

Technická zpráva



Akademie věd České republiky
Ústav teorie informace a automatizace AV ČR, v.v.i.

Demonstrátor Reed Solomonova kodéru a dekodéru s ethernetovým rozhraním implementovaný v FPGA

Ing. Antonín Heřmánek, PhD., Ing. Josef Dušek, Ing. Jan Kloub

hermanek@utia.cas.cz, +420-2-6605 2470

Obsah

1 Úvod	1
2 Popis	1
3 Struktura kodéru	2
3.1 Popis rozhraní kodéru a časování	3
4 Struktura dekodéru	5
4.1 Popis parametrů dekodéru, časování	5
5 Komunikační rozhraní	7
5.1 Popis ethernetového rozhraní	7
5.2 Komunikace s integrovaným obvodem	7
5.2.1 Cyklus sběrnice	8
5.2.2 Konfigurace a základní nastavení obvodu	8
5.3 Práce s rámcí	12
5.3.1 Přístup k datové struktuře rámce	13
5.3.2 Odesílání a přijímání rámců	14
6 Řízení komunikačního rozhraní	14
6.1 Mapování portů PicoBlaze na adresní prostor ethernetového čipu	14
6.2 Stavový automat	16
6.2.1 Čtení a zápis hodnot registrů ethernetového obvodu	16
6.2.2 Přerušení a jeho obsluha	16
6.2.3 Přenos rámců mezi ethernetovým obvodem a kodérem/dekodérem	18
6.2.4 Přepínání pamětí programu	19
6.3 Program procesoru	19
7 Komunikační protokol	20

8	Ověření funkce	20
8.1	Ověření funkčnosti dekodéru	22
8.2	Ověření funkčnosti kodéru	23
9	Výpis CDROM	25

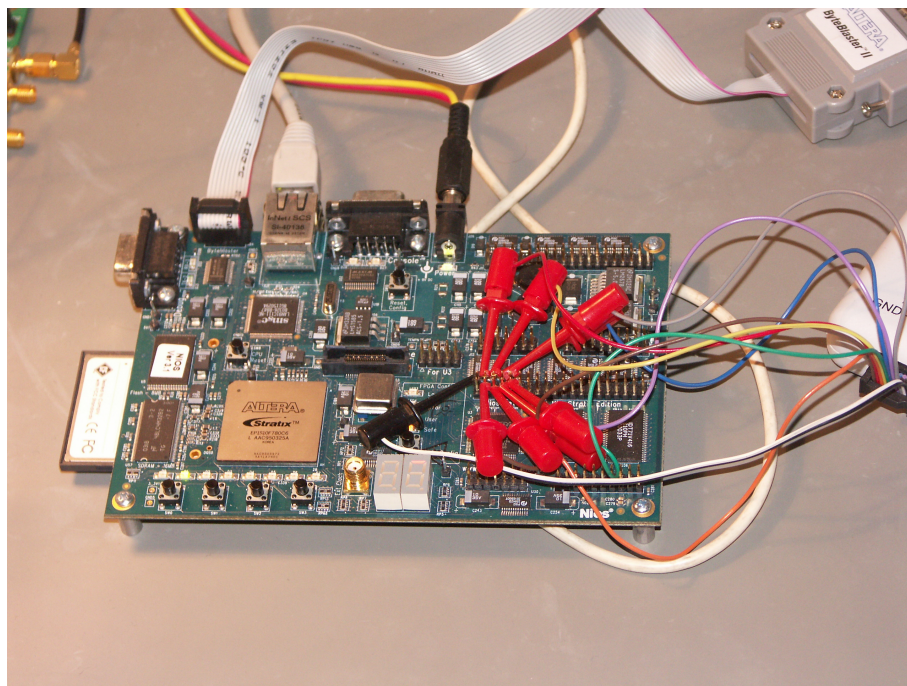
Revize

Revize	Datum	Autor	Popis změn v dokumentu
0	23.4.2007	Kloub	Vytvoření dokumentu
1			
2			
3			
4			

1 Úvod

Tato práce demonstuje funkci hardwarové implementace Reed Solomonova (RS) kodéru a dekodéru. Dekodér realizuje specifický kód RS(255,239,8). Vstupní a výstupní data kodéru a dekodéru jsou přenášena přes ethernetové rozhraní. Ethernetové rozhraní je implementováno pomocí obvodu SMSC 91C111, který je řízen a konfigurován procesorem PicoBlaze. Pro přenos dat je použit protokol UDP.

RS kodér, dekodér a řízení ethernetového rozhraní bylo implementováno v jazyce Handel-C na obvodch **Altera Cyclone EP1C20F400C7, Altera Stratix EP1S10F780C6 a Altera Stratix II EP2S180F1020C3**. Implementace byla odzkoušena na vývojových deskách Nios Development Board, Cyclone Edition; Nios Development Board, Stratix Edition a Stratix II EP2S180 DSP Development Board.



Obrázek 1: Vývojová deska Nios Development Board, Stratix Edition.

Tato práce je součástí výzkumného projektu Grantové agentury Akademie věd ČR číslo 1ET100750408 "Digitální video-senzorický systém záchraného robotu"

2 Popis

V této práci je popsána implementace systematického RS kódu nad tělesem GF(256) schopného opravy 8 chyb. Parametry jsou zvoleny s ohledem na normu IEEE 802.16 a to ($n = 255$, $k = 239$, $t = 8$), tedy délka kódového slova $n = 255$, počet informačních znaků $k = 239$, 16 znaků nesoucích zabezpečovací informaci. Galoisovo těleso je vytvořeno pomocí primitivního mnohočlenu

$$f(x) = x^8 + x^4 + x^3 + x^2 + 1 \quad (1)$$

Je zvolen generující mnohočlen ve tvaru

$$g(x) = (x + \alpha^0)(x + \alpha^1)(x + \alpha^2) \dots (x + \alpha^{2t-1}), \alpha = 2 \quad (2)$$

Dále je popsáno ethernetové rozhraní pro přenos dat. Pro příjem a vysílání rámců je použit obvod SMSC 91C111. Obvod je řízen a konfigurován pomocí procesoru PicoBlaze. Předávání dat ethernetových rámců je zajišťováno stavovým automatem, který zajišťuje vyšší propustnost přenosu dat. Data z příchozí fronty ethernetového obvodu jsou předána do paměti pro vstupní data kodéru či dekodéru. Výstupní data z kodéru či dekodéru jsou uložena do paměti předpřipraveného ethernetového rámce. Sestavený rámec je předán do odchozí fronty ethernetového obvodu a procesor zajistí jeho odeslání.

3 Struktura kodéru

Kodér pro systematický RS kód může být realizován pomocí lineárního zpětnovazebního registru LFSR (obr. 2), který provádí dělení generujícím mnohočlenem podle vztahu (3).

$$x^{n-k}m(x) = q(x)g(x) + p(x) \quad (3)$$

Na vstup registru přichází datové symboly určené k zakódování následované počtem $2t$ nulových symbolů (což odpovídá výrazu $x^{2t}m(x)$). Na vstup přichází koeficienty v pořadí m_k, \dots, m_1, m_0 . Všechny datové cesty na obrázku jsou osmibitové. Před zahájením dělení je potřeba vynulovat všechny registry. Po načtení všech n symbolů (tj. po n hodinových cyklech) budou registry obsahovat hodnoty odpovídající zbytku po dělení generujícím mnohočlenem.

K operaci zakódování jsou potřeba dvě operace nad konečným tělesem, a to sčítání a násobení. Sečtení dvou čísel v $GF(256)$ je provedeno pomocí bitové operace xor. Násobení dvou čísel je provedeno následovně. Čísla z $GF(256)$ lze vyjádřit pomocí mnohočlenu sedmého stupně s koeficienty z $GF(2)$:

$$A(x) = a_7x^7 + a_6x^6 + a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0; a_i \in GF(2), A \in GF(256)$$

Operace $C = A \cdot B$; $A, B, C \in GF(256)$, lze vyjádřit v polynomiální reprezentaci jako

$$C(x) = A(x) \cdot B(x) = A(x) \times B(x) \bmod f(x) = D(x) \bmod f(x),$$

kde polynomi $A(x), B(x), C(x), D(x)$ jsou definovány jako:

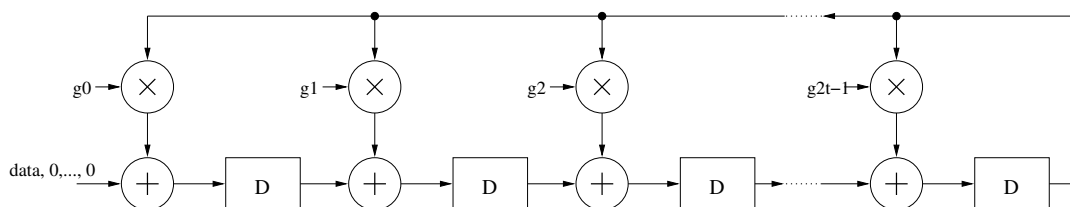
$$A(x) = a_7x^7 + a_6x^6 + \dots + a_0, \quad (4)$$

$$B(x) = b_7x^7 + b_6x^6 + \dots + b_0, \quad (5)$$

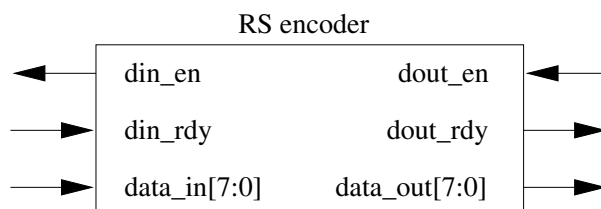
$$C(x) = c_7x^7 + c_6x^6 + \dots + c_0, \quad (6)$$

$$D(x) = d_{14}x^{14} + d_{13}x^{13} + \dots + d_0 \quad (7)$$

kde $a_i, b_i, c_i, d_j \in GF(2)$, mod je operace výpočet zbytku po dělení polynomem a \times je operace násobení polynomů. Uvedené vztahy pro násobení dvou čísel platí pro těleso $GF(256)$ s primitivním polynomem $f(x) = x^8 + x^4 + x^3 + x^2 + 1$.



Obrázek 2: RS kodér používající dělení generujícím polynomem.



Obrázek 3: Rozhraní kodéru RS(255,239,8).

$$\begin{aligned}
 d_{14} &= a_7 b_7 \\
 d_{13} &= a_6 b_7 + a_7 b_6 \\
 d_{12} &= a_6 b_6 + a_7 b_5 + a_5 b_7 \\
 d_{11} &= a_6 b_5 + a_5 b_6 + a_7 b_4 + a_4 b_7 \\
 d_{10} &= a_7 b_3 + a_6 b_4 + a_3 b_7 + a_4 b_6 + a_5 b_5 \\
 d_9 &= a_3 b_6 + a_6 b_3 + a_2 b_7 + a_7 b_2 + a_4 b_5 + a_5 b_4 \\
 d_8 &= a_3 b_5 + a_7 b_1 + a_1 b_7 + a_2 b_6 + a_5 b_3 + a_6 b_2 + a_4 b_4 \\
 d_7 &= a_4 b_3 + a_1 b_6 + a_0 b_7 + a_7 b_0 + a_2 b_5 + a_5 b_2 + a_3 b_4 + a_6 b_1 \\
 d_6 &= a_6 b_0 + a_4 b_2 + a_2 b_4 + a_1 b_5 + a_0 b_6 + a_3 b_3 + a_5 b_1 \\
 d_5 &= a_4 b_1 + a_1 b_4 + a_5 b_0 + a_3 b_2 + a_2 b_3 + a_0 b_5 \\
 d_4 &= a_1 b_3 + a_2 b_2 + a_0 b_4 + a_4 b_0 + a_3 b_1 \\
 d_3 &= a_1 b_2 + a_2 b_1 + a_3 b_0 + a_0 b_3 \\
 d_2 &= a_0 b_2 + a_1 b_1 + a_2 b_0 \\
 d_1 &= a_0 b_1 + a_1 b_0 \\
 d_0 &= a_0 b_0
 \end{aligned} \tag{8}$$

V druhém kroku se provede určení zbytku po dělení primitivním mnohočlenem, pomocí kterého získáme mnohočlen sedmého stupně. Pomocí operace "nalezení zbytku po dělení" se provede zobrazení patnácti binárních koeficientů d_{14}, \dots, d_0 na osm (c_7, \dots, c_0).

