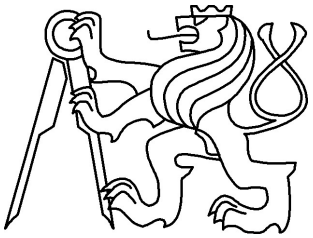
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAHE

FAKULTA ELEKTROTECHNICKÁ



Programové vybavenie testovacej platformy

BAKALÁRSKA PRÁCA

Ondrej Ille

Praha, 2014

Prehlásenie

Prehlasujem, že som predloženú prácu vypracoval samostatne a že som uviedol všetky použité informačné zdroje v súlade s Metodickým pokynom o dodržovaní etických princípov pri príprave vysokoškolských záverečných prác.

V Prahe dňa: ..... Podpis: .....

Kľúčové slová

FlexRay, LIN, CAN, automobilové distribuované systémy, programovateľná testovacia platforma

Poďakovanie

Rád by som sa poďakoval svojmu vedúcemu za vedenie a pomoc pri bakalárskej práci. Tiež by som rád poďakoval Jiřímu Blechovi, ktorý navrhoval vlastné zariadenie a poskytol mi užitočné rady pre programovanie zariadenia.

Abstrakt

Vzhľadom k rastúcemu použitiu distribuovaných systémov v automobilovom priemysle je nutné vyvinúť nové metódy testovania. Realizoval som pripojenie existujúceho FlexRay radiča, v podobe IP funkcie v hradlovom poli, k procesoru na testovacej platforme. Registre radiča som namapoval do adresného priestoru procesora cez rozhranie EMIF a umožnil tak jednoduché ovládanie radiča programom z procesora. Tiež som vytvoril VHDL komponenty zabezpečujúce variabilné pripojenie konfigurovaných radičov na fyzické budiče, zistenie počtu syntetizovaných radičov a zistenie bázových adries pre prácu s radičmi. Celý systém som sprevoznil na vyvíjanej testovacej platforme a implementoval som jednoduché funkcie v jazyku C pre prácu s implementovanými registrami.

Abstract

Due to increased usage of distributed systems in automotive, industry development of new test methods is needed. As first, I implemented connection between IP function of FlexRay controller in FPGA and processor on test platform. Registers of controllers were mapped into adress space of processor by using EMIF bus, enabling simple usage of controller. I created VHDL components enabling variable connection between controllers and physical layer transcievers, finding out number of synthetized controllers and reading base adresses of the controllers. Whole system was successfully tested on platform. and simple functions in C language were developed for control of implemented registers.

Obsah

Zoznam obrázkov

Zoznam tabuliek

1 Úvod

1.1 Úvod

Moderným trendom v automobilovej technike sa v posledných rokoch stáva použitie distribuovaných systémov pre riadenie periférií vo vozidle. Distribuované systémy sú aplikovateľné takmer v každom elektronickom, hydraulickom aj mechanickom systéme ktoré vozidlo obsahuje. Zastaranejší spôsob pripojenia periférií využíva mechanické prepojenie (volant, hydraulické brzdy), a prepojenie pomocou analógového elektrického signálu. Pri pripojení každého modulu cez samostatné vedenie, s rastúcim množstvom pripojených prvkov, rastie celková dĺžka použitých vodičov a tým rastie aj výrobná cena automobilu. Medzi ďalšie problémy patrí väčšia náchylnosť k rušeniu pri analógovom prenose ktoré spôsobuje chyby pri riadení. Pri digitálnom prenose so správnou voľbou prenosového formátu, spôsobu prístupu k médiu a riešením kolízií je chybovosť nižšia. Diagnostika týchto systémov je časovo menej náročná, je možné pripojiť sa cez merací modul a vyslať do uzlov dotazy na stav jednotiek a na základe odpovedí ľahko zistiť chybu. Je len málo oblastí, v ktorých "klasický" prístup porazí digitálnu komunikáciu v cene alebo spoľahlivosti (napr. elektronicky pripojené riadenie volantu je menej spoľahlivé ako priama mechanická väzba). Z uvedených dôvodov sa použitie distribuovaných systémov uprednostňuje a otvára veľa možností vývoja.

1.2 Vyvíjaný systém

S použitím distribuovaných systémov v automobiloch vzniká potreba správnej a rýchlej diagnostiky, ako v pri servise vozidla, tak aj pri výrobných testoch vozidla. Testovanie komunikácie každého uzlu pomocou fyzického pripojenia riadeného elementu je zdĺhavé a neefektívne. Preto je vhodné vytvoriť virtuálny uzol ktorý môžeme pripojiť

na miesto skutočného, konfigurovať ho, a testovať komunikáciu týmto spôsobom. Za predpokladu, že fyzická funkčnosť modulu je zaručená výrobcom, je takéto testovanie dostatočné a efektívne. V automobilovej technike dominujú tri zbernice, FlexRay, CAN a LIN. V tejto práci sa zameriame predovšetkým na zbernicu FlexRay, ale s myšlienkou neskoršieho rozšírenia systému aj pre zbernice LIN a CAN. Účelom je vytvoriť zariadenie ktoré bude pripojené na zbernicu a bude sa vedieť správať ako požadovaný uzol, radič. Radiče FlexRay budú syntetizovateľné do hradlového poľa a konfigurovateľné cez pamäťový priestor uživateľského rozhrania.

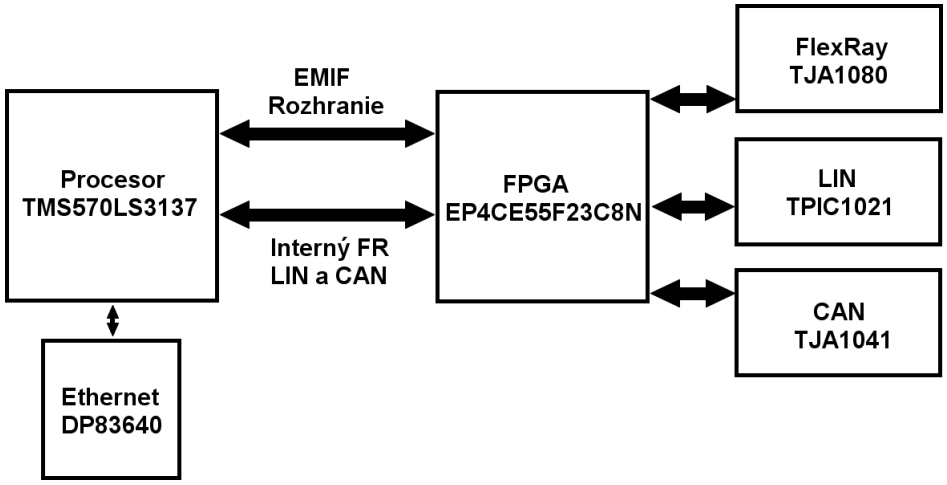
Do hradlového poľa sa z užívateľského prostredia nahrajú viaceré radiče FlexRay. Účelom mojej práce je vytvoriť systém, ktorý ponúkne prácu s IP funkciami radičov a ich vnútornými registrami bez priamej väzby na hardwarovú úroveň systému. Je potrebné vytvoriť sadu vzájomne prepojiteľných komponent (zabezpečujúce funkcie v bodoch 3 a 4) , definovať jednoduché rozhranie, po ktorom budú komunikovať, a zabezpečiť v prvom rade funkčnosť komponent na hardwarovej úrovni (viz Kapitola 7, Testovanie systému). Tieto komponenty musia spĺňať základné vlastnosti:

1. Ovládateľnosť zápisom do pamäte v užívateľskom prostredí (v programe pre procesor v jazyku C).

2. Kompatibilita s vlastným radičom FlexRay (radič ako IP funkcia prevzatý z [1]), teda mapovanie registrov radiča do rovnakého pamäťového priestoru ako vytvárané komponenty.

3. Možnosť vyčítať počet radičov FlexRay syntetizovaných v hradlovom poli a ich bázové adresy.

4. Možnosť prepojiť výstupné kanály radičov na vstupy viacerých fyzických budičov (viz 1.3).



Obr. 1.1: Blokové schéma produktu

1.3 Použité prostriedky

Fyzické zariadenie tvorí doska plošných spojov s 32 bitovým procesorom Texas Instruments (TMS570LS3137), hradlovým poľom FPGA (EP4CE55F23C8N), budiče fyzickej vrstvy (4xTJA1080 FlexRay, 4xTJA1041 Lin, 4xTPIC1021 CAN) a rozhranie Ethernet (DP83640). Blokové schéma výrobku sa nachádza na Obrázku 1.1. Procesor obsahuje aj vlastný radič FlexRay a cez FPGA sa dá pripojiť na fyzické budiče. Registre programovateľných radičov sú mapované do pameťového priestoru procesora a pristupuje sa k nim cez paralelné 16 bitové EMIF rozhranie.

1.4 Vývojové prostredie

Ako vývojové prostredie je použitý Quartus II verzia 9.1. Bloky radičov a registrov sú spracované v jazyku VHDL a prepojenie jednotlivých blokov je riešené v symbolickom editore Quartusu. Na testovanie je použitý ModelSim verzia 6.5b. Pre praktickú syntézu a testovanie obvodov je potom použitý vývojový kit Altera Cyclone II Starters Kit.

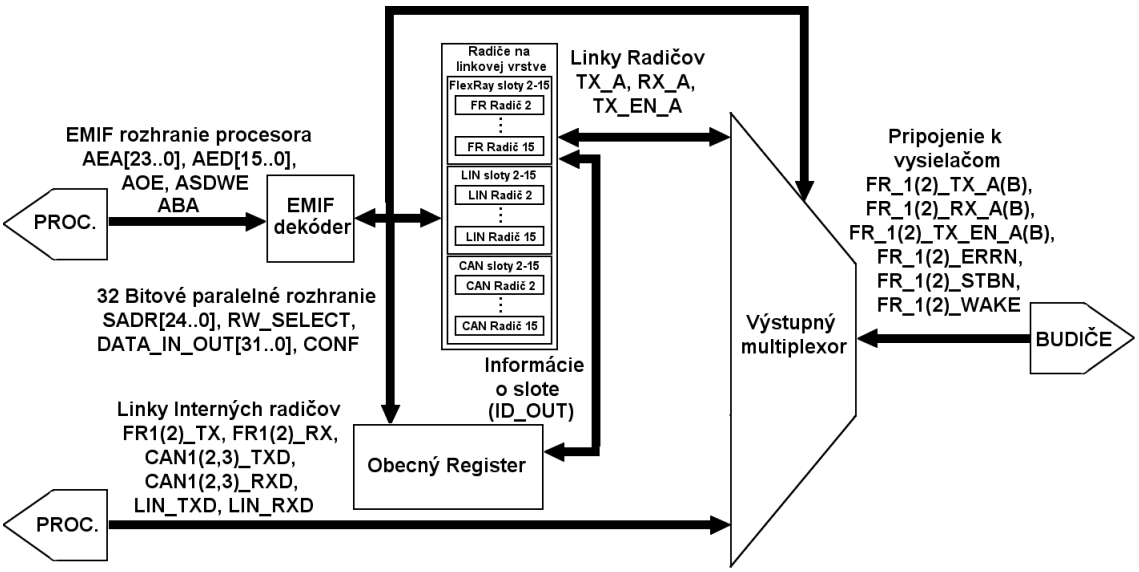
Primárnym nástrojom v overovaní funkčnosti obvodu je však Quartus Simulator Tool, ktorý umožňuje nadefinovať priebeh vstupných signálov a pozorovať výstupné signály. Simulácia je realizovaná tzv. po syntéze a započítava aj oneskorenie každého hradla a je primárnym spôsob testovania celého systému.

Na programovanie procesora Texas Instruments je použité vývojové prostredie Code Composer Studio. Konfigurácia FPGA je programovateľná cez USB Blaster (v budúcnosti plánované programovanie pomocou procesora) a nemení sa počas používania výrobku. Na programovanie procesora je použitý programátor s rozhraním JTAG. Na analýzu a testovanie som použil programový logický analyzátor Signal Tap II dostupný v Quartuse. Na vlastnú konfiguráciu procesora som potom použil program HalCoGen.

2 Štruktúra Systému v FPGA

2.1 Popis Systému

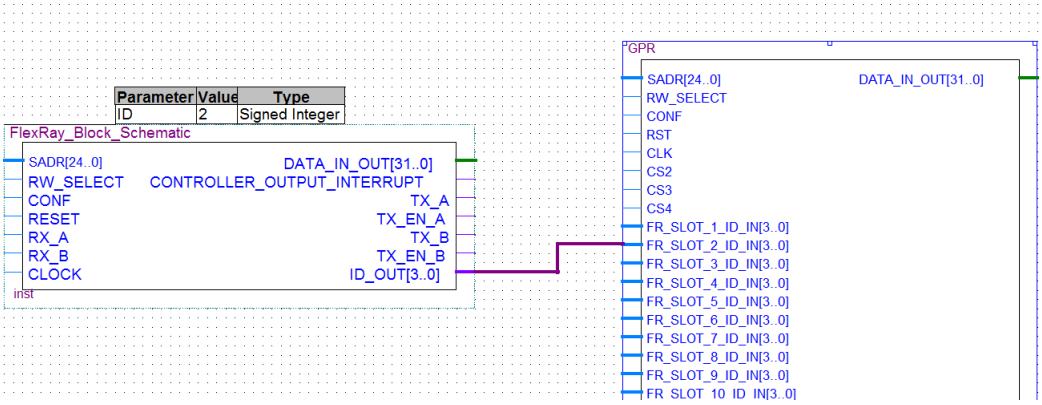
Obvod vytvorený v jazyku VHDL a symbolickom editore programu Quartus II sa skladá z komponent: FlexRay radič (viz Kapitola 3.), Obecný register (viz Kapitola 4.), Výstupný Multiplexor (viz Kapitola 5.), EMIF dekodér (viz Kapitola 6.). Blokové schéma systému spolu s názvami rozhraní, ktoré komponenty spájajú, sa nachádza na Obrázku 1.1 (už aj s uvažovaným rozšírením pre ďalšie zbernice). Popisy signálov rozhraní sa nachádzajú v nasledovných podkapitolách. S komponentmi sa pracuje ako s blokovými symbolmi v Quartuse. Komponenta FleRay radiča sa ďalej skladá z podkomponent ako napr. Injector, Controller Core. Adresa každej komponenty je určená samostatnou skupinou 4 bitov. Podrobný popis adresných bitov sa nachádza v podkapitole 2.4.



