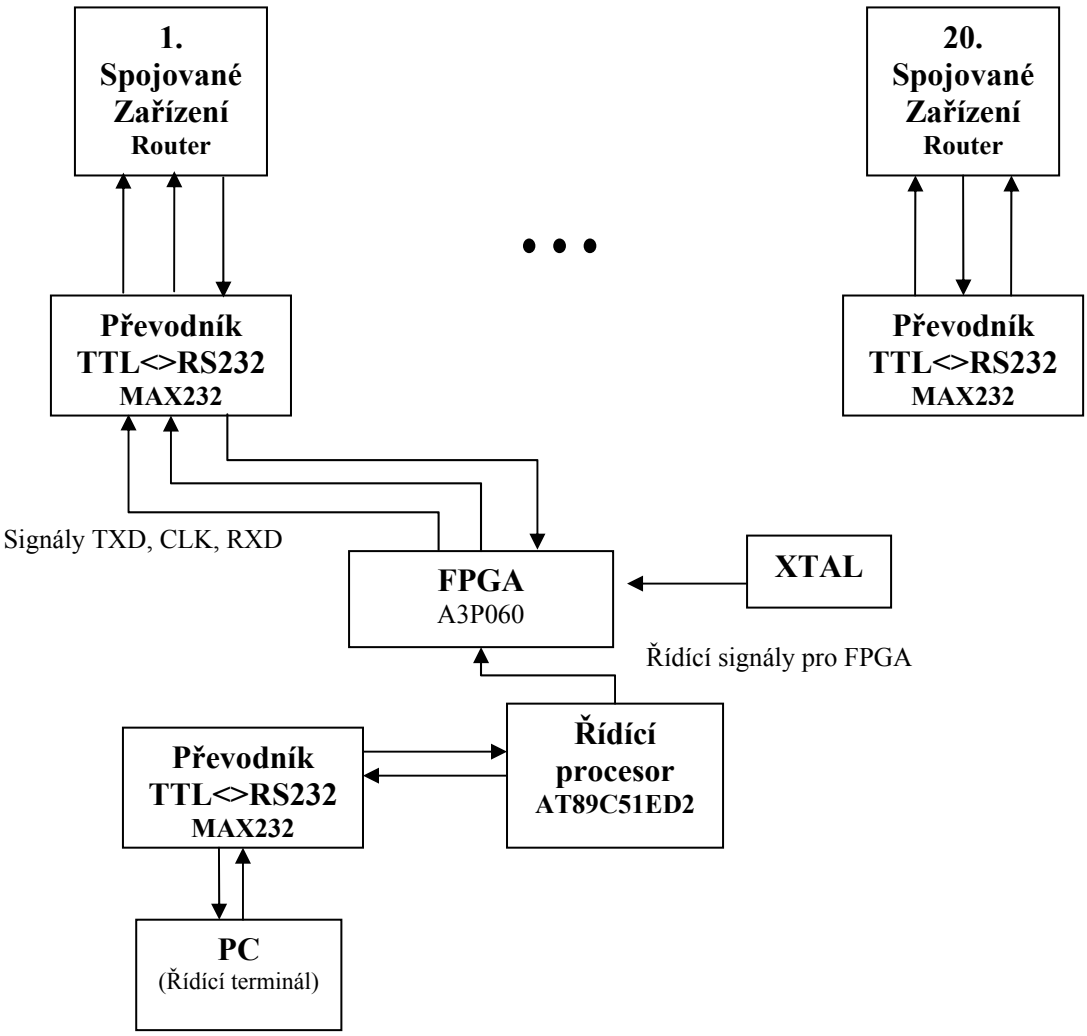
Obr. 14: Pouzdro a blokové schéma obvodu MAX232A

Zdroj: Katalogový list firmy MAXIM

4 Návrh řešení

4.1 Popis hardwarového návrhu

Nejdříve je důležité uvést blokové schéma zapojení (Obr. 15), které přiblíží funkci ASSSK 2 a zapojení obvodů.



Obr. 15: Blokové schéma zapojení

Popis blokového schématu zapojení:

- **Řídicí procesor** dostává informace z PC, které porty má spojit či rozpojit, a jakým hodinovým signálem má tyto porty taktovat. Tyto data zpracuje a vyšle řídící signály pro FPGA.
- **Obvod FPGA** realizuje samotné hardwarové propojení portů a zvolí požadované synchronizační hodiny pro sériový přenos. Získá je vydělením vstupního signálu z krystalového oscilátoru.
- **Převodník TTL<>RS232** převádí TTL logiku na logiku rozhraní RS232 pro sériovou linku a naopak.

Po ověření funkčnosti návrhu (kapitola 5.1 Ověření návrhu) bylo dalším krokem ve vývoji ASSSK 2 návrh DPS. Jako nástroj pro kresbu schématu a desky plošných spojů byl použit program EAGLE v. 4.16 (Easily Applicable Graphical Layout Editor).

Podrobné schéma zapojení je uvedeno v příloze 3. Schéma obsahuje jednu společnou sběrnici, která má v sobě obsaženy všechny potřebné signály a je identické s deskou plošných spojů, které se nechala vyrobit.

4.2 Popis programového vybavení ASSSK

Dále je nutné popsat komunikaci, která bude probíhat mezi řídícím procesorem a spínacím obvodem FPGA. Komunikace bude probíhat jen jednosměrně, pomocí čtyř signálů (řídíci signály na Obr. 15: Blokové schéma zapojení).

- **Signál Adr_blok (0..4)** – Signál (vektor) obsahuje adresu výstupu(buňky), který má být použit.
- **Signál Adr_F (0..4)** – Signál (vektor) obsahuje, který vstup se má použít.
- **Signál Wr_C** – Na nástupnou hranu signálu bude provedeno propojení TXD a RXD podle vektorů Adr_blok a Adr_F.
- **Signál Wr_F** – Na nástupnou hranu signálu bude provedeno připojení frekvence podle vektoru Adr_F na port určující vektor Adr_blok.

4.2.1 Software řídící jednotky AT89C51ED2

Program pro řídící procesor má za úkol komunikovat s obsluhou prostřednictvím terminálu a má řídit obvod FPGA tak, aby byla propojována potřebná rozhraní a frekvence dle pokynů obsluhy. Celý program je napsán v jazyce ANSI C

Program lze rozdělit na dvě části. Část pro komunikaci s terminálem a část pro obsluhu spojovacího obvodu FPGA. První část pro komunikaci s terminálem byla převzata a pozměněna pro nové potřeby z předchozího ASSSK. Druhou část bylo nutno celou naprogramovat.

Struktura programu

Program je rozdělen do souborů obsahujících jednotlivé funkce. Nejdříve jsou uvedeny soubory a jejich funkce, které byly převzaty z původního programu ASSSK 1 nebo byly změněny jen minimálně.

RS232.c – Tento soubor obsahuje základní rutiny pro komunikaci s nadřazeným PC. Tedy nastavuje potřebné parametry pro sériový asynchronní přenos. Stará se o řízení toku (flow control), aby nedošlo ke stavu, kdy zpracovávání znaku bude delší než doba příchodu dalšího znaku.

tcomand.c – Funkce tohoto souboru má za úkol analyzovat přijatý řetězec, poté vyhodnotit, zda se jedná o platný příkaz, a nakonec spustit příslušnou funkci, která bude reagovat na tento příkaz. Funkce zavádí do komunikace s obsluhou možnost zadávat příkazy ve zkrácené formě.

t_connect.c – Soubor s funkcí, která je pro celý systém nejdůležitější. Funkce změní systémové proměnné tak, aby bylo dosaženo propojení dvou portů.

eeprom.c – Většina předchozích souborů s funkcemi prováděly pouze nastavování vnitřních proměnných. Všechny tyto proměnné jsou definovány v tomto souboru. Součástí tohoto souboru jsou také funkce pro komunikaci s interní pamětí EEPROM a dovedou do ní zapsat systémové proměnné uložené v paměti RAM.

Nyní jsou uvedeny soubory, které jsou nové nebo byly hodně modifikovány.

fpga.c – Program obsahuje jedinou funkci fpga_switch, která provádí samotnou komunikaci s obvodem FPGA. Posílá mu potřebné informace co s čím spojit a jakou nastavit taktovací frekvenci. Tato funkce se volá pokaždé, při jakékoliv změně vnitřních proměnných.

main.c – Soubor obsahuje základní funkci main(), která je volána jako první po spuštění programu. Funkce provede inicializační rutiny a poté přejde do nekonečné smyčky. Všechny další funkce jsou volány na základě přerušení.

t_clock – V souboru je funkce, která nastaví vnitřní proměnné na požadovanou hodnotu frekvence pro příslušný port.

t_reset.c – Tento soubor obsahuje funkci, která převede všechny systémové funkce do základního stavu (nejsou spojeny porty, není nastavena žádná taktovací frekvence, je vymazán popis portů).