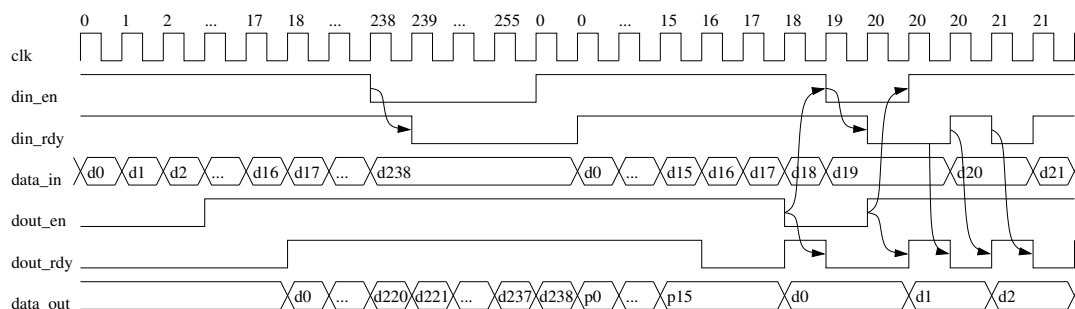
$$\begin{aligned}
 c_7 &= d_7 + d_{11} + d_{12} + d_{13} \\
 c_6 &= d_6 + d_{10} + d_{11} + d_{12} \\
 c_5 &= d_5 + d_9 + d_{10} + d_{11} \\
 c_4 &= d_4 + d_8 + d_9 + d_{10} + d_{14} \\
 c_3 &= d_3 + d_8 + d_9 + d_{11} + d_{12} \\
 c_2 &= d_2 + d_8 + d_{10} + d_{12} + d_{13} \\
 c_1 &= d_1 + d_9 + d_{13} + d_{14} \\
 c_0 &= d_0 + d_8 + d_{12} + d_{13} + d_{14}
 \end{aligned} \tag{9}$$

Násobení dvou čísel vyžaduje 77 hradel XOR, 64 hradel AND. V případě kodéru se provádí pouze násobení konstantou (tj. koeficienty generujícího mnohočlenu). Násobení konstantou vyžaduje méně hardwarových prostředků než obecná násobička. Například násobení číslem 59 vyžaduje pouze 28 hradel XOR.

3.1 Popis rozhraní kodéru a časování

Rozhraní IP kóru RS kodéru je symbolicky zobrazeno na obr. 3. Stručný popis jednotlivých portů je uveden v tabulce 1

RS kódér je vybaven osmibitovým vstupním a výstupním datovým portem a signály pro "hand-shaking" na straně vstupu a výstupu.



Obrázek 4: Časování RS kodéru

Popis chování a význam jednotlivých parametrů makro procedury

Chování kodéru je zachyceno na časových průbězích na obrázku 4, kde probíhá načítání vstupních symbolů (d0–d238) a současně vyčítání výstupních kódových symbolů (d0–d238, p0–p15). Na počátku je v proměnné `din_en` nastavená hodnota 1, kodér je schopen přijímat vstupní data. Po dobu, kdy `din_en` má hodnotu 1, kodér sleduje v každém cyklu `din_rdy` a za podmínky, že má hodnotu 1, načte vstupní symbol. Nejsou-li na vstupu `data_in` v dalším cyklu k dispozici nová data musí být vstup `din_rdy` nastaven na hodnotu 0. Platnost dat (jednoho vstupního symbolu) na `data_in` je omezena vždy jen na jeden cyklus. V cyklu následujícím po načtení předposledního vstupního symbolu kódového slova je přiřazena na výstup `din_en` hodnota 0 (na obrázku cyklus s číslem 238).

Na změnu stavu `din_en` musí také v následujícím cyklu zareagovat `data_in` nastavením na hodnotu 0 (což znamená zastavení dodávky nových symbolů). Poté, co kodér přijal prvních 17 symbolů, a za předpokladu, že `dout_en` má hodnotu 1, se nastaví `dout_rdy` na hodnotu 1 a na `dout` se začnou předávat načtené vstupní symboly. Zabezpečovací symboly se začnou předávat na výstup ihned po všech datových symbolech (na obrázku počínaje cyklem 0 uprostřed).

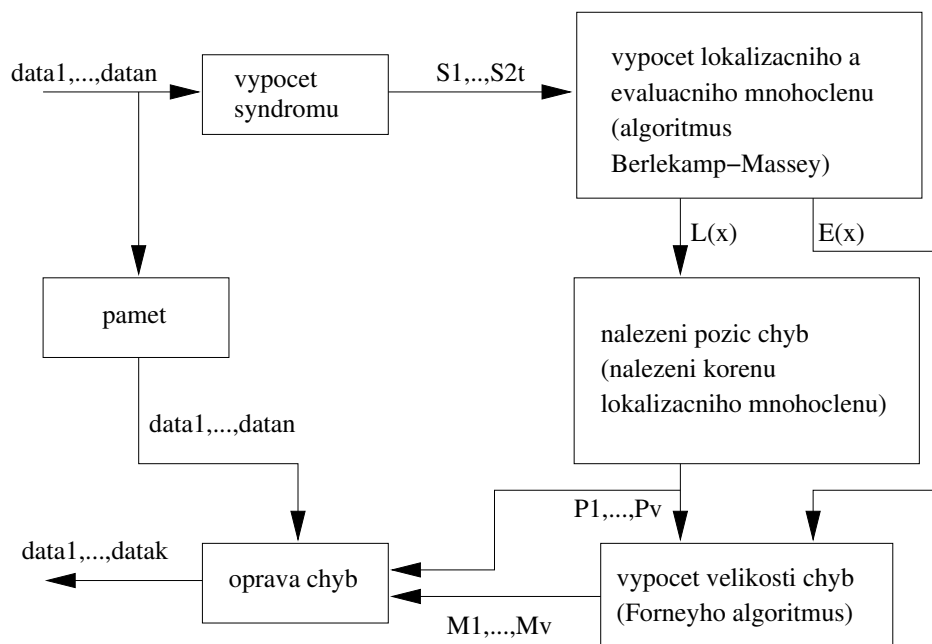
V případě, že kodér chce vysílat výstupní symboly a `dout_ready` má hodnotu 0, změní se stav `din_avail` na 0 a v následujícím cyklu přestane načítat vstupní data. Hodnota `din_avail` je změněna až poté co `dout_en` nabude hodnoty 1.

Shrnutí: `din_rdy` smí mít hodnotu 1 v cyklu následujícím po překlopení `din_en` do log. 1 (cyklus 0 uprostřed na obr. 4) a v dalších až do doby, kdy `din_en` nabude hodnoty 0 (cyklus 238 na obr. 4). Stejná pravidla platí pro `dout_rdy`, `dout_en`.

Každý hodinový cyklus je v horní části obrázku 4 označen číslem z rozsahu 0–255. Tímto číslem je vyjádřen průběh kódování (počet již zpracovaných vstupních symbolů a dodatečných nulových symbolů – viz obr. 2 a jeho popis). Pokud kodér nemůže z nějakého důvodu zpracovat nový vstupní symbol (když např. nejsou k dispozici žádné vstupní symboly nebo nelze předávat na výstup zakódované symboly), jeho vnitřní stav se nezmění. Tato situace nastává např. v cyklech označených čísly 20, 21. Kodér v té době přestane přijímat data buď z důvodu pozastavení výstupu symbolů (cykly s čísly 20), nebo kvůli neplatnosti vstupních dat (cykly s čísly 21). Během cyklů označených číslem 0 (uprostřed) proběhne

název	směr	šířka	popis
<code>din_en</code>	výstup	1	Indikace, že kodér je schopen přijímat nová data
<code>din_rdy</code>	vstup	1	Indikace, že data na vstupu <code>data_in</code> jsou platná
<code>data_in</code>	vstup	8	Vstupní data k zakódování
<code>dout_en</code>	vstup	1	Indikace, že je možné posílat zakódovaná data na výstup
<code>dout_rdy</code>	výstup	1	Indikace, že data na výstupu <code>data_out</code> jsou platná
<code>data_out</code>	výstup	8	Zakódovaná data.

Tabulka 1: Stručný popis vstupů a výstupů makro procedury RS kodéru.



Obrázek 5: Blokové schéma RS dekodéru.

nastavení počátečního stavu dekodéru a poté se začnou přijímat nové vstupní symboly.

4 Struktura dekodéru

Dekodér využívá paralelních výpočtů v jednotlivých krocích dekódování a je navržena pro kód RS(255, 239, 8).

Dekodér je založen na Berlekamp-Massey algoritmu pro určení lokalizačního a evaluačního mnohočlenu a na Forneyho algoritmu pro vyčíslení velikosti chyb. Blokové schéma dekodéru je zobrazeno na Obrázku 5.

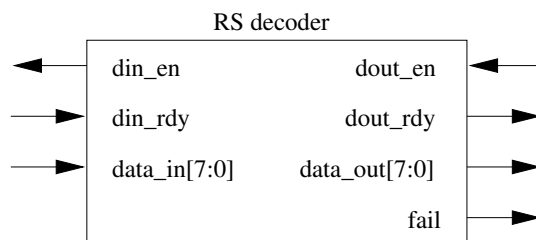
Příchozí symboly (na obrázku označeny jako $data_1, \dots, data_n$) jsou uschovávány v paměti RAM. Symboly přicházejí v pořadí, v jakém vycházejí z kodéru, tj. nejprve datové symboly následované kontrolními symboly. Z příchozích symbolů se počítají hodnoty syndromů. Z hodnot syndromů S_1, \dots, S_{16} se následně určí pomocí algoritmu Berlekamp-Massey lokalizační a evaluační mnohočlen. Po skončení algoritmu se pokračuje hledáním pozic chyb tak, že se postupným dosazováním všech prvků Galoisova tělesa hledají kořeny mnohočlenu. Po nalezení pozic chyb se pokračuje výpočtem velikostí chyb pomocí Forneyho algoritmu. Oprava dat v paměti je provedena za podmínky, že 1) došlo alespoň k jedné chybě a 2) celkový počet chyb není větší než počet opravitelných chyb daného kódu.

4.1 Popis parametrů dekodéru, časování

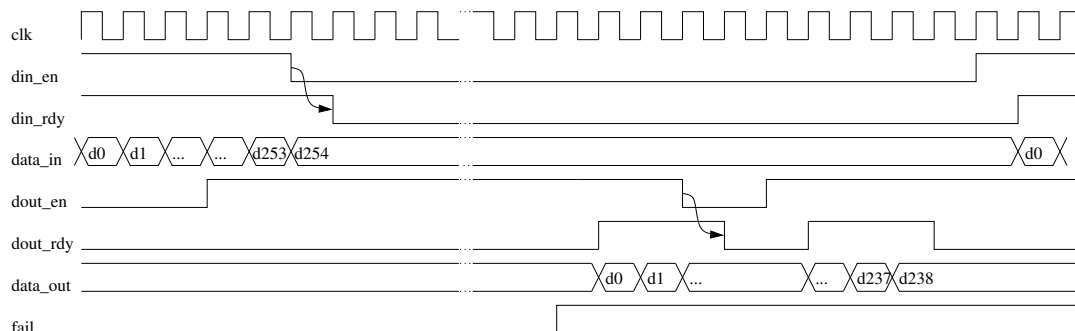
Makro procedura má definici portů uvedenou na obrázku 6 a tyto porty se shodují v sekvenční i paralelní verzi dekodéru.

V tabulce 2 jsou všechny parametry stručně popsány.

Časování kodéru je zachyceno na časových průbězích na obrázku 7, kde probíhá načítání vstupních symbolů (d0–d254) a následně výstup dekódovaných symbolů (d0–d238). Na počátku činnosti je v proměnné `din_en` nastavena hodnota 1, dekodér je schopen přijímat vstupní data. Po dobu nastavené jedničky kodér sleduje v každém cyklu `din_rdy` a za podmínky, že `din_rdy` má hodnotu 1, načte vstupní symbol. Nejsou-li na vstupu `data_in` v dalším cyklu k dispozici nová data, je `din_rdy` nastaven na hod-



Obrázek 6: Rozhraní dekodéru RS(255,239,8).



Obrázek 7: Časování RS dekodéru.

notu 0. Platnost dat (jednoho vstupního symbolu) na vstupu `data_in` je omezena vždy jen na jeden cyklus. V cyklu následujícím po načtení předposledního vstupního symbolu kódového slova je nastaven výstup `din_en` na hodnotu 0.