Obr. 2.1: Blokové schéma systému v hradlovom poli

2.1.1 ID Radičov, Sloty a Zapojenie radičov do Obecného registra

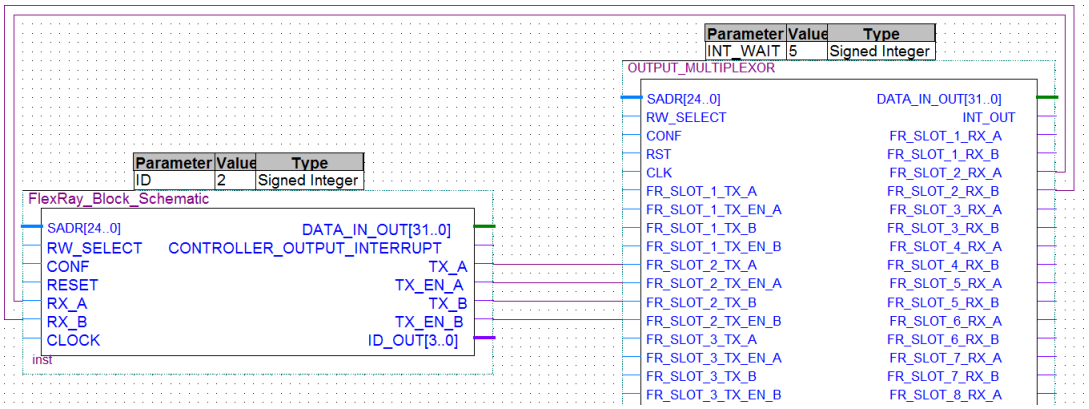
FlexRay radiče sú jednoznačne identifikované podľa ID, ktoré je možné meniť pre každý blok radiča v Quartuse, ako na Obrázku 3.1. ID musia byť vzájomne rozdielne a nadobúdať hodnoty od 1 do 15. Hodnota ID=0 je rezervovaná pre nepripojený radič (viz. ďaľej). ID radiča tvorí jeho unikátny identifikátor. ID je ako std_logic_vector z radiča vyvedené v podobe výstupu ID_OUT (opäť viz. Obrázok 3.1). Pomocou tohto ID komponenta Obecného registra rozoznáva o ktorý radič sa jedná. Na zoradenie radičov sú použité Sloty, číslované od 1 do 15. Jedná sa o virtuálny pojem, ktorý vyjadruje to do ktorého vstupu FR_SLOT_X_ID_IN komponenty Obecného registra (viz. Obrázok 4.1) je pripojený výstup ID_OUT. Napríklad pripojením ID_OUT radiča do vstupu FR_SLOT_1_ID_IN zaraďujeme radič do Slotu 1. Pri práci s obecným registrom potom čítame bázové adresy radičov v slotoch. Hodnotu ID je vhodné pre prehľadnosť voliť rovnakú ako hodnotu Slotu, do ktorého radič zapájame. Signály ID_OUT potom tvoria bity 15 až 12 bázovej adresy radiča (viz Tabuľka2.7). Pre názornosť je zobrazené zapojenie výstupu ID_OUT na nasledovnom Obrázku 2.2.



Obr. 2.2: Zapojenie výstupov ID_OUT

2.1.2 Zapojenie radičov do Výstupného multiplexora

Ďalej sa linky radiča TX_A(B), TX_EN_A(B), RX_A(B) zapoja do výstupného multiplexora tak, že sú vždy pripojené do portov FR_SLOT_X_TX_A(B), FR_SLOT_X_TX_EN_A(B), FR_SLOT_X_RX_A(B) podľa toho do akého Slotu sme ich zapojili pri Obecnom registri. Prepájanie liniek budičov na fyzické porty sa potom realizuje zápisom do registrov Výstupného multiplexora. K správnej funkčnosti systému je potrebné, aby sa dodržiavalo číslovanie slotov! Radič s ID =1 zapájame do slotu 1 atď... Pre názornosť je zobrazené zapojenie na Obrázku 2.3.



Obr. 2.3: Zapojenie FlexRay radiča do Výstupného multiplexora

2.1.3 Zapojenie interných radičov procesora

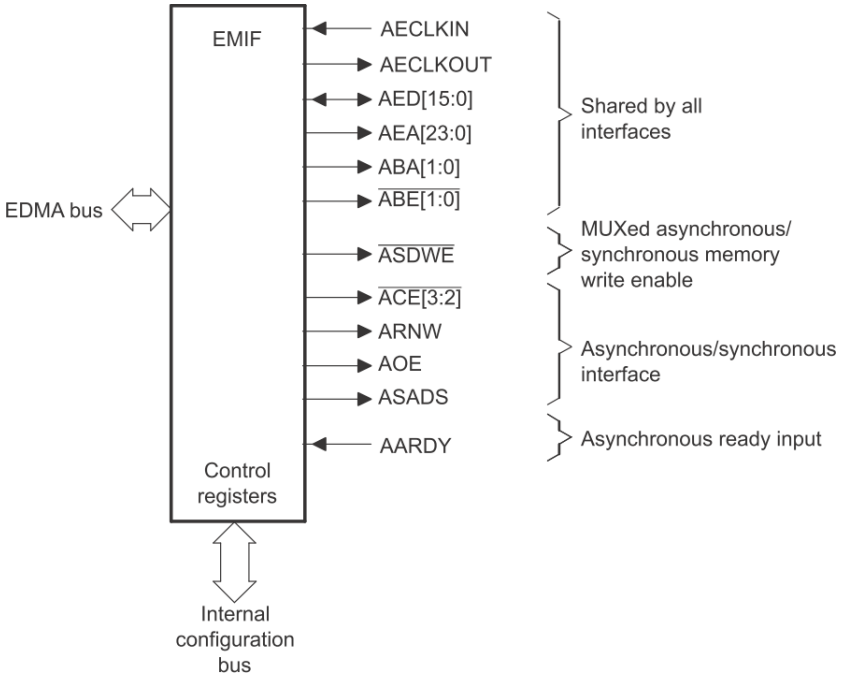
Okrem programovateľných radičov v FPGA je možné pracovať aj s radičmi v procesore, ktorý obsahuje jeden FlexRay radič s dvoma kanálmi, tri CAN radiče a jeden LIN radič. Konfigurácia stavových automatov týchto radičov v systéme nie je, je potrebné ich ovládať z procesora (viz. [13] a [14]) . Výstupné linky týchto radičov sú privedené na vstupy FPGA a sú prepojené do portov FR_INTERNAL_TX, FR_INTERNAL_TX_EN, FR_INTERNAL_RX, LIN_INTERNAL_RXD, LIN_INTERNAL_TXD, CAN_INTERNAL_TXD, CAN_INTERNAL_RXD komponenty Výstupného multiplexora. Pomocou registrov FR_INTERNAL,

LIN_INTERNAL, CAN_INTERNAL vo výstupnom multiplexore môžeme ovládať prepojenie týchto radičov na fyzické budiče. Do Obecného registra tieto radiče pripojené nie sú, pretože ich bázová adresa je konštantná a je možné ju z obecného radiča vyčítať bez ďalších informácií.

2.2 EMIF Rozhranie

Rozhranie EMIF zabezpečuje komunikáciu medzi procesorom a EMIF dekodérom, ktorý je implementovaný v FPGA. Jedná sa o zbernicu s 24 adresnými linkami a 16 dátovými linkami. Adresný priestor zbernice teda umožňuje používať 2^{24}adries. V implementovanom systéme je dostatok adries na pokrytie všetkých registrov. Preto zvolené adresovanie z pohľadu užívateľa, procesora, (viz 2.4) nie je volené kompaktne ale viac prehľadne. EMIF je poloduplexná zbernica, používa rovnaké linky na čítanie ako na zapisovanie. Umožňuje synchrónnu aj asynchrónnu komunikáciu (viz [5]). V našom systéme je použité asynchrónne pripojenie pamäte. Rozhranie umožňuje komunikáciu po 8 bitoch a po 16 bitoch, v našom systéme je použitá 16 bitová komunikácia. Na zápis 32 bitovej hodnoty sú použité 2 po sebe idúce 16 bitové zápisy rovnako ako na čítanie. Výstup EMIF z procesora je zobrazený na Obrázku 2.4 a popis signálov je na v Tabuľke 2.1. Podrobný spôsob prekladu adries procesora na adresné linky EMIF je na Obrázku 2.2. Kompatibilitu medzi EMIF a 32 bitovým paralelným rozhraním, na ktoré sú pripojené všetky linky obvodu zabezpečuje EMIF dekodér (viz. Kapitola 6).

Posledný bit Adresy procesora sa potom nachádza v signáli ABA[1]. V paralelnom 32 bitovom rozhraní (viz 2.3) sú signály AEA a ABA opäť zložené, a spodných 24 bitov adresy procesora odpovedá spodným 24 bitom adresy SADR(24..0) . Pretože radiče neposkytujú možnosť práce zo 16 alebo 8 bitmy je distribúcia spodných dvoch bitov zbytočná z hľadiska implementácie, ale potrebná z hľadiska simulácie. V prípade vynechania týchto bitov by sa všetky vyššie bity o dva posunuli a zadávanie dát do



Obr. 2.4: Rozhranie EMIF v procesore (prevzaté z [5] str. 11)

simulácií by bolo neprehľadné.

V procesore je pre asynchrónnu pamäť pripojenú na EMIF vyhradených až 3*16 MB adresného priestoru. Každý priestor má vlastný výberový signál (CS2, CS3 a CS4). Všetky CSn signály sú z procesora vyvedené do FPGA a je potrebné ich zapojiť na rovnako pomenované vstupy Informačného registra. Pri čítaní bázových adries radičov z Informačného registra sú potom vrátené adresy, ktorých horných 12 bitov zodpovedá použitému pamäťovému priestoru. Tiež je potrebné všetky tri CSn signály zapojiť do EMIF dekodéra a v jeho generickej premennej zvoliť hodnotu (2 pre CS2, 3 pre CS3, 4 pre CS4) podľa použitého pamäťového podpriestoru. EMIF dekodér potom bude pracovať iba s vybraným pamäťovým priestorom a nebude reagovať na zápis do ostatných priestorov.

EMIF zbernicu môžeme (podľa [5]) používať vo viacerých módoch. V našej implementácii pracujeme s asynchrónnym módom (Strobe) a s asynchrónnym módom s po-

Názov	Popis
CLK	Hodinový signál
AED[15:0]	16 bitová datová linka
AEA[23:0]	Adresná linka
ABA[1:0]	Adresné výstupy najnižších 2 bitov pre 16 a 8 bitový prenos
ABE[1:0]	V nule aktívne signály bajtového výberu.
ACEn	V nule aktívny výberový signál Cen
AARDY	Vstup použitý na vkladanie čakacích cyklov pre pomalé zariadenia
ARNW	Výber Zápis/čítanie pre asynchrónnu komunikáciu
AOE	Aktívny v nule, potvrdenie výstupu pre asynchrónnu komunikáciu
ASDWE	Aktívny v nule, impulz zápisu pre asynchrónnu komunikáciu

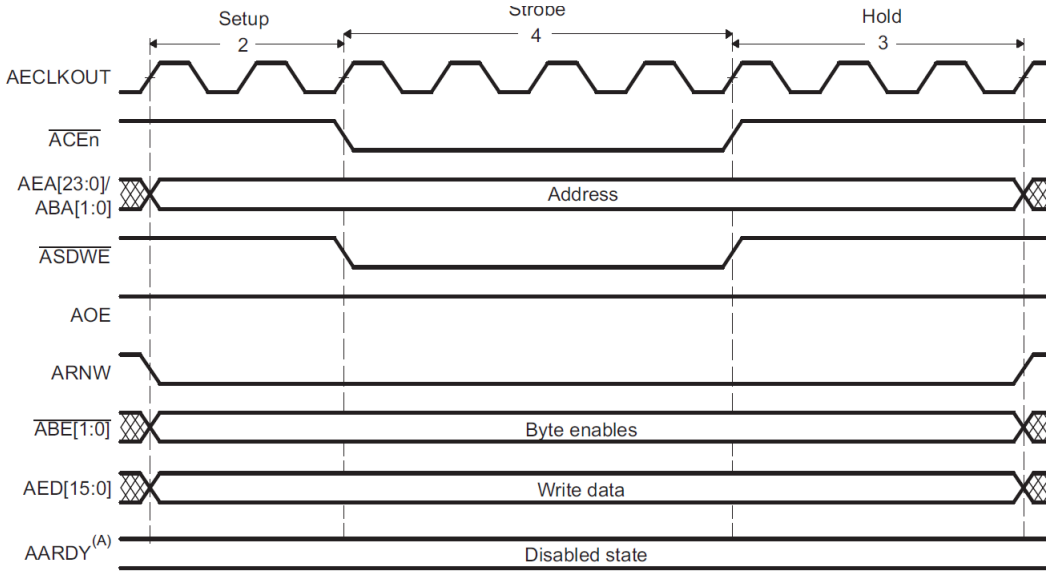
Tabuľka 2.1: EMIF Signály

Šírka Pamäťe	Maximálny počt Adresných Jednotiek na priestor jedného výberového signálu	Preklad adresy procesora na adresu EMIF	Reprezentuje
x8	64M	AEA[23:0] = A[25:2] ABA[1:0]=A[1:0]	Adresu Bajtu
x16	32M	AEA[23:0] = A[24:1] ABA[1:0]=A0	Adresu Pol-Slova

Tabuľka 2.2: Preklad adries procesora na Adresy EMIF

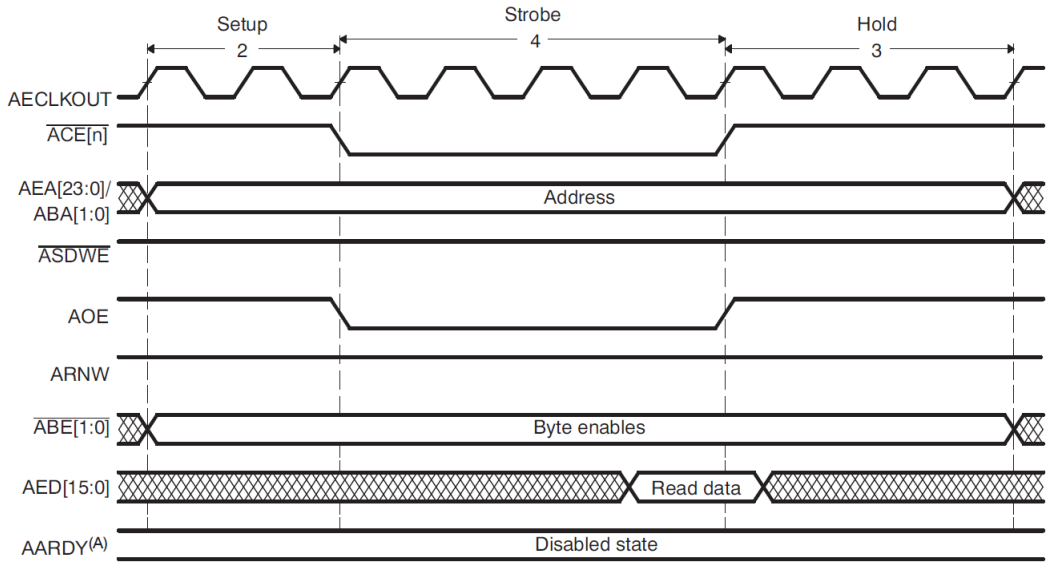
užitím vstupu “AARDY”. Nastavenie EMIF dekodéra pri použití módov je popísané v Kapitole 6. Časové diagramy pre každý z módov sú zobrazené v obrázkoch v nasledovných podkapitolách.

