VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

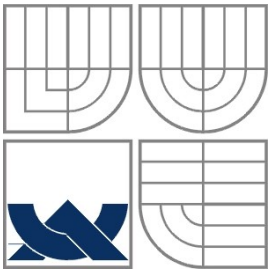
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

ROZHRANÍ ETHERNET PRO VÝUKOVOU HW/SW PLATFORMU FITKIT

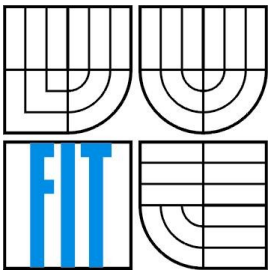
BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE RENÉ KOLAŘÍK
AUTHOR

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

ROZHRANÍ ETHERNET PRO VÝUKOVOU HW/SW PLATFORMU FITKIT

ETHERNET INTERFACE FOR TEACHING HW/SW PLATFORM FITKIT

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

RENÉ KOLAŘÍK

VEDOUCÍ PRÁCE
SUPERVISOR

Dr. Ing. OTTO FUČÍK

BRNO 2008

Abstrakt

Bakalářská práce se zabývá rozšířením výukové platformy FITKit o ethernetové rozhraní a volbou způsobu pro tento účelu. Dále se zabývá hardwarovou a softwarovou implementací zvoleného modulu a základních síťových protokolů.

Klíčová slova

ethernet, ENC28J60, FITKit, vestavěný systém, SPI

Abstract

Bachelor thesis deals with extending of the teaching platform FITKit with the ethernet interface and choosing appropriate concept for this purpose. Further deals with hardware and software implementation of the chosen module and some of the basic network protocols.

Keywords

ethernet, ENC28J60, FITKit, embedded system, SPI

Citace

René Kolařík: Rozhraní ethernet pro výukovou HW/SW platformu FITKit, bakalářská práce, Brno, FIT VUT v Brně, 2008

Rozhraní ethernet pro výukovou HW/SW platformu FITKit

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Dr. Ing. Otto Fučíka.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jméno Příjmení
Datum

Poděkování

Děkuji Dr. Ing. Otto Fučíkovi za odbornou pomoc při tvorbě práce, konzultace a rady týkající se vytváření technické dokumentace.

© René Kolařík, 2008.
Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.

Obsah

Obsah....1
1 Úvod....2
2 Ethernet....3
3 Návrh....4
3.1 Ethernetový kontrolér....4
3.2 FITKit a možnosti postupu....4
3.3 Protokoly vyšších vrstev....5
3.3.1 UDP protokol....5
3.3.2 IP protokol....5
3.3.3 ICMP protokol....5
3.3.4 ARP protokol....6
4 Modul SPINET....7
4.1 Popis modulu....7
4.2 Obvod ENC28J60....8
4.2.1 Příkazy SPI....8
4.2.2 Registry obvodu....9
4.2.3 Sdílená paměť....10
5 Realizace....12
5.1 Propojení....12
5.2 VHDL design....13
5.2.1 Entita spinet_adc....13
5.2.2 Celkovy design....14
5.3 Software mikrokontroléru....15
5.3.1 Knihovna enc28j60....15
5.3.2 Knihovna net....18
5.3.3 Vlastní program mikrokontroléru....19
5.4 Ukázkové aplikace....20
5.4.1 Ping....20
5.4.2 Skripty v PERLu....20
6 Závěr....21
Literatura....22
Seznam příloh....23

1 Úvod

Ethernet je protokolem síťové vrstvy a v instalacích lokálních sítí má dominantní postavení. Rozšíření výukoveho kitu o toto rozhraní položí základ implementacím protokolů vyšších vrstev a tím umožní ještě širší využití kitu.

V následujících kapitolách se nejprve zběžně popíšu některé vlastnosti ethernetu, poté se budu zabývat možnostmi přístupu k návrhu ethernetového kontroléru a možnými způsoby implementace ethernetového rozhraní pro FITKit. Dále popíšu modul zvolený k realizaci tohoto rozhraní a důvody jeho výběru. Následně proberu postup realizace hardwarového a softwarového propojení modulu s FITKitem, popíšu implementované knihovny a na závěr uvedu dvě drobné ukázky aplikace práce.

2 Ethernet

Ethernet je v současné době nejrozšířenějším standartem pro komunikaci na lokálních sítích. Z původního návrhu ethernetu, vyvinutého firmou Xerox, který používal tlustý koaxiální kabel a topologii lineární sběrnice, vznikl standart IEEE 802.3 používaný dnes.

Přístup zařízení sdílejících společné ethernetové médium je řízen pomocí algoritmu CSMA/CD. Ten je přibližně následující: zařízení čeká, dokud na médiu nedetekuje klid, poté začne s vysíláním. Pokud se na začátku vysílání vyskytne kolize (během tzv. kolizního okna), tzn. několik zařízení se pokusí vysílat společně, je jimi tato kolize detekována, všechna zařízení se stáhnou a čekají náhodnou dobu, než opět začnou vysílat. Pokud nenastane kolize, zařízení odvysílá svá data a určenou dobu čeká, než opět může začít vysílat. Zde se může vyskytnout ještě jeden problem a tím jsou tzv. pozdní kolize. To je kolize která se vyskytne po skončení kolizního okna, tj. v době, kdy zařízení očekává klid a předpokládá, že médium náleží jemu. Pozdní kolize jsou nežádoucí.

Formát odesílaných dat – ethernetového rámce – je následující: nejprve je odesláno sedm slabik tzv. preambule, což jsou slabiky o hodnotě 0xAA, nasleduje slabika tzv. *start of frame delimiter*, mající hodnotu 0xAB.