Dále je uvedena funkce souborů, které byly odstraněny z původního programu pro mikroprocesor. Tyto soubory obsluhovaly obvody, které již nejsou součástí nového zařízení ASSSK 2.

MT8816.c – Tento soubor obsahuje funkce, které zprostředkovávají komunikaci s procesorem a analogovým spínacím polem MT8816.

I2C.c – Zde jsou umístěny všechny funkce, které obsluhují sběrnici I2C, která pomáhá redukovat počet použitých pinů procesoru.

PCF8574.c – Zde se nacházejí funkce definující komunikaci s obvody PCF8574, které převádějí komunikaci po sběrnici I2C tam a zpět na jednotlivé bity. Jsou zde řešeny jak rutiny čtení, tak rutiny zápisu do těchto obvodů.

Podrobný popis těchto funkcí a zmíněných obvodů je uveden v Diplomové práci Ing. Davida Seidla [6].

Příkazový jazyk řídící jednotky

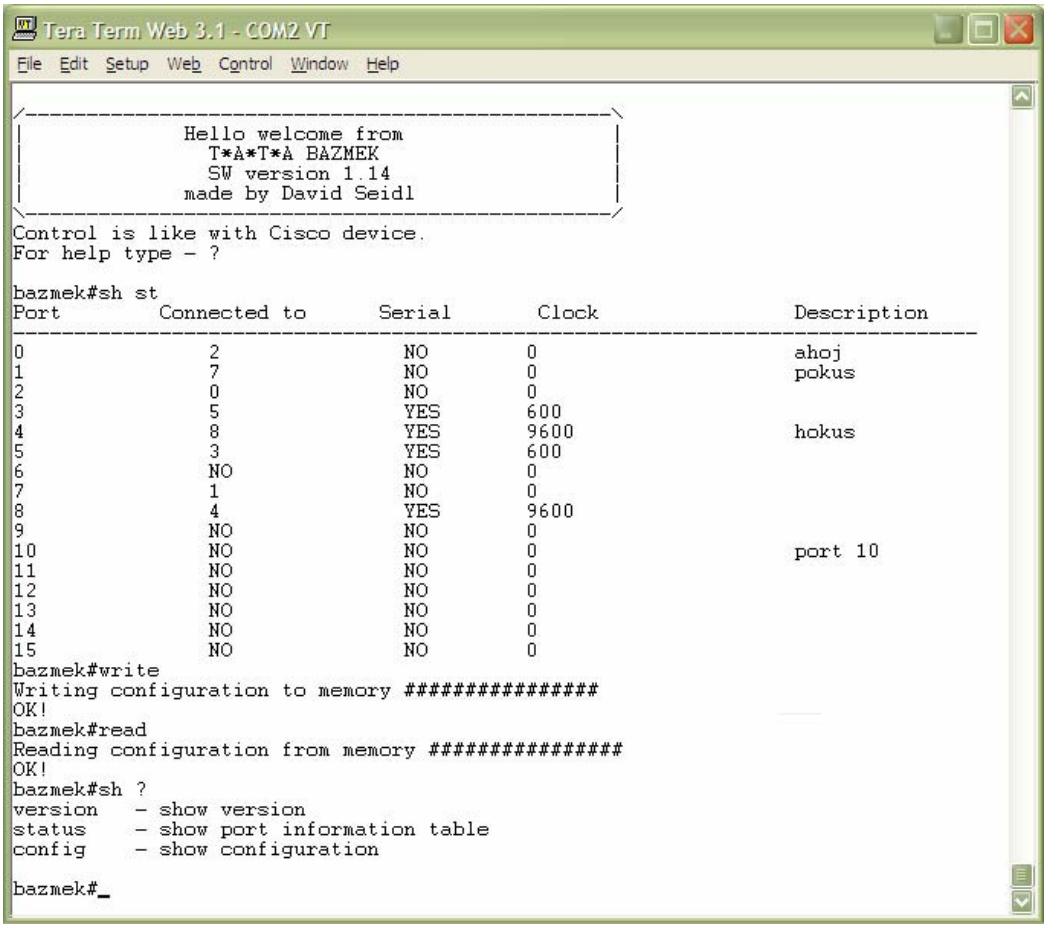
Komunikace probíhá prostřednictvím sériového terminálu. Aby bylo ovládání co nejjednodušší, bylo naprogramováno podobně jako u síťových zařízení Cisco.

Príkazy:

- **connect *a b*** - propojí port *a* s portem *b*, pokud již některý z portů není s něčím spojen
- **clock *a f* –** pokud je port a spojen s nějakým portem, je potom na tyto porty připojena frekvence *f*
- **reset –** příkaz nastaví všechny frekvence na 0, rozpojí všechny porty, a vymaže všechny popisy portů
- **description *a text* –** k portu *a* přiřadí popis *text* jehož maximální délka je 9 znaků
- **write –** zapíše aktuální konfiguraci do paměti EEPROM, tato konfigurace bude načtena po zanutí přístroje
- **read –** nastaví systém do stavu uloženého v EEPROM
- **no –** vložením **no** před příkaz se provede negace požadovaného příkazu. (např. no connect *a*, no description, no clock)

Terminálová aplikace obsahuje i funkci historie příkazů, která umožňuje opětovně vypsat již provedené příkazy. Další funkce, která má sloužit k většímu komfortu obsluhy, je funkce zkrácených příkazů. Tato funkce porovnává napsaný text s databází příkazů či parametrů daného příkazu, a pokud napsaný text jednoznačně definuje příkaz či parametr, je potom proveden. Tedy například příkaz *sh* je identický s příkazem *show* apod. Poslední důležitou funkcí je nápověda. Výpis nápovědy ke každému z příkazů je aktivován parametrem „?“. Tedy nápověda k příkazu *show* se vyvolá zadáním *show ?* či zkráceně *sh ?*.

Náhled do terminálové aplikace je naznačen v na Obr. 16. Je zde vidět uvítací hlášení, výstup příkazu *show status* (zkrácený na *sh st*), *read, write* a *sh ?*.



Obr. 16: Náhled do terminálové aplikace

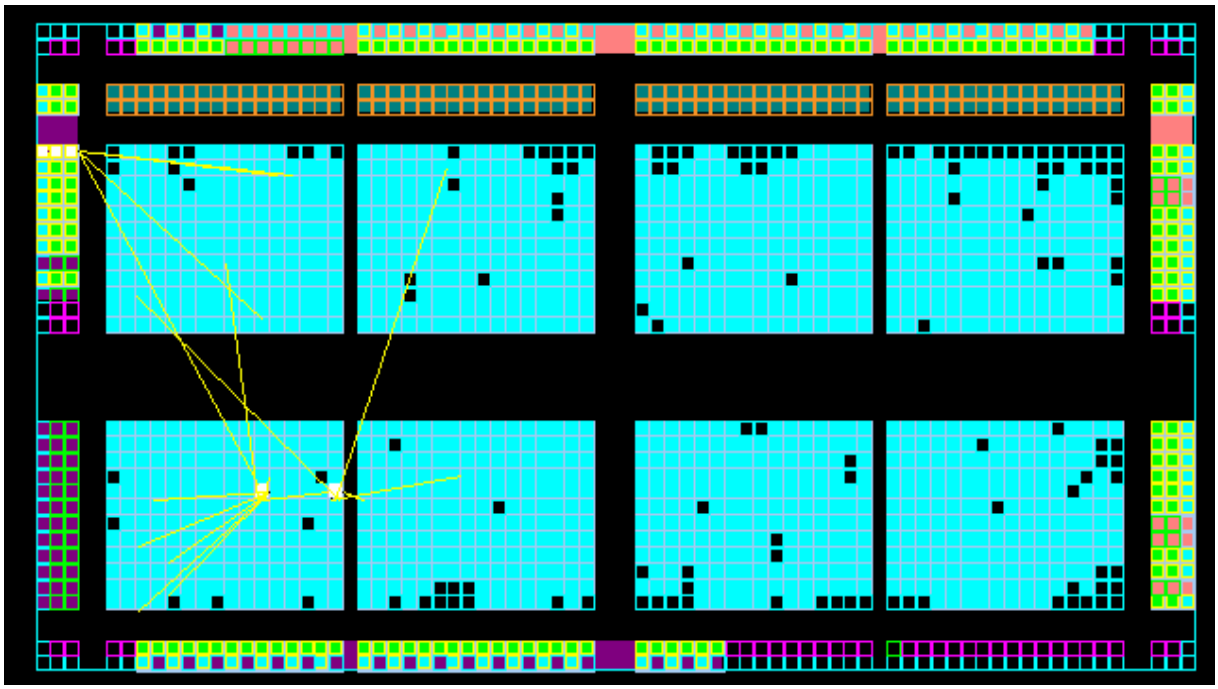
4.2.2 Návrh VHDL modelu

Základní vlastností FPGA je možnost naprogramovat propojení logických obvodů.