2.2.1 Asynchrónny zápis v móde “Strobe”



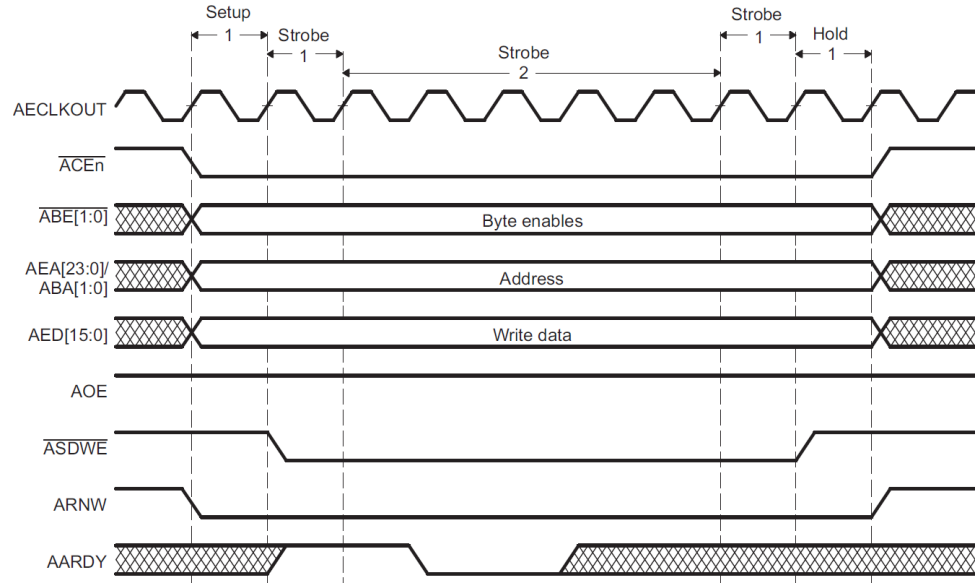
Obr. 2.5: Asynchrónny zápis v móde “Strobe” (prevzaté z [5])

2.2.2 Asynchrónne čítanie v móde “Strobe”



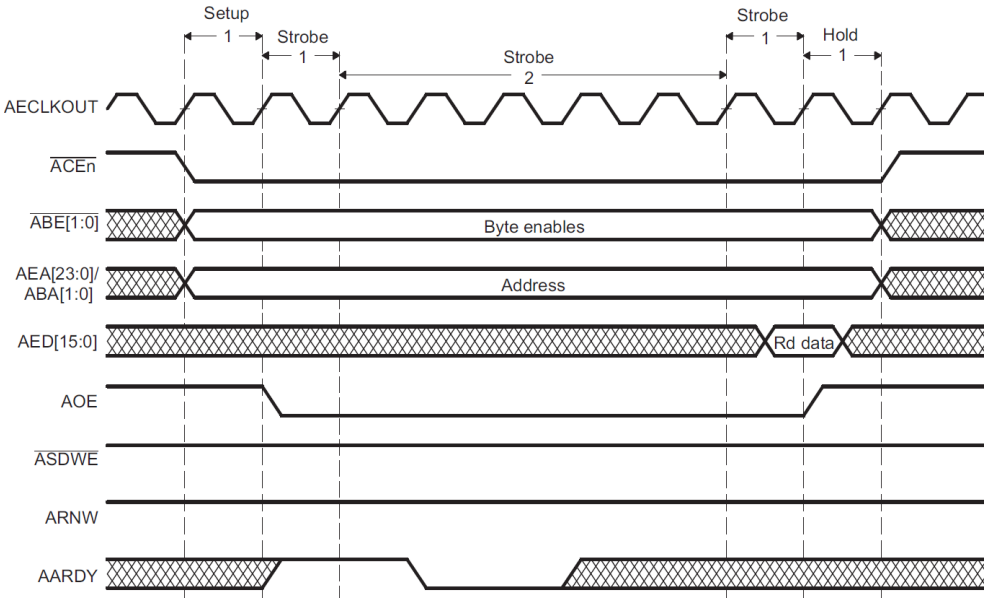
Obr. 2.6: Asynchrónne čítanie v móde “Strobe” (prevzaté z [5])

2.2.3 Asynchrónny zápis s použitím AARDY signálu



Obr. 2.7: Asynchrónny zápis s použitím AARDY signálu (prevzaté z [5])

2.2.4 Asynchrónne čítanie s použitím AARDY signálu



Obr. 2.8: Asynchrónne čítanie s použitím AARDY signálu (prevzaté z [5])

2.3 32 bitové paralelné rozhranie

32 bitové paralelné obojsmerné rozhranie zabezpečuje komunikáciu medzi EMIF dekodérom a komponentami systému ako FlexRay kontroléry, Obecný register a Výstupný multiplexor. Komponenty FlexRay radiča (Injector, Trigger...) sú v pôvodnej implementácii [1] kompatibilné s procesorom NIOS II a jeho zbernicou Avalon, preto je nami implementovaná komunikácia prispôsobená tomuto rozhraniu. Rozdielom oproti rozhraniu použitom v [1] je v dátových linkách a vynechaniu Výberových signálov . Namiesto Výberového signálu slúži adresa (SADR (24..0)) ako unikátny identifikátor obvodu. Je použitá iba jedna dátová linka, ktorá je obojsmerná (DATA_IN_OUT(31..0)). Zachovaný je asynchrónny reset, ktorý prenastaví registre v obvodoch na implicitné hodnoty. Ako signá,l ktorým sa spúšťa zápis alebo čítanie, je použitý signál CONF, ktorého logická jednotka po nastavenej adrese ukladá resp. číta do resp. z obvodu dáta. Popis signálov je v Tabuľke 2.3. Riadenie komunikácie zabezpečuje EMIF Dekodér (viz Kapitola 6.) na základe stavového automatu, ktorý je tiež popísaný v tejto kapitole. Prevod medzi 32 bitovým paralelným rozhraním a rozhraní v [1] je zabezpečený komponentami Adresného dekodéra a Dátového prepínača, obe sú popísané v Kapitole 3. Na fyzickej vrstve v jazyku VHDL je použitý prístup k datovej linke pomocou 3 stavového výstupu, aby sa zamedzilo kolíziám medzi jednotlivými komponentami.

Meno	Funkcia	Smer
SADR [24..0]	Adresa	Vstupný
RW_SELECT	Výber Zápis (Log 1) , Čítanie (Log 0)	Vstupný
CONF	Potvrenie platnosti dát a adresy (Log 1)	Vstupný
DATA_IN_OUT[31..0]	Obojsmerná datová linka	Obojsmerný
RST	Asynchrónny reset	Vstupný
CLK	Hodinový vstup	Vstupný

Tabuľka 2.3: Signály 32 bitového paralelného rozhrania

2.3.1 Zápis na 32 bitovom rozhraní

Zápis na 32 bitovom rozhraní je realizovaný nasledovným signálovým sledom:

1. Na začiatku zápisu:
   - SADR sa nastaví na platnú hodnotu
   - DATA_IN_OUT sa nastaví na platnú hodnotu
   - CONF sa stane aktívny (aktívny v jednotke)
   - RW_SELECT sa nastaví na logickú 1.
2. Počas priebehu zápisu:
   - Čítač čaká vopred nastavený počet hodinových cyklov, dokiaľ sa dáta nezapíšu.
3. Na konci zápisu:
   - SADR sa nastaví na neplatnú hodnotu (vysoká impedancia)
   - DATA_IN_OUT sa nastaví na neplatnú hodnotu (vysoká impedancia)
   - CONF sa stane neaktívny (aktívny v jednotke)
   - RW_SELECT sa nastaví neplatnú hodnotu (vysoká impedancia)

2.3.2 Čítanie na 32 bitovom rozhraní

Čítanie na tejto linke je realizované nasledovným signálovým sledom:

1. Na začiatku čítania:

- -SADR sa nastaví na platnú hodnotu
- -DATA_IN_OUT sa nastaví na neplatnú hodnotu
- -CONF sa stane aktívny (aktívny v jednotke)
- -RW_SELECT sa nastaví na logickú 0.

2. Počas priebehu zápisu:

- -Čítač čaká vopred nastavený počet hodinových cyklov, dokiaľ obvod nedoručí dáta na linku DATA_IN_OUT

3. Po doručení dát:

- -Čítač čaká predom nastavený počet hodinových cyklov, dokiaľ dáta nie sú spracované (v našom systéme EMIF dekodérom, ktorý ich ďalej posiela cez EMIF rozhranie do procesora)

4.Po uplynutí doby čakania na odber načítaných dát:

-SADR sa nastaví na neplatnú hodnotu (vysoká impedancia)

-DATA_IN_OUT sa nastaví na neplatnú hodnotu (vysoká impedancia)

-CONF sa stane neaktívny (aktívny v jednotke)

-RW_SELECT sa nastaví na neplatnú hodnotu (vysoká impedancia)

2.4 Adresovanie v systéme

Obvody v procesore sú pripojené do EMIF priestoru medzi bázové adresy 0x60000000 a 0x6FFFFFFF. Význam jednotlivých bitov adresy sa nachádza v Tabuľke 2.4. Hodnoty adresy pre cieľovú komponentu sú v nasledujúcich sekciách.

ADR [31:24]	ADR [23:20]	ADR [19:16]	ADR [15:12]	ADR [11:8]	ADR [7:0]
Podľa použitého CSn signálu	Druh Komponenty	ID Komponenty	Časť komponenty	Vnútorné registre komponent	

Tabuľka 2.4: Adresovanie v procesore

2.4.1 Bity 31 až 24

Pamäťový priestor procesora pre periférie pripojené cez EMIF je rozdelený do troch podpriestorov, z ktorých každý je vyberaný vlastným Výberovým signálom podľa použitých hodnôt v Tabuľke 2.5. Je treba si uvedomiť, že napriek tomu, že zvolený Interface

môže adresovať až 3*16 MB v našej implementácii sa zaobídeme iba s 16MB a preto stačí používať iba jednu hodnotu pre najvyšších 8 bitov a k nej patriaci výberový signál pri zapájaní obvodu v FPGA.

ADR [31:24]	
Hodnota	Význam
0x00 až 0x63	CS2
0x64 až 0x67	CS3
0x68 až 0x6B	CS4
0x6C až 0x6F	Rezervované

Tabuľka 2.5: Význam bitov 31 až 24

2.4.2 Bity 23 až 20

Bity 23 až 20 udávajú druh komponenty, do ktorej chceme zapisovať. Použiteľné hodnoty sú uvedené v Tabuľke 2.6.

ADR [23:20]	
Hodnota	Význam
0x1	Obecný Register Slotov
0x2	Výstupný multiplexor
0x3	FlexRay Radič
0x4	CAN radič
0x5	LIN radič

Tabuľka 2.6: Význam bitov 23 až 20

2.4.3 Bity 19 až 16

Bity 19 až 16 sú použité pri zápise do vlastných radičov zberníc a udávajú ID radiča, do ktorého sa zapisuje. ID radiča je zadané ako generická hodnota pri vytváraní schémy v Quartuse. Keďže navrhnutý systém umožňuje zvoliť si ID v rozsahu 1 až 15 a následne pripojiť výstup ID_OUT do slotov 1 až 15, pre prehľadnosť je vhodné voliť hodnotu ID rovnakú, ako je hodnota Slotu, do ktorého ho pripájame. Bity 19 až 16 adresy radiča potom nadobúdajú rovnakú hodnotu v decimálnej sústave ako číslo slotu, do ktorého

je tento radič pripojený, a teda pri znalosti zapojenia radiča do Slotu nebude nutné používať obecný register pre vyčítanie jeho bázovej adresy, všetky bity okrem bitov 19 až 16 sú totiž pre radič zrejmé z ostatných tabuliek.

ADR [19:16]	
Hodnota	Význam
0x1	Rezervované
0x2	ID = 2
...	...
0xF	ID = 15

Tabuľka 2.7: Význam bitov 19 až 16

2.4.4 Bity 15 až 12

Bity 15 až 12 sú zatiaľ v systéme implementované iba pre FlexRay radič, a udávajú časť radiča, ku ktorej registrom chceme pristupovať. Hodnoty týchto 4 bitov kopírujú rozdelenie hodnôt v [1] (str. 23.) na pozícii najvyšších bitov. Hodnoty sú uvedené v Tabuľke 2.8.

