VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

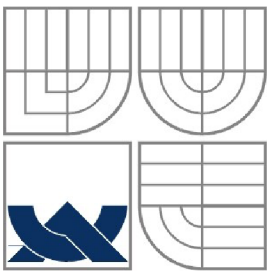
ROZHRANÍ IDE PRO PLATFORMU FITKIT

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

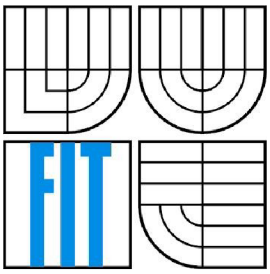
AUTOR PRÁCE
AUTHOR

Stanislav Sigmund

BRNO 2007



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

ROZHRANÍ IDE PRO PLATFORMU FITKIT

IDE INTERFACE FOR HW/SW PLATFORM FITKIT

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

Stanislav Sigmund

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. Zdeněk Vašíček

BRNO 2007

Zadání bakalářské práce

Řešitel: **Sigmund Stanislav**
Obor: Informační technologie
Téma: **Rozhraní IDE pro platformu FITKit**
Kategorie: Počítačová architektura
Vedoucí: Vašíček Zdeněk, Ing., UPSY FIT VUT

Pokyny:

1. Seznamte se s SW/HW platformou FITkit.
2. Seznamte se s rozhraním IDE, protokolem ATA/ATAPI a strukturou FAT/FAT32.
3. Navrhněte řadič IDE, který bude možné integrovat do stávajícího systému a pomocí kterého bude možné komunikovat s pevným diskem případně jiným zařízením obsahujícím rozhraní IDE.
4. Implementujte navržený řadič ve VHDL.
5. Navrhněte a implementujte sadu základních knihovních funkcí pro práci s pevným diskem jako je čtení a zápis sektoru, zjištění seznamu adresářů, souborů apod.
6. Demonstrujte funkčnost na vzorové aplikaci.

Literatura: http://www.fit.vutbr.cz/kit

Licenční smlouva

Licenční smlouva je uložena v archívu Fakulty informačních technologií Vysokého učení technického v Brně.

Abstrakt

Tato práce se zabývá praktickou realizací řadiče IDE rozhraní uvnitř jednotky FPGA a připojením pevného disku. Kromě řadiče byla vytvořena řada knihovních funkcí pro práci s diskem na nejnižší úrovni a řada funkcí zpřístupňující souborový systémem FAT32. Řadič je určen pro výukovou platformu FITKit, kde značně rozšiřuje kapacitní možnosti kitu.

Klíčová slova

FAT, FAT32, MBR, Partition, BSP, FITKit, IDE, ATA

Abstract

This thesis deals with an implementation of IDE controller on FPGA unit and hard drive connection. Besides, a lot of library functions were created that provide basic control of a hard drive and make accessible file system FAT32. Proposed controller was designed for platform FITKit and to enlarge memory capacity of this kit.

Keywords

FAT, FAT32, MBR, Partition, BSP, FITKIT, IDE, ATA

Citace

Stanislav Sigmund: Rozhraní IDE pro platformu FITkit, bakalářská práce, Brno, FIT VUT v Brně, 2007

Rozhraní IDE pro platformu FITkit

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením ing. Zdeňka Vašíčka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Stanislav Sigmundí
15. května 2007

Poděkování

Tímto děkuji za pomoci s platformou FITKit, s programovacím jazykem VHDL a radami s bakalářskou prací vedoucímu projektu, ing. Zdeňku Vašíčkovi.

Obsah

1 Úvod

Platforma FITKit, která je vyvíjená na Ústavě počítačových systémů (UPSY), je určena především pro výuku formou praxe. K tomu je vybavena mikroprocesorem a jednotkou FPGA, umožňující snadné propojení komponent a snadnou implementaci případných řadičů externích zařízení. Může sloužit pro zpracování zvuku, sběr dat (napětí, teplota), řešení výpočtů, ale i např. zpracování obrazu.

Tyto aplikace však pracují s velkým objemem dat. Platforma FITKit může sice být propojena přímo s počítačem, ale maximální přenosová rychlost je velice omezena komunikačním sériovým rozhraním a platforma se stává závislá na počítači. Hlavní uplatnění je právě v její možné samostatnosti. Propojení s velkokapacitním zařízením, jako je pevný disk platforma získává velký úložní prostor s dostatečnou přenosovou kapacitou. Jako velmi vhodné řešení se jeví použití pevného disku nebo některé paměťové karty, podporující rozhraní IDE.

Rozhraní IDE je velmi rozšířené a přitom nenáročné, umožňuje připojit velkou škálu pevných disků, jednotek CD-ROM a také některé paměťové karty. Pokud se použije běžně používaný souborový sytém, jako je FAT (dnes hlavně FAT32), jsou data snadno přenositelná a zpracovatelná.

Cílem bakalářské práce je implementovat jednoduché úložiště dat, které je přenositelné a umožňuje pohodlně pracovat s daty na něm. To nám poskytuje právě disk s IDE rozhraním a souborovým systémem FAT32. Data lze zpracovávat na počítači a přenášet mezi počítačem a platformou.

Práce je členěna následovně. První kapitola obsahuje stručný popis použitých komponent pro tuto BP (platforma FITKit a disk s IDE rozhraním). Druhá až čtvrtá kapitola popisuje samotné IDE rozhraní, práci s ním a jeho implementaci na platformě FITKit. Pátá kapitola stručně popisuje souborový systém FAT32 a šestá pak práci s ním pomocí vytvořených knihovních funkcí. Poslední kapitola nabízí další možná rozšíření a vylepšení řadiče i knihovních funkcí.

1 Popis použitých komponent

1.1 Výuková platforma FITKit

Výuková platforma FITKit je určena pro získání praktických zkušeností s vestavěnými systémy, které jsou stále více používány v našem okolí (mobilní telefony, MP3 přehrávače). K tomuto účelu je vybavena jednotkou FPGA (polem programovatelných hradel), na které je možno vytvářet různá jednodušší i složitější zapojení číslicových obvodů na jediném čipu. Toto zapojení lze programově velmi jednoduše změnit, umožňující tak rychlé úpravy výsledného obvodu.