Nejprve je nutné celý blok popsat pomocí syntetizovatelné podmnožiny jazyka VHDL. Popis hardwaru pomocí programovacího jazyka umožňuje srozumitelný zdrojový text na poměrně vysoké úrovni abstrakce. Chování můžeme ověřit pomocí simulátoru (Obr. 22). Pro programování a simulaci byl použit program ModelSim XE.

Dalším krokem je převod VHDL popisu do konfiguračních dat pro programovatelné hradlové pole. Nejprve se provede syntéza – převod VHDL na obecné a technologicky nezávislé schéma. Schéma je pak v následujících krocích (mapování na technologii, rozmístění, propojení a generování bitového toku) převedeno přímo na konfiguraci hradlového pole. Nakonfigurované hradlové pole s naším obvodem můžeme znázornit graficky. Tyto kroky byly provedeny ve vývojovém prostředí Libero IDE v7.2.

Blokové schéma programovatelného hradlového pole ProAsic3 A3P060 (výrobce Actel) s rozmístěnými prvky obvodu je na Obr. 17. Uprostřed jsou světle modře označeny použité versa tiles a černě nepoužité. Na okraji jsou zeleně označeny použité vstupy a výstupy, tmavě zelené jsou paměťové buňky RAM bloku, který v programu nevyužívám.



Obr. 17: Blokové schéma programovatelného hradlového pole ProAsic3 A3P060 s rozmístěnými prvky obvodu

Dále jsou ve vývojovém prostředí Libero zajímavé statistiky, které nám ukazují zpoždění mezi jednotlivými propojení vstupů a výstupů obvodu FPGA. Tyto statistiky jsou přístupné po převedení schématu na konfiguraci hradlového pole. Jedna ze statistik je například zpoždění FPGA mezi propojovanými signály. V Tabulka 5 jsou uvedeny dvě propojení s největším a dvě s nejmenším zpožděním.

Zdrojový pin	**Cílový pin**	**Zpoždění (ns)**	
		Min	**Max**
Txd_in(1)	rxd_out(18)	8,276	13,583
Txd_in(17)	rxd_out(7)	9,790	13,483
Txd_in(16)	rxd_out(3)	7,676	10,863
Txd_in(7)	rxd_out(18)	7,831	10,862

Tabulka 5: Zpoždění mezi spojenými signály

4.2.2.1 Popis základních principů jazyka VHDL

Návrh je typicky rozdělený do několika bloků. Tyto bloky jsou pak spojeny dohromady a tvoří kompletní návrh. VHDL návrh může být kompletně popsán v jediném souboru, nebo rozložen do několika menších souborů. Každý blok v jazyce VHDL má dvě základní části: deklaraci entity a tělo architektury. Deklarace entity popisuje vstupy a výstupy bloku, tělo architektury definuje její funkci.

Architektura může být psána různými styly jazyka VHDL. Nejčastěji se označují jako styl behaviorální, styl dataflow (tok dat) a styl strukturální.

Pojem **behaviorální styl** se obvykle nazývá **proces** (*process*). Význačným rysem procesu je jeho algoritmický charakter, čímž se text procesu podobá počítačovým programovacím jazykům. Tato podoba odpovídá skutečnosti, je-li v procesu zpracovávána proměnná. Při zpracovávání signálů se provede predikce a signál je nastaven na požadovanou hodnotu až při ukončení procesu. Každý proces se provede jednou při startu programu, a potom jen dojde-li ke změně hodnot nastavených v citlivostním seznamu proměnných nebo signálů. Pomocí tohoto stylu je například popsán obvod D, který slouží k dělení frekvence vstupního signálu.

Ukázka D obvodu popsaného v jazyce VHDL pomocí procesu

```
entity delicka is                                  --začátek entity delicka
  port(                                            --začátek deklarace vst/vyst portů
  vst_CLK : in std_logic;                          --vstupní signál vst_CLK
  vyst_CLK : inout std_logic);                     --výstupní signál vyst_CLK
end entity;                                        --konec  entity

architecture archdelicka of delicka is             --tělo architektury vycházející z entity delicka
 begin
     process(vst_CLK)                              --proces, který je citlivý na změnu signálu
                                                   vst_CLK
       begin
        if rising_edge(vst_CLK) then               --na nástupnou hranu vst_CLK se provedou
                                                   následující příkazy
vyst_CLK<=  not vyst_CLK ;                         --vystupu vyst_CLK se přiřadí negovaný
                                                   vst_CLK
if vyst_CLK='U' then vyst_CLK<='0'; end if;        --pokud je vst_CLK nedefinovaný pak
                                                   vyst_CLK přiřaď ‚0‘ (nutné pro simulaci)
         end if;                                   --ukončení příkazů reagujících na nástupnou
                                                   hranu signálu vst_CLK
     end process;                                  --ukončení procesu
  end architecture;                                --ukončení těla architektury
```

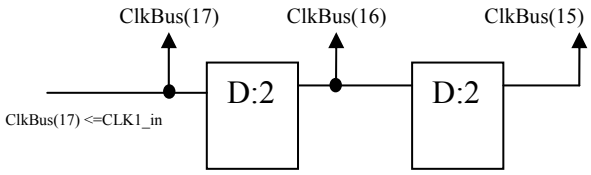
Styl **dataflow** neprovádí příkazy sekvenčně, ale souběžně (paralelně) na rozdíl od sekvenčních příkazů používaných v procesech. U souběžných příkazů nezávisí výsledek na pořadí jejich zápisu v textu.

Strukturální popis je třetím význačným stylem. Spočívá ve vkládání komponent (*component*) do kódového útvaru. Strukturální styl se nejčastěji používá pro popis propojení dílčích bloků nižší úrovně. V našem příkladě jsou propojeny entity dělička.

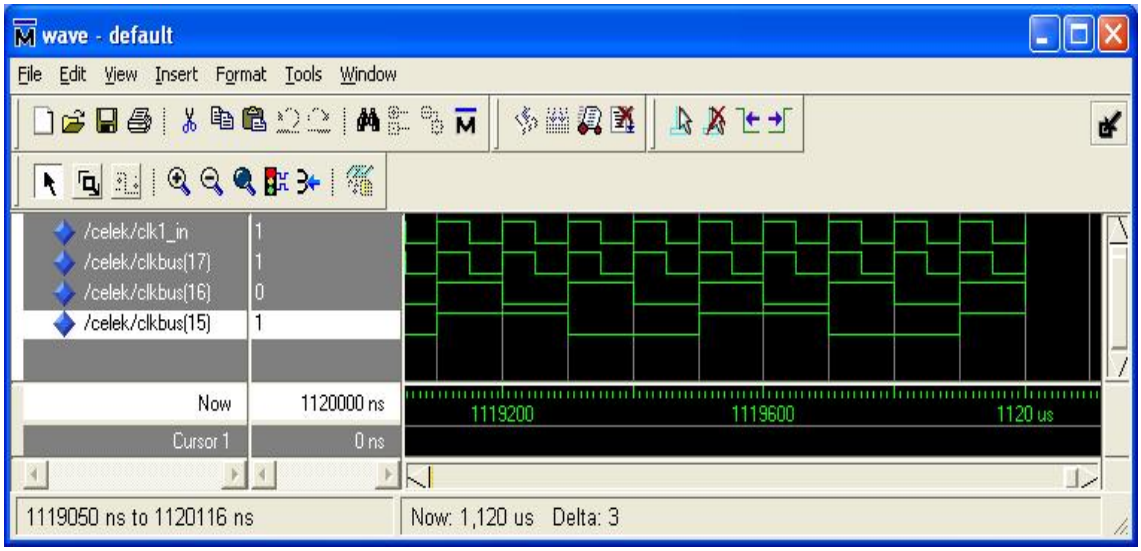
U těchto dvou stylů nezáleží na pořadí příkazů (všechny probíhají souběžně). Příklad použití je ukázán na zapojení dvou entit delička (obvod D) za sebou (Obr. 18).

```
entity celek is                                  --začátek deklarace entity celek
   port(clk0_in:in std_logic;                    --vstupní signál clk_0, který chceme dělit
);
end entity;                                      --konec deklarace entity

architecture archcelek of celek is               --tělo architektury vycházející z entity celek
  component delicka is                           --pro použití entity delička je nutné ji uvést jako
                                                 komponentu
       port(
       vst_CLK : in std_logic;
       vyst_clk : inout std_logic);
     end component;

  signal ClkBus :std_logic_vector (17 downto 0); --deklarace vnitřního vektoru

ClkBus(17)<=clk1_in;                             --styl dataflow, na signál ClkBus(17) připoj
                                                 vstupní signál clk1_in

  D01: delicka port map (ClkBus(17),ClkBus(16)); -- styl strukturální popis, vytvoření deličky
                                                 D01, která má jako vstup signál ClkBus(17) a
                                                 výstup ClkBus(16), popis jejího chování je
                                                 v entitě dělička
  D02: delicka port map (ClkBus(16),ClkBus(15)); --vytvoření deličky D01, která má jako vstup
                                                 signál ClkBus(1) a výstup ClkBus(15)
                                                 --obě tyto děličky pracují současně nezávazně
                                                 na sobě
end architecture;                                --ukončení popisu architektury archcelek
```