ADR [15:12] - Iba pre FlexRay	
Hodnota	Význam
0x4	Jadro Radiča
0x5	Trigger
0x6	Generátor časových značiek
0x7	Injector
0x8	Data Logger
0x9	Multiplexor

Tabuľka 2.8: Význam bitov 15 až 12

2.4.5 Bity 11 až 0

Bity 11 až 0 sú určené pre adresovanie vnútorných registrov každej komponenty či už radičov, Obecného Informačného registra alebo Výstupného Multiplexora. Adresovanie prebieha po 32 bitoch a teda každá ďalšia adresa je na pozícii o 0x4 vyššie ako

predchádzajúca. Keďže použité radiče umožňujú iba prácu s celými 32-bitovými hodnotami, nie je možné adresovať priestor medzi týmito adresami. Implementovaný systém (v kompatibilite s [1]) nepoužíva najspodnejšie dva bity adresy prichádzajúcej z procesora, a preto pri prístupe napr. na adresu s offsetom 0x9,0xA,0xB od bázovej adresy (namiesto správneho 0x8) sa bude stále pristupovať k celému 32 bitovému registru na adrese 0x8. Na prístup k nasledujúcemu registru je použitá až adresa s 0ffsetom 0xC. Význam týchto bitov je individuálny, pre nami implementované komponenty je vysvetlený v mapách registrov v nasledujúcich kapitolách a pre Komponenty FlexRay radiča v [1].

2.5 Postup zapojenia systému

Pre správne pripojenie systému je potrebné poznať schému dosky, resp. na ktoré vstupy FPGA sú privedené požadované výstupy procesora a z ktorých výstupov FPGA sú vedené signály na fyzické budiče. Je možné vytvoriť v quartuse tzv. Pin Assignments a ďalej pracovať iba s priradenými menami signálov, alebo pracovať priamo s číslom pinu v tvare : (písmeno),(číslo), kde písmeno a číslo udávajú pozíciu pinu v BGA puzdre. Schéma dosky sa dá nájsť v [2] a zoznam dôležitých pripojených signálov je v prílohe B. Na realizáciu jednoduchého systému v Quartuse postupujeme nasledovne:

1. Vložíme bloky: EMIF dekodér, Obecný register, Výstupný multiplexor a FlexRay radič.

2. Pripojíme signály vstupné signály EMIF dekodéra na input piny v Quartuse. Jedná sa o signály : AED[15..0], CS2, CS3, CS4, ACE, ABE[3..0], AEA [23..0], ABA [1..0], AOE, ASDWE, RNW. Ako mená vstupných pinov použijeme mená odpovedajúce menám vstupov EMIF dekodéra v Prílohe B.

3. Pripojíme 32 bitovú paralelnú zbernicu a to nasledovne: Port DATA_IN_OUT

EMIF dekodéra pripojíme na rovnako menné vstupy v komponentách FlexRay radiča, Obecného registra, Výstupného multiplexora. Rovnako postupujeme pri ostatných signáloch 32 bitového paralelného rozhrania, ktoré sú popísané v 2.3.

4. Nastavíme ID Flexray radiča a jeho výstup ID_OUT zapojíme do Obecného registra do vstupu FR_SLOT_X_ID_IN kde X je rovné hodnote ID.

5. Signály CS2,CS3,CS4 z bodu 2 tiež pripojíme do rovnako menných vstupov obecného registra.

6. Linky FR_TX_A(B), FR_TX_EN_A(B), FR_RX_A(B) pripojíme do Výstupného multiplexoru do portov FR_SLOT_X_TX_A(B), FR_SLOT_X_TX_EN_A(B), FR_SLOT_X_RX_A(B), kde X číslo slotu resp. hodnota ID zvolená vo FlexRay radiči.

7. Pripojíme výstupný multiplexor na piny v Quartuse, vstupné na vstupné, výstupné na výstupné, ktoré budú pripojením Výstupného multiplexora na fyzické budiče. Pre FlexRay sa jedná o signály Výstupného multiplexora: FR_1(2)_A(B)_TX, FR_1(2)_A(B)_TX_EN, FR_1(2)_A(B)_EN, FR_1(2)_A(B)_BGE, FR_1(2)_A(B)_STBN, FR_1(2)_A(B)_WAKE, FR_1(2)_A(B)_RX, FR_1(2)_A(B)_RX_EN, FR_1(2)_A(B)_ERRN. Pre CAN sú to signály: CAN_1(2,3,4)_RXD, CAN_1(2,3,4)_ERRN, CAN_1(2,3,4)_WAKE, CAN_1(2,3,4)_STB, CAN_1(2,3,4)_TXD, CAN_1(2,3,4)_EN. Pre LIN sa jedná o signály: LIN_1(2,3,4)_TXD, LIN_1(2,3,4)_WAKE, LIN_1(2,3,4)_EN, LIN_1(2,3,4)_RXD. Ako mená výstupných Quartus pinov opäť volíme rovnako menné signály z Prílohy B.

8. Pripojíme signály výstupného Multiplexora : FR_INTERNAL_A(B)_RX, FR_INTERNAL_ A(B)_TX, LIN_INTERNAL _1_RX, LIN_INTERNAL_1_TX, CAN_INTERNAL_1(2,3)_RX, CAN_INTERNAL_1(2,3)_TX na vstupné a výstupné quartusovské piny. Názvy pinov znova volíme podľa prílohy B.

9. Ak chceme využiť možnosť zachytávať prerušenie fyzickej vrstvy, pripojíme na vstupné piny do procesora výstup FR_INTERRUPT_OUT.

10. Zapojíme hodinové signály CLK, všetky obvody by mali mať jeden zdroj hodinového signálu. Tiež môžeme zapojiť signály asynchrónneho resetu do procesora.

11. Všetky nepoužité porty je vhodné potom pripojiť na zem (GND), alebo ich nechať nezapojené a nastaviť v Quartuse ich automatické uzemnenie.

12. Výsledná schéma v Quartus potom môže vyzerať ako príklad na priloženom CD.

2.6 Príklad adresovania

Pre názornú ukážku adresovania komponent v systéme (viz ďalšie Kapitoly a [1]) použijeme následovný príklad:

2.6.1 Príklad 1

Máme zapojený systém, v ktorom je obecný register, výstupný multiplexor a tri výstupné radiče zapojené do slotov 2,3,4 s ID=2,3,4. Chceme systém adresovať do priestoru 0x60000000 až 0x63000000. Použitý bude teda signál CS2. Do políčka EMIF, USED_CHIP_SELECT dáme 2. Jednotlivé komponenty v systéme potom majú adresy:

0x60100000 - Bázová adresa (BA) Obecného registra.

0x60200000 - BA Výstupného Multiplexora.

0x60320000 - BA FlexRay kontroléru v Slote 2.

0x60324000 - BA Jadra FlexRay kontroléru v Slote 2.

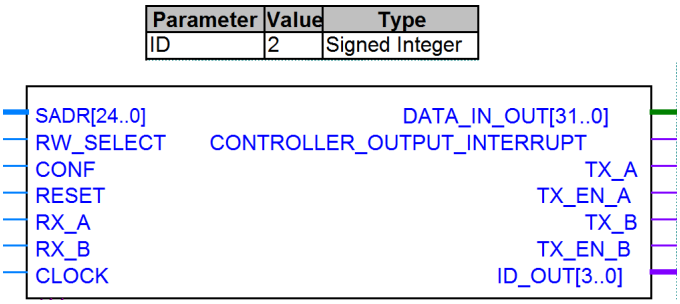
0x60327000 - BA Injectoru FlexRay kontroléru v Slote 2.

0x60326000 - BA Generátora časových značiek FlexRay kontroléru v Slote 2.

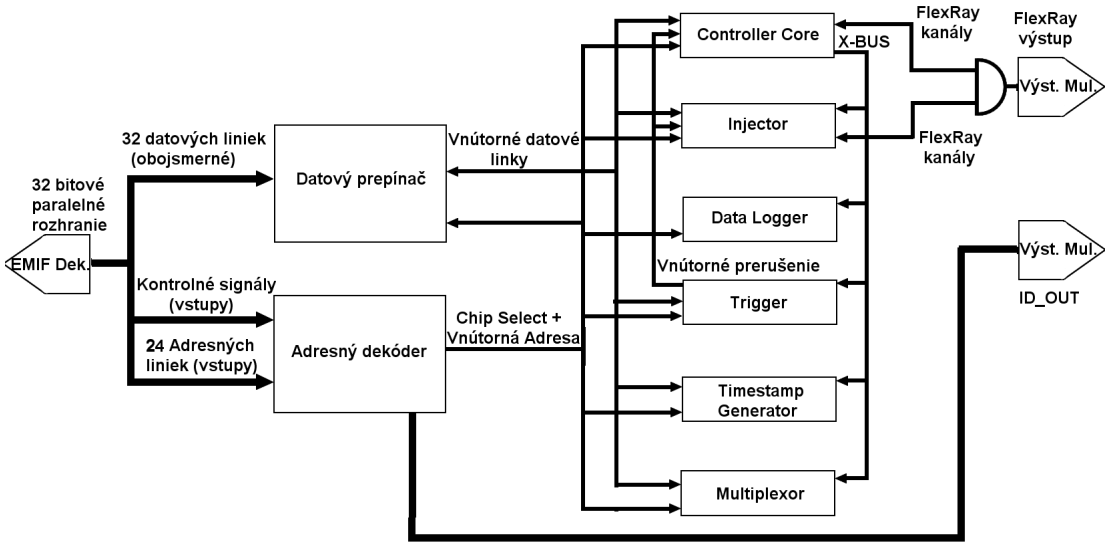
3 FlexRay radič

3.1 Popis radiča

Programovateľný FlexRay radič sa skladá z niekoľkých podčastí. Tie poskytujú nadštandardnú funkcionalitu oproti bežnému FlexRay radiču. Okrem vlastných funkčných podčastí sa tu nachádza Adresný dekodér, ktorý je tvorený kombinačnou logikou ktorá obsluhuje signály CSC, SWR, SRD a Dátový prepínač, ktorý obsluhuje prenos dát. Z vlastných funkčných častí sa tu nachádzajú bloky : FlexRay core, Injector, Timestamp generator, Logger a Trigger. Zdieľajú 32 bitovú dátovú linku kompatibilnú s Avalon zbernicou a sú vyberané na základe posledných 12-tich bitov adresy. Funkčné časti zdieľajú zbernicu X-BUS, ktorá distribuuje dáta z jadra radiča do všetkých ostatných častí. Podrobný popis funkcionality jednotlivých častí sa nachádza v [1] vrátane popisu vnútorných registrov. Blokové schéma radiča sa nachádza na Obrázku 3.2 .Každý blok má vlastný výberový signál, a spoločne zdieľajú signály SWR (indikuje zápis, aktívny v 1) , SRD (indikuje čítanie aktívny v 1). Signál RST je asynchrónny reset blokov. Tento signál je vyvedený na výstup, aby sa mohol radič pred konfiguráciou z procesora uviesť do východzieho stavu. Radič je zobrazený Obrázku 3.1.



Obr. 3.1: FlexRay Radič



Obr. 3.2: Blokové schéma FlexRay radiča

3.2 Popis signálov

Popis signálov FlexRay radiča sa nachádza v nasledujúcej tabuľke (spoločné signály pre 32 bitové paralelné rozhranie v 2.3):

Meno	Funkcia	Smer
RX_A(B)	Príjmacia linka FlexRay kanálu A(B)	Vstupný
TX_A(B)	Vysielacia linka kanálu A(B)	Výstupný
TX_EN_A(B)	Enable linka kanálu A(B)	Výstupný
ID_OUT[3..0]	Výstup ID, podľa volenej hodnoty, zapája sa do obecného registra	Výstupný
CONTROLLER_OUTPUT _INTERRUPT	Prerušenie z jadra FlexRay kontroléra	Výstupný

Tabuľka 3.1: Signály FlexRay radiča

3.3 Controller Core

Jadro radiča je hlavná komponenta radiča, ktorá sa stará o vysielanie FlexRay dát na linkovej vrstve. Okrem štandardných signálov (viz začiatok kapitoly) sa tu nachádzajú

vlastné výstupy linkovej vrstvy, výstupná zbernica X_BUS, ktorá distribuuje informácie do všetkých ostatných blokov v radiči, 64 bitový vstup timestamp, ktorý obsahuje časový údaj o dobe behu radiča. Tiež je tu vstupná linka prerušenia, ktorá prichádza z komponenty Trigger obstarávajuca generovanie prerušenia.

3.4 Injector

Injector alebo "vstrekovač" je komponenta umožňujúca na linkovú vrstvu nahrať požadovanú skupinu bitov a umožňuje tak priamo testovať kolízne situácie. Výstup Injectora a FlexRay radiča sú prevedené cez logický súčin (AND hradlo) a tvoria výstup linkovej vrstvy FlexRay z radiča.

3.5 Data logger

Data logger je komponenta ktorá umožňuje zaznamenávanie časových značiek rôznych udalostí. Zaznamenáva tak napríklad nábežné hrany signálov z FlexRay liniek. Užívateľ potom môže vyčítať časový údaj, kedy udalosť nastala.

3.6 Trigger

Trigger "spínač" je komponenta, ktorá generuje prerušenie pri zaznamenaní konkrétnej skupiny bitov na FlexRay zbernici. Informácie získava z X_BUS zbernice a je schopná reagovať na udalosti ako prijatie nadefinovaného vzoru, chyba v rámci alebo dosiahnutie určitého času, počítaného pomocou generátora časových značiek.

3.7 Datový prepínač

Datový prepínač je komponenta FlexRay radiča zabezpečujúca kompatibilitu medzi obojsmernou dátovou linkou použitou v [1] a dátovou linkou použitou medzi komponen-

tami radiča. Prepínač je naprogramovaný na základe vzoru obojsmernej dátovej zbernice [12]. Na vstupe sú dátové linky každej časti FlexRay kontroléra, výberové signály a signál, ktorý indikuje zápis alebo čítanie (viz. Kapitola 2). Na základe aktívneho výberového signálu a hodnoty RW_SELECT sa buď čítané dáta prenesú na obojsmernú zbernicu, alebo dáta z obojsmernej zbernice sa prenesú na zapisovaciu dátovú linku.

3.8 Adresný dekodér

Adresný dekodér je komponenta, ktorá nastavuje výberové signály pre všetky ďalšie časti FlexRay radiča podľa adresy na vstupe, SADR(24..0). Tiež aktivuje obvody iba ak sú dáta na vstupe určené iba pre daný radič. Kontroluje správnosť adresy na základe nastavenej generickej hodnoty ID pre vstupný radič. Napriek tomu, že SADR(24..0) obsahuje spodných 24 bitov adresy procesora, Adresný dekodér ďalej do podkomponent neprepúšťa spodné dva bity (Komponenty sú adresovateľné iba 32 bitovo). Adresy teda priraďuje podľa vzoru : INPUT_ADRESSS(9..2)–>OUTPUT_ADRESS(7..0).