Na změnu na `din_en` musí také v následujícím cyklu zareagovat `data_in` nastavením na 0 (zastavením dodávky nových symbolů). V průběhu načítání vstupních dat je zahájeno i dekódování. Předtím, než dekodér začne posílat na výstup dekódovaná data, nastaví se `fail` buď na hodnotu 0, když dekodér byl schopen provést opravu, nebo na hodnotu 1 v případě, že je překročena samoopravná schopnost kódu a žádná oprava nebyla provedena (nastalo více než 8 a méně než 17 chyb). Dekódované symboly se začnou předávat na výstup `dout` a zároveň `dout_rdy` se nastaví na hodnotu 1 jen v případě, že `dout_en` má hodnotu 1, nebo v cyklu, ve kterém dojde ke změně z 1 na 0.

název	směr	šířka	popis
<code>din_en</code>	výstup	1	Indikace, že dekodér je schopen přijímat nová data
<code>din_rdy</code>	vstup	1	Indikace, vstupní data jsou platná
<code>data_in</code>	vstup	8	Symboly určené k dekódování
<code>dout_en</code>	vstup	1	Indikace, že je možné posílat na výstup dekódované symboly
<code>dout_rdy</code>	výstup	1	Oznámení platnosti výstupních symbolů
<code>data_out</code>	výstup	8	Dekódované symboly
<code>fail</code>	výstup	1	Indikace překročení samoopravné schopnosti kódu (tj. nastalo více než 8 chyb)

Tabulka 2: Stručný popis vstupů a výstupů makro procedury RS dekodéru.

5 Komunikační rozhraní

V této kapitole bude popsáno, jakým způsobem se ovládá obvod SMSC LAN91C111 a jak s jeho pomocí pracovat s ethernetovými rámci.

5.1 Popis ethernetového rozhraní

Obvod SMSC LAN91C111 umožňuje komunikovat přenosovou rychlostí 10/100 Mb/s. Má integrovanou linkovou (MAC) a analogovou (PHY) část fyzické vrstvy. Obrázek 8 znázorňuje strukturu obvodu (blok TR. označuje připojený transformátor). Obvod je vybaven 32-bitovou datovou sběrnici. Pro systémy s nižší datovou šířkou než je 32 bitů, lze přenos dat uskutečnit 16-bitově nebo 8-bitově. Šířka datového přenosu se nastaví pomocí signálů nBE0 - nBE3. Příslušné nastavení signálů nBE0 - nBE3 je uvedeno v tabulce 3.

nBE0	nBE1	nBE2	nBE3	Způsob přístupu
0	0	0	0	32 bitový přístup
0	0	1	1	16-bitový přístup k nižšímu slovu
1	1	0	0	16-bitový přístup k vyššímu slovu
0	1	1	1	8-bitový přístup k bytu 0
1	0	1	1	8-bitový přístup k bytu 1
1	1	0	1	8-bitový přístup k bytu 2
1	1	1	0	8-bitový přístup k bytu 3

Tabulka 3: Nastavení šířky datové sběrnice pro komunikaci s ethernetovým obvodem SMSC LAN91C111

Linková vrstva (MAC) je vnitřně 32-bitová. Má sadu 16-bitových registrů, rozdělených do 4 bank. Pomocí těchto registrů se obvod nastavuje, řídí a vyčítá datový prostor. Registry jsou mapovány na vnější paměťový prostor. Adresu a příslušnou banku registru znázorňuje tabulka 4. Pro příchozí a odchozí data je zde integrována paměť 8KB typu fronta (FIFO).

Fyzická vrstva (PHY) obsahuje také sadu registrů pro řízení komunikace. S touto vrstvou se komunikuje přes sériové rozhraní, které je vyvedeno na vnější vývody obvodu. Jedná se o takzvané MII (Media Independent Interface) rozhraní. Toto rozhraní je také mapováno do jednoho registru (MGMT - Management Interface) v linkové vrstvě - lze tedy k fyzické vrstvě přistoupit pomocí hostitelského systému.

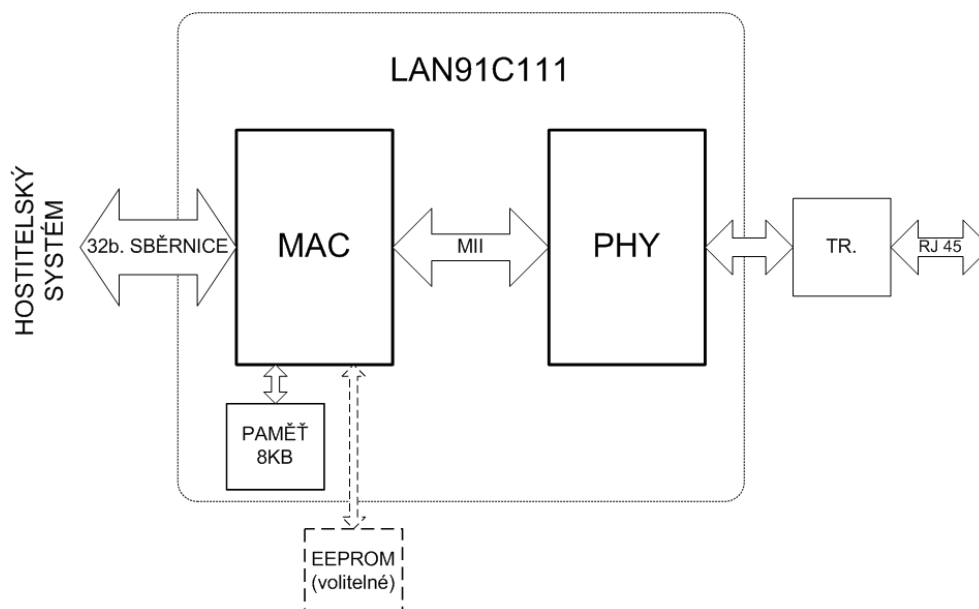
V dalším textu budu linkovou vrstvu označovat zkratkou MAC a fyzickou vrstvu zkratkou PHY.

5.2 Komunikace s integrovaným obvodem

Hostitelský systém může s obvodem komunikovat synchronním i asynchronním způsobem. Základem komunikačního rozhraní je datová a adresová sběrnice s doprovodnými kontrolními signály.

Adresová sběrnice je 15-bitová (signály označené jako A15 - A1) a dělí se na část adresující samotný obvod v hostitelském systému a část adresující registry obvodu. Vnější adresové signály A15 - A4 adresují obvod v hostitelském systému - jedná se o takzvanou základní adresu. Výchozí nastavení základní adresy je 300h. Tato adresa se dá měnit nastavením hodnoty registru BASE (viz 4) v MAC vrstvě obvodu. Adresové signály A3 - A1 adresují jednotlivé vnitřní registry obvodu. Mapa registrového pole je uvedena v tabulce 4 a na jednotlivé registry se budu v textu odkazovat označením v této tabulce. Obvod obsahuje celkem 4 banky registrů po 8 registrech. Aktuální banku registrů určuje registr BANK. Tento registr je přístupný bez ohledu na aktuální zvolenou banku.

Obsah jednotlivých registrů po zapnutí systému je dán výchozím nastavením od výrobce. Je-li nutné mít obsah registrů nastaven již po zapnutí systému (převážně v případech, kdy je nutné mít nastavenou základní adresu obvodu), lze jejich obsah uložit do externí paměti, zpravidla typu EEPROM. V našem případě není tato paměť na vývojové desce připojena a nastavení musí provést hostitelský systém.



Obrázek 8: Bloková struktura obvodu SMSC LAN91C111

Adresa	Banka 0	Banka 1	Banka 2	Banka 3
0	TCR	CONFIG	MMU COMMAND	MT0-1
2	EPH STATUS	BASE	PNR	MT2-3
4	RCR	IA0-1	FIFO PORTS	MT4-5
6	COUNTER	IA2-3	POINTER	MT6-7
8	MIR	IA4-5	DATA	MGMT
A	RPCR	GENERAL PURPOSE	DATA	REVISION
C	RESERVED	CONTROL	INTERRUPT	ERCV
E	BANK	BANK	BANK	BANK

Tabulka 4: Mapa registrů MAC vrstvy obvodu SMSC LAN91C111

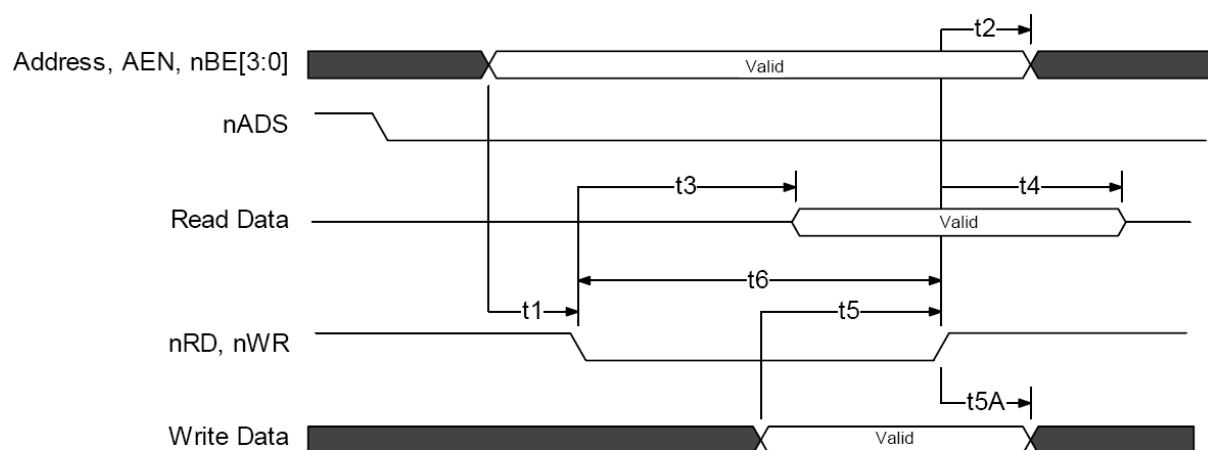
5.2.1 Cyklus sběrnice

Obvod podporuje asynchronní (obr. 9), synchronní (obr. 10) a blokový (obr. 11) mód přenosu. Zachytávání adresy je ve všech módech stejné. Signály ADS (ADDRESS STROBE) a AEN (ADDRESS ENABLE) určí platnost adresy. Signál ADS slouží pro zachycení adresy a je řízený úrovní. Signál AEN spouští dekódování adresy. Asynchronní mód je řízen signály pro čtení a zápis dat - RD (READ) a WR (WRITE). Synchronní mód je řízen pomocí signálu určující počátek cyklu sběrnice CYCLE, který je aktivní v logické nule. Blokový mód je uvozen signálem cyklu sběrnice a signálem pro blokový přenos DATACS. Blokový přenos je taktován hodinami obvodu (25 MHz).

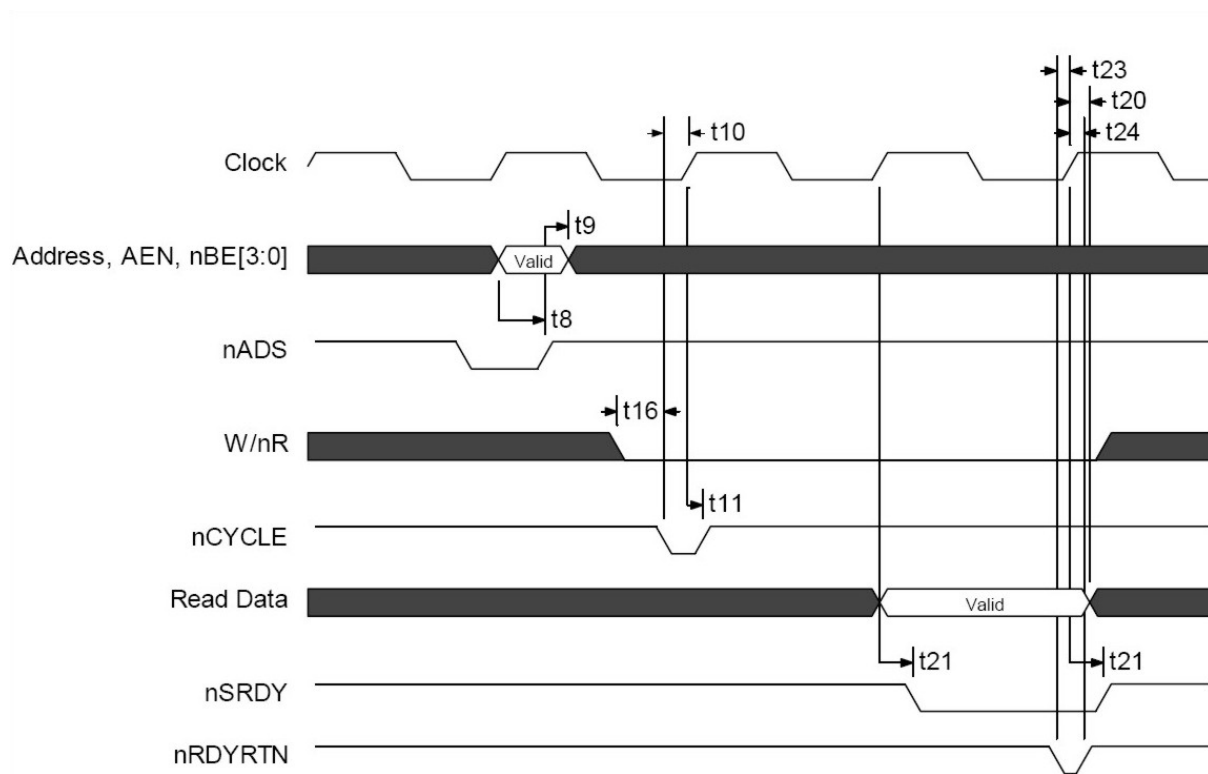
5.2.2 Konfigurace a základní nastavení obvodu