Ukázka simulace propojení těchto komponent je na Obr. 19.



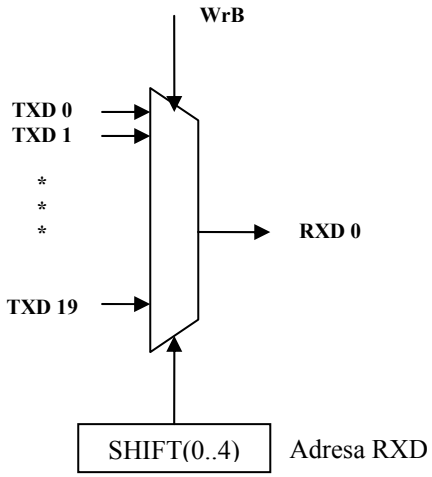
Obr. 18: Blokové schéma propojení děliček dvěma



Obr. 19: Simulace propojených komponent dělička v programu ModelSim XE III 6.0d

4.2.2.2 Blokové schéma spínání v programu pro FPGA

Základní součástí programu je buňka s multiplexorem, který bude provádět samotné propojení signálů. Těchto buněk je v programu dvacet (pro každý výstupní port jedna).



Obr. 20: Blokové schéma buňky pro spínání RXD s TXD

TXD 0 .. 19 jsou vstupní signály do obvodu FPG. Výstupní signál RXD neboli buňka, která bude použita, se vybere pomocí 20 bitového vnitřního signálu obsahujícího vždy jen jednu jedničku. Její pozice určuje, která buňka má propojit TXD s RXD a to podle signálu s aktuální adresou txd (Adr_TXD).

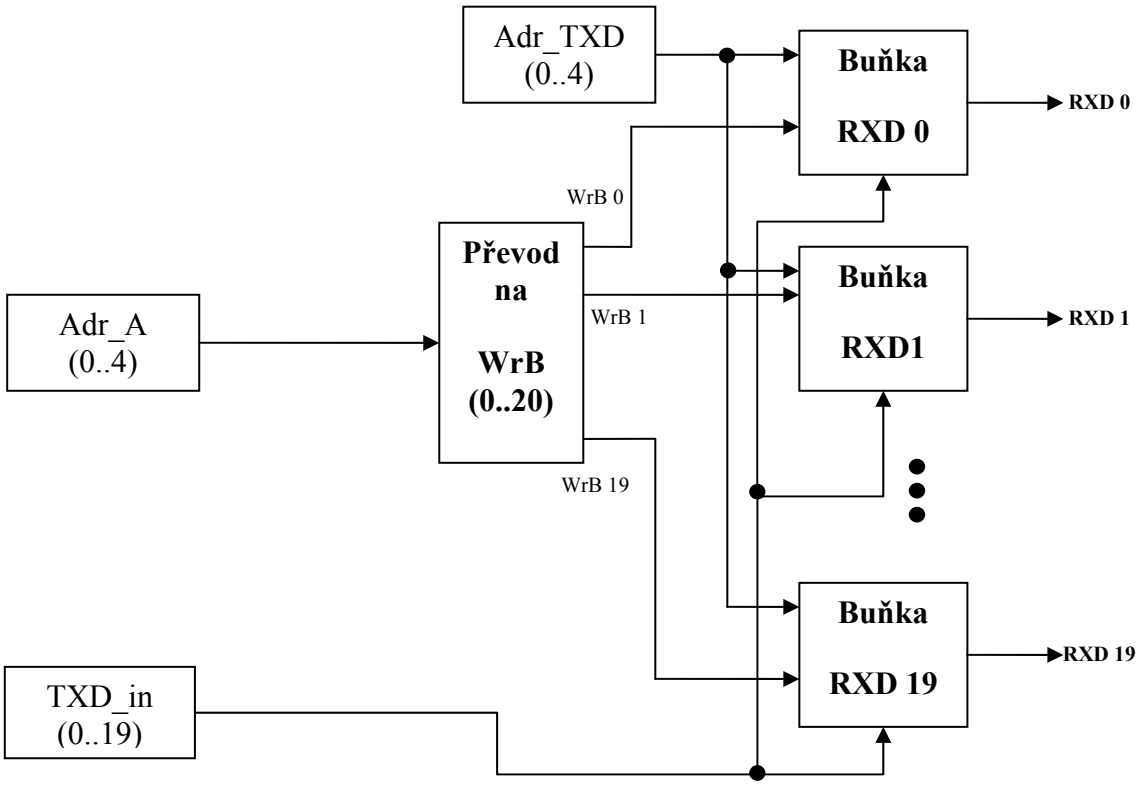
Buňky pro spínání hodinových signálů jsou stejné. Pouze místo vstupů TXD jsou vstupy frekvencí. Tyto frekvence vzniknou podělením vstupu FPGA z krystalového oscilátoru. Výstupem je signál CLK.

Buňka pro spojení TXD s RXD popsána pomocí jazyka VHDL

```
entity bunka_rxd is
port(
Adresa_TXD : in std_logic_vector(4 downto 0);      --vstupní vector do buňky s adresou TXD
TXD :in std_logic_vector(19 downto 0);             --vstupní vector do buňky nesoucí signály TXD
Y :out std_logic;                                  --výstupní signál RXD
WrB :in std_logic);                                --zápisový bit, na nástupnou hranu se provede
                                                   spojení

end entity;

---Spojování portu
architecture archbunka_rxd of bunka_rxd is         --začátek těla arhitekry archbunka
signal TXD_in :std_logic_vector(4 downto 0);       --vnitřní pomocný signál s adresou TXD
begin
P1: process(WrB)                                   --tento proces se bude vykonávat jen při
                                                   změně hodnoty signálu WrB

begin
if  rising_edge(WrB) then
```

```
TXD_in<=Adresa_TXD;                    -- při příchodu nástupné hrany WrB, se do
                                       vnitřního vektoru TXD_in zapíše vektor
                                       Adresa_TXD, který má jít na vystup
end if;
end process;

                                       --Tato část programu probíhá souběžněpořád
                                       od vzniku buňky
with TXD_in select                     --podmínka, kdy podle  TXD_in vybíráme, který
                                       signál se připojí na výstup.
Y<=  TXD(0) when "00000",
TXD(1) when "00001",
TXD(2) when "00010",
*
*
TXD(19) when "10011",
'0' when others;                       --pokud je TXD_in vetší než 19 (10011), dojde
                                       k rozpojení neboli na výstup se začne pouštět
                                       ‚0‘
end architecture;
```

Blokové schéma celého programu pro spojování TXD s RXD je na Obr. 21.