Poté jsou odesílána vlastní data ethernetového paketu. Šest slabik cílové hardwarové adresy, následuje šest slabik adresy zdrojové. Dále je odesláno dvouslabikové pole označující buď typ dat přenášených uvnitř paketu, nebo jejich délku. Následuje odeslání vlasních dat uvnitř paketu. Dolní hranice velikosti ethernetového paketu je 64 slabik, horní pak záleží na hodnotě MTU, maximálně však 1518 slabik včetně CRC. To je čtyřslabikové, je vloženo na konec paketu a vypočítává se z celého jeho obsahu.

3 Návrh

3.1 Ethernetový kontrolér

Strukturu samotného ethernetového kontroléru můžeme rozdělit zhruba do tří hlavních částí:

Část spravující linkovou vrstvu – MAC. Ta má za úkol např. řízení přístupu k médiu (u ethernetu je to uváděné CSMA/CD) nebo také adresování jednotlivých zařízení sítě.

MAC část býva propojena obvykle pomoci standartizovaného rozhraní MII – media independent interface s částí spravující vrstvu fyzickou – PHY. Ta je napojena na vlastní médium a obstarává např. kódování a dekódování odesílaných a přijímaných informací apod. Určuje také standarty na kterých je kontrolér schopný komunikovat (10Base-T, 100Base-TX atd.).

Třetí částí jsou externí obvody jako ethernetová trafa a různé pasivní součástky. Také bych sem zařadil části obvodů nebo obvody definující rozhraní kontroléru a určující tak, jakým způsobem s ním hostující systém bude komunikovat (PCI, SPI ...).

Problém návrhu ethernetového kontroléru, nebo obecně jakéhokoliv složitějšího systém a jeho implementace v systému cílovém spádá do oblasti HW/SW codesignu – tzn. máme za úkol nalezení optimálního rozdělení návrhu mezi hardwarové komponenty a software. Roli v rozhodování hrají různé faktory - např. dostupnost, cena a vhodnost již existujících hardwarových modulů. Dále charakteristiky cílového systému, dostupná rozhraní a přidělené zdroje, jako je výpočetní síla, velikost paměti, velikost hradlového pole atd. Je vhodné zvážit také požadavky na případné budoucí změny nebo rozšiřování navrhovaného systému.

3.2 FITKit a možnosti postupu

Jádrem FITKitu je MCU. Ten bude k realizaci ethernetového kontroléru určitě použit. Dále se můžeme se rozhodovat, zda využijeme hradlové pole spojené sběrnici SPI s MCU. V případě využití pouze mikrokontroléru, bude výsledný systém složený z obslužného programu v MCU a kontroléru připojeného na

rozhraní JP9. Rozhodování dále bude spočívat v tom, jestli kontrolér sestavíme sami z dodaných MAC a PHY obvodů, nebo pořídíme modul již hotový. Pravděpodobně by bylo také možné implementovat MAC logiku do MCU, otázka ale je, jak náročné by to bylo na zdroje.

Jinou, pravděpodobně lepší, variantou je využití FPGA. Tím nám bude umožněno přenést sem část ovládání kontroléru a uvolnit tak prostředky MCU. Poté se opět můžeme rozhodovat, jak implementujeme hardwarovou část kontroléru, která bude v tomto případě připojená na rozhraní JP10.

3.3 Protokoly vyšších vrstev

Cílem mé bakalářské práce je i to, aby na jejím konci byl pro FITKit k dispozici soubor alespoň základních funkcí síťového rozhraní. K tomu je ale třeba navíc k ethernetovému kontroléru a jeho obslužným rutinám implementovat i vyšší protokoly ISO/OSI modelu.

3.3.1 UDP protokol

UDP se jako bezrelační protokol s minimalistickou hlavičkou hodí pro použití jako základní protokol transportní vrstvy v aplikacích běžících na mikrontroléru. Absence nutnosti udržovat relaci zjednodušuje návrh a šetří zdroje.

3.3.2 IP protokol

Implementace tohoto prokolu síťové vrstvy je nutná, pokud chceme nasledně využívat kit v IP sítích. Což určitě chtít budeme.

Implementuji protokol IP verze čtyři a pravděpodobně si problém zjednoduším absencí fragmentace paketů.

3.3.3 ICMP protokol

Tento protokol je součástí IP protokolu a jeho využití není bezpodmínečně nutné pro funkčnost, ale je vhodné implementovat alespoň základní typy zpráv tohoto protokolu, jako je:

- **echo reply** - čímž si zajistíme možnost kontrolovat síťové spojení s FITKitem

- **destination unreachable** - minimálně proto, že server by měl odeslat tuto zprávu klientovi, pokud se pokusí odeslat data na uzavřený port.

3.3.4 ARP protokol

Univerzálnost protokolu IP, který lze provozovat na různých druzích linkových protokolů přináší nutnost překladu z adres IP na adresy použitého protokolu linkové vrstvy. Toto obstarává protokol ARP.

V případě, že je třeba odeslat data na zadanou IP adresu, je nejdříve otestováno, zda se daná adresa nachází v ARP cache. Pokud ne, je na všesměrovou adresu linkové vrstvy odeslán „ARP request“, který obsahuje zadanou IP adresu cíle, IP adresu odesilatele a jeho adresu linkové vrstvy.

Cíl si uloží do své ARP cache údaje o odesilateli a pošle mu zpět „ARP reply“ se svými údaji.