Jednotka FPGA je řízena 16-bitovým procesorem (též osazeným na platformě) a komunikující s ní přes sériové rozhraní SPI. Jelikož jednotka FPGA obsahuje pouze volatilní paměť, je platforma osazena pamětí FLASH o kapacitě 256kB, v které je uloženo nastavení jednotky FPGA, umožňující tak případnou mobilitu platformy (nezávislost na počítači).

Procesor komunikuje s počítačem přes rozhraní RS232, simulovaného rozhraním USB, je tedy použitelný i na systémech bez rozhraní RS232 a poskytuje platformě napájení dostatečné pro většinu aplikací.

Platforma má k dispozici dva A/D a dva D/A převodníky na procesoru použitelné pro práci s analogovým signálem (např. Audio). Na jednotce FPGA pak 50-pinovou patici pro připojení dalších zařízení (je použit i pro připojení IDE konektoru), konektor pro výstup VGA signálu na monitor, výstup na rozhraní RS232 a dva porty PS/2. Platforma je osazena LCD displejem, maticovou klávesnicí a dynamickou pamětí 8MB. Vše je připojeno taktéž k jednotce FPGA.

Uvedené součásti platformy poskytují zázemí i pro náročnější aplikace. Více informací můžete nalézt na stránkách platformy [5].

1.2 ATA zařízení s IDE rozhraním

IDE rozhraní je určeno pro připojení ATA zařízení, kterými jsou nejčastěji pevné disky. Teoreticky mohou být k tomuto rozhraní připojeny i jiná zařízení, jako jsou scanery a tiskárny. Toho se však nevyužívá). Umožňuje připojení i zařízení ATAPI (paketová komunikace), jako jsou jednotky CD/DVD – rom, neboť jsou kompatibilní se standardem ATA.

Vyskytují se různá rozšíření, která jsou však zpětně kompatibilní a definují především nové vlastnosti. Při návrhu řadiče a knihoven bylo v této práci použito rozšíření ATA-2 [1].

Zpětná kompatibilita a jednoduchost rozhraní přispěla k velkému rozšíření především v osobních počítačích (PC), a i když je již v dnešních dobách vytlačováno novými rozhraními, jako je např. SATA, je stále přítomna na většině počítačů. Díky tomu je na trhu velká nabídka různých zařízení.

Na jedno rozhraní IDE mohou být současně připojeny až dvě zařízení, označované jako MASTER (disk 0, nejčastější) a SLAVE (disk 1). Rozlišeny mohou být buď externím nastavením, nebo pozicí na kabelu.

Přenosové rychlosti jsou, vzhledem k jednoduchosti rozhraní, poměrně velké, i když nedosahují přenosových rychlostí jiných rozhraní (např. SATA, SCSI).

Velké rozšíření rozhraní a velká nabídka zařízení stejně jako jednoduchost rozhraní byly hlavními kritérii volby rozhraní pro platformu FITKit. Rozhraní IDE je plně splňuje.

2 Řadič IDE rozhraní

2.1 Komunikační protokol IDE rozhraní

Implementovaný řadič vychází ze standartu ATA-2 [1], který je podporován většinou starších disků. V této práci byl použit disk Seagate ST34311A s kapacitou 4,3GB, který podporuje standardy ATA-1, 2 a 3. Standardy ATA jsou zpětně kompatibilní a vyšší verze mají funkčně jen menší odlišnosti. Definují především další přenosové režimy. Bude s nimi tedy počítáno v rozšíření řadiče.

2.1.1 Použité signály

Komunikace probíhá pomocí šestnácti obousměrných třístavových datových vodičů, dvou signálů pro výběr přijímacího obvodu (CS0 a CS1) a tří adresních signálů (DA2, DA1, DA0), které společně se signály pro výběr přijímacího obvodu tvoří adresu registru na zařízení, a signálů pro čtení a zápis z/do registru (IOR, IOW). Dále je použit signál RESET, který slouží pro hardwarový restart zařízení.

Lze též využít signál IORDY pro pozdržení komunikace s kratší délkou cyklu, ale pro PIO módy 0-2 je jeho použití pouze volitelné (řadič je zatím konfigurován na PIO 0). Při dodržení časování není potřeba. Je doporučen až při PIO módech 3 a výše. Ve stávajícím řadiči není součástí časování a jeho použití je zvažováno v rozšířené verzi řadiče.

Signály DMARQ (žádost o přenos) a DMACK (potvrzení přenosu) slouží ke komunikaci v DMA režimech a jejich využití je plánováno pro rozšíření řadiče. Signál INTRQ (žádost o přerušení) je možno použít v přerušovacím systému platformy, ale při použití PIO módů a vzhledem k maximální rychlosti, nemá zatím reálné uplatnění.

2.1.2 Přenosové režimy

Základní komunikační režim zařízení je PIO (programovatelný vstup/výstup). Přenos je řízen řadičem IDE rozhraní. Tento režim je velmi jednoduchý a umožňuje adresaci všech registrů na ATA zařízení, je však potřeba účasti procesoru a nedosahuje příliš vysokých přenosových rychlostí.

Tento režim se může nacházet v různých módech ([1] jich definuje pět), PIO mód 0 má nejdelší časování (a tedy i nejnižší přenosovou rychlost), ale je zpětně kompatibilní i s nejstaršími ATA zařízeními. Měl by výchozím módem, než je zjištěn a nastaven vyšší PIO mód na zařízení. Vyšší PIO módy umožňují kratší časy v časování, ale až na některé menší změny v komunikaci (neplatný signál IOCS16, prakticky nutnost použít řízení pomoci IORDY) jsou zpětně kompatibilní s režimem PIO 0.

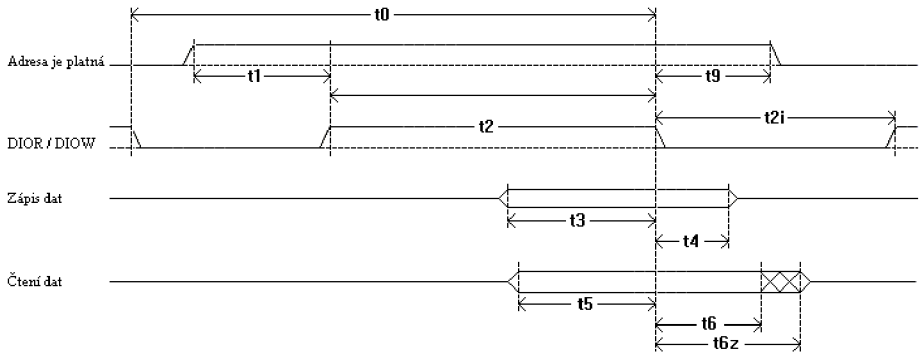
Přenos je osmi-bitový, kromě přístupu na datový registr, ten je šestnácti-bitový, ale pro nižší PIO módy jej lze změnit taktéž na osmi-bitový. Je nutné, aby řadič byl vždy nastaven na PIO mód stejný, nebo nižší, než je nastavený PIO mód na zařízení.