Po zapnutí se obvod nachází ve výchozím nastavení. Pro zahájení komunikace je nutné provést inicializaci skládající se z následujících kroků:

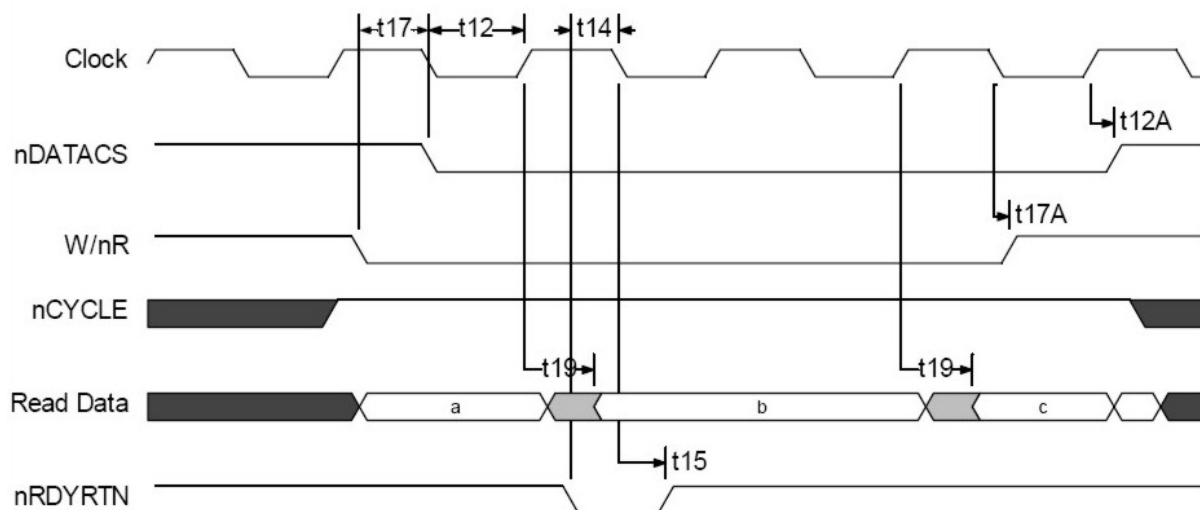
1. Nastavení MAC adresy. Tato adresa je uložena v registrech IA0-1, IA2-3 a IA4-5 ve vrstvě MAC.
2. Povolení komunikace MAC vrstvy s PHY vrstvou. Toto se provádí zápisem do registru "Control Register" PHY vrstvy - bit "MII_DIS". Dále umožnění předávání příslušných přerušovacích signálů



Obrázek 9: SMSC LAN91C111, sběrníkový cyklus, asynchronní přístup (převzato z [15])



Obrázek 10: SMSC LAN91C111, sběrníkový cyklus, synchronní přístup - čtení (převzato z [15])



Obrázek 11: SMSC LAN91C111, sběrníkový cyklus, blokový přístup – čtení (převzato z [15])

<Idle>	<Start>	<Read>	<Write>	<PHY Addr.>	<REG.Addr.>	<Turnaround>	<Data>
IDLE	ST[1:0]	READ	WRITE	PHYAD[4:0]	REGAD[4:0]	TA[1:0]	D[15:0]

Obrázek 12: Struktura sériového rámce pro komunikaci S PHY vrstvou obvodu SMSC LAN91C111 (převzato z [15]).

od PHY vrstvy. Nastavíme tedy "Interrupt Register" a "Interrupt Mask Register". Zápis do těchto registrů se provede přes sériové rozhraní PHY vrstvy. Toto rozhraní je přístupné přes registr MGMT vrstvy MAC. Formát sériového rámce je zobrazen na obrázku 12 a význam jednotlivých bitových polí rámce je vysvětlen v tabulce 5. Bližší informace lze nalézt v dokumentaci k obvodu [15].

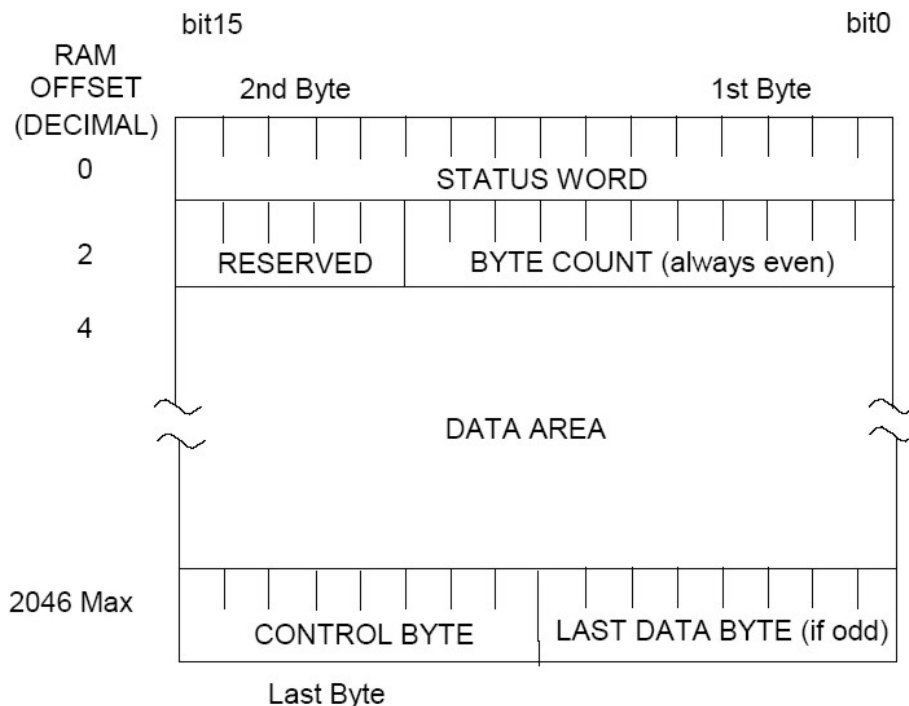
- Pro uvolnění všech front rámce je proveden resetovací příkaz. Příkazy obvodu se provádějí zápisem do registru MMUCR. Příkazy jsou reprezentovány číselně od 0 do 7. Příkaz pro uvolnění front je označen pod číslem 2. Význam jednotlivých příkazů a jejich číselná reprezentace je uvedena v tabulce 6. Bližší specifikaci lze nalézt v dokumentaci k obvodu [15]. Po zápisu do tohoto registru, se začne daný příkaz provádět. Provádění příkazu je indikováno bitem BUSY v registru MMUCR.
- Pro automatické uvolňování paměti po odeslání rámce, lze nastavit bit AUTORELEASE v registru CTR.
- Pro nastavení rychlosti komunikace přes ethernetové rozhraní lze využít automatického módu nastavením bitu ANEG v registru RPCR.
- Příjem rámců se povoluje nastavením bitu TXENA v TCR registru na hodnotu 1.
- Odesílání rámců se povoluje nastavením bitu RXEN v RCR registru na hodnotu 1.
- Dále nastavíme masku přerušení v INTERRUPT registru. Tento registr se dělí logicky na tři části označované v dokumentaci [15] k obvodu zkratkami ACK, IST a MSK. IST a ACK jsou umístěny na stejné adrese bytu. Pokud se čte z této adresy, je přečtena hodnota IST a pokud se zapisuje, tak je

Symbol	Význam	Popis
IDLE	Klidový stav	Tyto bity detekují klidový stav. Sériový rámec není přijat, pokud není přijato minimálně 31 za sebou jdoucích úrovní v logické 1.
ST[1:0]	Start bity	Pokud je detekována posloupnost ST[1:0]=01, jedná se o začátek sériového rámce.
READ	Čtení	
WRITE	Zápis	
PHYAD[4:0]	Fyzická adresa	Adresa PHY vrstvy. K obvodu lze připojit přes sériové rozhraní obvodu jinou fyzickou vrstvu, která je přístupná přes tuto adresu. Integrovaná PHY vrstva má adresu 00000.
REGAD[4:0]	Adresa registru	Adresa vnitřního registru PHY vrstvy, ke kterému se přistupuje. Pokud REGAD[4:0] nabývá hodnot 00000 až 11110, je zapisováno či čteno z konkrétního registru určeného touto adresou. Je-li REGAD[4:0] rovna 11111, je zapisováno či čteno ze všech registrů v jednom sériovém cyklu.
TA[1:0]	Čas změny směru přenosu	Toto bitové pole zajišťuje korektní přepnutí směru sériové sběrnice při čtení obsahu registru vrstvy PHY. Po zápisu adresy registru je nutné přepnout směr sběrnice a vyčíst hodnotu registru ze vstupu MDI. Pokud se bude číst (pole READ=1), pak TA[1:0]=Z0, kde Z znamená vysokou impedanci. Pokud se bude zapisovat (pole WRITE=1) tak TA[1:0]=10.
D[15:0]	Data	16-bitové pole pro čtená či zapisovaná data z registru určeným polem REGAD[4:0]

Tabulka 5: Popis významu jednotlivých polí sériového rámce pro komunikaci s vrstvou PHY ethernetového obvodu.

Číslo příkazu	Význam příkazu
0	žádná operace (NOP)
1	příkaz pro alokaci paměti pro odchozí rámec
2	uvolnění front pro příchozí a odchozí rámce, vymazání všech příslušných přerušení a vynulování ukazatelů do front pro rámce
3	odebrání rámce z vrcholu příchozí fronty
4	odebrání rámce z vrcholu příchozí fronty a uvolnění paměti alokované tímto rámcem
5	uvolní paměť specifikovaného (v registru PNR) rámce
6	zařadí specifikovaný (v registru PNR) rámec do odchozí fronty
7	resetuje odchozí frontu

Tabulka 6: Čísla příkazů obvodu SMSC LAN91C111 a jejich význam



Obrázek 13: Struktura datového prostoru obvodu SMSC LAN91C111 pro odeslané a přijaté rámce (převzato z [15])

zapsáno do ACK. IST obsahuje informaci o tom, k jaké události při přerušení došlo. ACK slouží k potvrzování přerušení. MSK nastavuje masku přerušení.

- Pro obsluhu některých přerušení je nutné je nejprve potvrdit zápisem nuly na příslušnou bitovou pozici stavového slova ACK. Jedná se o přerušení označovaná v dokumentaci jako MDINT (přerušení od PHY vrstvy), ERCV_INT (překročení stanovené délky rámce), RX_OVRN (při příchodu rámce nebyl dostatek paměti pro jeho uložení), TX_EMPTY_INT (odchozí fronta je prázdná), TX_INT (indikuje chybu při odesílání - například kolize). Význam jednotlivých přerušení je blíže popsán v dokumentaci k tomuto obvodu [15].

5.3 Práce s rámcí

V obvodu je integrováno 8KB paměti typu fronta (FIFO) pro příchozí i odchozí rámce. Každý rámec alokuje prostor 2KB. Tento prostor je tvořen datovou strukturou, která se skládá ze stavových slov a vlastních dat rámce. Datová struktura pro rámce je zobrazena na obrázku 13

Pole "STATUS WORD" obsahuje informace o stavu přijatého nebo odeslaného rámce. Pro přijatý rámec obsahuje následující informace:

- zda došlo ke špatnému zarovnání datového rámce (celkový počet přijatých bitů není dělitelný osmi),
- rámec byl přijat jako broadcast,
- CRC rámec neodpovídá přijatým datům,
- zda byl přijatý rámec delší než maximální délka, kterou stanovuje norma IEEE 802.3 (1518 bytů),
- zda byl přijatý rámec kratší než minimální délka, kterou stanovuje norma IEEE 802.3 (64 bytů),
- rámec byl přijat jako multicast.

Při odesílání rámce je pole "STATUS WORD" zkopírováno do registru EPHSR. Toto pole obsahuje informace o stavech, které při odesílání mohly nastat:

- úspěšné odeslání,
- při odesílání došlo ke kolizi,
- došlo ke ztrátě nosné,
- odeslaný rámeček byl broadcast,
- odeslaný rámeček byl multicast.

V poli "BYTE COUNT" je uložen počet bytů uložených ve struktuře včetně samotných doplňujících polí. "CONTROL BYTE" obsahuje informaci o tom zda pole "LAST DATA BYTE" je platné (pokud byl přijat lichý počet bytů). Pokud odesíláme rámeček, pak v tomto poli určíme, zda je počet bytů lichý a také zda se mají odesílaná data doplnit o CRC automaticky nebo zda již bylo CRC zadáno hostitelským systémem.