3.9 Syntéza FlexRay radiča

Syntéza radiča spolu s pomocnými obvodmi bola realizovaná na vývojovej doske Altera Cyclone II Starters Kit. Výsledky sú zobrazené v nasledovnej tabuľke:

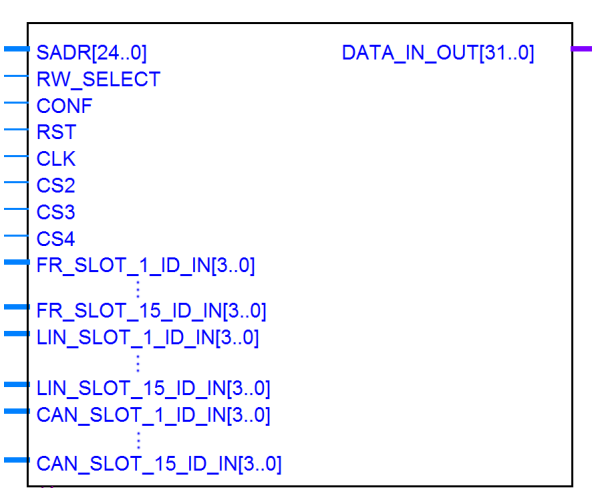
Počet Logických elementov	Počet kombinačných funkcií	Počet Registrov
12 207 (65 %)	10 803 (58%)	5977

Tabuľka 3.2: Výsledky syntézy FlexRay radiča

4 Obecný register

4.1 Popis Obecného registra

Obecný register je komponenta, ktorá umožňuje užívateľovi, programátorovi, z procesora vyzistiť aktuálny stav nahraných radičov. Užívateľ tak môže pracovať s výrobkom, ktorý pred ním niekto nakonfiguroval a nasyntetizoval radiče bez toho, aby poznal ich vlastné usporiadanie. Nemusí sám spájať bloky a syntetizovať obvod do FPGA ale stačí mu vyčítať informácie z registrov tejto komponenty. Obecný register poskytuje informácie o počte pripojených radičov, bázové adresy radičov pripojených do jednotlivých slotov (viz. 2.1), umožňuje vyzistiť do ktorých slotov sú radiče pripojené, takže programátor nemusí prechádzať všetky sloty a zisťovať, či je nejaký radič v danom slote pripojený. Počet pripojených radičov je spočítaný a uložený do registra vždy po spustení obvodu. Pri prípadnej chybe môže užívateľ register prekonfigurovať tým, že znova spustí prepočítanie pripojených radičov. Komponenta je navrhovaná s budúcou rozšíriteľnosťou pre CAN a LIN radiče.



Obr. 4.1: Obecný Register

4.2 Popis signálov

Popis signálov Obecného registra sa nachdáza v nasledujúcej tabuľke (spoločné signály pre 32 bitové paralelné rozhranie v 2.3):

Meno	Funkcia	Smer
CS2,CS3,CS4	Chip Select signály z procesra	Vstupný
FR_SLOT_X_ID_IN[3..0]	ID linky FlexRay radiča v slote X	Vstupný
LIN_X_ID_IN[3..0]	ID linky LIN radiča v slote X	Vstupný
CAN_X_ID_IN[3..0]	ID linkyCAN radiča v slote X	Vstupný

Tabuľka 4.1: Signály 32 Obecného registra

4.3 Mapa registrov

Mapa registrov sa nachádza na Obrázku 4.2 a význam registrov je popísaný v nasledujúcich podkapitolách.

4.3.1 DEVICE_ID

Udáva ID komponenty. Vždy vracia hodnotu 0xAAAABBBB. Register je určený iba na čítanie. Dotazom na hodnotu registra je možné testovať prítomnosť komponenty v implementovanom systéme.

4.3.2 FR_COUNT, LIN_COUNT, CAN_COUNT

Registre vracajú počet pripojených radičov (danej zbernice podľa mena registra) pripojených do slotov každého radiča. Vždy vracia hodnotu počtu pripojených radičov. V prípade zápisu do registra COUNT_AGAIN sa počty radičov prepočítajú a hodnoty sa aktualizujú. Radič je považovaný za pripojený do slotu X, ak na vstupe BUS_SLOT_X_ID_IN je privedený výstup ID_OUT radiča zbernice a tento radič má nastavené ID v rozmedzí 1 až 15. V prípade pripojenia radiča s ID=0 radič nie je rozpoznaný.

Offset	Meno	R/W	Default value
0x00	DEVICE ID	R	0xAAAAAAAA
0x04	FR_COUNT	R	Počet pripojených radičov pre každú zbernicu
0x08	LIN_COUNT	R	
0x0C	CAN_COUNT	R	
0x10	FR_1_BASE	R	0xFFFFFFFF
0x14	FR_2_BASE	R	0xFFFFFFFF
.	.	.	.
.	.	.	.
.	.	.	.
0x48	FR_15_BASE	R	0xFFFFFFFF
0x4C	LIN_1_BASE	R	0xFFFFFFFF
0x50	LIN_2_BASE	R	0xFFFFFFFF
.	.	.	.
.	.	.	.
.	.	.	.
0x84	LIN_15_BASE	R	0xFFFFFFFF
0x88	CAN_1_BASE	R	0xFFFFFFFF
0x5C	CAN_2_BASE	R	0xFFFFFFFF
.	.	.	.
.	.	.	.
.	.	.	.
0xC8	CAN_15_BASE	R	0xFFFFFFFF
0xCC	COUNT_AGAIN	W	0xFFFF0000
0xD0	FR_ACTIVE	R	0xFFFF0000
0xD4	LIN_ACTIVE	R	0xFFFF0000
0xD8	CAN_ACTIVE	R	0xFFFF0000
0xDC	FR_INTERNAL_BASE	R	0xFFF7_C800
0XE0	CAN_INTERNAL_1_BASE	R	0xFFF7_DC00
0XE4	CAN_INTERNAL_2_BASE	R	0xFFF7_DE00
0XE8	CAN_INTERNAL_3_BASE	R	0xFFF7_E000
0XEC	LIN_INTERNAL_1	R	0xFFF7_E400

Tabuľka 4.2: Mapa Registrov Obecného registra

4.3.3 FR_BASE_1 až FR_BASE_15

Vracia bázové adresy radičov syntetizovaných v FPGA a pripojených do slotov 2 až 15. Ak nie je do slotu pripojený žiadny radič alebo je pripojený radič s ID=0, register vracia hodnotu 0xFFFFFFFF. S Adresou sa ďalej pracuje podľa [1] (Offset pre požadovaný register) a podľa Kapitoly 2, bity 11 až 8 na výber komponenty.

4.3.4 LIN_BASE_1 až LIN_BASE_15

Vracia bázové adresy LIN radičov pripojených do Slotov 2 až 15. Ak nie je do slotu pripojený žiadny radič alebo je pripojený radič s ID=0, register vracia hodnotu 0xFFFFFFFF.

4.3.5 CAN_BASE_1 až CAN_BASE_15

Vracia bázové adresy CAN radičov pripojených do Slotov 2 až 15. Ak nie je do slotu pripojený žiadny radič alebo je pripojený radič s ID=0, register vracia hodnotu 0xFFFFFFFF.

4.3.6 COUNT_AGAIN

Zápisom hodnoty 0xFFFFFFFF do registra sa spustí v komponente procedúra, ktorá znovu prepočíta pripojené radiče a výsledok uloží do registra BUS_COUNT, kde BUS je druh zbernice. Po spočítaní a uložení sa register znovu samostatne vynuluje.

4.3.7 FR_ACTIVE_SLOTS, LIN_ACTIVE_SLOTS a CAN_ACTIVE_SLOTS

Čítaním hodnôt vracia informácie o jednotlivých slotoch pre každú zbernicu. V prípade, že je do Slotu zapojený radič s ID=1 až 15, je bit tohto slotu v logickej 1. V prípade že je do Slotu zapojený radič s ID=0 alebo žiadny radič, je bit tohto slotu v logickej 0. Poradie bitu pre slot je dané číslom slotu. Bit 0 teda nemá význam, bity 1 až 15 majú hodnotu podľa pripojených radičov a bity 16 až 31 sú vždy v 1.

4.3.8 FR_INTERNAL, CAN_INTERNAL, LIN_INTERNAL

Vracia bázové adresy interných FlexRay, CAN a LIN radičov procesora. Registre sú zavedené iba pre možnosť získať adresy všetkých radičov, ktoré sa v systéme nachádzajú. Adresa sa dá jednoducho nájsť v [13] bez čítania dát z FPGA. Na tento radič sa nevzťahuje adresný systém v 2.4. Register teda poskytne bázovú adresu s, ktorou sa

ďalej pracuje podľa [13].

4.4 Syntéza Obecného registra

Syntéza Obecného registra bola realizovaná na vývojovej doske Altera Cyclone II Starters Kit. Výsledky sú zobrazené v nasledovnej tabuľke:

Počet Logických elementov	Počet kombinačných funkcií	Počet Registrov
487 (3 %)	446 (2%)	160

Tabuľka 4.3: Výsledky syntézy Obecného registra

5 Výstupný Multiplexor

5.1 Popis Výstupného multiplexora

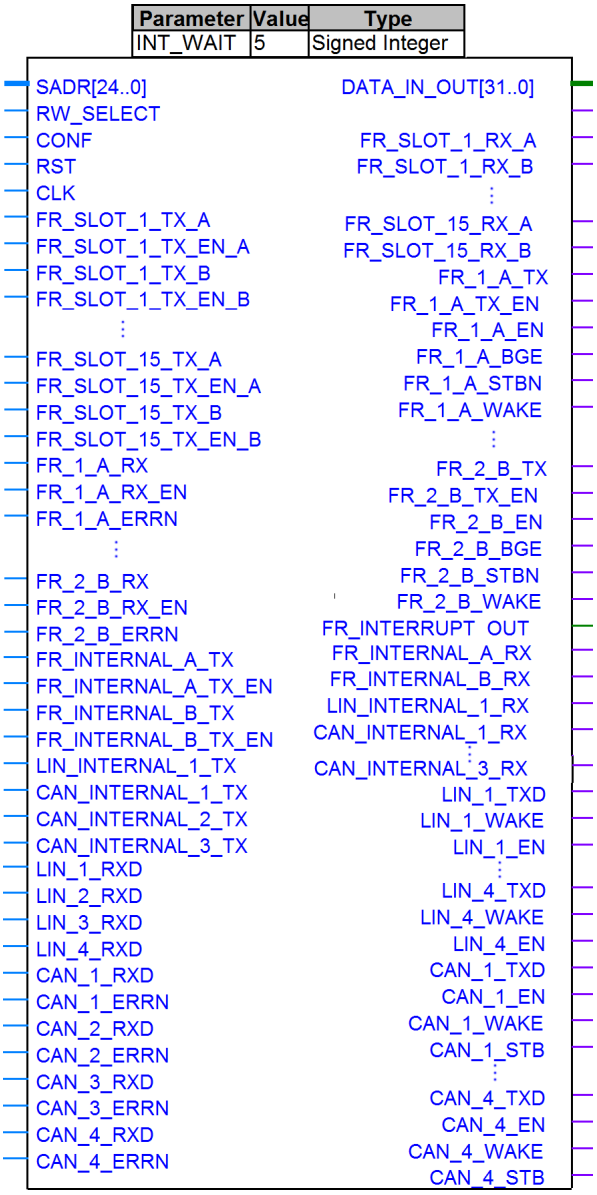
Výstupný register umožňuje prepájať logické radiče na linkovej vrstve na budiče fyzických vrstiev, konfigurovať fyzické budiče a generovať prerušenie pri zachytení vybranej udalosti prichádzajúcej od fyzického budiča. Rovnako ako do Obecného registra sa radiče pripájajú do slotov. Oproti Obecnému Registru je potrebné pripojiť aj signály vlastných liniek (TX_A, TX_EN_A,RX_A,TX_B, TX_EN_B,RX_B) z radiča, nielen výstupy ID_OUT radiča. Vstupy, do ktorých sa signály zapájajú podľa cieleného Slotu, sú uvedené v popise signálov. Výstupný multiplexor je zobrazený na obrázku 5.1 a popis jeho signálov je na obrázku 5.1

5.2 Popis signálov

Popis signálov výstupného multiplexora sa nachádza v nasledujúcej tabuľke (spoločné signály pre 32 bitové paralelné rozhranie v Tabuľke 2.3):

Meno	Funkcia	Smer
FR_SLOT_X_TX_A(B)	Signál TX_A(B) z radiča v Slote X	Vstupný
FR_SLOT_X_TX_EN_A(B)	Signál TX_EN_A(B) z radiča v Slote X	Vstupný
FR_SLOT_RX_A(B)	Signál RX_A(B) do radiča v Slote X	Výstupný
FR_Y_A(B)_TX	Pripojenie budiča Y, kanálu A(B), signál TX	Výstupný
FR_Y_A(B)_TX_EN	Pripojenie budiča Y, kanálu A(B), signál TX_EN	Výstupný
FR_Y_A(B)_EN	Pripojenie budiča Y, kanálu A(B), signál EN	Výstupný
FR_Y_A(B)_BGE	Pripojenie budiča Y, kanálu A(B), signál BGE	Výstupný
FR_Y_A(B)_STBN	Pripojenie budiča Y, kanálu A(B), signál STBN	Výstupný
FR_Y_A(B)_WAKE	Pripojenie budiča Y, kanálu A(B), signál WAKE	Výstupný
FR_Y_A(B)_RX	Pripojenie budiča Y, kanálu A(B), signál RX	Vstupný
FR_Y_A(B)_RX_EN	Pripojenie budiča Y, kanálu A(B), signál RX_EN	Vstupný
FR_Y_A(B)_ERRN	Pripojenie budiča Y, kanálu A(B), signál ERRN	Vstupný
FR_INTERRUPT_OUT	Prerušenie generované obvodom	Výstupný

Tabuľka 5.1: Signály Výstupného multiplexora



Obr. 5.1: Výstupný multiplexor

5.3 Mapa registrov

Mapa registrov sa nachádza v Tabuľke 5.2 a význam registrov je popísaný v nasledujúcich podkapitolách.