Další přenosovým režimem je režim DMA, který je použit některými příkazy (mají varianty pro PIO i DMA režim). Tento režim je určen pouze pro přenos dat přes datový registr a neumožňuje tedy přístup k ostatním registrům. Je řízen ze strany ATA zařízení a umožňuje rychlejší přenos dat bez asistence procesoru. Přenos je vždy 16-bitový a časován je signály DMARQ a DMACK. Rozlišujeme singleword a multiword (přenáší několik slov během 1 cyklu) DMA módy.

Řadič, implementovaný na platformě FITKit používá pouze PIO mód 0, další přenosové režimy (vyšší PIO, DMA) jsou předmětem rozšíření, popsaného v kapitole 7.

2.1.3 Zápisový a čtecí cyklus v PIO módu

Na obrázku 2.1 je vyznačeno časovaní zápisového/čtecího cyklu, časování vychází z popisu standardu ATA-2 [1]. Je odstraněn popis signálů IORDY (není použit) a IOCS16 (pouze informativní, řadič jej nepoužívá, při vyšších přenosových módech je neplatný):



Obrázek 2.1: Časování zápisu/čtení v PIO módech

Operaci čtení/zápisu vždy předchází nastavení adresy registru signály CS1, CS0, DA2, DA1 a DA0. Po ustálení adresy (odpovídá času t1) je signálem IOW (zápis) nebo IOR (čtení) zahájena zvolená operace. Tyto signály musí být nastaveny minimálně po dobu t2, která je závislá na šířce přenášených dat (viz tabulka 2.1).

Data pro operaci zápis musí být platná na výstupu minimálně po dobu t3 před a minimálně po dobu t4 po ukončení operace (navrácení signálu IOR/IOW do výchozího stavu). Při operaci čtení jsou platná data přítomná na výstupu zařízení min. po dobu t5 před a po dobu t6 po ukončení operace. Od ukončení předchozí operace do ukončení aktuální operace musí uběhnout minimálně doba t0, tato hodnota tedy určuje maximální přenosovou rychlost.

	PIO módy	Mód 0	Mód 1	Mód 2	Mód 3	Mód 4
t0	doba cyklu (min.)	600	383	240	180	120
t1	platná adresa do začátku operace (min.)	70	50	30	30	25
t2	min. doba nastavení IOR/IOW - 16 bitů	165	125	100	80	70
	- 8 bitů	290	290	290	80	70
t2i	min. doba nečinnosti IOR/IOW	-	-	-	70	25
t3	min. doba platnosti dat před ukončením op. zápisu	60	45	30	30	20
t4	min. doba platnosti dat po ukončení op. zápisu	30	20	15	10	10
t5	min. doba platnosti dat před ukončením op. čtení	50	35	20	20	20
t6	min. doba platnosti dat po ukončení op. čtení	5	5	5	5	5
t6z	max. doba přechodu zařízení do stavu vys. impedance	30	30	30	30	30
t9	min. doba platnosti adresy po ukončení operace	20	15	10	10	10

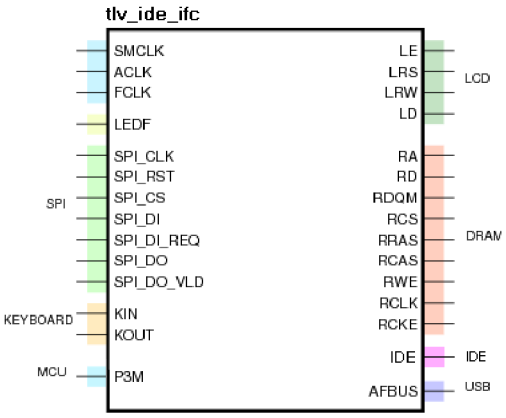
Tabulka 2.1: Doby časování a jejich omezení v PIO módech 0 – 4

Doba t6z určuje maximální čas, za který přejdou datové vodiče zařízení do stavu vysoké impedance po ukončení operace čtení. V daném zapojení řadiče nemá tato doba vliv. Doba t2i je použita až u PIO módů 3 a výše.

2.2 Implementace řadiče

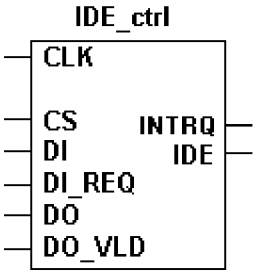
2.2.1 Top-level entita tlv_ide_ifc

Tato entita (viz obrázek 2.2) zastřešuje komunikační vývody FPGA s IDE rozhraním, definuje port IDE, Neobsahuje však definici portů VGA, PS2 a RS232, neboť sdílejí stejné vývody s IDE rozhraním. Je určena pro aplikace, které pracují s neupravenou verzí konektoru IDE rozhraní.



Obrázek 2.2: Komponenta tlv_ide_ifc

2.2.2 Komponenta ide_ctrl



Obrázek 2.3: Komponenta IDE_ctrl

Tato komponenta představuje samotný řadič IDE rozhraní, je určena k přímému propojení s vývody na entitě fpga_ide. Je řešena konečným automatem.

Vývody CS,DI,DO,DI_REQ a DO_VLD jsou signály vnitřního SPI rozhraní, připojují se na signály SPI_ top-level entity. Signál hodin CLK by měl být společný se signálem SPI_CLK, platí tedy pro něj, že frekvence hodin by měla být min. 4x větší, než u hodin samotného SPI rozhraní (jeho nejvyšší frekvence je 3.6864MHz). Je ovšem nutné nastavit komponentě generickou vlastnost CLK_PERIOD na délku periody hodin v ns. Komponenta na základě této hodnoty určí časování jednotlivých fází komunikace (časy odpovídají PIO módu 0 při 8-bitovém přenosu).