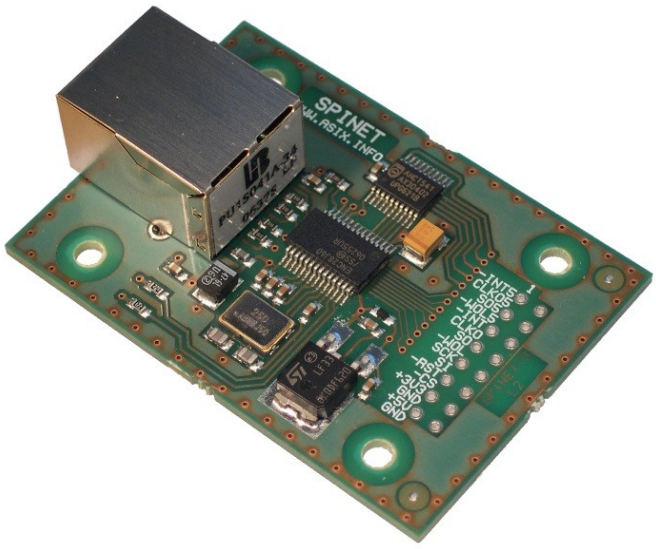
4 Modul SPINET

4.1 Popis modulu

SPINET je modul postavený na obvodu ENC28J60 firmy Microchip. Výrobcem je firma Asix. Umožňuje snadnou správu síťové a fyzické vrstvy připojení pomocí rozhraní SPI.
Důvody, které mě vedly k volbě tohoto modulu jsou následující:

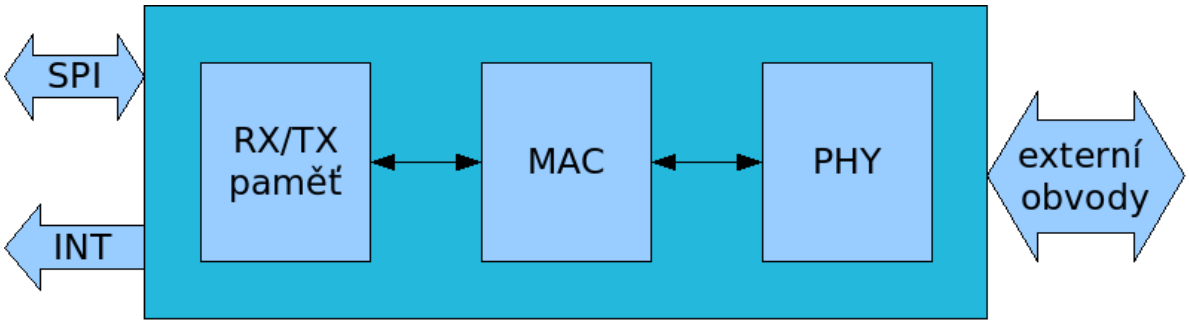
- jedním z bodů zadání je implementace s přihlédnutím ke zdrojům, použití externího modulu ušetří hlavně prostředky FPGA, ve kterém tak zbyde místo i pro implementaci jiných kontrolérů než jen ethernetového
- hlavní součást modulu, obvod ENC28J60 by mohl být, díky svým rozměrům, vhodným adeptem pro umístění na některou další verzi FITKitu. Potom, podle případného umístění, pokud by byl umístěn paralelně s FPGA a flash pamětí, stačilo by upravit knihovnu enc28j60, ale pokud by byl umístěn až za FPGA nemusela by se knihovna upravovat skoro vůbec (až na části, které jsou nyní implementovány kvůli kompatibilitě s erraty – ty by mohly být odstraněny).

Modul umožňuje napájení pomocí 3,3 a 5 voltů. Výstupy jsou zdvojené pro obě úrovně a vstupy jsou 5V-tolerantní. Další vlastnosti jsou určeny obvodem ENC28J60, který je jádrem modulu.



4.2 Obvod ENC28J60

Tento obvod zastává funkci ethernetového kontroléru. Na obrázku můžeme vidět zjednodušené blokové schéma.



Obvod zajišťuje jednoduché odesílání a přijímání paketů, vyhovuje standartu IEEE 802.3. Může pracovat v jak poloduplexním, tak v plně duplexním režimu. Obvod volitelně automaticky zajistí doplnění dat do povinných 64 slabik ethernetového rámce. Podporuje také automatický výpočet hodnoty CRC. Obsažený PHY modul pracuje ve standartu 10BASE-T.

Další zajímavé vlastnosti obvodu jsou např. možnost filtrovat příchozí pakety nebo přítomnost integrovaného DMA modulu, umožňujícího rychlé přesuny v paměti.

4.2.1 Příkazy SPI

Komunikace s obvodem a přenos dat jsou řešeny výhradně pomocí sběrnice SPI a volitelně pomocí šesti zdrojů přerušení vyvedených na jeden vývod.

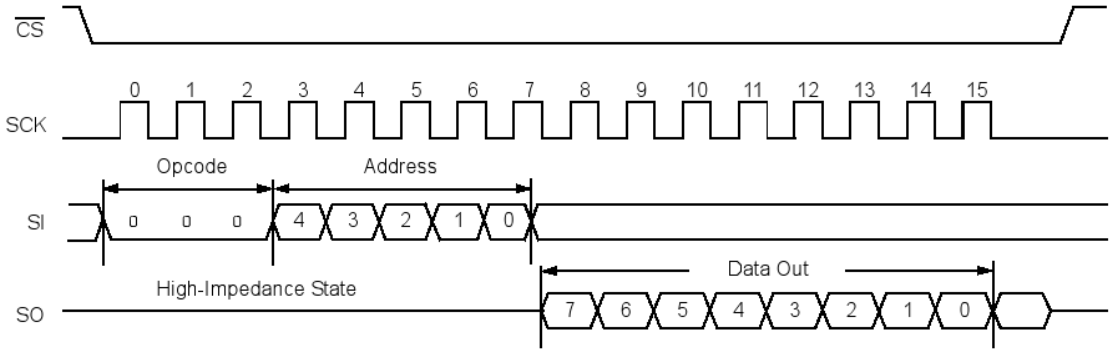
Rozhraní SPI není konfigurovatelné, signál CS je aktivní v nule, data jsou vzorkována náběžnou stranou hodin, které mají klidový stav v nule.
Pro ovládání obvodu lze použít tyto příkazy SPI:

- zápis do kontrolního registru / čtení z kontrolního registru
- nastavení / smazání bitů kontrolního registru
- zápis do sdílené paměti / čtení ze sdílené paměti
- softwarový reset

4.2.2 Registry obvodu

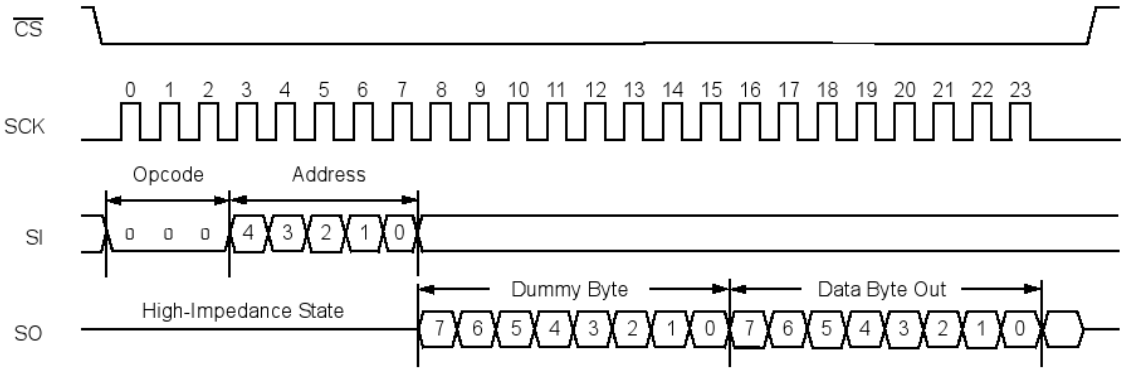
Registry obvodu se dělí na tři hlavní skupiny, podle způsobu, jakým z nich lze číst popř. do nich zapisovat data.

- **Registry skupiny ETH**, začínající písmenem 'E' – postup čtení i zápisu je standartní tzn. při čtení se odešle slabika kombinující operační znak a adresu a hned po ní jsou obvodem na sběrnici vysunuta data, při zápisu následuje slabiku s operačním znakem a adresou slabika s daty. Viz obrázek z datasheetu:



Zápis do registru ETH

- **Registry MAC a MII**, začínající na 'MA' nebo 'MI' – postup zápisu je stejný jako u předchozí skupiny, čtení se však liší tím, že po odeslání slabiky s operačním znakem a adresou, je nejprve obvodem vysunuta 'dummy' slabika a až poté slabika s daty. Viz obrázek z datasheetu:



Zápis do registru MAC a MII

- **Registry PHY** – začínající na 'PH' – zápis a čtení registrů PHY části obvodu se neprovádí přímo pomocí SPI příkazů, ale pomocí registrů MII rozhraní, kdy se nejprve do registru MIREGADR zapíše adresa zvoleného PHY registru, poté pokud zapisujeme data, vložíme je do dvojice registrů MIWRH:MIWRL, pokud čteme data, máme je po skončení operace uložena v dvojici registrů MIRDH:MIRDL (podrobněji viz datasheet).

4.2.3 Sdílená paměť

Obvod obsahuje 8 Kbyte paměti, která může byt podle potřeby rozdělena na část pro ukladání došlých paketů a část pro ukládání dat k odeslání

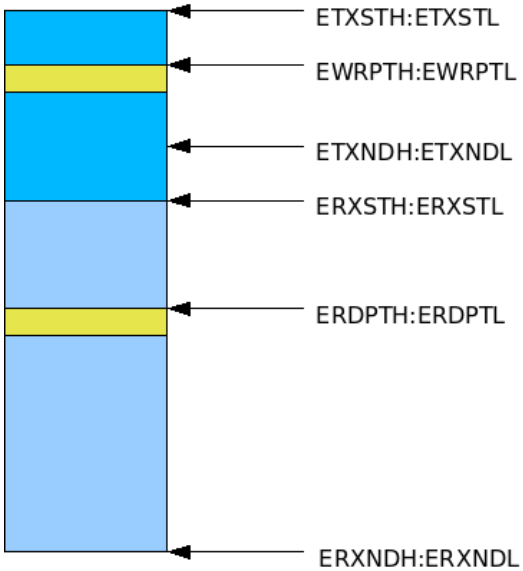


Schéma paměti obvodu

Část určená pro ukládání došlých paketů je vymezena dvojicí registrových párů – ERXSTH:ERXSTL pro uložení začátku a ERXNDH:ERXNDL pro uložení konce. Ty je třeba na začátku práce s obvodem inicializovat na požadované hodnoty. Aktuálni pozici čtecího ukazatele určuje pár ERDPTH:ERDPTL. Přijímací paměť funguje jako cyklická fifo fronta.

Zbytek sdílené paměti, nevyužívaný přijímací pamětí je využit pro odesílací paměť. Není třeba jej inicializovat. Při odesílání paketu pouze uložíme počáteční adresu odesílaných dat do ETXSTH:ETXSTL a adresu poslední slabiky do ETXNDH:ETXNDL. Data mezi těmito dvěma ukazateli budou odeslána jako

jeden paket. Do paměti zapisujeme ždy na místo určene registrovým párem EWRPTH:EWRPTL – zápisový ukazatel.