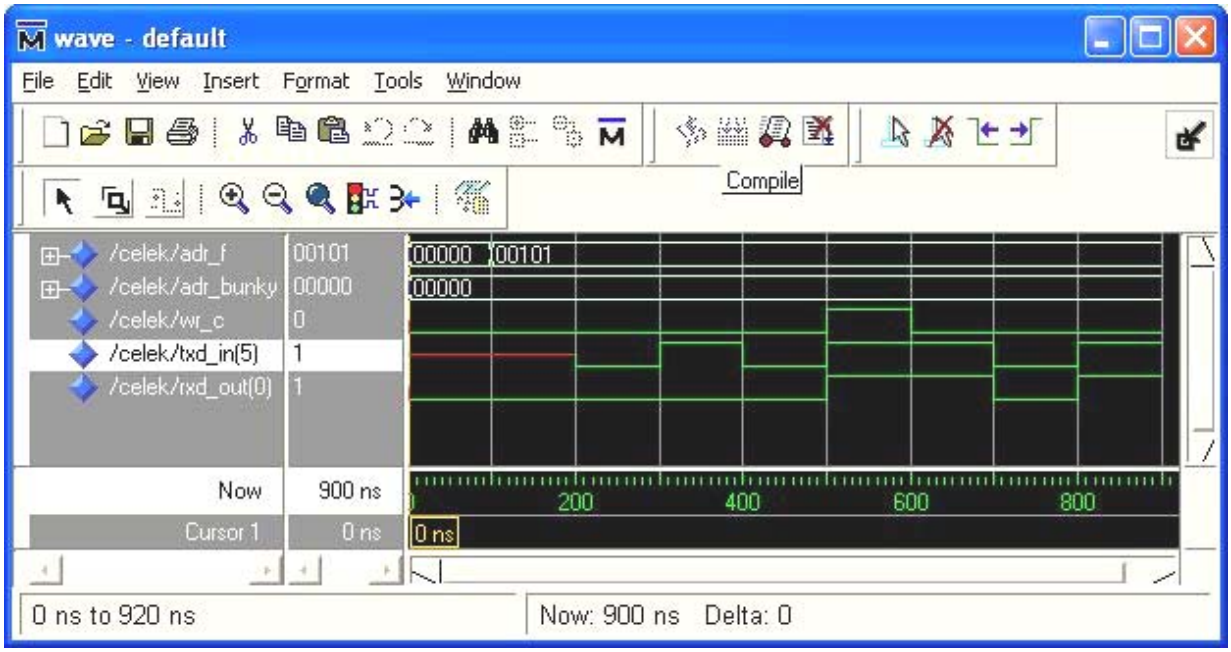
Obr. 21: Blokové schéma celého programu pro spojování TXD s RXD

Ukázka volání *bunka_RXD* z hlavní části programu v jazyce VHDL

```
entity celek is
  port(
   clk0_in,clk1_in:in std_logic;  --vstupy Xtal
   Clk_Out : out std_logic_vector (19 downto 0);          --porty ven
   Adr_Bunky : in std_logic_vector (4 downto 0);          --výběr výstupního portu(buňky)
   Adr_F  : in std_logic_vector (4 downto 0);             --výběr vstupu(frekvence nebo TXD)
   Wr_F   : in std_logic;                                 --zapisový signál pro frekvenci
   Wr_C   : in std_logic;                                 --zapisový signál pro spojení
   txd_in : in std_logic_vector(19 downto 0);             --vstupní vektor se signály TXD
   rxd_out : out std_logic_vector(19 downto 0)            --výstupní vektor se signály RXD
   );
end entity;

architecture archcelek of celek is
  signal WrAdr_C :std_logic_vector (5 downto 0);          --vnitřní pomocný vektor

  begin
   G2: for i in 0 to 19 generate
      K2: bunka_rxd port map (Adr_F,txd_in,rxd_out(i),Load_C(i));      --vygeneruje 20
    end generate;                                                      nezávislých buněk RXD

   WrAdr_C<=Wr_C&Adr_Bunky;                  --spojení signálu Wr_C s vektorem Adr_Bunky
   with WrAdr_C select                        -- vyber určitého signálu Load
    Load_C <= X"00001" when "100000",        --0
              X"00002" when "100001",        --1
              X"00004" when "100010",        --2
              X"00008" when "100011",        --3
              X"00010" when "100100",        --4
              X"00020" when "100101",        --5
              X"00040" when "100110",        --6
              X"00080" when "100111",        --7
              X"00100" when "101000",        --8
              X"00200" when "101001",        --9
              X"00400" when "101010",        --10
              X"00800" when "101011",        --11
              X"01000" when "101100",        --12
              X"02000" when "101101",        --13
              X"04000" when "101110",        --14
              X"08000" when "101111",        --15
              X"10000" when "110000",        --16
              X"20000" when "110001",        --17
              X"40000" when "110010",        --18
              X"80000" when "110011",        --19
              X"00000" when others;          --při příchodu jiného vektoru WrAdr_C pošli do
                                             vektoru load_C vektor se samými nulami
end architecture;
```

Simulace propojení portu 5 s portem 0

K propojení signálů txd_in(5) a rxd_out(0) se nejdříve musí nastavit 5 bitové vektory na požadované hodnoty, adr_f na hodnotu 5_D $(00101)_B$ a adr_bunky na 0_D $(00000)_B$. K propojení dojde až po nástupné hraně signálu wr_c. Ukázka simulace tohoto propojení je na Obr. 22. Aby mohla probíhat komunikace mezi požadovanými porty, musí se také spojit txd_in(0) a rxd_out(5). Postup bude stejný, jen se prohodí adr_bunky s adr_f. K rozpojení by došlo zasláním hodnot adr_f = 32_D $(11111)_B$ a adr_bunky = 0_D $(00000)_B$.



Obr. 22: Ukázka simulace v programu ModelSIM XE

4.3 Rozpočet návrhu řešení

Důležitým kritériem pro výběr z možností řešení byla cena obvodů a součástek. Odhad doby strávené nad přepracováním návrhu a jeho realizací je cca 85 hodin čistého času.

	cena/kus	kusů	celkem s DPH	poznámky
Výroba plošného spoje 400x200 mm	4000	1	**4000**	
µP AT89C51RD2	198	1	**198**	
Patice na uP 89S51-24PI	39	1	**39**	
FPG ProAsic3 A3P060	600	1	**600**	FPG
MAX 232 CPE	23	21	**483**	2x transmitter/receiver RS232
Zdroj	300	1	**300**	
PLD-10S	2	20	**40**	2 x 5 kontaktů
LPV 10	5	20	**100**	Konektor pro plochý kabel (male)
Plochý kabel	15	3	**45**	
Krabice 19" 2U s 16xCAN25	1250	1	**1250**	
Konektor D-SUB-25 femasle	30	20	**600**	canon 25 pro plochý kabel
Drobné součástky	800	1	**800**	R,C, stabilizátory napětí,…
Celkem			**8455**	

Tabulka 6: Rozpočet návrhu řešení

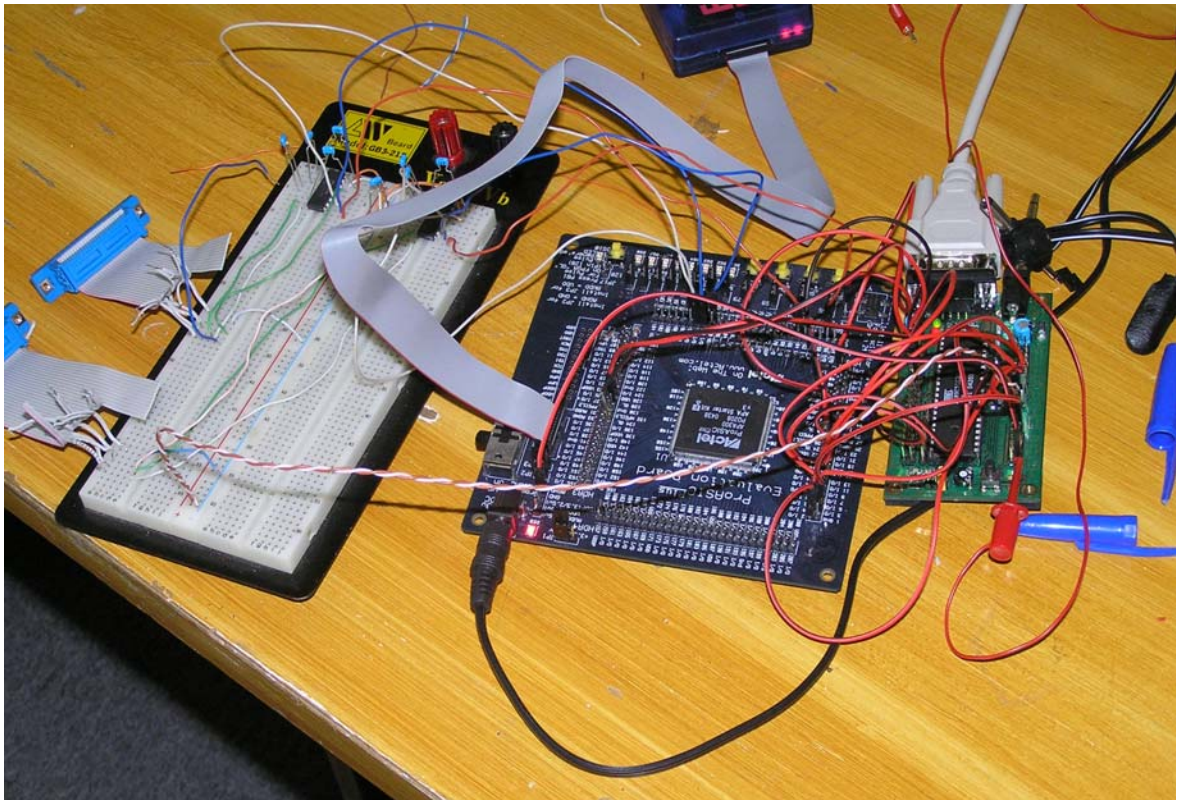
Všechny použité součástky byly zakoupeny ve firmě GM electronic, s.r.o.. Pouze FPGA ProAsic A3P060 byl věnován od firmy Phobos, s.r.o., která tento obvod poskytla za účelem popsání a odzkoušení v této diplomové práci. Dále výroba desky plošných spojů a její osazení bylo provedeno firmou Marpos, s.r.o.. Podrobný seznam použitých součástek a jejich cen je v příloze 2.