Port IDE slouží k přímému připojení k top-level entitě, nebo nepřímo přes komponentu, vybírající konfiguraci kabelu (je plánovaná).

Signál INTRQ je vyveden přímo z portu IDE. Jedná se o žádost o přerušení, která je vyvolaná IDE zařízením, aktivní v 1. Generování přerušení lze zakázat nastavením registru Device Control. Žádost je nulována čtením STATUS registru zařízení, nulování je nutné pouze, pokud chceme využívat přerušení (neovlivňuje chod zařízení).

SPI rozhraní využívá dva SPI dekodéry s osmi-bitovou adresou:

Jméno portu	Šířka dat	Adresní prostor	vlastnosti
Komunikační port (zápis/čtení registrů na IDE zařízení)	16	00H – 1FH	nižších 5 bitů adresy (1FH) určuje cílový registr na IDE zařízení
Řídící port (řízení komunikace)	8	20H – 21H	je využit pouze zápis, bit číslo 0 řídí signál RESET

Tabulka 2.2: Komunikační porty řadiče (rozhraní SPI)

3 Komunikace s ATA zařízením

Tato kapitola popisuje samotnou práci s ATA zařízením pomocí jeho registrů. Také popisuje nejpoužívanější příkazy.

3.1 Popis registrů

Komunikace probíhá čtením/zápisem do/z registrů zařízení. V tabulce 3.1 jsou znázorněny adresy jednotlivých registrů. Některé registry sdílejí stejnou adresu. Jeden registr je však pouze pro čtení, druhý pouze pro zápis. K adresaci registru slouží pětice adresních bitů. Komponenta ide_ctrl umožňuje jejich přímé použití přičtením k její bázové adrese. Při přístupu na adresu mimo povolený rozsah by zařízení nemělo reagovat. Všechny registry kromě datového jsou osmi-bitové, datový registr pracuje běžně se šestnácti-bitovou šířkou, ale lze jej do osmi-bitového režimu přepnout (nedoporučuje se, nelze použít v DMA režimech a vyšších PIO módech).

adresa		registr	
H	B	čtení	zápis
10	10000	Data (16 bitů)	Data (16 bitů)
11	10001	Error	Features
12	10010	Sector Count	Sector Count
13	10011	LBA bity 7-0 (Sector Number)	LBA bity 7-0 (Sector Number)
14	10100	LBA bity 15-8 (Cylinder Low)	LBA bity 15-8 (Cylinder Low)
15	10101	LBA bity 23-16 (CylinderHigh)	LBA bity 23-16 (CylinderHigh)
16	10110	LBA bity 27-24 + cislo disku (Device/Head)	LBA bity 27-24 + cislo disku (Device/Head)
17	10111	Status	Command
0E	01110	Alternate Status	Device Control
0F	01111	Adress (zastaraly)	-

Tabulka 3.1: Adresy registrů ATA zařízení

3.1.1 Registr Data

Registr, určený pouze pro přenos dat oběma směry, data se přenáší po sektorech (512B).

3.1.2 Registr Status

Vrací aktuální stav IDE zařízení. Jeho čtení nuluje signalizaci přerušení (signál INTRQ).

7	6	5	4	3	2	1	0
BSY	DRDY	DF	DSC	DRQ	CORR	IDX	ERR

Tabulka 3.2: Bitová mapa registru Status

Význam jednotlivých bitů:

- BSY (Busy) – zařízení je zaneprázdněno, mělo by se zamezit používat jiný registr, než registr Status (a Alternate Status).
- DRDY (Device Ready) – zařízení připraveno, mělo by reagovat na příkazy
- DF (Device Fault) – selhání zařízení, chování není definovatelné (vážná porucha zařízení)
- DSC (Device Seek Complete) – hledání ukončeno, může se přistupovat k datům
- DRQ (Data Request) – jsou požadována data (při zápisu do zařízení), nebo data jsou nachystána (minimálně pro 1 sektor). Platí pro přenos dat přes registr Data.
- CORR (Corrected Data) – nalezena opravitelná chyba, data jsou opravena – přenášená data mohou být poškozená
- IDX (Index) – přejetí Indexu pod hlavou, v dalších specifikacích již zrušeno (nemá význam)
- ERR (Error) – chyba zařízení, bližší informace lze získat čtením registru Error (jeho čtením je bit ERR nulován).

3.1.3 Registr Alternate Status

Vrací stejná data jako registr Status, ale zařízení na jeho čtení nereaguje (nenuluje signalizaci přerušení).

3.1.4 Registr Error

Chybový stav zařízení.

7	6	5	4	3	2	1	0
r	UNC	MC	IDNF	MCR	ABRT	TKN0NF	AMNF

Tabulka 3.3: Bitová mapa registru Error

Význam jednotlivých bitů:
- r – rezervováno
- UNC (Uncorrectable Data) – nalezena neopravitelná chyba
- MC (Medium Changed) – médium bylo vyměněno – pouze pro vyměnitelná média
- IDNF (ID Not Found) – sektor nebyl nalezen
- MCR (Media Change Requested) – bylo zažádáno o výměnu média, např. stiskem tlačítka
- ABRT (Aborted Command) – příkaz byl zrušen, např. neznámý příkaz, nebo jej nelze provést
- TKN0NF (Track 0 Not Fount) – stopa číslo 0 nenalezena
- AMNF (Adress mark Not Found) – nenalezena adresa po nalezení pole s ID sektoru – špatně zapsaný sektor, nebo jeho poškození

3.1.5 Registr Sector count

Pro příkazy čtení/zápisu sektorů z/do zařízení udává počet sektorů, přenášených v jednom příkazu. Číslo 0 znamená 256. V průběhu přenosu je zde zapsán počet zbývajících sektorů. Některé příkazy jej používají pro zápis parametrů.

3.1.6 Registr Command

Zápisem do tohoto registru dáme povel k provedení příkazu. Je nutné, aby zařízení bylo připravené k plnění příkazu (není nastaven bit BSY a je nastaven bit DRDY popř. DSC v registru Status).

V případě, kdy příkaz není podporován, je vykonáván jiný příkaz, zařízení je v chybovém stavu (bit ERR registru Status je nastaven) nebo příkaz nelze vykonat (špatné číslo sektoru), je příkaz zrušen a je nastavena chyba ABRT. Pokud je zařízení ve stavu chyby (příznak ERR), je nutné přečíst registr Error.