5.3.1 Přístup k datové struktuře rámce

Datovou strukturu je nejdříve nutné alokovat v paměti obvodu pro ethernetovou komunikaci. Při příjmu rámce dojde k alokaci automaticky a přijatý rámeček je zařazen do příchozí fronty (označováno jako FIFO FOR RX). Datová struktura pro odesílaný rámeček musí být alokována za pomoci hostitelského systému následujícími kroky:

1. Do registru MMUCR zapíšeme příkaz pro alokaci paměti pro odesílaný rámeček. Jedná se o příkaz číslo 1 (význam jednotlivých příkazů je uveden v tabulce 6).
2. Po dokončení alokace přečteme výsledek operace z registru PNR, kde bit FAIL indikuje, zda došlo k bezchybné alokaci. Pokud nedošlo k chybě, obsahuje vyšší byte registru identifikační číslo nově alokované struktury.
3. Nižší byte tohoto registru určuje, s kterou datovou strukturou se pracuje ve frontě pro odesílání (označována jako FIFO FOR TX). Chceme-li následně pracovat s právě alokovanou strukturou, zapíšeme její identifikační číslo do tohoto nižšího bytu.
4. Při neúspěšné alokaci je třeba uvolnit paměť obvodu a operaci opakovat.

Chceme-li pracovat s alokovanými strukturami a přistupovat k jejich datům, musíme tak učinit pomocí ukazatele do datové struktury. Ukazatelem je v tomto případě registr PTR. K vlastním datům pak přistupujeme přes datový registr DATA.

Obsah registru PTR obsahuje následující informace o přístupu k datům. Určuje:

- Zda pracujeme s frontou pro odesílané či přijímané rámce. Pokud pracujeme s frontou pro přijaté rámce, pracujeme vždy s prvním rámcem ve frontě. Pokud pracujeme s frontou pro odesílání, pracujeme s rámcem, který je určen v registru PNR svým identifikačním číslem.
- Zda se má ukazatel do oblasti dat automaticky zvýšit při každém přístupu (ukazatel se příslušně zvýší podle šířky datového přístupu).
- Zda se bude číst či zapisovat do datové oblasti.
- Vlastní ukazatel do datové oblasti přistupovaného rámce.

5.3.2 Odesílání a přijímání rámců

Příjem rámce vyvolá přerušení, toto přerušení trvá, dokud není fronta pro příjem rámců prázdná. Po zpracování přijatých dat můžeme rámec z paměti obvodu uvolnit. Uvolnění paměti a odebrání rámce z fronty se provede zápisem příslušného příkazu do registru MMUCR (příkaz číslo 4 viz tabulka 6).

Pokud chceme odeslat rámec musíme pro něj alokovat paměť způsobem, jak bylo popsáno výše. Po vyplnění datové struktury rámce, zařadíme rámec do odchozí fronty odešleme zápisem příkazu do registru MMUCR (příkaz číslo 6 viz tabulka 6). Pokud nemáme zapnuto automatické uvolnění odeslaných rámců (v registru CTR), musíme tuto paměť uvolnit (příkaz číslo 5 viz tabulka 6).

6 Řízení komunikačního rozhraní

Řízení komunikačního rozhraní zajišťuje procesor PicoBlaze (dále v textu budu označovat procesor PicoBlaze pouze jako procesor), protože řízení komunikačního rozhraní má sekvenční charakter. Ethernetový obvod (SMSC 91C111) komunikačního rozhraní je vybaven 32-bitovou obousměrnou datovou sběrnici. Procesor disponuje pouze vstupním a výstupním 8-bitovým portem. Je tedy nutné nalézt způsob, jak zajistit komunikaci mezi ethernetovým obvodem a procesorem.

Šířka datové sběrnice ethernetového obvodu lze nastavit pro 8-bitový režim přenosu dat, jak bylo popsáno v kapitole 5. Nyní zbývá vyřešit řízení směru toku dat mezi obousměrnou sběrnici ethernetového obvodu a porty procesoru. Toto vyřešíme implementací řídicího stavového automatu (dále v textu budu označovat řídicí stavový automat pouze jako automat), který bude zajišťovat přepínání směru datové sběrnice a generování příslušného cyklu sběrnice ethernetového obvodu.

Procesor bude pomocí stavového automatu číst či zapisovat hodnoty registrů ethernetového obvodu. Tento automat se bude také využívat při vyčítání či zápisu rámců kóděrem/dekóděrem. Při komunikaci s kóděrem/dekóděrem bude datová sběrnice přepnuta do 16-bitového režimu přenosu dat pro zvýšení propustnosti systému.

Při startu systému procesor provede inicializaci ethernetového obvodu a další řízení toku dat bude kontrolováno pomocí přerušení. Přerušení je generováno tehdy, pokud nastala nějaká událost v ethernetovém obvodu (typicky událost příchodu rámce) nebo pokud je kódér/dekódér připraven poslat ethernetový rámec.

Jelikož maximální velikost programu je u této verze procesoru omezená datovou šířkou čítače instrukcí, je program uložen ve dvou pamětech, mezi kterými se přepíná pomocí vykonávaného programu. Inicializační program je v jedné paměti a obsluha řízení toku dat je v paměti druhé.

Obsluha přerušení je zajištěna vykonáním příslušné posloupnosti instrukcí programu procesoru. Výhodou této realizace je možnost snadné modifikace příslušného kódu programu při potřebě změnit chování řízení toku dat.

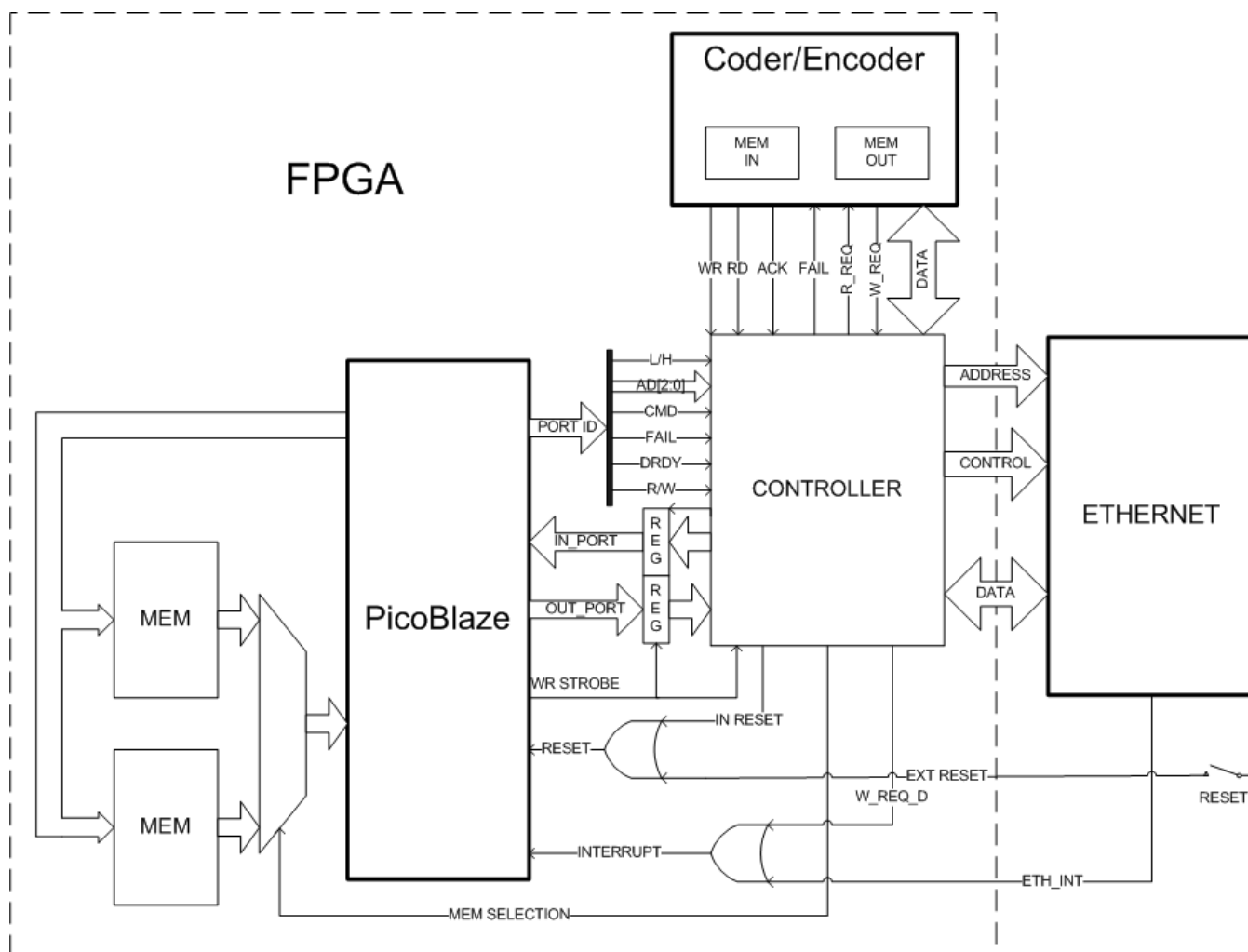
Základní blokové schéma propojení procesoru, ethernetového obvodu a kóděru/dekóděru je znázorněno na obrázku 14 (řídicí stavový automat je označen jako CONTROLLER).

6.1 Mapování portů PicoBlaze na adresní prostor ethernetového čipu

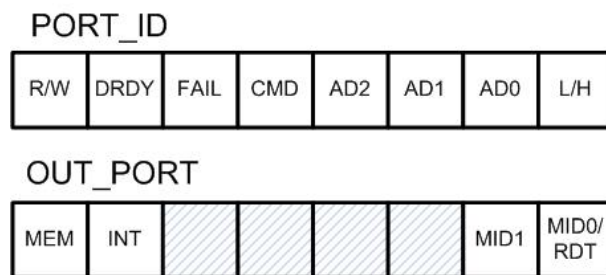
Adresa ethernetového obvodu se dělí na základní adresu a adresu vnitřního registrového pole obvodu jak bylo popsáno v kapitole 5. Základní adresa je nastavena na pevnou hodnotu, která se nemění. Procesor musí tedy adresovat pouze v rámci registrového pole obvodu.

Toto pole je rozděleno na jednotlivé banky registrů. Přístupná je vždy pouze jedna banka, která obsahuje osm 16-bitových registrů. Adresa určí daný registr, který je však 16-bitový. Procesor pracuje pouze s 8-bitovými hodnotami.

Možnost pracovat s 8-bitovými hodnotami lze vyřešit v kombinaci s řízením šířky datové sběrnice a přistupovat tak ke konkrétnímu bytu adresovaného registru. Při 8-bitovém přístupu je byte určen signály nBE0 - nBE3 (viz tabulka 3). Bereme-li v úvahu, že registr je 16-bitový, musíme určit, ke kterému bytu



Obrázek 14: Propojení procesoru PicoBlaze s ethernetovým čipem SMSC LAN91C111



Obrázek 15: Význam jednotlivých bitů rozhraní při komunikaci se stavovým automatem

přístupujeme. Pro určení konkrétního bytu v bance registrů potřebujeme 4-bitovou adresu. Pro předávání adresy automatu využijeme rozhraní procesoru pro určení čísla portu (PORT_ID) a pro předávání hodnot rozhraní OUT_PORT a IN_PORT.

6.2 Stavový automat

Jak bylo řečeno, pro zprostředkování komunikace mezi ethernetovým obvodem, procesorem a kódérem/dekódérem bylo zapotřebí navrhnout stavový automat. Procesor řídí automat pomocí výstupního portu (OUT_PORT) a rozhraní pro určení čísla portu (PORT_ID). V následujícím textu popíšeme funkci automatu a význam jednotlivých bitových polí rozhraní PORT_ID a OUT_PORT pro jeho ovládání. Uspořádání jednotlivých bitových polí je zobrazeno na obrázku 15, jejich význam je popsán v tabulce 7. Dekódování výstupu rozhraní je spuštěno doprovodným signálem pro operaci zápisu (WRITE_STROBE).