Oba dva ukazatele, jak zápisový, tak čtecí, mohou být (a ve výchozím nastavení jsou) nastaveny tak, aby se po přečtení, nebo zapsaní jedné slabiky automaticky inkrementovaly. Což umožňuje plynulé čtení bez nutnosti neustále odesílat operační kód pro čtení nebo psaní.

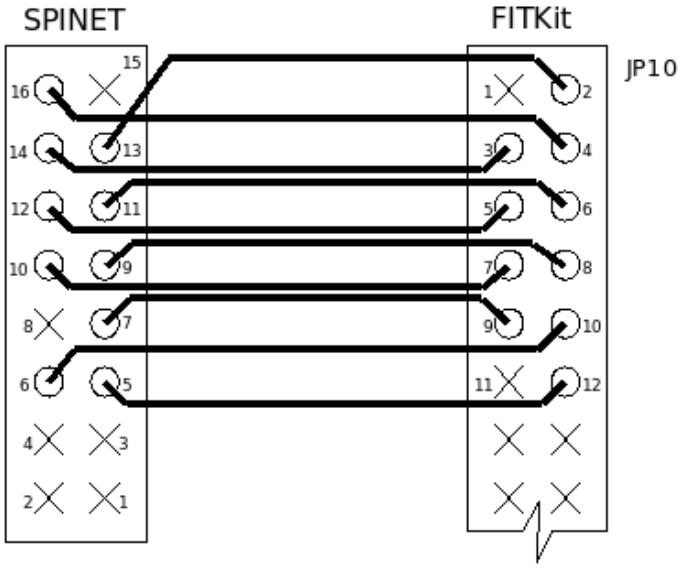
Dalším ukazatelem, který není na obrázku zobrazen, je ERXRDPTH:ERXRDPTL. Tento ukazatel označuje začátek nezpracovaných dat a brání tak jejich přepsání. Je třeba jej po zpracování určité časti dat přesunout za zpracovaná data, v opačném případě by obvodu brzy došla paměť.

5 Realizace

5.1 Propojení

Modul SPINET není připojen přímo k mikrokontroléru, ale přes FPGA je připojen na rozhraní JP10. Využívá zde dostupných 3,3V a taktéž zapojeny jsou jen 3,3V výstupy.

Toto řešení je vhodné zejména proto, že umožňuje do budoucna možnost přenést část funkčnosti, obstarávané mikrokontrolérem do FPGA a uvolnit tak paměťové a výpočetní prostředky mikrokontroléru.



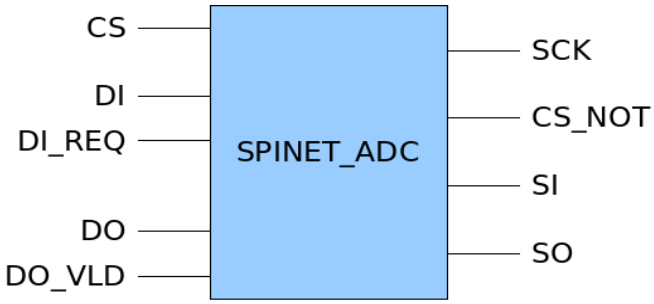
SPINET pin	*FITKit JP10 pin*	*Název*	*Top modul*
13	2	3,3V	-
14	3	GND	-
16	4	GND	-
12	5	-CS	X0
11	6	-RST	X1
10	7	SI	X2
9	8	SCK	X3
7	9	SO	X4
6	10	CLKO	X5
5	12	-INT	X7

5.2 VHDL design

5.2.1 Entita spinet_adc

Pro ovládání vstupů modulu SPINET jsem se rozhodl, jako základní stupeň, použít již existující entitu SPI dekodéru a nahradit její paralelní výstup generováním signálů SPI.

Tato upravená entita umožňuje ovládat modul pomocí interní seriové sběrnice, stejně, jako ostatní kontroléry (LCD, klavesnice).



Vstupy jsou tedy zachovány, vystupy jsou následující:

- **SCK** – taktovací signál pro SPI přenos – je generován na základě signálu DI_REQ a DO_VLD, kdy při nástupem signálu DO_VLD do '1' je signál SCK nastaven na '1' a nástupem DI_REQ do '1' je signál nastaven do '0'. Takto můžeme zrekonstruovat původní SPI takt přichazející z mikrokontroléru a jen v době, kdy to skutečně potřebujeme, tj. v době přenosu dat. Vyplývá z toho ale také, že všechny přenosy mezi modulem SPINET a mikrokontrolérem musí být uskutečňovány v módu R i W, tj. první odeslaná slabika musí být 0x13.
- **CS_NOT** – je prodloužením signálu CS do modulu
- **SI / SO** – vstup a výstup seriových dat

Zbývající vstupy a výstupy modulu jsou zapojeny následovně

- **-RST** – na tento výstup je přípojen negovaný signál resetu používany uvnitř FPGA
- **-INT** – vstup přerušení je nezapojen
- **CLKO** – výstup hodinového signálu z modulu je nezapojen, ve své práci jej nevyužívám

5.2.2 Celkovy design

V práci jsem použil kromě výše zmíněné entity, ještě kontrolér LCD a kontrolér klávesnice. V případě, kdy bych chtěl využít přerušení modulu, bylo by třeba použít ještě kontrolér přerušení.