3.1.7 Registr Device Control

Zápisem je možné provést softwarový reset, nebo povolit/zakázat generování přerušení:

7-3	2	1	0
rezervováno	SRST	nIEN	0

Tabulka 3.4: Bitová mapa registru Device Control

- SRST – nastavením na 1 je zařízení drženo ve stavu softwarového resetu (pokud jsou přítomna dvě zařízení, platí pro obě)
- nIEN – pokud je nastaven na 0, jsou přerušení zakázána (výchozí hodnota je 1)

3.1.8 Registr Features

Je použit k nastavování vlastností zařízení jako parametr (příkaz Set Features).

3.1.9 Registry adres

Tato BP používá výhradně adresace pomocí logických adres LBA28 (LBA48 implementována zatím není), adresování CHS (Cylindr, Hlava, Sektor) není použito, tyto registry jsou využity k zápisu nejnižších 24-bitů adresy sektoru.

Registr Device/Head určuje zbývající 4 bity adresy, systém adresace a aktivní disk:

7	6	5	4	3-0
r	LBA	r	D	LBA27-LBA24

Tabulka 3.5: Bitová mapa registru Device/Head

- LBA – 1 = LBA, 0 = CHS
- D – aktivní disk, 0 = MASTER, 1 = SLAVE

3.2 Práce se zařízením

3.2.1 Inicializace

Inicializace zařízení musí být zahájena hardwarovým resetem – nastavením signálu RESET na hodnotu 0 po dobu 25ms. Poté proběhne diagnostika zařízení, kterou provádí zařízení MASTER,

pokud je zařízení přítomné. Během diagnostiky je nastaven stav BSY registru Status. Zařízení poté provede některé operace (např. roztočení ploten a nalezení stopy 0) a po ukončení vlastní inicializace zruší stav BSY a nastaví DRDY popř. DSC. Pokud došlo k chybě, je nastaven stav ERR. Diagnostika může trvat i několik vteřin, je tedy nutné počítat např. s pravidelným nulováním WatchDog časovače, pokud je přítomen a zapnut (to je i případ platformy FITKit).

Dále je potřeba zjistit typ a vlastnosti zařízení, např. příkazem IDENTIFY. Pomocí něj je možné zjistit počet sektorů na zařízení, typ zařízení (ATA, ATAPI, vyměnitelné), módy přenosu a podporované služby. Následovat by mělo nastavení vyžadovaných vlastností (S.M.A.R.T., odemknutí zařízení, pokud bylo zamknuto heslem, nastavení spotřeby atd.).

Všechny tyto operace provádí funkce IDE_HDD_Setup v jednotce ide.h, která je součástí BP.

3.2.2 Zápis a čtení dat

Přenos dat je zahájen některou z operací READ/WRITE SECTORS popř. verze s povoleným opakováním (zařízení se pak pokouší vyžadovaný sektor přečíst/zapsat několikrát za sebou, jestliže neuspělo). Přenos dat využívají i operace READ/WRITE BUFFER a IDENTIFY, ale ty přenášejí pouze 1 sektor a nepoužívají adresu sektoru a počet sektorů k přenesení. Pro novější verzi řadiče je uvažováno o příkazech READ/WRITE DMA.

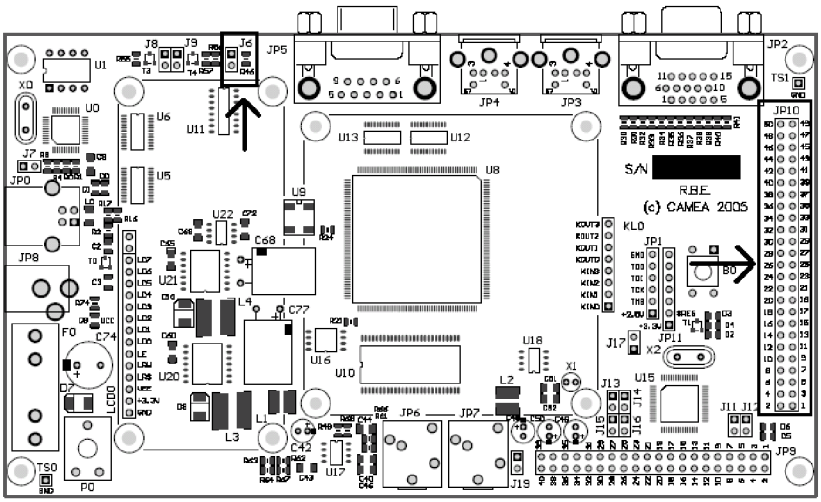
Před spuštěním příkazu je potřeba nastavit adresu sektoru a také registr Sector Count na počet sektorů, které chceme zapisovat/číst po sobě.

Po zadání příkazu je třeba vyčkat stavu, kdy je zrušen stavový bit BSY a nastaveny bity DRDY a DRQ. Pokud je nastaven bit ERR, je přenos zrušen. Následuje přenos dat jejich zápisem/čtení do/z registru Data. Během této doby by mělo být přistupováno pouze k registru Status (nebo Alternate status) a případně Sector Count (pouze čtení – počet zbývajících sektorů). Po ukončení přenosu by neměl být nastaven bit DRQ.

4 Technické řešení

4.1 Umístění konektoru

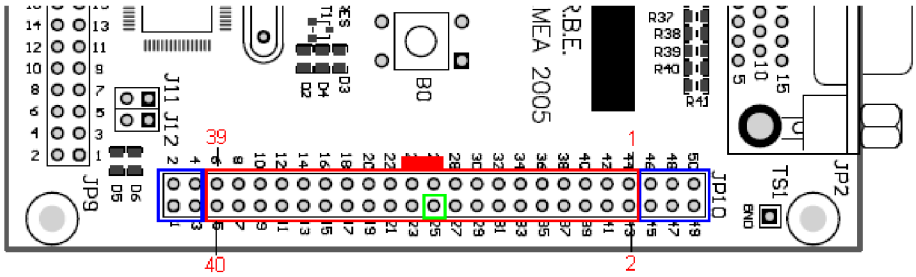
Pro IDE rozhraní je na platformě vyhrazena 50-pinová patice JP10, nacházející se na spodní straně platformy, jak ukazuje obrázek 4.1. Protože IDE rozhraní používá 40-pinový konektor, je nutné odpojit výstup na VGA, PS2 a RS232 rozpojením propojky J6 (taktéž označena na obrázku 4.1), neboť tyto výstupy jsou připojeny na část vývodů patice JP10.