Automat řídí následující operace:

- čtení hodnoty z registrů ethernetového obvodu
- zápis hodnoty do registrů ethernetového obvodu
- čtení stavu přerušení
- přepnutí paměti programu procesoru
- přenos rámců mezi kódérem/dekódérem a ethernetovým obvodem

6.2.1 Čtení a zápis hodnot registrů ethernetového obvodu

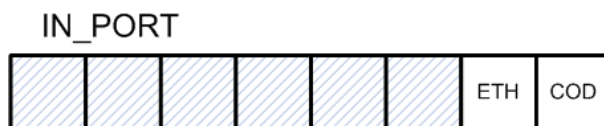
Vstupní a výstupní operace u procesoru trvá pouze dva hodinové cykly. Cyklus sběrnice se skládá z více hodinových cyklů. Automat zachytí hodnotu adresy (bity AD[2:0] a L/H), na kterou se přistupuje, a je proveden příslušný cyklus sběrnice čtení či zápisu (v našem případě je použit asynchronní přístup, který byl zmíněn v kapitole 5 na obrázku 9). Při vstupní operaci musí procesor nejdříve informovat automat o tom, že chce číst z daného registru, a hodnotu přečíst po dokončení cyklu sběrnice. Automat přečte hodnotu z daného místa a uloží ji do registru připojeného na vstupní port IN_PORT. V době, kdy je prováděn cyklus sběrnice, vykonává procesor prázdné operace. Počet prázdných operací musí být spočten podle počtu taktů čtecího cyklu sběrnice. Poté procesor přečte správnou hodnotu ze vstupního portu.

6.2.2 Přerušení a jeho obsluha

Procesor má pouze jeden vstup pro přerušení. V našem případě je třeba obsluhovat dvě přerušení - přerušení od ethernetového obvodu a od kódéru/dekódéru. Signál požadavku kódéru/dekódéru a přerušení ethernetového obvodu je spojeno do jediného vstupu pomocí logického součtu. Procesor při detekci

R/W	Bit určuje, zda se bude číst či zapisovat hodnota registru ethernetového obvodu. Při zápisu je hodnota určena rozhraním OUT_PORT. Při čtení je proveden čtecí cyklus sběrnice ethernetového obvodu a hodnota z čteného registru je uložena do registru připojeného na vstupní port IN_PORT.
DRDY	Nastavením tohoto bitu procesor předává kontrolu nad přenosem rámce kodéru/dekodéru. Je-li nastaven bit RDT na hodnotu 1, bude se provádět zápis rámce kodérem/dekodérem do ethernetového obvodu. Nemá-li bit RDT nastaven, pak se bude provádět čtení rámce kodérem/dekodérem.
FAIL	Bit je nastaven na hodnotu 1, pokud nedošlo k alokaci paměti v ethernetovém obvodu pro odesílaný rámec (typicky při požadavku kodéru/dekodéru odeslat rámec).
CMD	Bit indikuje příkaz určený hodnotou rozhraní OUT_PORT. Příkaz pro přepnutí paměti (určenou bity MID1 a MID0) programu procesoru je indikován bitem MEM a příkaz pro připravení stavu přerušení na vstupní port procesoru je indikován bitem INT.
AD[2:0]	Adresní bity registru ethernetového obvodu.
L/H	Určuje, zda se pracuje s horním či spodním bytem registru (L/H=1 pro horní byte a L/H=0 pro spodní byte).
MEM	Indikuje s bitem CMD přepnutí paměti programu procesoru. Bity MID1 a MID0 určí, která paměť se má připojit. Po přepnutí paměti je procesor resetován.
INT	Indikuje s bitem CMD načtení stavu přerušení do registru připojeného na vstupní rozhraní procesoru IN_PORT. Nejnižší bit hodnoty přečtené ze vstupního rozhraní určuje, zda je přerušení generováno ethernetovým obvodem. Druhý bit určuje, zda přerušení pochází od kodéru/dekodéru (viz obrázek 16).
MID1	Ve spojení s příkazem pro přepnutí paměti programu určuje identifikátor cílové paměti.
MID0/RDT	Ve spojení s příkazem pro přepnutí paměti programu určuje identifikátor cílové paměti. Ve spojení s bitem DRDY určuje směr přenosu rámce mezi kodérem/dekodérem a ethernetovým obvodem.

Tabulka 7: Význam bitových polí pro řízení automatu



Obrázek 16: Příslušnost bitových polí při čtení stavu přerušení

COD	Kodér/dekodér generuje přerušení.
ETH	Ethernetový obvod generuje přerušení.

Tabulka 8: Význam bitových polí stavu přerušení

přerušení zjistí pomocí příkazu pro zjištění stavu přerušení, které signály jsou nastaveny. Tento příkaz je proveden pomocí výstupních rozhraní procesoru, jak bylo popsáno výše. Po dekódování příkazu je do registru připojeného na vstupní port nahrán stav přerušení. Příslušnost jednotlivých bitových polí je znázorněna na obrázku 16, jejich význam je popsán v tabulce 8.

6.2.3 Přenos rámců mezi ethernetovým obvodem a kodérem/dekodérem

Přenos rámců mezi ethernetovým obvodem a kodérem/dekodérem není zajišťováno procesorem. Procesor pouze připraví pro tento přenos podmínky.

Pokud dojde k přijetí rámce přes ethernetové rozhraní, procesor nastaví registr PTR ethernetového obvodu na začátek rámce uloženém ve frontě, nastaví jej pro automatickou inkrementaci (viz kapitola 5) a předá kontrolu kodéru/dekodéru (indikováno nastavením hodnoty bitu DRDY do logické 1 na rozhraní PORT_ID a doplněno informací o směru - bitem MID0/RDT). Kodér/dekodér provádí čtení pouze nastavováním signálu RD (Read - čtení). Signál RD nesmí nastavit znovu dříve, než dojde k dokončení cyklu sběrnice ethernetového obvodu, a tak k uložení čtené hodnoty na výstup rozhraní mezi automatem a kodérem/dekodérem.

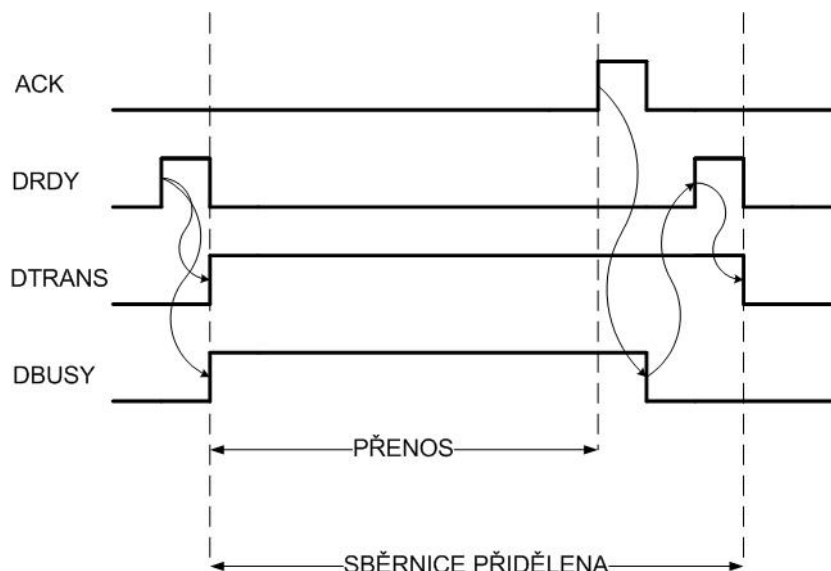
Od této chvíle procesor nemůže komunikovat s ethernetovým obvodem, protože sběrnice je směrována ke kodéru/dekodéru. Předání sběrnice je indikováno vnitřním stavem automatu - buď jej označovat jako DTRANS. Přenos po sběrnici je indikován vnitřním stavem automatu DBUSY. Indikace tohoto stavu je připojena na vstupní port procesoru. Procesor tak čeká na ukončení přenosu.

Ukončení přenosu kodéru/dekodéru je signalizováno nastavením hodnoty signálu ACK do logické 1. Po potvrzení hodnotou signálu ACK je nulován vnitřní stav DBUSY. Procesor pak opět převezme kontrolu nad sběrnici (opět pomocí nastavení bitu DRDY na rozhraní PORT_ID). Průběh a závislost jednotlivých signálů a stavů znázorňuje obrázek 17.

Obdobným způsobem je obsloužen požadavek kodéru/dekodéru na odeslání rámce přes ethernetové rozhraní. Procesor se pokusí alokovat paměť pro nový rámeček. Pokud dojde k úspěšné alokaci rámce, je nastaven registr POINTER ethernetového rámce na začátek alokované paměti a je nastaven také pro autoinkrementaci. Posloupnost signálů je totožná jako při předávání sběrnice pro čtení rámce (viz obrázek 17). Zápis dat je prováděn obdobným způsobem jako při čtení, tentokrát pomocí signálu WR (Write - zápis). Hodnota vystavená kodérem/dekodérem je zachycena automatem a uložena do registru ethernetového obvodu (v tomto případě registru DATA).

Pokud nedojde k úspěšné alokaci paměti pro nový rámeček, je o tom kodér/dekodér informován nastavením hodnoty signálu FAIL do logické 1.

Rámce jsou přenášeny tak, jak jsou uloženy v paměti ethernetového obvodu. Kodér/dekodér tedy musí velikost rámce, a tím i délku přenosu, určit podle pole "BYTE COUNT" (viz kapitola 5).



Obrázek 17: Průběh signálů při předání sběrnice kodéru/dekodéru

6.2.4 Přepínání paměti programu

Procesor zajišťuje inicializaci ethernetového obvodu a řízení toku dat mezi kodérem/dekodérem a ethernetovým obvodem. Program pro inicializaci zabírá 166 instrukcí, což je poměrně hodně vzhledem k tomu, že paměť programu může obsahovat maximálně 256 instrukcí. Proto bylo nutné tuto paměť programu zvětšit.

Zvětšení paměti programu jsem zajistil připojením druhé paměti a vytvořením prostředků pro přepínání. Mezi těmito paměťmi se dle potřeby přepíná. Přepnutí je prováděno příkazem pro automat. Příkaz pro přepnutí paměti je indikován nastavením bitových polí CMD a MEM. Paměť, na kterou se má přepnout, určuje bitové pole MID1 a MID0/RDT. Pro určení dvou pamětí je zapotřebí pouze bitového pole MID0/RDT. Pro případ, že by se připojovalo pamětí více (v tomto případě maximálně 4), je zde pro tento účel připraveno bitové pole MID1.

Po dekódování tohoto příkazu automat pomocí multiplexoru přepne paměť a resetuje procesor, pomocí resetovacího vstupu procesoru. Resetovací signál musí být nastaven minimálně dva hodinové takty. Pomocí logického součtu je s vnějším resetovacím signálem spojen reset od automatu. Zjednodušené schéma připojení dvou pamětí a resetovacího signálu je zobrazen na obrázku 14.

6.3 Program procesoru

Program procesoru bude komunikovat převážně s ethernetovým obvodem. Tato komunikace se skládá převážně z čtení a zápisu hodnot registrů ethernetového obvodu. Proto jsou v programu vytvořeny procedury pro čtení a zápis hodnot registrů MAC vrstvy a proceduru pro zápis do registrů PHY vrstvy ethernetového obvodu, které se budou v programu volat s příslušnými parametry. Těmto procedurám jsou předávány parametry pomocí v programu pevně stanovených registrů. Při programování musíme brát tedy ohled na speciální význam registrů při volání těchto procedur.

S využitím těchto procedur je po startu systému zinicializován ethernetový obvod jak bylo popsáno v kapitole 5. Po dokončení inicializace je přepnuta paměť programu. V této paměti je uložen hlavní program a rutina obsluhy přerušení. Hlavním programem je zde nekonečná smyčka, v které není vykonáván žádný užitečný kód, ale čeká pouze na přerušení. Přerušení je vyvoláno buď ethernetovým obvodem nebo kodérem/dekodérem, který bude odesílat ethernetový rámec.

Po příchodu přerušení a spuštění obslužné rutiny je přečten stav přerušení a na základě této infor-

mace se rutina větví na obsluhu přerušení ethernetovým obvodem a obsluhu požadavku kodéru/dekodéru (větev RS - viz 18).

Jedná-li se o přerušení od kodéru/dekodéru, je spuštěna alokace paměti v ethernetovém obvodu. Při neúspěšné alokaci je automat signalizován tento stav nastavením bitu FAIL (automat tento signál předá kodéru/dekodéru) a je ukončena obslužná rutina. Při úspěšné alokaci je nastaven obsah registru PNR hodnotou identifikátoru nového rámce. Poté je nastaven registr POINTER tak, aby ukazoval na začátek alokované paměti, a aby docházelo při zápisu k automatické inkrementaci tohoto registru.

Procesor předá řízení sběrnice kodéru/dekodéru pomocí nastavení bitu DRDY a čeká na dokončení přenosu dat od kodéru/dekodéru - čte hodnotu vstupního portu, na který je v tomto případě připojena indikace stavu automatu DBUSY. Po dokončení přenosu procesor převezme řízení sběrnice opětovným nastavením bitu DRDY. Průběh a závislost jednotlivých signálů a stavů automatu znázorňuje obrázek 17.

Rámec je poté odeslán tak, že je zařazen do fronty odchozích rámců v ethernetovém obvodu.