Offset	Meno	R/W	Počiatočná hodnota
0x00	DEVICE ID	R	0xBBBBBBBB
0x04	FR_1_A_CONFIG	R/W	0x0000000F
0x08	FR_1_B_CONFIG	R/W	0x0000000F
0x0C	FR_2_A_CONFIG	R/W	0x0000000F
0x10	FR_2_B_CONFIG	R/W	0x0000000F
0x14	LIN_1_CONFIG	R/W	0x00000000
0x18	LIN_2_CONFIG	R/W	0x00000000
0x1C	LIN_3_CONFIG	R/W	0x00000000
0x20	LIN_4_CONFIG	R/W	0x00000000
0x24	CAN_1_CONFIG	R/W	0x00000000
0x28	CAN_2_CONFIG	R/W	0x00000000
0x2C	CAN_3_CONFIG	R/W	0x00000000
0x30	CAN_4_CONFIG	R/W	0x00000000
0x34	INT_VECTOR	R/W	0x00000000
0x38	FR_SLOT_1_TO_4	R/W	0x00000000
0x3C	FR_SLOT_5_TO_8	R/W	0x00000000
0x40	FR_SLOT_9_TO_12	R/W	0x00000000
0x44	FR_SLOT_13_TO_15	R/W	0x00000000
0x48	RESERVED	R/W	0x00000000
0x4C	CAN_INTERNAL	R/W	0x00000000
0x50	LIN_INTERNAL	R/W	0x00000000
0x54	FR_INTERNAL	R/W	0x00000000

Tabuľka 5.2: Mapa Registrov Výstupného multiplexora

5.3.1 DEVICE_ID

Udáva ID typu komponenty. Vždy vracia hodnotu 0xBBBBBBBB. Register je určený iba na čítanie.

5.3.2 FR_X_Y_CONFIG

Konfiguruje fyzické budiče pre Flexray, nastavuje vyvolávanie prerušenia. K dispozícii sú 4 budiče, dva s kanálmi A a B (FR_1_A_CONFIG, FR_1_B_CONFIG, FR_2_A_CONFIG, FR_2_B_CONFIG). Nastavuje vstupy WAKE, STBN, BG, EN na požadovanú hodnotu. Presný popis funkcionality je uvedený v [9]. Význam jednotlivých bitov je nasledovný:

- Bit 0 - nastavuje ENABLE_INPUT (EN) vstup budiča. Aktívny v logickej 1. Musí byť nastavené na 1 aby budič príjmal na FlexRay kanáli.

- Bit 1 - nastavuje BUS_GUARDIAN (BG) vstup budiča. Aktívny v logickej 0. V aktívnom stave vypína vysielanie. Implicitne nastavený na 1.
- Bit 2 - nastavuje STANBY_INPUT (STBN) vstup budiča. Aktívny v logickej 0. Nastavuje režim nízkej spotreby. Implicitne nastavený na 1.
- Bit 3 - nastavuje WAKE vstup budiča. Aktívny v logickej 0. Implicitne nastavený na 1.
- Bit 4 - nastavuje vyvolanie prerušenia indikáciou chyby na ERRN výstupe budiča. V logickej 1 je prerušenie vyvolané. Implicitne nastavený na 0.
- Bit 5 - nastavuje vyvolanie prerušenia prijatím dát na RX_EN výstupe budiča. V logickej 1 je prerušenie vyvolané. Implicitne nastavený na 0.

5.3.3 LIN_X_CONFIG

Konfiguruje fyzické budiče pre LIN, tento radič neposkytuje možnosť vyvolávať prerušenie. K dispozícii sú 4 budiče, LIN_1 až LIN_4. Nastavuje vstupy WAKE, EN na požadovanú hodnotu. Presný popis funkcionality je uvedený v [11]. Význam jednotlivých bitov je nasledovný:

- Bit 0 - nastavuje ENABLE_INPUT (EN) vstup budiča. Aktívny v logickej 1. Aktivuje LIN budič. Implicitne nastavený na 1.
- Bit 1 - nastavuje BUS_GUARDIAN (BG) vstup budiča. Aktívny v logickej 0. Implicitne nastavený na 1.

5.3.4 CAN_X_CONFIG

Konfiguruje fyzické budiče pre CAN, nastavuje vyvolávanie prerušenia. K dispozícii sú 4 budiče, CAN_1 až CAN_4. Nastavuje vstupy WAKE, STBN, EN na požadovanú hod-

notu. Presný popis funkcionality je uvedený v [10]. Význam jednotlivých bitov je nasledovný:

- Bit 0 - nastavuje ENABLE_INPUT (EN) vstup budiča. Aktívny v logickej 1. Aktivuje CAN budič. Implicitne nastavený na 1.

- Bit 1 - nastavuje STANBY_INPUT (STBN) vstup budiča. Aktívny v logickej 0. Implicitne nastavený na 1.

- Bit 2 - nastavuje WAKE vstup budiča. Aktívny v logickej 0. Implicitne nastavený na 1.

- Bit 4 - nastavuje vyvolanie prerušenia indikáciou chyby na ERRN výstupe budiča. V logickej 1 je prerušenie vyvolané. Implicitne nastavený na 0.

5.3.5 INT_VECTOR

Je register, ktorý obsahuje zdroj prerušenia. Vždy daný bit je vyhodený v logickej 1, ak daný signál je zdrojom prerušenia. Implicitne je register nastavený na 0x00000000. Po prečítaní zdroja prerušenia je nutné zapísať do registra ľubovoľnú hodnotu ktorá register vynuluje aby sa mohlo čítať ďalšie prerušenie. Na správnu funkčnosť tohto registra je tiež nutné najprv nastaviť vyvolávanie prerušenia zápisom do registrov FR_CONFIG, CAN_CONFIG, LIN_CONFIG. Priradenie bitov k zdrojom prerušenia je v nasledovnej Tabuľke:

Číslo bitu	Zdroj prerušenia
0	FR_1_A_ERRN
1	FR_1_A_RX_EN
2	FR_1_B_ERRN
3	FR_1_B_RX_EN
4	FR_2_A_ERRN
5	FR_2_A_RX_EN
6	FR_2_B_ERRN
7	FR_2_B_RX_EN
8	CAN_1_ERRN
9	CAN_2_ERRN
10	CAN_3_ERRN
11	CAN_4_ERRN

Tabuľka 5.3: Mapa Registrov Výstupného multiplexora

5.3.6 FR_SLOT_X_TO_Y

Nastavuje pripojenie radičov v slotoch na fyzické budiče (TJA1080). Registre sú celkovo 4 (FR_SLOT_1_TO_4, FR_SLOT_5_TO_8, FR_SLOT_9_TO_12, FR_SLOT_13_TO_15). Každý kanál môže byť pripojený zvlášť, to znamená, že pripojenie radiča do jedného slotu ešte neznamená že oba jeho kanály budú pripojené k rovnakému vysielaču. Vždy 4 združené bity udávajú pripojenie radiča v určitom slote. Podľa hodnoty zapísanej do týchto 4 bitov sa potom udáva, do ktorého fyzického budiča je radič v danom slote pripojený. V prípade, že je viacero radičov pripojených na jeden fyzický budič, vysielané linky sú pripojené na AND hradlo a výstup hradla je pripojený na budič. V tabuľkách 5.4 a 5.5 je uvedený popis hodnôt pre pripojenie k požadovanému budiču a priradenie skupín bitov pre nastavenie požadovaného slotu.

5.3.7 CAN_INTERNAL, LIN_INTERNAL, FR_INTERNAL

Registre nastavujú presmerovanie interných radičov procesora na fyzické budiče. Princíp zápisu do registrov je podobný ako pri FlexRay radičoch v Slotoch. Aj pre radiče ostatných zberníc (v našom prípade CAN, LIN radič je iba 1), platí že ako presmeru-

jeme viacero radičov na jeden fyzický budič, je vstup na budiči logickým súčinom liniek pripojených radičov. Celý popis registrov, rozdelenia bitov a hodnôt je uvedený v nasledovných tabuľkách.

Názov Registra	Bity 31:28	Bity 27:24	Bity 23:20	Bity 19:16	Bity 15:12	Bity 11:8	Bity 7:4	Bity 3:0
FLEXRAY_SLOT_1_TO_4	Slot 4 Kanál B	Slot 4 Kanál A	Slot 3 Kanál B	Slot 3 Kanál A	Slot 2 Kanál B	Slot 2 Kanál A	Slot 1 Kanál B	Slot 1 Kanál A
FLEXRAY_SLOT_5_TO_8	Slot 8 Kanál B	Slot 8 Kanál A	Slot 7 Kanál B	Slot 7 Kanál A	Slot 6 Kanál B	Slot 6 Kanál A	Slot 5 Kanál B	Slot 5 Kanál A
FLEXRAY_SLOT_9_TO_12	Slot 12 Kanál B	Slot 12 Kanál A	Slot 11 Kanál B	Slot 11 Kanál A	Slot 10 Kanál B	Slot 10 Kanál A	Slot 9 Kanál B	Slot 9 Kanál A
FLEXRAY_SLOT_13_TO_15	Nepoužité		Slot 15 Kanál B	Slot 15 Kanál A	Slot 14 Kanál B	Slot 14 Kanál A	Slot 13 Kanál B	Slot 13 Kanál A
FR_INTERNAL	Nepoužité						FR Internal 2	FR Internal 1
CAN_INTERNAL	Nepoužité					CAN Internal 3	CAN Internal 2	CAN Internal 1
LIN_INTERNAL	Nepoužité							LIN Internal 1

Tabuľka 5.4: Význam bitov registra FR_SLOT_X_TO_Y

Hodnota	FlexRay budiče	CAN Radič	LIN radič
0x0	Slot nezapojený	Interný radič nezapojený	Interný radič nezapojený
0x1	FlexRay 1 Kanál A	Budič 1	Budič 1
0x2	FlexRay 1 Kanál B	Budič 2	Budič 2
0x3	FlexRay 2 Kanál A	Budič 3	Budič 3
0x4	FlexRay 2 Kanál B	Budič 4	Budič 4

Tabuľka 5.5: Hodnoty bitov pre registre pripojenia na fyzické budiče

5.4 Syntéza Výstupného multiplexora

Syntéza Výstupného multiplexora bola realizovaná na vývojovej doske Altera Cyclone II Starters Kit. Výsledky sú zobrazené v nasledovnej tabuľke:

Počet Logických elementov	Počet kombinačných funkcií	Počet Registrov
1395 (7 %)	1348 (7%)	633

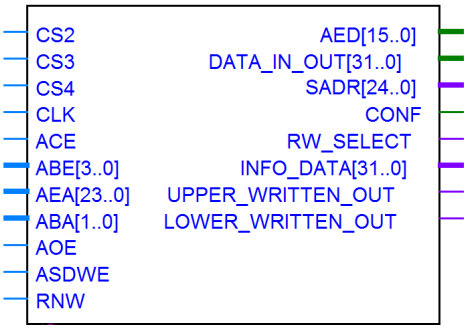
Tabuľka 5.6: Výsledky syntézy výstupného multiplexora

6 EMIF dekodér

6.1 Popis dekodéru

EMIF dekodér je komponenta, ktorá zabezpečuje prevod medzi EMIF rozhraním a 32 bitovým paralelným rozhraním. Dekodér je zobrazený na obrázku 6.1 a funguje ako stavový automat, ktorý je popísaný na v nasledujúcej podkapitole. Dekodér pri zápise z procesora do vnútorného registra INTERNAL_DATA nahrá dáta z dvoch po sebe idúcich EMIF zápisov a tie potom posunie na 32 bitové paralelné rozhranie medzi obvodmi. Pri čítaní vyberie, ktoré polslovo (16 bitov) je na EMIF adresované bitom AEA(0) a to poskytne procesoru. EMIF dekodér tiež nastavuje, do ktorého adresného podpriestoru EMIF v procesore obvody mapujeme, a to voľbou generickej hodnoty USED_CHIP_SELECT.

Parameter	Value	Type
USED_CHIP_SELECT	0	Signed Integer
WRITE_COUNTER_LIMIT	5	Signed Integer
READ_COUNTER_LIMIT	5	Signed Integer
READ_WAIT_COUNTER_LIMIT	3	Signed Integer



Obr. 6.1: Emif dekodér

6.2 Popis signálov

Popis signálov EMIF dekodéra sa nachdáza v nasledujúcej tabuľke (spoločné signály pre 32 bitové paralelné rozhranie v 2.3):

Meno	Funkcia	Smer
CS2,CS3,CS4	Chip Select signály z procesra	Vstupný
ACE	Chip Enable Signál	Vstupný
ABE[3..0]	Byte Enable signál, nepoužitý	Vstupný
AEA[23..0]	Adresné linky EMIF	Vstupný
ABA[1..0]	Bity 0 a 1 adresy v procesore	Vstupný
AOE	Output Enable signál (akt. V Log.0)	Vstupný
ASDWE	Write signál (akt. V Log.0)	Vstupný
RNW	Read, Write signál, nepoužitý	Vstupný
AED[15..0]	Datové linky EMIF	Obojsmerný
INFO_DATA[31..0]	Vnútorných dáta na zápis	Výstupný*
UPPER_WRITTEN_OUT	Zapísané horné pol slovo	Výstupný*
LOWER_WRITTEN_OUT	Zapísané dolné pol slovo	Výstupný*
STATE_OUT	Stav stavového automatu	Výstupný*

Tabuľka 6.1: Signály EMIF dekodéra

6.3 Konfigurácia EMIF dekodéru

EMIF dekodér môžeme používať pre prácu v módoch “Strobe” a “s použitím pinu AARDY”. Časové diagramy pre každý mód sú v Kapitole 2. V móde “Strobe” trvá obdobie “Strobe”(viz. Obrázok 2.5) 4 hodinové cykly EMIF. V móde “s použitím pinu AARDY” je možné predĺžiť obdobie “Strobe” podľa vložených cyklov na pin AARDY. Tento mód je vhodné použiť pre veľmi pomalý hardware, kedy pri čítaní dát obvody nie sú schopné dáta dodať v dobe 4 hodinových cyklov. Použitie tohto módu sa nastavuje hodnotou TRUE pre parameter USE_READY_OUTPUT. Je vhodné dodať, že výhradne na zápis nie je potrebné používať mód “ použitím pinu AARDY”, pri zápise sú dáta najprv uložené do EMIF dekodéra (register INTERNAL_DATA) a až následne spracované 32 bitovým paralelným rozhraním.