Obrázek 4.1: Umístění patice JP10 a propojky J6 na platformě FITKit

Pokud je plánováno použití některého z těchto výstupů, bude nutné velmi upravit konektor kabelu. Tato úprava je plánována do budoucna.

Dále je nutné dodržet přesné umístění konektoru, neboť piny 1-4 na J10 jsou propojeny s vývody napájení. V této bakalářské práci je počítáno s připojením kabelu na vývody 5-44 patice JP10, jak ukazuje obrázek 4.2:

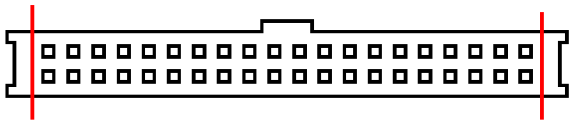


Obrázek 4.2: Umístění konektoru IDE na patici JP10

Na obrázku jsou též zaznačena čísla pinů na kabelu, obvykle vodič propojený s pinem č. 1 je obarven na červeno. Konektor je pootočen o 180º oproti běžné poloze, neboť tato poloha vyhovuje případnému umístění disku pod platformou FITKit. Pokud bude vyžadována jiná poloha konektoru, lze ji snadno změnit přečíslováním vývodů. Pro různé polohy konektoru a případné varianty jeho konfigurace je plánována komponenta, která se předřadí samotnému řadiči IDE rozhraní.

4.2 Úpravy konektoru na kabelu

Běžný konektor na kabelu IDE zařízení je po stranách opatřen nálitky, které slouží pro udržení celistvosti konektoru, zejména při jeho vyjímaní. Ovšem tyto nálitky se nacházejí v místě konektorem nevyužitých okolních pinů a znemožňují jeho zasunutí do patice. Proto je nutné tyto nálitky odbrousit (viz Obrázek 4.3). Tento úkon ovšem není problematický a přináší především zvýšení nutné opatrnosti při vyjímání z patice.



Obrázek 4.3: Přečnívající části IDE konektoru

5 Souborový systém FAT

Souborový systém FAT (File Alocation Table) uchovává informace o alokovaném prostoru v tabulkách. Při implementaci souborového systému FAT jsem vycházel z [2] a [3].

5.1 Master Boot Record MBR

Tabulka MBR se nachází v sektoru 0 a popisuje jednotlivé oddíly, nacházející se na disku. Na začátku sektoru se nachází 446B spustitelného kódu. Za touto oblastí se nachází čtyři 16-bytové popisy jednotlivých oddílů. Sektor je uzavřen znakem AA55H, který označuje MBR sektor. Struktura tMBR je definovaná v knihovně fat.h.

Jednotlivé popisy mají strukturu, uvedenou v tabulce 5.1. Jsou též popsány strukturou tPartition. Jelikož tato práce předpokládá použití pouze adresního režimu LBA, jsou pro nás důležité pouze vlastnosti type a offset. Oddíly bez rozšíření o LBA jej ovšem také mohou používat.

Offset	Délka [B]	Popis	Vlastnost
00H	1	Stav(0 = neaktivní, 80H = aktivní)	Active
01H	1	Začátek – číslo hlavy	BHead
02H	2	Začátek – cylindr/sektor	BCylSec
04H	1	Typ oddílu (tabulka 5.2)	type
05H	1	Konec – číslo hlavy	EHead
06H	2	Konec – cylindr/sektor	ECylSec
08H	4	Logické číslo počátečního sektoru	offset
0CH	4	Počet sektorů oddílu	sectors

Tabulka 5.1: Popis jednoho oddílu v záznamu MBR

typ	popis
00H	Neznámý/žádný
01H	FAT12
04H	FAT16 (menší než 32MB)
05H	Rozšířený oddíl DOS
06H	FAT16 (větší než 32MB)
0BH	FAT32
0CH	FAT32 s LBA rozšířením (služby 13H BIOS)
0EH	FAT16 s LBA
0FH	Rozšířený oddíl DOS s LBA

Tabulka 5.2: Typy oddílů FAT

Rozšířený oddíl DOS umožňuje větší počet oddílů na disku, než jsou základní čtyři. Ukazuje na strukturu, podobnou MBR, která obsahuje popis dalšího oddílu FAT a případně odkaz na další

rozšířený oddíl. Offset je ovšem vztažen k pozici aktuální tabulky, je tedy nutné ji přičíst k pozici, udávané v položce offset. Poslední rozšířený oddíl obsahuje již pouze poslední oddíl FAT.

5.2 FAT Boot Record

Podrobněji bude popisován systém FAT32, ostatní systémy pouze okrajově. První sektor oddílu FAT je Boot Record (viz tabulka 5.3). Obsahuje všechny potřebné údaje ke správnému nastavení oddílu. V knihovně fat.h je popsán strukturou tBPB.

Offset	Velikost [B]	Popis	Vlastnost
00H	3	Adresa spustitelné oblasti	JumpCode
03H	8	Jméno systému, který oddíl vytvářel	OEMName
0BH	2	Počet bytů na sektor (512,1024,2048,4096)	BytsPerSecL + BytsPerSecH
0DH	1	Sektorů na cluster	SecPerClus
0EH	2	Rezervovaných sektorů (běžně 32)	RsvdSecCnt
10H	1	Počet FAT tabulek (běžně 2)	NumFATs
11H	2	Počet položek v Root Directory (pro FAT32 = 0)	RootEntCntL + RootEntCntH
13H	2	Celkový počet sektorů. = 0: údaj je v TotSec32	TotSec16L + TotSec16H
15H	1	Typ disku, = F8H: pevný	Media
16H	2	Počet sektorů na FAT t. = 0: údaj je v FATSz32	FATSz16
18H	2	Sektorů na stopu	SecPerTrck
1AH	2	Počet hlaviček	NumHeads
1CH	4	Počet skrytých sektorů	HiddSec
20H	4	Celkový počet sektorů. = 0: údaj je v TotSec16	TotSec32
24H	4	Počet sektorů na FAT t. = 0: údaj je v FATSz16	FATSz32
28H	2	Aktivita FAT tabulek, bit 7 = 1: pouze jedna aktivní FAT tabulka (její číslo v bitech 0-3), = 0: mirroring	ExtFlags
2AH	2	Verze FAT, měla by být = 0	FSVer
2CH	4	Číslo clusteru kořenového adresáře	RootClus
30H	2	Číslo informačního sektoru (obvykle 1)	FSInfo
32H	2	Číslo záložního boot sektoru	BkBootSec
34H	12	Rezervováno	Reserved
40H	1	Písmeno disku, pevné disky začínají od 80H	DrvNum
41H	1	Rezervováno	Reserved1
42H	1	Následující 3 položky jsou přítomné:	BootSig
43H	4	Číslo svazku, pro identifikaci	VolID
47H	11	Jméno svazku	VolLab
52H	8	Typ systému, obvykle „FAT32 „ - neurčuje typ FAT	FilSysType