5 Oživení a otestování funkčnosti ASSSK

Před samotným vyrobením desky plošných spojů bylo zapotřebí provést ověření funkčnosti návrhu.

5.1 Ověření návrhu

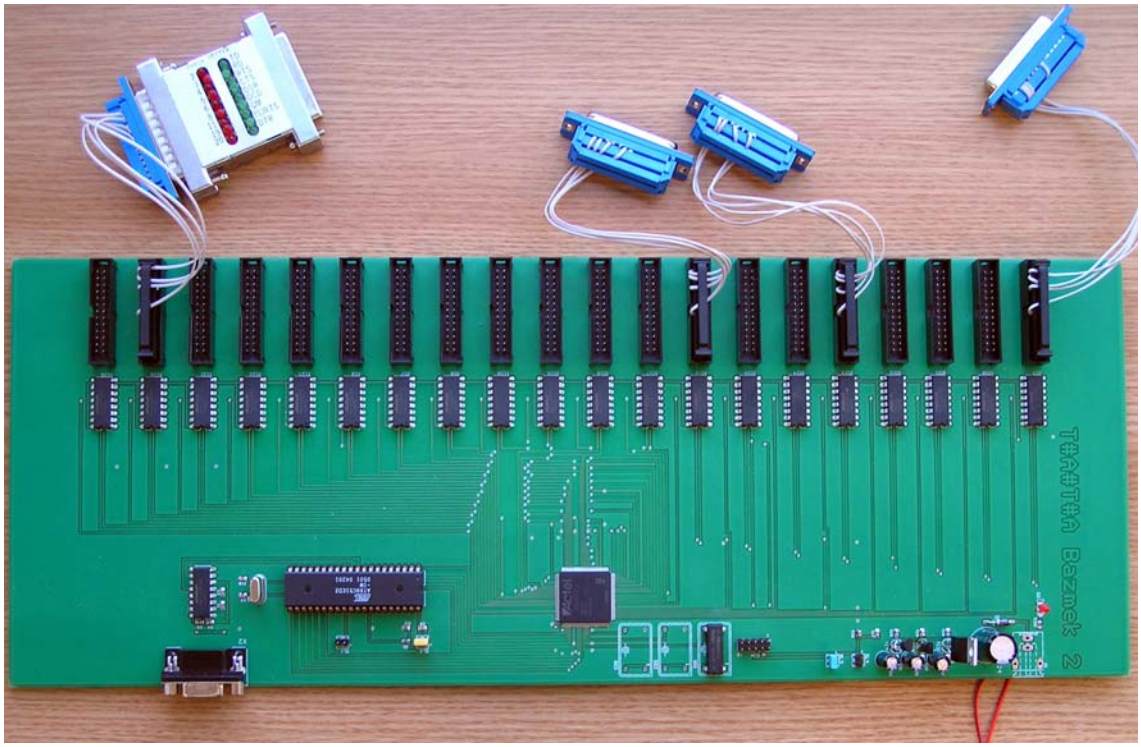
Návrh zařízení byl otestován pomocí sériově vyráběných vývojových kitů **ProASIC^{PLUS} Starter Kit** a **RD2 Kit**. ProAsic Starter Kit obsahuje FPGA obvod APA300, ten je velmi podobný obvodu A3P060, který byl vybrán pro spínání. Jediným rozdílem je, že toleruje vstupní 5V TTL logiku, proto bude simulovat funkčnost návrhu zcela stejně jako vybraný obvod. RD2 Kit obsahuje procesor stejný jako je v návrhu řešení, a proto nám také zcela vyhovuje. Zapojení obvodu MAX232 bylo provedeno na nepájivém kontaktním poli. Ukázka zapojení je na následujícím obrázku Obr. 23.



Obr. 23: Ukázka propojení vývojových kitů (zprava: RD2 Kit, ProASIC^{PLUS} Starter Ki, nepájivé kontaktní pole se zapojenými obvody MAX232a konektory RS232)

Po naprogramování a odzkoušení jednotlivých obvodů testovací zapojení ihned fungovalo. Potom byl proveden test přímo na propojení routerů. Výsledky byly uspokojivé a nejvyšší synchronizační frekvence, při které routery spolu ještě komunikovaly, byla 300 kHz.

Po vyrobení a osazení DPS byla provedena optická kontrola správnosti zapojení a přeměření výstupního napětí ze stabilizátorů, aby nedošlo k poškození obvodů. Poté následovala zkouška funkčnosti jednotlivých obvodů. Ukázka desky plošných spojů je na Obr. 24



Obr. 24:Deska plošných spojů

Chyby vyskytující se na DPS

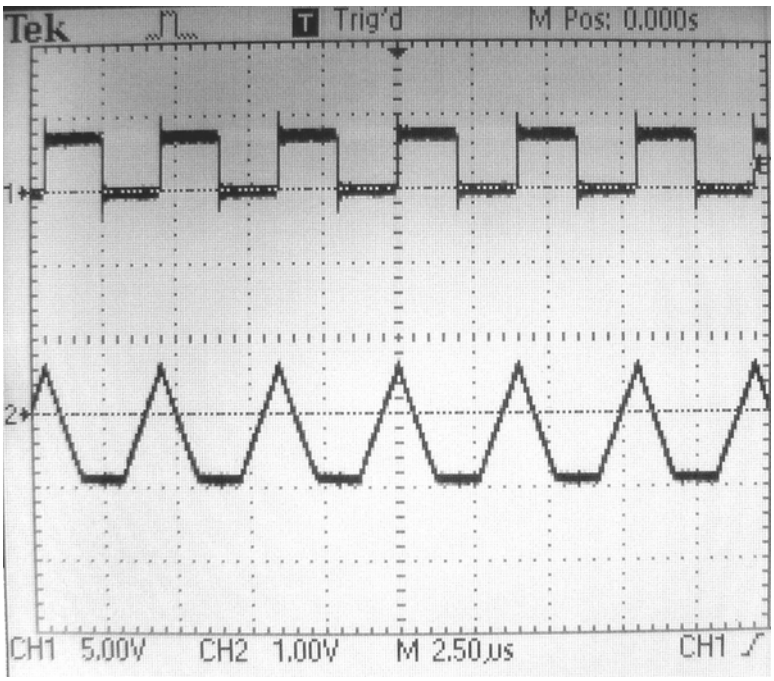
- Mikroprocesor fungoval, dokud nebyl nahrán program do FPGA. Tuto chybu způsobilo stejné pojmenování pinů pro komunikaci PC s mikroprocesorem a výstupů z FPGA v programu Eagle, který tyto piny automaticky propojil. Chyba byla odstraněna přerušením těchto dvou cest na DPS.
- Po odstranění této chyby se testovala komunikace procesoru s obvodem FPGA. Ta neprobíhala správně (FPGA nereagovalo na pokyny mikroprocesoru). Testováním se zjistilo, že v syntéze programu ve vývojovém prostředí Libero musí

být na těchto pinech obstarávajících komunikaci nastaven pull-down rezistor (rezistor zajišťující implicitní stav - logickou nulu)