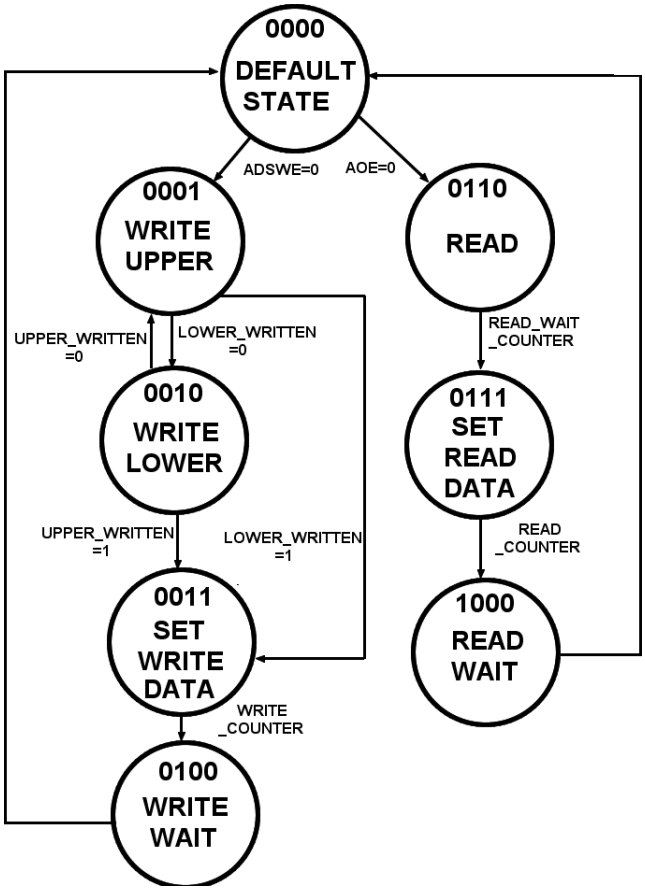
Na konfiguráciu časovania sú implementované tri časovače. Časovače sú zazna-

čené v prechodoch medzi stavmi stavového automatu. Počítajú hodinové cykly a pri dosiahnutí limitnej hodnoty (nastaviteľná v Quartuse) prejdú na ďalší stav. WRITE_COUNTER počíta cykly po nastavení hodnoty z EMIF datovej linky na DATA_IN_OUT, počas čakania musia obovdy hodnotu zapísať do vnútorných registrov. READ_WAIT_COUNTER počíta po nastavení adresy SADR, RW_SELECT a CONF pre čítanie a po uplynutí musia byť dáta nastavené z obvodu na DATA_IN_OUT vstupe EMIF dekodéra. Čítače WRITE_COUNTER a READ_WAIT_COUNTER je vhodné nastaviť na rovnakú limitnú hodnotu. Čítač READ_ COUNTER počíta cykly po prečítaní dát EMIF dekodérom z DATA_IN_OUT a po nastavení na AED. Po uplynutí musí mať procesor z AEA dáta prečítané.

Ladenie časovacích konštánt bolo najprv realizované experimentálne na základe výstupných dát zo simulácií. Snaha bola o čo najväčšiu podobnosť medzi predpísanými signálovými sledmi zbernice a výsledkami simulácie. Doladenie konštánt bolo realizované pri testovaní na skutočnom zariadení na základe výstupov logického analyzátora v Quartuse. Nastavené hodnoty konštánt sú v Quartus projekte v implementovanom systéme.

6.4 EMIF dekodér ako stavový automat

Emif dekodér funguje ako stavový automat typu Mealy, teda jeho výstup závisí nielen na stave, ale aj na vstupe stavového automatu. Stavový automat má cesty, ktorými môže dospieť do pôvodného stavu: Zápis z EMIF na paralelné rozhranie a čítanie do EMIF z paralelného rozhrania. Stavový automat implementuje signálové sledy zobrazené v Kapitole 2 pre EMIF a pre 32 bitové paralelné rozhranie. Popis stavového automatu je na obrázku 6.2



Obr. 6.2: EMIF dekodér ako stavový automat

6.5 Syntéza EMIF dekodéra

Syntéza EMIF dekodéra bola realizovaná na vývojovej doske Altera Cyclone II Starters Kit. Výsledky sú zobrazené v nasledovnej tabuľke:

Počet Logických elementov	Počet kombinačných funkcií	Počet Registrov
350 (2 %)	271 (1%)	256

Tabuľka 6.2: Výsledky syntézy EMIF dekodéra

7 Testovanie systému

7.1 Úvod k Simuláciám

Na simulovanie funkčnosti obvodov bol použitý Simulator Tool programu Quartus. Simulácia bola vykonávaná po kompilácii a syntéze obvodu tak ako na cieľovom výrobku. Simulácia po syntéze započítava aj oneskorenie hradla a preto vyhovuje časovým požiadavkám systému. Pre doladenie prípadne nefunkčného systému je možné meniť čakacie doby jednotlivých rozhraní na dáta (generické hodnoty EMIF dekodéra) a dobu zotrvania prerušenia v aktívnej hodnote. Obvody sú simulované samostatne tak, že vstupy simulácie obvodu v systéme vždy zodpovedajú výstupom predchádzajúceho odvodu (napr. EMIF dekodéra). Všetky použité signálové sledy odpovedajú popisu EMIF rozhrania a 32-bitového paralelného rozhrania.

7.2 Simulácie Obvodov

7.2.1 Simulácia FlexRay radiča

Pre overenie správnosti pripojenia pamäťového priestoru FlexRay kontroléru vykonáme jednoduchú simuláciu, v ktorej do podkomponenty Injectora zapíšeme do registra z Offsetom 0x08 (register LENGTH podľa [1]). V jednom zbernicovom cykle 32 bitového paralelného rozhrania zapíšeme na túto adresu hodnotu 4 a v ďalšom zbernicovom cykle prečítame hodnotu, ktorá sem bola zapísaná. Takýmto spôsobom overíme zápis aj čítanie a funkčnosť komponent Datového prepínača a Adresného dekodéru, ktoré sa v FlexRay radiči nachádzajú. Výsledná simulácia je na priloženom CD (Simulácie Komponent/Výsledky/ControllerExtendedWrite.wvf). Adresa je 0x031702 , 0 -rezervovaná, 3 - FlexRay radič, 1 - ID_Radiča, tiež možno vidieť v simulácii výstup ID_OUT, 7 - Odpovedá Injectoru, 2 - offset v 32 bitovom rozhraní (2*4=0x8= offset v procesore).

Pri simulácii sú napodobnené podmienky na 32 bitovom paralelnom rozhraní, ktoré vytvára pri prístupe EMIF dekodér. Pre účely simulácie boli z komponenty vyvedené aj nadštandardné výstupy, ktoré spolu zo znalosťou štruktúry radiča dávajú prehľad o vnútornom dianí v obvode. Jedná sa napríklad o výberové signály podkomponent (SCS_INJECTOR...).

7.2.2 Simulácia Obecného Registra

Na Obecnom registri vykonáme dve simulácie. V prvej simulácii pripojíme 4 virtuálne FlexRay radiče nadefinovaním nenulových vstupov na vstupoch FR_SLOT_1(2,3,4)_ID-_IN a čítame hodnotu z registra FR_ACTIVE (Offset v 32 bitovom rozhraní 0x32). Komponenta vracia logickú jednotku na pozícách 1 až 4 výstupných dát a správne signalizuje pripojené radiče. Výsledok simulácie je na priloženom CD Simulácie (Komponent/ Výsledky/ GPRActiveSlotsSimulation.wvf).

V Druhej simulácii čítame počet pripojených radičov pri rovnakých vstupných dátach. Adresa je 0x012032 (1 - Obecný register, 2 - ľubovoľná hodnota, 0 - ľubovoľná hodnota, 32 - offset). Na výstupnej datovej linke dostaneme aktívny iba bit 3 datového výstupu, ktorý vyjadruje počet pripojených radičov. Výsledok simulácie je na priloženom CD Simulácie (Komponent/Výsledky/GPR_FRCount.wvf).

7.2.3 Simulácia Výstupného Multiplexora

Na Výstupnom multiplexore prevedieme simulácie, ktoré overia jeho pripojenie do adresného priestoru aj jeho funkčnosť. Najjednoduchšia simulácia je čítanie ID komponenty, Adresovanie obdobné ako v predchádzajúcich simuláciách a výsledok na CD v (Simulácia Komponent/ Výsledky/ GPR_FRCount.wvf). Ďalej simulácia logického súčinu medzi dvoma pripojenými kanálmi. Najprv je do registra FR_1_TO_4 CONFIG (offset 0xE) zapísaná hodnota 0x00000011 (prvý Slot, kanál A na výstup 1 , prvý Slot kanál

B na výstup 1). Ďalej sú na Slot 1 kanály A, B privedené dva rôzne hodinové signály. Výsledný signál na výstup FR_1_A je logickým súčinom týchto dvoch signálov. Výsledok simulácie je na priloženom CD Simulácie (Komponent/ Výsledky/ OutMuxAndingChannels.wvf). Ako poslednú simuláciu prevedieme simuláciu generovania prerušenia a čítania vektoru prerušenia. Tu ako prvé nakonfigurujeme generovanie prerušenia zápisom hodnoty 0x00000010 do registra FR_1_A_CONFIG (prerušenie linkou FR_1_ERRN), potom simulujeme podnet prerušenia logickou nulou na vstupe FR_1_ERRN. Vyvolané prerušenie je logickou jednotkou na výstupe FR_INTERRUPT OUT. Prerušenie zostáva v logickej jednotke po dobu, ktorá je konfigurovateľná pre blok výstupného multiplexora (generická hodnota INT_WAIT). Po nastavení výstupu prerušenia prečítame register INT_VECTOR, v ktorom je uložená hodnota 1, odpovedajúca zdroju prerušenia FR_1_ERRN podľa popisu registra INT_VECTOR. Výsledok simulácie je na priloženom CD Simulácie (Komponent/ Výsledky/ GPR_INTERRUPT.wvf).

7.2.4 Simulácia EMIF dekodéra

Pri EMIF dekodéri Simulujeme zápis do a čítanie z procesora. Pre lepšiu prehľadnosť sú v simulácii zobrazené aj nadbytočné dáta ktoré nie sú výstupom EMIF dekodéru (INTERNAL_DATA). V tomto prípade sa jedná o použitie EMIF dekodéra v móde “Strobe”. EMIF dekodér je konfigurovaný nasledovne: WRITE_COUNTER_LIMIT=1, READ_COUNTER_LIMIT=0, READ_WAIT_COUNTER_LIMIT=2. V prípade zápisu sa jedná o dva po sebe idúce zbernicové cykly. Dáta z EMIF rozhrania sa zapisujú do vnútorného registra INTERNAL DATA a po zapísaní horného pol Slova aj dolného pol Slova sú dáta posunuté na výstup, v tomto prípade linku DATA_IN_OUT. Výsledok simulácie je na priloženom CD Simulácie (Komponent/ Výsledky/ EMIF_Zapis.wvf). Ako ďalšie je Simulované čítanie, pri ktorom sa dáta posúvajú z 32 bitového paralelného rozhrania na EMIF rozhranie. Jedná sa iba o jeden zbernicový cyklus EMIF, v ktorom je vybrané

požadované polSlovo podľa hodnoty AEA[0]. Pre názornosť sú v simulácii prečítané po sebe obe polslová a tiež je zobrazený výstup stavového automatu. Výsledok simulácie je na priloženom CD Simulácie (Komponent/ Výsledky/ EMIF_Citanie.wvf).

7.3 Testovanie systému na zariadení

Po striedavých úpravách a testovaní sa podarilo implementovaný systém úšpešne oživiť na zariadení. Bolo treba odskúšať časovanie EMIF zbernice na strane procesora, a rovnako na strane EMIF dekodéru. Finálne nastavenie pre EMIF dekodér je v projekte Quartusu na priloženom CD. Počiatočné testovanie bolo realizované s frekvenciou hodín pre EMIF a pre FPGA 20 Mhz.

V jazyku C som zostavil sadu funkcií ktorá umožňoavala testovať zápisy a čítanie do registrov a jednoducho vypisovať na sériový port zariadenia.

Maximálna frekvencia ktorú sa podarilo dosiahnuť pri priamom vedení hodinového signálu do procesora je 25 Mhz (25 Mhz frekvencia EMIF a 25 Mhz frekvencia hodinového signálu do FPGA). Z tohto pohľa sa jedná o funkčnosť aj za hranicu limitnej hodnoty keďže maximálna frekvencia hodinvého signálu vstupujúceho do FPGA je 20 Mhz.

Snaha o zvýšenie funkčnej frekvencie viedla k použitiu fázového závesu. Na vstup FPGA bol privedený hodinový signál s frekvenciou 25 Mhz a vynásobený fázovým závesom na 30 Mhz. V procesore bol ako hoidnov signál EMIF zbernice použitý signál 30 Mhz pre zachovanie synchrónnosti. Bohužial sa však pri iteračnom teste čítanie (jedna z funkcií v jazyku C), hádzalo chybné dáta pre približne každý piaty až desiaty test čítania. Dôvodom maximálnej frekvencie 25 Mhz bolo horné obmedzenie FPGA. EMIF sám o sebe je možné použiť aj pre vyššie frekvencie, avšak bez FPGA.