Pokud je přerušení vyvoláno ethernetovým obvodem, je přečten registr INTERRUPT (část označovaná jako IST). Na základě obsahu registru INTERRUPT zjistíme, zda došlo k přijetí rámce, nebo došlo k jiné události. Pokud nastala jiná událost, obslužná rutina skončí. V případě potřeby tyto události obsloužit, lze snadno doplnit obslužnou rutinu o příslušný kód.

Dojde-li k přijetí rámce, pak je registr POINTER nastaven tak, aby ukazoval na začátek paměti příchozího rámce, a aby docházelo při čtení k automatické inkrementaci tohoto registru. Poté dojde k předání řízení sběrnice kodéru/dekodéru stejným způsobem, jak zde již bylo popsáno.

Po dokončení přenosu je rámec uvolněn z paměti ethernetového obvodu a obslužná rutina skončí.

Struktura obslužné rutiny je znázorněna pomocí vývojového diagramu na obrázku 18.

7 Komunikační protokol

Data mezi kodérem/dekodérem a uživatelem jsou předávána pomocí ethernetových rámců. Data jsou přenášena pomocí protokolu UDP. Protokol je implementován pouze částečně. Hlavička IP nesmí obsahovat přídatné možnosti (IP hlavička musí mít velikost 20 bytů).

Návrh nepodporuje protokol ARP, proto vysílající strana musí do záznamu ARP tabulky nastavit příslušnou MAC adresu (MAC adresa a je nastavena v inicializačním programu). Na IP adrese nezáleží - návrh pro odpověď použije cílovou IP adresu z příchozího rámce.

Příchozí rámce jsou filtrovány pomocí procesoru. Procesor zkontroluje typ rámce a cílový port. Pokud rámec není typu UDP nebo je cílový port rozdílný od nastavené hodnoty obsažené v programu, pak je rámec zahozen a není na něj nijak odpovězeno.

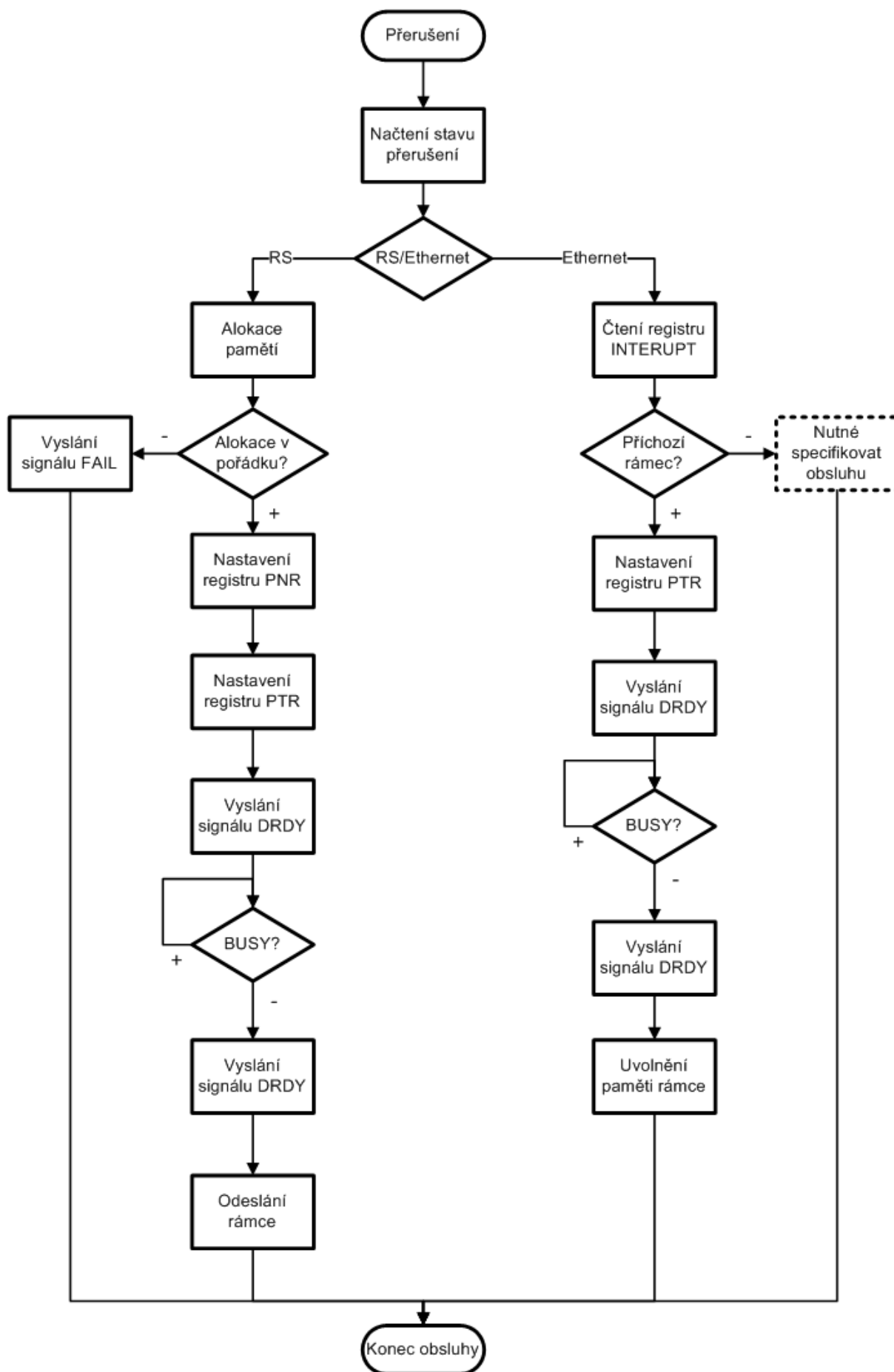
Rámec obsahující data je pomocí automatu uložen do paměti. Z rámce jsou načteny potřebné informace (MAC adresy, IP adresy a UDP porty) pro sestavení rámce, který bude poslán zpět vysílající straně se zpracovanými daty.

Kodér/dekodér uloží výstupní data do paměti. Po uložení jsou data doplněna o řídicí slova obvodu SMSC 91C111 (viz. 13), ethernetovou hlavičku, IP hlavičku a UDP hlavičku. Takto vytvořený rámec je předán do výstupní fronty ethernetového obvodu a za pomoci procesoru je odeslán.

Pokud příchozí rámec obsahuje více vstupních řetězců symbolů (pro kodér $N * 239$ bytů a pro dekodér $N * 255$ bytů) bude i výsledný rámec s odpovědí obsahovat N odpovídajících řetězců symbolů. Počet obsažených řetězců je omezen na maximální počet roven pěti (omezeno maximální velikostí ethernetového rámce).

8 Ověření funkce

Ověření dekodéru a kodéru je možné nahráním příslušného bitstreamu (.sof), který se nahraje do je jedné z vývojových desek:



- Nios Development Board, Cyclone Edition (osazena obvodem Altera Cyclone EP1C20F400C7)
- Nios Development Board, Stratix Edition (osazena obvodem Altera Stratix EP1S10F780C6)
- Stratix II EP2S180 DSP Development Board (osazena obvodem Altera Stratix II EP2S180F1020C3)

Vstupní data pro kodér/dekodér jsou předávána pomocí ethernetového rozhraní za pomoci UDP protokolu. Po nahrání příslušného bitstreamu má vývojová deska nastavenou MAC adresu na 00:07:ed:0a:05:f3 a přijímá na portu číslo 2000.

Soubory s definicemi vstupních dat a jejich ověření pro kodér i dekodér lze vygenerovat pomocí skriptu v Matlabu. Tento skript je uložen v souboru

`\matlab\datsrcencdec.m.`

8.1 Ověření funkčnosti dekodéru

Funkci dekodéru lze ověřit nahráním bitstreamu do příslušného obvodu vývojové desky.

Bitstramy pro jednotlivé vývojové desky jsou uloženy v souborech uvedených v tabulce 9.

Cílová vývojová deska:	Cesta:
Nios Development Board, Cyclone Edition	<code>\pb_eth_rsdec\pblaze_eth\CYCLONE\pblaze_eth.sof</code>
Nios Development Board, Stratix Edition	<code>\pb_eth_rsdec\pblaze_eth\STRATIX\pblaze_eth.sof</code>
Stratix II EP2S180 DSP Development Board	<code>\pb_eth_rsdec\pblaze_eth\STRATIXII\pblaze_eth.sof</code>

Tabulka 9: Cesty k bitstreamům dekodéru pro jednotlivé vývojové desky.

Tyto bitstreamy lze do vývojové desky nahrát pomocí downloaderu vývojového prostředí Quartus II (viz obrázek 19).

Po nahrání bitstreamu vývojová deska čeká na příchozí ethernetový rámec obsahující data a po přijetí zobrazí na sedmi-segmentovém displeji počet přijatých řetězců o délce 255 bytů, zpracuje data a vyšle je přes ethernetové rozhraní. Rámec musí být ve formátu UDP a číslo cílového portu rovno 2000. Vývojová deska nemá přiřazenu žádnou IP adresu a nepodporuje protokol ARP. Pro komunikaci s vývojovou deskou je nutné přidat statický záznam v ARP tabulce. IP adresu zvolíme libovolnou a MAC adresa je 00:07:ed:0a:05:f3. V prostředí MS Windows vytvoříme statický záznam příkazem `arp -s XX.XX.XX.XX 00-07-ed-0a-05-f3`, kde XX.XX.XX.XX je libovolná zvolená adresa (viz obrázek 20).

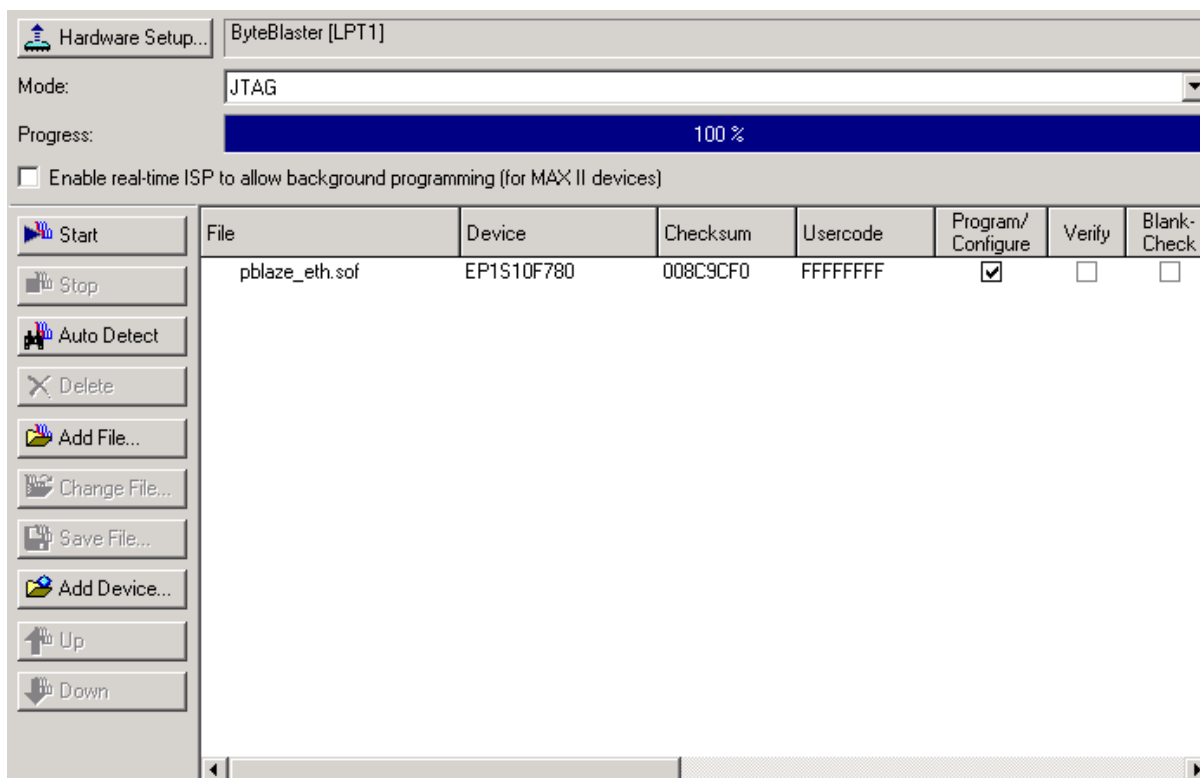
Data pro dekodér lze zaslat pomocí jednoduchého programu uloženého ve spustitelném souboru `\sendudp\Release\sendup.exe`. Užití programu je následující:

```
sendup.exe <cílová IP adresa> <cílový port> [Zdrojová IP adresa] <vstupní soubor>
          <výstupní soubor>
```

Cílová IP adresa je zvolená IP adresa v záznamu ARP tabulky, cílový port je v našem případě roven 2000, zdrojová IP adresa je nepovinný parametr (pokud není zadána je nastavena automaticky), vstupní soubor obsahuje binární data se vstupními řetězci a do výstupního souboru se uloží přijatá data (použití programu viz obrázek 21).