Tabulka 5.3: První sektor FAT oddílu (Boot Record)

5.3 Informační struktura

Informační struktura, pokud se nachází na oddílu (FSInfo > 0), obsahuje některé informativní údaje.

Offset	Velikost [B]	Popis
1E8H	4	Počet volných clusterů, = -1: neznámý
1ECH	4	Číslo clusteru, který byl naposledy alokován (zrychlení vyhledávání)

Tabulka 5.4: Některé položky informační struktury

Tyto údaje jsou pouze informativní a nemusejí být správné.

5.4 Základní informace

Celkový počet sektorů oddílu je uložen v položce TotSec16, pokud = 0 (malá kapacita), je použita hodnota TotSec32. Podobně velikost FAT tabulky v sektorech je uložena v položce FATSz16, pokud = 0 pak v položce FATSz32. V následujících výpočtech budou výsledné hodnoty označovány jako Sectors a FATSz.

Nejprve je třeba zjistit počet položek, alokovaných zvlášť pro kořenový adresář(pouze FAT12 a FAT16):

$$RootDir = \frac{RootEntCnt * 32 + BytsPerSec - 1}{BytsPerSec} \quad (1)$$

Pro systém FAT32 bývá RootEntCnt = 0, Tedy i RootDir = 0.

Poté je nutné zjistit číslo sektoru clusteru č. 2 (první cluster v systému):

$$ClusStart = (NumFATs * FATSize) + RsvdSecCnt + RootDir \quad (2)$$

Nesmíme zapomenout, že čísla sektorů jsou vztažena k začátku oddílu.

Následuje výpočet počtu clusterů v oddíle:

$$Clusters = \frac{Sectors - ClusStart}{SecPerClus} \quad (3)$$

Typ FAT tabulky by měl být určen právě z počtu clusterů, nikoliv tedy ze jména svazku (FilSysType) nebo jiných typických znaků. Pokud je počet clusterů menší než 4085, jedná se o FAT12, je-li větší nebo roven 4085 a menší, než 65525, jedná se o FAT16 a je-li větší nebo roven 65525, jedná se o FAT32.

Číslo sektoru, na kterém se nachází cluster, lze spočítat ze vztahu (4):

$$Sector = ((Cluster - 2) * SecPerClus) + ClusStart \quad (4)$$

5.5 Adresářová struktura

Adresáře jsou podobné souborům, liší se pouze svojí délkou (adresáře mají vždy nulovou délku) a atributem DIRECTORY. Přístup k nim je však podobný.

Adresář obsahuje jednotlivé položky, dlouhé 32B, které popisuje struktura tDirEntry (tabulka 5.5). Položky jsou řazeny za sebou a konec adresáře je označen zarážkou.

Kořenový adresář se ve FAT32 nachází na clusteru č. RootClus.

Offset	Velikost [B]	Popis	Vlastnost
00H	8	Jméno položky, prázdná místa jsou vyplněna znakem 32	Name
08H	3	Koncovka položky, prázdná místa jsou vyplněna znakem 32	Ext
0BH	1	Atributy položky (viz tabulka 5.7)	Attr
0CH	1	Rezervováno	NTRes
0DH	1	Čas vytvoření – setiny sekundy (pro 2 sekundy)	CrtTimeTenth
0EH	2	Čas vytvoření – granularita jsou 2 sekundy	CrtTime
10H	2	Datum vytvoření	CrtDate
12H	2	Datum posledního čtení/zápisu	LstAccDate
14H	2	Číslo prvního clusteru (horní polovina)	FstClusH
16H	2	Čas poslední změny (včetně vytvoření souboru)	WrtTime
18H	2	Datum poslední změny (včetně vytvoření souboru)	WrtDate
1AH	2	Číslo prvního clusteru (dolní polovina)	FstClusL
1CH	4	Délka souboru	FileSize

Tabulka 5.5: Jedna položka v adresáři

Typ položky je určen prvním znakem jména:

- E5H – volná položka
- 05H – normální položka, první znak = E5H
- 00H – prázdná položka, za ní se nenachází žádná jiná položka (zarážka)
- jiná hodnota – normální položka

Formát datumu a času:

CrtDate, LstAccDate, WrtDate			CrtTime, WrtTime		
Rok	Měsíc	Den	Hodiny	Minuty	Vteřiny
(Bity 15 – 9) + 1980	Bity 8 – 5	Bity 4 – 0	Bity 15 – 11	Bity 10 – 5	(Bity 4 – 0) * 2

Tabulka 5.6: Formát datumu a času, použitý v informacích o položce

Atributy položky:

Hodnota	Popis
01H	Pouze pro čtení
02H	Skrytý soubor/adresář
04H	Systémový soubor/adresář
08H	Jméno svazku (smí být pouze v kořenovém adresáři)
10H	Adresář
20H	Archivní soubor/adresář (byl změněn)
0FH	Dlouhé jméno