7.4 Modelový program pre procesor

Pre ukážkovú prácu s implementovanými komponentami je pre užívateľa zostavený názorný program v pseudojazyku podobnom jazyku C. Predpokladáme, že procesor disponuje 32 bitovým zápisom v tvare Write(BASE_ADRESS,OFFSET,VALUE) a čítaním v tvare Read(BASE_ADRESS, OFFSET) :Value . Nasledujúci kód tak nastaví fyzické budiče TJA1080, prečíta bázové adresy syntetizovaných radičov a nastaví pripojenie prvých dvoch slotov na oba fyzické budiče.

```
int frDeviceCount; //Počet FlexRay zariadení int frDeivceMask;
int genRegID; //ID Obecného registra
int *frDevBaseAdr; //Pointer pole bázových adries(BA) kontroléra
int outMuxID //ID Výstupného multiplexora
const GPR_BASE = 0x6010000;
const OUT_MUX_BASE = 0x60200000;
main {
genRegID=Read(GPR_BASE,0x0); //ID Obecného registra
if(genRegID==0xAAAAAAAAA) //Ak je ID vrátené správne
{ frDeviceCont=Read(GPR_BASE,0x04); //Počet FlexRay kontrolérov
frDeviceMask=Read(GPR_BASE,0xD0); //Sloty pre zariadenia
int i=0;
if(frDeviceCount>0){
for(i=1;i<15;i++){ if(frDevMask(i)=1){
*frDevBaseAdr=Read(GPR_BASE,0x0C+i*4); //BA kontroléra v i-tom Slote
frDevBaseAdr++; //navýši pointer pre uloženie ďalšej adresy
} } }
```

```
outMuxID=Read(OUT_MUX_BASE,0x00); //Prečíta ID výstupného multip-
lexora
if(outMuxID==0xBBBBBBBB){ //Ak je ID vrátené správne
Write(OUT_MUX_BASE,0x04,0xF); //Konfiguruje fyzické budiče FlexRay
Write(OUT_MUX_BASE,0x08,0xF);
Write(OUT_MUX_BASE,0x0C,0xF);
Write(OUT_MUX_BASE,0x10,0xF);
write(OUT_MUX_BASE,0x38,0x1; //Slot 1 Kanál A na Vysielač 1 A
                             |(0x2<<4) //Slot 1 Kanál B na Vysielač 1 B
                             |(0x3<<8) //Slot 2 Kanál A na Vysielač 2 A
                             |(0x4<<12));//Slot 2 Kanál B na Vysielač 2 B
}else { //Výstupný multiplexor nie je zapojený }
}else { //Obecný multiplexor nie je zapojený }
//Kód na konfiguráciu flexRay radičov}
```

8 Záver

Účelom mojej práce bolo navhrnúť spôsob pripojenia IP funkcie FlexRay radiča k procesoru pre konkrétny vyvíjaný výrobok, implementovať jeho zaradenie do pamäťového priestoru procesora a následne v simuláciách a v reále overiť funkčnosť navrhnutého systému.

V mojej práci som navrhol spôsob pripojenia IP funkcií FlexRay radiča k procesoru a tiež k fyzickým budičom, ktoré sa nachádzajú na vyvíjanej testovacej platforme. Overil som funkčnosť systému najprv v simuláciách a potom na reálnom zariadení. Vytvoril som VHDL kódy, ktoré spolu so súbormi Quartus symbolov implementujú požadovanú funkcionalitu. Priradil som piny zariadenia k ich funkcii pre zjednodušenia práce v systéme. Odladil som prístup k registrom radičov v FPGA z procesora. Pre ďalšiu prácu je teda k dispozícii projekt v Quartuse, na ktorý je možné nadviazať pri ďalšom rozširovaní možností použitia testovacej platformy. Týmto som splnil zadanie svojej bakalárskej práce. Do budúcna by som rád implementovaný systém rozšíril aj o radiče ďalších zberníc. Pre testovaciu platformu je plánované použitie operačného systému, ktorý by umožňoval komunikáciu s PC a nahranie konfigurácie FPGA priamo cez procesor bez použitia USB blasteru.

Na mojej práci pozitívne hodnotím zvolený formát systému a implementáciu funkcionality komponent. Negatívne hodnotím voľbu obojsmernej datovej zbernice s trojstavovým výstupom, odladenie trojstavového výstupu bolo veľmi komplikované. Lepšie by bolo použiť oddelenú linku pre dáta na zápis a dáta na čítanie. Obojsmernú dátovú linku som volil z dôvodu jednoduchej pripojiteľnosti v schematickom editore Quartusu. Napriek tejto komplikácii hodnotím moju bakalársku prácu ako vhodne a úspešne implementovanú.

Literatúra

[1] Martin Paták, Methods for Testing FlexRay Start-Up Mechanism, Diploma thesis , Prague 2012

[2] Jiří Blecha, Programovatelná testovací platforma, Diplomová práce, Praha 2014

[3] FlexRay Consortium, FlexRay Communications System Protocol specification, Version 2.1 Revision A, 2005

[4] Jiří Pinker, Martin Koupa, Číslicové systémy a jazyk VHDL, BEN 2006

[5] Texas Instruments, TMS320DM647/DM648 DSP External Memory Interface (EMIF) User's guide, 2008

[6] Lupini, C.A.: Vehicle Multiplex Communication, SAE International 2004, ISBN 0-7680-1218-X

[7] Schäuffele, J., Zurawka, T.: Automotive Software Engineering, SAE International 2005, ISBN 0-7680-1490-5

[8] Texas Instruments, Datasheet k procesoru TMS570LS3137, Revision A, 2012

[9] Philips Semiconductors, Datasheet k obvodu TJA1080, Revision 1, 2006

[10] NXP Semiconductors, Datasheet k obvodu TJA1041, Revision 6, 2007

[11] Texas Instruments, Datasheet k obvodu TPIC1021, Revision 1, 2005

[12] Altera, Bidirectional Bus Example, http://www.altera.com/support/examples/vhdl/v_bidir.html

[13] Texas Instruments, TMS570LS31x/21x 16/32-Bit RISC Flash Microcontroller, Reference Manual, 2012

[14] Altera, Datasheet k obvodu EP4CE55F8N, 2012

Príloha A - Obsah priloženého CD

CD:

- /Pin Assignments - Obsahuje mená pinov priradené k číslam pinov v BGA puzdre.
- /VHDL Kódy - Obsahuje kódy naprogramovaných komponent spolu s použitými zdrojmi z [1].
- /Projekt Code Composer - Obsahuje projekt s programom pre procesor v jazyku C.
- /Projekty Quartus - Obsahuje hlavný quartus v ktorom je aj implementovaný príkladný systém v schéme : System_Device_Implementation.bdf
- /Simulácie Komponent - Obsahuje zdrojové simulačné súbory pre vyvíjané komponenty a

Príloha B - Zapojenie pinov obvodu EP4CE55F23C8N

Meno signálu	Pin FPGA	Meno signálu	Pin FPGA
CAN1_EN	,PIN_Y8	FRAY1.2_STBN	,PIN_W17
CAN1_ERR	,PIN_AB7	FRAY1.2_TXD	,PIN_AB19
CAN1_RXD	,PIN_W8	FRAY1.2_TXEN	,PIN_AA18
CAN1_STB	,PIN_AB8	FRAY1.2_WAKE	,PIN_V16
CAN1_TXD	,PIN_AA8	FRAY2.1_BGE	,PIN_Y22
CAN1_WAKE	,PIN_V8	FRAY2.1_EN	,PIN_W22
CAN2_EN	,PIN_V10	FRAY2.1_ERRN	,PIN_AA20
CAN2_ERR	,PIN_AA9	FRAY2.1_RXEN	,PIN_AA21
CAN2_RXD	,PIN_AB9	FRAY2.1_STBN	,PIN_Y21
CAN2_STB	,PIN_W10	FRAY2.1_TXD	,PIN_W21
CAN2_TXD	,PIN_Y10	FRAY2.1_TXEN	,PIN_W19
CAN2_WAKE	,PIN_V9	FRAY2.1_WAKE	,PIN_AB20
CAN3_EN	,PIN_V11	FRAY2.2_BGE	,PIN_U21
CAN3_ERR	,PIN_AB10	FRAY2.2_EN	,PIN_T19
CAN3_RXD	,PIN_AB13	FRAY2.2_ERRN	,PIN_U20
CAN3_STB	,PIN_AA13	FRAY2.2_RXD	,PIN_T18
CAN3_TXD	,PIN_V12	FRAY2.2_RXEN	,PIN_V22
CAN3_WAKE	,PIN_AA10	FRAY2.2_STBN	,PIN_U19
CAN4_EN	,PIN_V13	FRAY2.2_TXD	,PIN_T20
CAN4_ERR	,PIN_Y13	FRAY2.2_TXEN	,PIN_U22
CAN4_RXD	,PIN_AB14	FRAY2.2_WAKE	,PIN_V21
CAN4_STB	,PIN_W14	LIN1_EN	,PIN_Y4
CAN4_TXD	,PIN_AA14	LIN1_NWAKE	,PIN_AB3
CAN4_WAKE	,PIN_W13	LIN1_RXD	,PIN_AA4
FPGA_CLK	,PIN_A11	LIN2_EN	,PIN_V6
FPGA_RESET	,PIN_G5	LIN2_NWAKE	,PIN_AA5
FRAY1.1_BGE	,PIN_W15	LIN2_RXD	,PIN_AB5
FRAY1.1_EN	,PIN_AA16	LIN2_TXD	,PIN_AB4
FRAY1.1_ERRN	,PIN_V14	LIN3_EN	,PIN_AA6
FRAY1.1_RXD	,PIN_Y15	LIN3_NWAKE	,PIN_W6
FRAY1.1_RXEN	,PIN_AB15	LIN3_RXD	,PIN_V7
FRAY1.1_STBN	,PIN_AA15	LIN4_EN	,PIN_W7
FRAY1.1_TXD	,PIN_AB16	LIN4_RXD	,PIN_AA7
FRAY1.1_TXEN	,PIN_V15	LIN4_TXD	,PIN_AB6
FRAY1.1_WAKE	,PIN_Y14	LIN4_WAKE	,PIN_Y7
FRAY1.2_BGE	,PIN_Y17	MCU_CAN1_EN	,PIN_A8
FRAY1.2_EN	,PIN_AA19	MCU_CAN1_ERRN	,PIN_B13
FRAY1.2_ERRN	,PIN_AA17	MCU_CAN1_RXD	,PIN_E11
FRAY1.2_RXD	,PIN_AB18	MCU_CAN1_STB	,PIN_B10
FRAY1.2_RXEN	,PIN_AB17	MCU_CAN1_TXD	,PIN_E12

Tabuľka 8.1: Pripojenie signálov FPGA (1. časť)

Meno signálu	Pin FPGA	Meno signálu	Pin FPGA
MCU_CAN1_WAKE	,PIN_C10	MCU_EMIF_DATA[12]	,PIN_A20
MCU_CAN2_EN	,PIN_H19	MCU_EMIF_DATA[13]	,PIN_C20
MCU_CAN2_ERR	,PIN_J19	MCU_EMIF_DATA[14]	,PIN_C19
MCU_CAN2_RXD	,PIN_J20	MCU_EMIF_DATA[15]	,PIN_B20
MCU_CAN2_STB	,PIN_H21	MCU_EMIF_DATA[2]	,PIN_B1
MCU_CAN2_TXD	,PIN_J21	MCU_EMIF_DATA[3]	,PIN_B3
MCU_CAN2_WAKE	,PIN_H20	MCU_EMIF_DATA[5]	,PIN_M21
MCU_CAN3_EN	,PIN_D1	MCU_EMIF_DATA[6]	,PIN_M19
MCU_CAN3_ERR	,PIN_B2	MCU_EMIF_DATA[7]	,PIN_H22
MCU_CAN3_RXD	,PIN_C2	MCU_EMIF_DATA[8]	,PIN_F19
MCU_CAN3_STB	,PIN_D2	MCU_EMIF_DATA[9]	,PIN_D21
MCU_CAN3_TXD	,PIN_C3	MCU_EMIF_NCS[2]	,PIN_E6
MCU_CAN3_WAKE	,PIN_E4	MCU_EMIF_NCS[3]	,PIN_E3
MCU_EMIF_ADDR[0]	,PIN_C18	MCU_EMIF_NCS[4]	,PIN_C4
MCU_EMIF_ADDR[1]	,PIN_A18	MCU_EMIF_NDQM[0]	,PIN_A10
MCU_EMIF_ADDR[10]	,PIN_B14	MCU_EMIF_NDQM[1]	,PIN_A9
MCU_EMIF_ADDR[11]	,PIN_D13	MCU_EMIF_NOE	,PIN_B8
MCU_EMIF_ADDR[12]	,PIN_A13	MCU_EMIF_NWAIT	,PIN_C22
MCU_EMIF_ADDR[13]	,PIN_D10	MCU_EMIF_NWE	,PIN_F2
MCU_EMIF_ADDR[14]	,PIN_B9	MCU_FR_1_EN	,PIN_E2
MCU_EMIF_ADDR[15]	,PIN_A7	MCU_FRAY_1_ERRN	,PIN_C8
MCU_EMIF_ADDR[16]	,PIN_A6	MCU_FRAY_1_RX_EN	,PIN_B7
MCU_EMIF_ADDR[17]	,PIN_D7	MCU_FRAY_1_RXD	,PIN_B6
MCU_EMIF_ADDR[2]	,PIN_E16	MCU_FRAY_2_ERRN	,PIN_B18
MCU_EMIF_ADDR[20]	,PIN_B5	MCU_FRAY1_STBN	,PIN_E1
MCU_EMIF_ADDR[21]	,PIN_A3	MCU_FRAY1_TXD	,PIN_D6
MCU_EMIF_ADDR[3]	,PIN_A16	MCU_FRAY1_TXEN	,PIN_A4
MCU_EMIF_ADDR[4]	,PIN_A15	MCU_FRAY1_WAKE	,PIN_F7
MCU_EMIF_ADDR[5]	,PIN_A14	MCU_FRAY2_EN	,PIN_B17
MCU_EMIF_ADDR[6]	,PIN_D18	MCU_FRAY2_RXD	,PIN_E13
MCU_EMIF_ADDR[7]	,PIN_D17	MCU_FRAY2_RXEN	,PIN_B19
MCU_EMIF_ADDR[8]	,PIN_B16	MCU_FRAY2_STBN	,PIN_C17
MCU_EMIF_ADDR[9]	,PIN_C15	MCU_FRAY2_TXD	,PIN_E14
MCU_EMIF_ADDR[18]	,PIN_A5	MCU_FRAY2_TXEN	,PIN_C13
MCU_EMIF_BA[0]	,PIN_C7	MCU_FRAY2_WAKE	,PIN_A19
MCU_EMIF_BA[1]	,PIN_B4	MCU_LIN_EN	,PIN_E15
MCU_EMIF_DATA[0]	,PIN_E5	MCU_LIN_NWAKE	,PIN_A17
MCU_EMIF_DATA[1]	,PIN_C1	MCU_LIN_RXD	,PIN_B15
MCU_EMIF_DATA[10]	,PIN_B22	MCU_LIN_TXD	,PIN_D15
MCU_EMIF_DATA[11]	,PIN_B21		

Tabuľka 8.2: Pripojenie signálov FPGA (2. časť)