Dalšími částmi návrhu jsou čtyři soubory

- arch_spinet_ifc.vhd
- fpga_xc3s50_spinet.ucf
- fpga_xc3s50_spinet.vhd
- tlv_spinet_ifc.vhd

Tyto čtyři soubory vznikly upravením odpovídajících souborů, majících ve svém názvu '*ide*' místo '*spinet*'. Rozdíl mezi původními soubory a v návrhu použitými je, že ze sběrnice X bylo odděleno prvních osm signálu a byly osamostatněny. Jméno bylo pro zachování orientace ponecháno jako X0 – X7 a dále zbylým signálům byl taktéž ponechán původní index, tj. sběrnice X je v tomto designu definována jako *std_logic_vector(45 downto 8)*. Takto upraven je i vektor sběrnice IDE. Jeho prvních osm signálu není označeno jako v předchozím případě, ale je pojmenováno, podle signálu modulu SPINET se kterými jsou spojeny.

5.3 Software mikrokontroléru

5.3.1 Knihovna enc28j60

Tato knihovna obstarává základní funkce pro použití modulu SPINET a komunikaci s obvodem ENC28J60 přes rozhraní SPI. Obsažené funkce lze rozdělit do tří skupin.

První jsou funkce nejnižší úrovně, zajišťující zápis, čtení hodnot registrů a nastavování a mazání jednotlivých bitů registrů ETH, MAC a MII části modulu a dále zápis a čtení registrů PHY části modulu.

Zde se vyskytl asi největší problém v průběhu realizace a sice to, že funkce pro zápis do některých registrů někdy nefungovaly správně nebo vůbec. Tuto nefunkčnost přikládám chybě v obvodu, o které píšou errata a která se vztahuje k mnou používané revizi (rev. 4) obvodu. Touto chybou je nutnost, aby SPI hodinový signál běžel na frekvenci alespoň osm MHz, v opačném případě je zápis a čtení registrů MAC a MII nespolehlivé.

V případě nemožnosti zapojení uvedených osmi Mhz (což u FITKitu nejde, maximální rychlost SPI je 3,6864 Mhz), uvádějí errata jako možnost řešení použití výstupu CLKO k taktování hodin SPI. Problém jsem nakonec obešel řešením, které jsem nalezl víceméně experimentálně. Tímto řešením je implementace funkce pro zápis do registru ve dvou verzích. První verze standartně odešle první slabiku, tj. operační znak plus adresu registru a poté slabiku s daty a ukončí přenos. Druhá verze po odeslání slabiky s daty odešle dvakrát slabiku nulovou.

Dále bylo nutné upravit pořadí zápisu do registrů v inicializační funkci, aby všechny registry byly správně zinicializovány. Pokud si například v současné funkční konfiguraci spustíme na FITKitu příkaz '*debug_init*' a poznamenáme si správne hodnoty registrů, nasledně zkusíme ve funkci nic_init pozměnit pořadí jejich inicializace, přeložíme, nahrajeme a opět spustíme, je pravděpodobné že nedostaneme uplně stejné výsledky.

Přesto, že nakonec jsem nalezl funkční pořadí inicializace a funkce, které vedou ke správnému zapisu do registrů, existují minimálně dva registry, do kterých se mi nepodařilo zapsat – MAADR5 a MAADR6. Tyto dva registry udržují poslední dvě slabiky z hardwarove adresy. Ta bude prot vždy končit dvěma nulami. Toto je tedy moje řešení, ale případných budoucích verzích, by

však bylo vhodné upravit VHDL design a využít řešení se signálem CLKO. Funkce náležící do této skupiny jsou tyto:

- **enc28_ctrlreg_rd** – čtení z kontrolního registru – ETH, MAC, MII
- **enc28_ctrlreg_wr** – zápis do kontrolního registru, verze, kde odeslání slabiky s daty následuje odeslání dvou nulových slabik
- **enc28_ctrlreg_wrE** – zápis do kontrolního registru, standartní verze, snažím se používat tuto, někdy to ale nefunguje a je třeba použít předchozí
- **enc28_ctrlreg_bfs** – nastavení bitů v kontrolním registru podle zadané masky
- **enc28_ctrlreg_bfc** – smazání bitů v kontrolním registru podle zadané masky
- **enc28_phyreg_rd** – čtení registru PHY
- **enc28_phyreg_wr** – zápis do registru PHY

Další skupina funkcí zajišťuje obslužné funkce sdílené TX / RX paměti, jako je čtení a zápis jedné slabiky, čtení a zapis bufferu, přesuny zápisových a čtecích ukazatelů v paměti apod. Důležitou vlastností těchto funkcí je, že nechávají otevřené SPI připojení k paměti a to proto, že tím využívají kombinace vlastnotí paměťových ukazatelů, které se automaticky inkrementují a příkazů SPI pro zápis a čtení, kde pokud pokračujeme s dodáváním hodin, obvod stále vysunuje další data, tzn. ušetříme čas na odesílání adresy zařízení v FPGA a operačního kodů a inkrementaci ukazatelů. Nyní funkce z této skupiny:

- **netbuf_putc** – zápis slabiky na aktuální pozici zápisového ukazatele
- **netbuf_getc** – čtení slabiky z aktuální pozice čtecího ukazatele
- **netbuf_write** – zápis pole určené délky na aktuální pozici zápisového ukazatele
- **netbuf_read** – čtení pole určené délky z aktuální pozice čtecího ukazatele
- **netbuf_wr_seek** – přesun zápisového ukazatelem
- **netbuf_rd_seek** – přesun čtecího ukazatele
- **netbuf_close** – uzavření paměti po skončení zápisu nebo čtení

Poslední skupinou jsou funkce rozhraní knihovny, konkrétně:

- **nic_init** – funkce inicializuje modul do poloduplexního režimu, nastavuje všechny potřebné kontrolní registry a registry ukazatelů do paměti, nastavuje také registry udržující MAC adresu modulu, přesněji první čtyri, do zbylých dvou nelze zaručit zapsání správné hodnoty, tak je nechávám ve stavu jakém jsou po resetu, tj. v nule
- **nic_transalloc** – přesune zápisovy pointer (EWRPT) na začátek TX paměti a zapíše nulový kontrolní byte, ovlivňující odesílání každého paketu
- **nic_transmit** – nastaví ukazatel na konec TX paměti (ETXND) na konec zapsaných dat a data odešle
- **nic_receive** – funkce zjistí, zda přišly nové pakety, pokud ano, vrátí velikost prvního na řadě
- **nic_recfree** – po zpracování každého paketu je nutné volat tuto funkci, která označí paket jako zpracovaný tím, že přesune ERXRDPT ukazatel na jeho konec (a začátek dalšího paketu) a sníží interní počítadlo nezpracovaných paketů o jedna

Typické použití této knihovy, metodou dotazování, je následující:

- inicializujeme modul pomocí *nic_init*
- odesílání:
  1. funkcí *nic_transalloc* připravíme odesílací paměť
  2. zapisujeme do ní pomocí zápisových funkci s předponou '*netbuf_*'
  3. když jsme hotvi, odešleme zapsané množství dat pomocí funkce *nic_transmit*
- příjem:
  1. příjem dat realizujeme nejprve dotazem funkcí *nic_receive*, čímž zjistíme, jestli přišly nové pakety
  2. případě, kladné odpovědi, získáme pomocí čtecích funkcí s předponou '*netbuf_*' požadovaná data
  3. když jsme se zpracováním paketu hotovi, je nutno zavolat funkci '*nic_recfree*', která jej označí jako zpracovaný

5.3.2 Knihovna net

Knihovna funkcí net využívá ke své funkci předchozí knihovnu. Zatímco předchozí knihovna se obstarává vstup a výstup dat do a z modulu, knihovna net by měla jejímu uživateli a jeho programům umožnit jednoduchou (prozatím jen základní) komunikaci pomocí ethernetu a vyšších protokolů.
Dvě hlavní funkce rozhraní knihovny net jsou:

- **net_init** – tato funkce inicializuje jednak samotný modul a dále datovou strukturu obsahující informaci o síťovém rozhraní
- **net_idle** – funkce určená k periodickému volání, zkontroluje došlé pakety, v případě jejich přítomnosti je první z nich zpracován a na konci zpracování uvolněn

Pro komunikaci FITKitu s okolím jsou určeny dvě funkce, jedna pro odesílaní a jedna pro příjem.

- **net_send** – funkce odesílá ze zadaného portu data dané délky na zadanou cílovou ip adresu a port určeným protokolem
- ***udp_listen*** – funkce sloužící k zaregistrování callback funkce na daný port, tato funkce má nasledně možnost zpracovávat data přenášena UDP protokolem, parametrem je daný port a ukazatel na funkci

Funkce určená k zaregistrování na udp port by měla vypadat následovně:

```
void funkce( unsigned char *  zdrojova_ip,
             unsigned int     zdrojovy_port,
             unsigned int     delka_udp_dat );
```

V parametrech *zdrojova_ip* a *zdrojovy_port* bude funkci předána IP adresa a port odesilatele dat, v parametru *delka_udp_dat* bude předána délka UDP paketu bez hlavičky, tj. dat která má funkce k užití. Data opět načítame standartně pomocí *netbuf_read* nebo *netbuf_getc*, s tím že při zavolání funkce je čtecí ukazatel umístěn na začatku UDP dat.

Knihovna dále definuje pro každý použitý protokol hlavičky, funkce potřebné pro jejich uložení do odesílaného paketu, nebo načítání z příchozích paketů.

Důležitou částí této knihovny je také implementace arp resolveru. Ten překládá adresy IP protokolu na adresy MAC. Používá při tom arp cache. Pokud záznam v cache nenalezne, odešle automaticky „ARP request“. Zde je ovšem problém. Pokud ve funkci *net_send* odešleme data na adresu, která není v arp cache, odešle se sice ARP zpráva s dotazem, ale aby se adresa do cache dostala, musí alespoň jednou proběhnout funkce *net_idle*, která příjme ARP odpověď a umístí ji tam. Pro uspěšné odeslání je pak třeba volání funkce *net_send* zopakovat. To je celkem závažný nedostatek, jeho vyřešení jsem neimplementoval, avšak v budoucí verzi knihovny, bych funkci *net_send* upravil tak, aby nezapisovala data přímo do paměti modulu, ale do paměti mikrokontroléru a až poté by byly ve funkci *net_idle* přesunuty do odesílací paměti a odeslány. Tím by se zajistilo, že pakety určené adresám, pro které se nenalezne záznam v ARP cache, budou moct být odloženy do doby než přijde odpovídající paket ARP odpovědi.

5.3.3 Vlastní program mikrokontroléru

V programu jsem využil výše zmíněný VHDL design a knihovny pro sestavení celku, který by mohl alespoň naznačovat některé využití mé práce.

Program se skládá ze dvou hlavních částí. V první části funguje FITKit jako klient. Po jejím zapnutí pomocí příkazu *udpkey_on*, jsou po stiknutích jednotlivých kláves odesílány jejich ASCII kódy na adresu a port zvolenou příkazy *dst_ip* a *dst_port*.