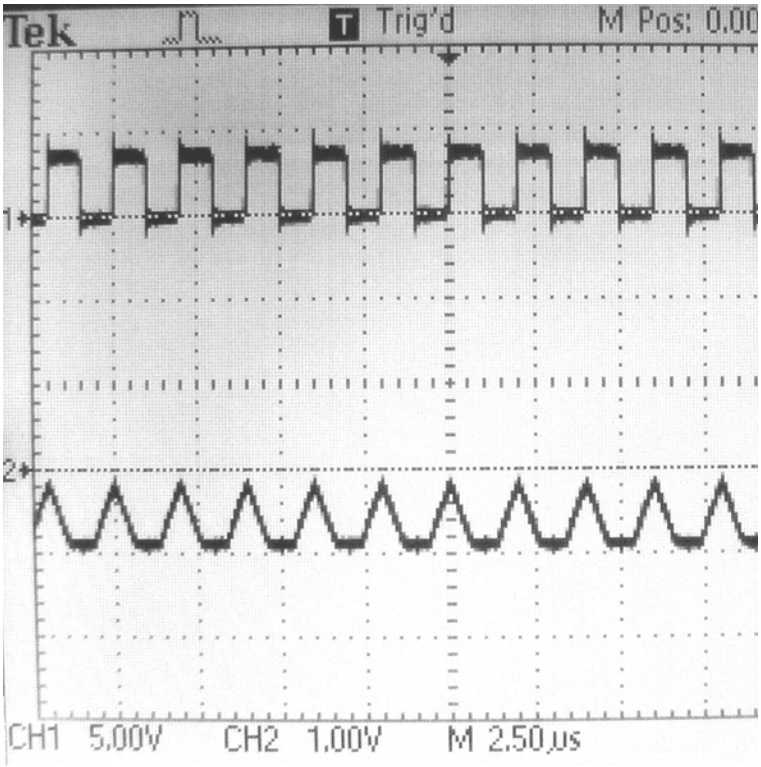
– Další chyba byla, že při kresbě schématu a kresbě plošného spoje se použily dvě rozdílné knihovny obsahující pouzdro s konektoren pro připojení rozhraní RS232. Tím se změnilo zapojení pinů. Chyba byla vyřešena tak, že konektory nejsou propojeny plochým 25-ti pinovým kabelem, ale jen dráty spojující příslušné piny.

5.2 Testování funkčnosti na routerech a jeho výsledky

Po odstranění výše uvedených chyb byl proveden test přímo na propojení routerů. Výsledky byly uspokojivé a nejvyšší přenosová rychlost, při které routery spolu ještě komunikovaly, byla 250 kb/s. Při vyšších frekvencích obvody MAX232 začaly zkreslovat převáděný signál. Převod na logickou nulu je již při datovém toku 250 kb/s hodně zkreslen a má tvar trojúhelníku (Obr. 25). Při datovém toku 500 kb/s již MAX232 nepřevádí logické úrovně (Obr. 26).



Obr. 25:Snímek z osciloskopu při převodu z TTL na RS232 logiku obvodem MAX232 (datový tok 250 kb/s)



Obr. 26: Snímek z osciloskopu při převodu z TTL na RS232 logiku obvodem MAX232 (datový tok 500 000 bps)

Dále byl změřen rozdíl v odezvě routerů při průchodu přes ASSSK a bez něj. Výsledky uvádím v Tabulka 7.

Při průchodu přes ASSSK 2			Při přímém propojení routerů (bez ASSSK 2)		
ASSSK frekvence [bps]	ping [ms]	ztrátovost [%]	Router [bps]	Ping [ms]	ztrátovost [%]
250 000	9,1		250 000	8,5	
62 500	28,6	0	64 000	28	0
32 250	50,7	0	38 400	41	0
16 125	98,5	0			
8 063	193	0			
4 031	387	0			
2 016	782				

Tabulka 7: Rychlost odezvy při různých přenosových rychlostech

Z této tabulky vyplývá, že při průchodu paketů přes ASSSK 2 doba odpovědi (ping) zůstává skoro stejná. Maximální dosažená přenosová rychlost byla 250 000 bps.

Závěr

Cílem práce bylo přepracovat původní dálkově řízené spojovací pole WAN a LAN portů, které vyrobil v rámci své Diplomové práce Ing. David Seidl [6]. Nové spojovací pole umožňuje spínání více portů a nezávislé stanovení hodinové frekvence na spojených portech navýšení přenosové rychlosti.

V úvodu práce byl popsán princip fungování virtuální laboratoře a využití předchozího ASSSK 1 (automatický systém spojování síťových konfigurací).

Další kapitola se zabývá popisem spínaných rozhraní a vysvětluje proč byl z ASSSK 1 vyřazen modul pro spínání portů Ethernetu.

Třetí kapitola obsahuje možnosti z výběru obvodů, které jsou schopny provádět naše spínání. Dále byl do této kapitoly zahrnut popis mikroprocesoru, který provádí komunikaci s terminálovým prostředím (PC). Součástí byl také uveden popis obvodů, které byly vybrány pro převod logiky rozhraní RS232 na TTL logiku a stabilizaci napětí.

Následující kapitola byla v této práci stěžejní a vysvětlila jakým principem bylo prováděno samotné spínání. Zabývala se jak hardwarovým propojením, tak i softwarovým řešením. Byly v ní uvedeny ukázky ze zdrojových textů v jazyce VHDL a také popis funkcí používaných v mikroprocesoru. Dále kapitola uvádí stručný cenový přehled hlavních součástí ASSSK 2.

V závěru práce byly uvedeny problémy při uvádění desky plošných spojů do provozu. Je zde také uvedeno na jakých vývojových deskách byl návrh odzkoušen. Byl tam také popsán způsob testování funkčnosti a prezentovány dosažené výsledky.

Další možností přepracování ASSSK 2 je použití jiných obvodů pro převod logiky rozhraní RS232 do TTL logiky za účelem zvýšení přenosových rychlostí. Dále se uvažuje o vypuštění mikroprocesoru, jehož činnost by kompletně převzal obvod FPGA. Programovatelné hradlové pole má kapacitu na to, aby bylo schopno zprostředkovat komunikaci s terminálem a také realizovat samotné spínání.

Diplomová práce splnila všechny požadavky, které na ni byly kladeny v úvodu. Automatický systém spojování síťových konfigurací se bude využívat stejně jako jeho předchůdce ve virtuální laboratoři síťových technologií realizované na katedře informatiky VŠB-TU Ostrava v rámci Regional Cisco Academy. V této laboratoři by se zařízení mělo začít využívat v září 2007.

Literatura

[1] ECKEL, B. 2000. *Myslíme v C++ knihovna programátora.* Praha: Granada Publishing, spol.s.r.o, 2000, 556s..ISBN 80-247-9009-2

[2] GRYGÁREK, P.: *Zkušenosti z nasazení virtuální laboratoře počítačových sítí a další směry jejího rozvoje*, Sborník semináře Technologie pro e-vzdělávání, FEL ČVUT Praha, katedra počítačů, 2006 11 s. ISBN 80-01-03512-3

[3] KAINKA, B. 1998. *Využití rozhraní PC.* Praha: HELL, 2000, 133s. ISBN 80-86167-13-5

[4] MANN, B. 2003. *C pro mikrokontroléry*. Praha : BEN – technická literatura, 2003, 280 s. ISBN80-7300-077-6

[5] NĚMEC, *P. Virtuální síťová laboratoř.* Diplomová práce, FEI VŠB-TU Ostrava, květen 2005.

[6] SEIDL, D. *Automatizovaný systém pro správu síťových konfigurací.* Diplomová práce, FMMI VŠB-TU Ostrava, květen 2005.