V souboru `\sendudp\Release\decdata.bin` jsou zakódovaná data s vnesenou chybou (2 řetězce o délce 255 bytů, kde je v každém řetězci záměrně vneseno 8 chybných bytů), který lze použít pro ověření funkce dekodéru. Přijatá data lze porovnat s nezakódovanými daty (2 řetězce o délce 239 bytů bez vnesené chyby) uložené v souboru `\sendudp\Release\encdata.bin`.

Pomocí zasílání dat přes ethernetové rozhraní lze dekodovat rychlostí až 1,64 MB/s.



Obrázek 19: Dialogové okno downloaderu vývojového prostředí Quartus II.

Obvod	Cyclone	Stratix	Stratix II
LEs (ALUTs)	9,889 / 20,060 (49 %)	9,889 / 10,570 (94 %)	9,750 / 143,520 (7 %)
Bitů paměti	78,440 / 294,912 (27 %)	78,440 / 920,448 (9 %)	78,440 / 9,383,040 (< 1 %)
f _{max} [MHz]	53.17	57.65	66.62

Tabulka 10: Parametry HW návrhu dekodéru

8.2 Ověření funkčnosti kodéru

Kodér lze ověřit stejným způsobem jako dekodér, s tím rozdílem, že přijímá řetězce dlouhé 239 bytů a vysílá zakódované řetězce o délce 255 bytů (239 původních bytů a 16 zabezpečovacích bytů).

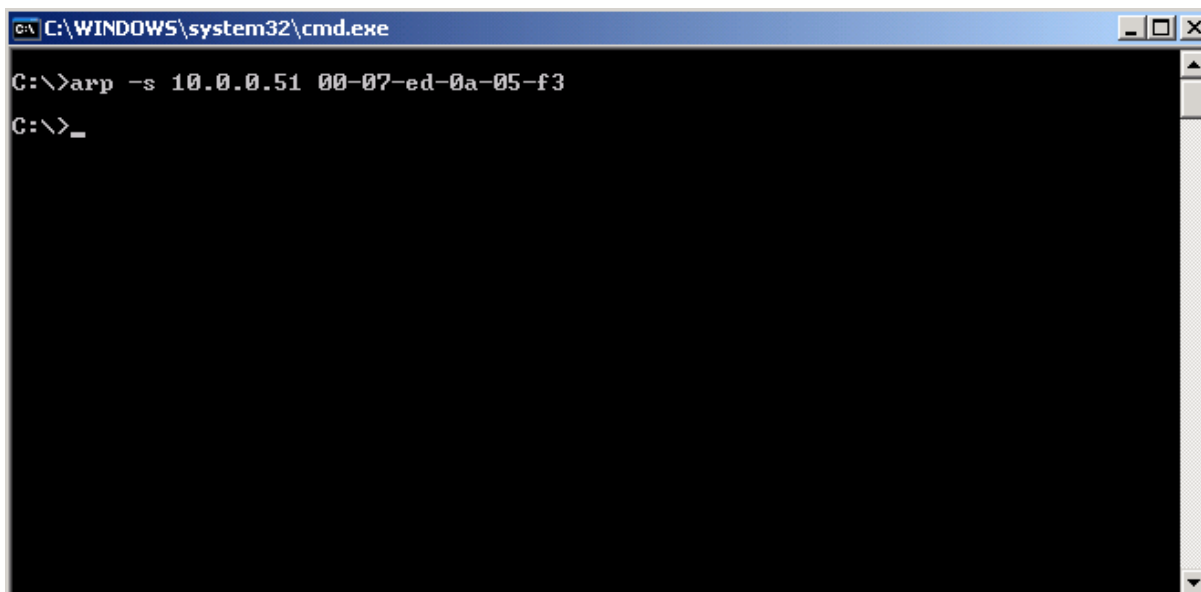
Bitstramy pro jednotlivé vývojové desky jsou uloženy v souborech uvedených v tabulce 11.

Cílová vývojová deska:	Cesta:
Nios Development Board, Cyclone Edition	\pb_eth_rsenc\pblaze_eth\CYCLONE\pblaze_eth.sof
Nios Development Board, Stratix Edition	\pb_eth_rsenc\pblaze_eth\STRATIX\pblaze_eth.sof
Stratix II EP2S180 DSP Development Board	\pb_eth_rsenc\pblaze_eth\STRATIXII\pblaze_eth.sof

Tabulka 11: Cesty k bitstreamům kodéru pro jednotlivé vývojové desky.

Pro ověření funkce kodéru je možné pomocí výše zmíněného programu zaslat data pro kodér, která jsou uložena v souboru \sendudp\Release\encdata.bin. Přijatá data můžeme pro ověření porovnat se souborem \sendudp\Release\decdata_ok.bin.

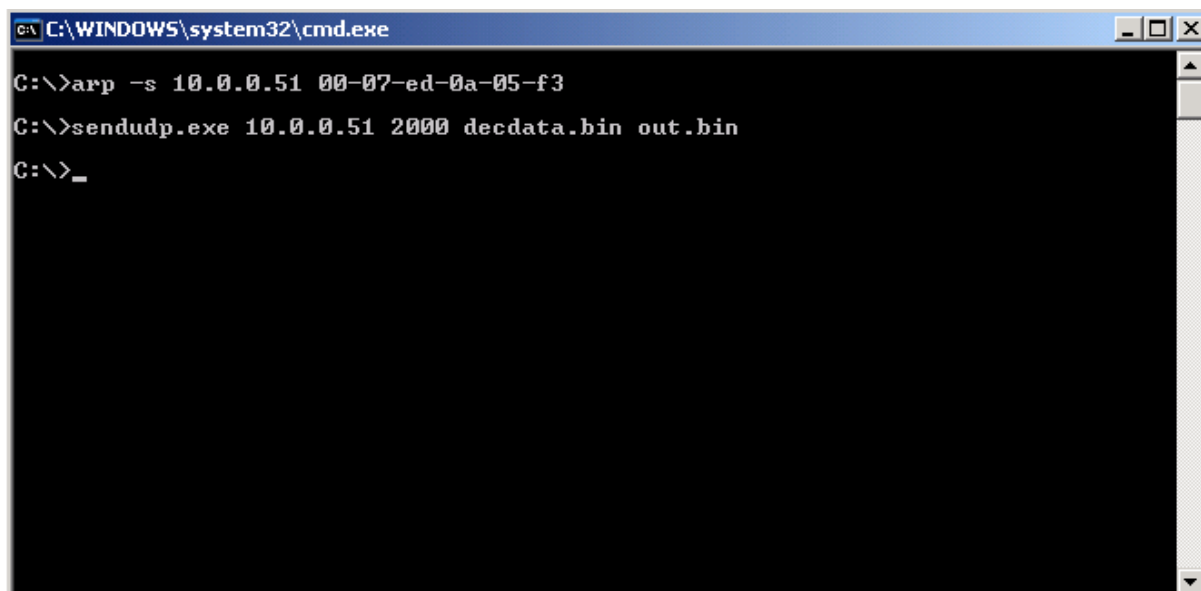
Pomocí zasílání dat přes ethernetové rozhraní lze kódovat rychlostí až 1,7 MB/s.



Obrázek 20: Vytvoření statického záznamu ARP tabulky v prostředí MS Windows.

Obvod	Cyclone	Stratix	Stratix II
LEs (ALUTs)	3,684 / 20,060 (18 %)	3,688 / 10,570 (35 %)	3,607 / 143,520 (3 %)
Bitů paměti	74,352 / 294,912 (25 %)	74,352 / 920,448 (8 %)	74,352 / 9,383,040 (< 1 %)
f_{\max} [MHz]	57.35	58.73	71.25

Tabulka 12: Parametry HW návrhu kodéru



Obrázek 21: Příklad použití programu pro odesílání dat.

9 Výpis obsahu CD-ROM

Na CD se nachází text dokumentu, zdrojové kódy pro Matlab (funkce pro kodér, dekodér, skripty pro vygenerování testovacích vektorů a referenčních dat pro ověření funkce jednotky RS kódu v FPGA), a soubory pro naprogramování FPGA.

Příložené CD má následující adresářovou strukturu:

```
.
|-- matlab/                Popis kodéru a dekodéru v Matlabu, soubory
|                          pro vygenerování referenčních výsledků.
|-- pb_eth_rsdec/          Projekt nástroje DK pro ověření funkce dekodéru.
|  |
|  '-pblaze_eth
|  |
|  |   |-CYCLONE           Bitstream pro desku osazenou obvodem Cyclone
|  |   |-STRATIX           Bitstream pro desku osazenou obvodem Startix
|  |   '-STRATIX_II        Bitstream pro desku osazenou obvodem Stratix II
|-- pb_eth_rsenc/          Projekt nástroje DK pro ověření funkce kodéru.
|  |
|  '-pblaze_eth
|  |
|  |   |-CYCLONE           Bitstream pro desku osazenou obvodem Cyclone
|  |   |-STRATIX           Bitstream pro desku osazenou obvodem Startix
|  |   '-STRATIX_II        Bitstream pro desku osazenou obvodem Stratix II
|-- sendudp/               Zdrojové kódy programu pro odesílání binárních dat
|  |                       pomocí UDP protokolu.
|  '--Release              Spustitelný program a referenční data
|-- text                   Text dokumentu
'-- readme.txt
```

Reference

- [1] Hanzo, L.; Liew, T. H.; Yean, B. L. Turbo Coding, Turbo Equalisation and Space-Time Coding for Transmission over Fading Channels. John Wiley, 2002 ISBN 0-470-84726-3
- [2] Hrdina, Z.; Vejražka, F. Digitální rádiová komunikace. Praha: Vydavatelství ČVUT, 1994. ISBN 80-01-01059-7
- [3] Hlavička, J.; Racek, S.; Golan, P.; aj. Číslicové systémy odolné proti poruchám. Praha: Vydavatelství ČVUT, 1992. ISBN 80-01-00852-5
- [4] Adámek, J. Kódování a teorie informace. Praha: České vysoké učení technické, 1991. ISBN 80-01-00661-1
- [5] Sklar, B.; Harris, F. J. The ABCs of Linear Block Codes. IEEE Signal Processing Magazine, July 2004, Vol. 21, s. 14 – 35, ISSN: 1053-5888
- [6] Němec, K. Protichybové kódové zabezpečení s Bose-Chaudhury-Hocquenhemovými kódy [online]. Dostupné na WWW: <http://www.elektrorevue.cz/clanky/05015>
- [7] Slanina, M.; Říčný, V. Reed-Solomonovy kódy v časové a ve frekvenční oblasti [online]. Dostupné na WWW: <http://www.elektrorevue.cz/clanky/06011>
- [8] Paar, Ch.; Fleischmann, P.; Roelse, P. Efficient Multiplier Architectures for Galois Fields. IEEE Transactions on Computers, February 1998, vol 47, no 2, s. 162 – 170, ISSN: 0018-9340
- [9] Handel-C Language Reference Manual [online]. Dostupné na WWW: <http://www.celoxica.com/techlib/files/CEL-W0410251JJ4-60.pdf>
- [10] IEEE 802.16-2004 [online]. Dostupné na WWW: <http://standards.ieee.org/getieee802/download/802.16-2004.pdf>
- [11] Xilinx. *Embedded Processing* [online]. Dostupné na WWW: <http://www.xilinx.org/>
- [12] Xilinx. *PicoBlaze for CoolRunner-II Application Note* [online]. Dostupné na WWW: <http://www.xilinx.com/bvdocs/appnotes/xapp387.pdf>
- [13] Altera. *Nios Development Board Reference Manual*, Stratix Edition [online]. Dostupné na WWW: http://www.altera.com/literature/manual/mnl_nios2_board_stratix_1s10.pdf
- [14] Altera. *Stratix II EP2S180 DSP Development Board Reference Manual* [online]. Dostupné na WWW: http://www.altera.com/literature/manual/mnl_stx2_pro_dsp_dev_kit_ep2s180.pdf
- [15] SMSC. *SMSC LAN91C111 Datasheet* [online]. Dostupné na WWW: <http://www.smsc.com/main/datasheets/91c111.pdf>
- [16] Altera. *Stratix Device Handbook* [online]. Dostupné na WWW: http://www.altera.com/literature/hb/stx/stratix_handbook.pdf
- [17] Altera. *Stratix-II Device Handbook* [online]. Dostupné na WWW: http://www.altera.com/literature/hb/stx2/stratix2_handbook.pdf