V druhé části je FITKit serverem. Na UDP portu 12345 je pomocí funkce *udp_listen* zaregistrována funkce, která po obdržení UDP zprávy s čtyřmi znaky, tvořícími slovo „*temp*“, odešle jako odpověd hodnotu teploty, naměřené FITKitem.

Seznam příkazů pro ovládání programu:

- *setip x.x.x.x* – nastaví ip adresu FITKitu a zobrazí na displeji
- *cache* – vypíše obsah ARP cache na terminál
- *dstip x.x.x.x* – zvolí cílovou IP adresu, k odesílání dat
- *dstport* x – zvolí cílový port k odesílání dat

- *udpkeyon* – zapne odesílání ASCII kódu stisknuté klávesy po síti
- *udpkeyoff* – vypne odesílání ASCII kódu stisknuté klávesy po síti
- *init* – inicializace modulu s vypisem inicializovaných hodnot – tento příkaz není třeba explicitně volat, je pouze pro kontolu správné inicializace registrů

5.4 Ukázkové aplikace

Ověřit funkčnost programu a knihoven můžeme tím, že si zkusíme z počítače na stejné síti poslat na FITKit ping. Dále také k tomuto účelu předkládám dva jednoduché skripty v PERLu.

5.4.1 Ping

FITKit odpovídá na ICMP zprávu „echo request“ zprávou „echo reply“.

5.4.2 Skripty v PERLu

server.pl

Tento skript funguje jako server, naslouchající na zvoleném UDP portu. Volba portu se provádí pomocí parametru '*--port=x*', pokud není zadán, použije se výchozí volba portu 12345. Pokud je na FITKitu tato funkce zapnuta, ten po stisknutí tlačítka odešle na port předem zvolený příkazem '*dst_port x*' na adrese předem zvolené příkazem '*dst_ip x.x.x.x*' UDP zprávu, obsahující ASCII kód zmáčknuté klávesy. Naslouchající skript stisknutou klávesu následně zobrazí.

client.pl

Skript spustíme pomocí parametru '*--ip=x.x.x.x*' určujícího na jakou adresu bude odeslán požadavek a '*--port=x*', který specifikuje port na této adrese. Po spuštění skript zkontroluje parametry a na zvolenou adresu/port odešle zprávu s požadavkem na odeslání hodnoty teploty. Je to jednoduchá UDP zpráva obsahující pouze čtyři znaky - „temp“. Na UDP portu 12345 FITKitu naslouchá funkce, která po obdržení tohoto dotazu odešle zpět klientovi hodnotu naměřené teploty jako ascii řetězec. Skript hodnotu obdrží a vytiskne. Tato ukázková aplikace využívá funkčnost knihovny thermometer.

6 Závěr

V mé práci jsem se snažil o nalezení nejvhodnějšího způsobu rozšíření platformy FITKit o rozhraní ethernet. Jako vhodné řešení jsem nakonec zvolil modul SPINET s obvodem ENC28J60. Dále jsem se pokusil o implementaci alespoň těch nejzákladnějších protokolů, aby bylo případnému uživateli této práce umožněna co nejjednodušší komunikace po ethernetové síti. To se povedlo, přestože je tato komunikace prozatím jen na primitívní úrovni a s omezeními. Za největší problém svého řešení považuji nedokonale vyřešenou ARP cache.

Co se týka možného pokračování této práce, nabízí se zde mnoho možností. První oblastí je úprava a rozšíření knihovny enc28j60 pro komunikaci s obvodem a knihovny net pro komunikaci po síti. Obvod totiž nabízí široké možnosti konfigurace, které nejsou bezpodmínečně nutné pro standartní provoz, ale v určitých specialních aplikacích by našly využití. Těmi jsou například různé filtry příchozích paketů, možnost plně duplexní komunikace, sedmi slabikovy status vektor, zapisovaný po odeslání paketu, DMA modul apod. Do této oblasti spadá také využití přerušení.

Knihovna net určitě také potřebuje ještě nejaký čas práce, ať už je to kvuli dopracování odpovídající implementace IP protokolu a nebo implementaci prokolů nových, jako je například TCP nebo DHCP.

Druhý směr, kterým by se případné pokračování této práce mohlo ubírat je možnost implementace části 'stacku' v FPGA. Zde si myslím, že přenesení zpracování IP vrstvy z mikrokontroléru do hradlového pole by bylo přínosem. Nedokážu však posoudit, nakolik by to bylo náročné na zdroje FPGA.

Třetím způsobem pokračování této práce by mohlo využití FITKitu s modulem SPINET v praktickém zapojení. Jako zajímavá by se mi např. jevila realizace nějakého druhu ethernet převodníku, kdy by jsme použili dva FITKity, připojené každý jinému počítači nebo switchi a vzájemně je nějakým způsobem spojili (opticky nebo jakkoliv) a přenášeli tak spojení mezi dvěma sítěmi.

Literatura

1. Andrew S. Tannenbaum, *Computer Networks*. Prentice Hall 2003
2. Heather Osterloh, *TCP/IP – Kompletní průvodce*. SoftPress 2003
3. Wikipedie - Ethernet, www stránky elektronické encyklopedie (5/2008)
   http://en.wikipedia.org/wiki/Ethernet
4. Wikipedie – MAC, www stránky elektronické encyklopedie (5/2008)
   http://en.wikipedia.org/wiki/Media_Access_Control
5. Wikipedie - PHY, www stránky elektronické encyklopedie (5/2008)
   http://en.wikipedia.org/wiki/PHY
6. Datasheet obvodu ENC28J60
   http://www.microchip.com
7. Ukázkové příklady modulu SPINET
   http://www.asix.cz

Seznam příloh

1. CD se zdrojovými kódy a dokumentací