[7] *Katalogový list obvodu ProASIC®3 Flash Family FPGAsMT8816*. [online], Květen 2007. Dostupný na WWW: http://www.actel.com/documents/PA3E_DS.pdf

[8] *Katalogový list mikroprocesoru ATMEL 89c51ED2.* [online], Květen 2007. Dostpný na WWW: http://www.atmel.com/dyn/resources/prod_documents/doc4235.pdf

[9] *Katalogový list obvodu MAX232.* [online], Květen 2007, Dostupný na WWW: http://www.gme.cz/linkydok/ba061a6fae3c24e059d276d37bb1c598.pdf

[10] *Katalogový list obvodu LM317.* [online], Květen 2007, Dostupný na WWW: http://www.gme.cz/linkydok/73cced39270b94add15a748326590995.pdf

[11] PROCHÁZKA, P. *Popis jazyka VHDL.* [on-line], květen 2007, Dostupný na WWW: http://prochazka.d2.cz/vhdl.php

[12] PECH, J. *Programovatelné logické obvody* [online], květen 2007, Dostupný na WWW: http://sweb.cz/fpga/

Seznam Příloh:

Obsah přiloženého CD:

Text diplomové práce

Schéma zapojení a schéma desky plošných spojů

Zdrojový kód pro mikroprocesor

Zdrojový kód pro FPGA

Fotodokumentace

Příloha 1:

Přehled jednotlivých pinů připojených k FPGA

	Název portu	Číslo pinu	Jméno banky	I/O standard	Výstupní proud (mA)	Rezistor pull
1	Wr_C	1	Bank1	LVTTL	--	Down
2	Wr_F	2	Bank1	LVTTL	--	Down
3	Adr_Bunky[0]	3	Bank1	LVTTL	--	Down
4	Adr_Bunky[1]	4	Bank1	LVTTL	--	Down
5	Adr_Bunky[2]	5	Bank1	LVTTL	--	Down
6	Adr_Bunky[3]	6	Bank1	LVTTL	--	Down
7	Adr_Bunky[4]	7	Bank1	LVTTL	--	Down
8	Adr_F[0]	8	Bank1	LVTTL	--	Down
9	Adr_F[1]	12	Bank1	LVTTL	--	Down
10	Adr_F[2]	13	Bank1	LVTTL	--	Down
11	Adr_F[3]	14	Bank1	LVTTL	--	Down
12	Adr_F[4]	15	Bank1	LVTTL	--	Down
13	xtal0_in	18	Bank1	LVTTL	--	None
14	txd_in[0]	38	Bank1	LVTTL	--	None
15	txd_in[1]	39	Bank1	LVTTL	--	None
16	txd_in[2]	40	Bank1	LVTTL	--	None
17	txd_in[3]	41	Bank1	LVTTL	--	None
18	txd_in[4]	42	Bank1	LVTTL	--	None
19	txd_in[5]	43	Bank1	LVTTL	--	None
20	txd_in[6]	44	Bank1	LVTTL	--	None
21	txd_in[7]	49	Bank1	LVTTL	--	None
22	txd_in[8]	51	Bank1	LVTTL	--	None
23	txd_in[9]	53	Bank1	LVTTL	--	None
24	txd_in[10]	55	Bank1	LVTTL	--	None
25	txd_in[11]	58	Bank1	LVTTL	--	None
26	txd_in[12]	59	Bank1	LVTTL	--	None
27	txd_in[13]	60	Bank1	LVTTL	--	None
28	txd_in[14]	61	Bank1	LVTTL	--	None
29	txd_in[15]	65	Bank1	LVTTL	--	None
30	txd_in[16]	66	Bank1	LVTTL	--	None
31	txd_in[17]	67	Bank1	LVTTL	--	None
32	txd_in[18]	78	Bank0	LVTTL	--	None
33	txd_in[19]	79	Bank0	LVTTL	--	None
34	rxd_out[0]	80	Bank0	LVTTL	12	None
35	rxd_out[1]	83	Bank0	LVTTL	12	None
36	rxd_out[2]	84	Bank0	LVTTL	12	None
37	rxd_out[3]	85	Bank0	LVTTL	12	None
38	rxd_out[4]	86	Bank0	LVTTL	12	None
39	rxd_out[5]	87	Bank0	LVTTL	12	None
40	rxd_out[6]	88	Bank0	LVTTL	12	None
41	rxd_out[7]	89	Bank0	LVTTL	12	None
42	rxd_out[8]	90	Bank0	LVTTL	12	None

	Název portu	Číslo pinu	Jméno banky	I/O standard	Výstupní proud (mA)	Rezistor pull
43	rxd_out[9]	91	Bank0	LVTTL	12	None
44	rxd_out[10]	92	Bank0	LVTTL	12	None
45	rxd_out[11]	93	Bank0	LVTTL	12	None
46	rxd_out[12]	94	Bank0	LVTTL	12	None
47	rxd_out[13]	101	Bank0	LVTTL	12	None
48	rxd_out[14]	102	Bank0	LVTTL	12	None
49	rxd_out[15]	103	Bank0	LVTTL	12	None
50	rxd_out[16]	104	Bank0	LVTTL	12	None
51	rxd_out[17]	105	Bank0	LVTTL	12	None
52	rxd_out[18]	106	Bank0	LVTTL	12	None
53	rxd_out[19]	111	Bank0	LVTTL	12	None
54	Clk_Out[0]	112	Bank0	LVTTL	12	None
55	Clk_Out[1]	113	Bank0	LVTTL	12	None
56	Clk_Out[2]	114	Bank0	LVTTL	12	None
57	Clk_Out[3]	115	Bank0	LVTTL	12	None
58	Clk_Out[4]	116	Bank0	LVTTL	12	None
59	Clk_Out[5]	120	Bank0	LVTTL	12	None
60	Clk_Out[6]	121	Bank0	LVTTL	12	None
61	Clk_Out[7]	122	Bank0	LVTTL	12	None
62	Clk_Out[8]	123	Bank0	LVTTL	12	None
63	Clk_Out[9]	124	Bank0	LVTTL	12	None
64	Clk_Out[10]	125	Bank0	LVTTL	12	None
65	Clk_Out[11]	126	Bank0	LVTTL	12	None
66	Clk_Out[12]	127	Bank0	LVTTL	12	None
67	Clk_Out[13]	129	Bank0	LVTTL	12	None
68	Clk_Out[14]	130	Bank0	LVTTL	12	None
69	Clk_Out[15]	131	Bank0	LVTTL	12	None
70	Clk_Out[16]	132	Bank0	LVTTL	12	None
71	Clk_Out[17]	133	Bank0	LVTTL	12	None
72	Clk_Out[18]	137	Bank0	LVTTL	12	None
73	Clk_Out[19]	138	Bank0	LVTTL	12	None

Příloha 2:

Seznam použitých součástek

název	hodnota	package	library	počet	cena	cena celkem	koupeno
Kondenzátory							
C1	1u	C0805K	rcl	84	2,547	213,948	GME
C28	0,33UF	C0805K	rcl	1	3	3	GME
C25	100M	E2,5-7	rcl	3	1	3	GME
C5	100n	C0805K	rcl	8	1,547	12,376	GME
C24	1G/25V	E5-13	rcl	1	5	5	GME
C9	22p	C0805K	rcl	2	1,5	3	GME
Odpory							
R2	2k2	R0805	rcl	22	1	22	GME
R4	3k9	R1206	rcl	20	1	20	GME
R1	470R	R0805	rcl	1	2	2	GME
Max232							
IC5	MAX232	DIL16	maxim	21	19,004	399,084	GME
Porty a jumpery							
PORT0		MA13-2	con-lstb	20	4,96	99,2	GME
SL1	Loader	02P	con-amp-	1	1	1	GME
SL2	Reset	02P	con-amp-	1	1	1	GME
SL3	Vjtag (Co)	02P	con-amp-	1	1	1	GME
SL4	Vpump (NoCo	) 02P	con-amp-	1	1	1	GME
SV4	Flash Pro	MA05-2	con-lstb	1	4	4	GME
Napěťové regulátory							
IC4	1,5 V	78LXX	v-reg	1	7,5	7,5	GME
IC3	3,3 V	78LXX	v-reg	1	22	22	GME
IC2	5 v	78LXX	v-reg	1			GME
Konektory							
X1	napájecí kon.	737992	-5 con-conr	1	5	5	GME
X2	CAN 9		con-subd	1	30	30	GME
CAN25	samořezný konektor pro ploch. kab.			20	30	600	
Samořezný konektor do PCB, 20pinů				20	15	300	
Diody							
D1	1N4148	DO35-1	0 diode	1	1	1	GME
D2	HLMP6	HLMP6	display-hp	1	3	3	GME
Krystaly, oscilátory							
QG1	4,096MHz	DIL14S	crystal	1	88	88	GME
Q1	18,432MHz	HC49U7	0 crystal	1	9,9	9,9	GME
QG2	nepájet	DIL14S	crystal	1	0	0	GME
QG3	nepájet	DIL14S	crystal	1	0	0	GME

FPGA							
ProAsic3	A3P060	TQFP144	4 pes	1	zdarma	0	Phobos
Mikroprocesor							
Patice		DIL40		1	22,9	22,9	GME
IC1	AT89C52P	DIL40	atmel	1	196	196	GME
Krabice							
U-RE4622	16xCAN25 otvor			1	246	246	GME
U-RE4032	2U 19"Rack			1	1007	1007	GME
Deska plošných spojů a její osazení							
DPS	400 x 200 mm			1	4200	4200	
Zdroj							
Stejnosměrný	9V			1	300	300	
CELKEM				**245**		**7828,9**	