Tabulka 5.7: Atributy položky, kromě dlouhého jména jsou všechny komutativní

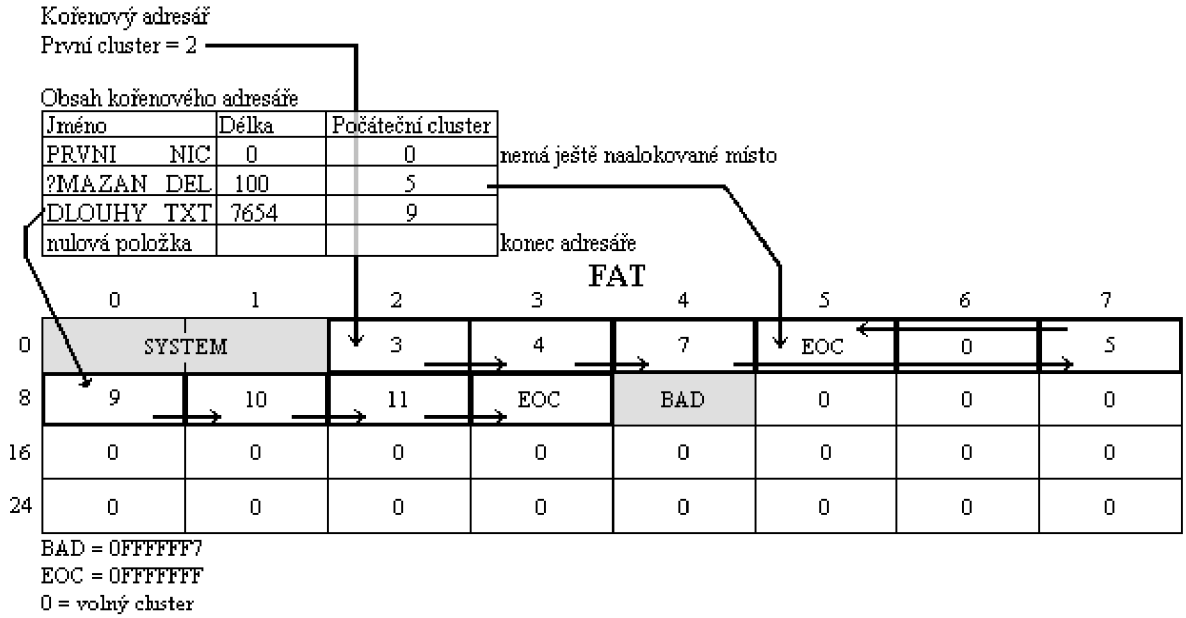
5.6 Tabulka FAT

FAT tabulky se nacházejí za oblastí rezervovaných sektorů. První sektor tabulky (FATNum) se dá vypočítat pomocí vztahu (5):

$$FATSector = RsvdSecCnt + FATNum * FATSz \quad (5)$$

Pro FAT32 je tabulka tvořena polem položek o velikosti 4B, z nichž je použito 28bitů, nejvyšší 4bity jsou rezervovány a neměly by se měnit. Pro velikost sektoru 512B (nejpoužívanější) se tedy v jednom sektoru nachází 128 položek.

Každá položka obsahuje číslo následujícího clusteru souboru/adresáře. Hodnota větší nebo rovna 0FFFFFF8H (nejnižších 28bitů = 0) znamená konec řetězce, řetězec dále již nepokračuje a aktuální cluster je poslední. Číslo prvního clusteru řetězu je uloženo v položce adresáře v FstClusL a FstClusH. Soubor o nulové délce, který dosud nemá alokována data, má číslo 0.
Hodnota 0FFFFFF7H znamená vadný cluster. Hodnota 0 označuje volný cluster.
Clustery číslo 0 a 1 nesmějí být použity.



Obrázek 5.1: Příklad tabulky FAT

Na obrázku 5.1 je ukázka tabulky FAT (v tomto případě FAT32) s velikostí 2KB na cluster. Kořenový adresář má přidělených 5 clusterů (2,3,4,7,5) a obsahuje 3 položky (zbytek místa by měl být vynulován, nulová položka je poslední položkou v adresáři).

První položka obsahuje soubor PRVNI.NIC (název by měl být psán velkými písmeny, FAT je necitlivá na velikost písmen). Ten však nemá ještě přiděleno místo, neboť jej dosud nepotřeboval, a tak má jako počáteční cluster hodnotu 0.

Druhá položka byla smazána, avšak předtím měla přidělena cluster číslo 5 (a pravděpodobně i 6, což vysvětluje skok v řetězci adresáře).

Třetí položka má přiděleny 4 clustery (8,9,10,11), aby mohla obsahovat všech 7654B dat.

6 Knihovní funkce pro práci s diskem

6.1 Knihovna ide.h (práce s diskem)

Tato knihovna obsahuje funkce pro práci s diskem na nejnižší úrovni. Definuje bázové adresy datového a řídícího portu na SPI, bázové adresy registrů na disku (konstanty, začínající na REG_, např. REG_DATA), čísla příkazů, komunikační struktury pro příkazy IDENTIFY a SECURITY a informační strukturu tIDE_HDD, která obsahuje informace o discích. Taktéž definuje jednotlivé příznaky registrů Error (IERR_) a Status popř. Alternate status (STATUS_). Poskytuje ale sadu funkcí, které práci s diskem usnadňují a umožňují již určitou míru abstrakce.

6.1.1 Základní funkce

Základní funkce umožňují přístup přímo k řadiči rozhraní.

void IDE_Hard_Reset()

Provede hardwarový reset (nastaví signál RESET na hodnotu 0 po dobu minimálně 25ms).

unsigned short IDE_Read_Word()
void IDE_Write_Word(unsigned short data)

Funkce přečte/zapíše 16-bitové slovo z/do registru Data.

unsigned char IDE_Read_Reg(unsigned char addr)
void IDE_Write_Reg(unsigned char addr,unsigned char value)

Funkce přečte/zapíše znak z/do registru, určeného adresou addr (lze použít konstanty, začínající na REG_).

6.1.2 Inicializace řadiče

K inicializaci jsou určeny následující funkce

void IDE_HDD_Setup_Init()

Zinicializuje stavové proměnné řadiče a provede hardwarový reset. Měla by předcházet funkci IDE_HDD_Setup.

int IDE_HDD_Setup()

Čeká na dokončení diagnostiky disků (po vypršení času vrací IDE_SETUP_WAIT, je třeba opakovat, dokud nevrátí IDE_SETUP_DONE). Po jejím dokončení zjistí stav jednotlivých disků (zatím detekuje pouze disk č. 0 – MASTER, detekce disku č. 1 nebyla testována) a provede jejich identifikaci, jejich stav je uložen v poli IDE_HDD[2] (pro oba disky) typu tIDE_HDD.

6.1.3 Práce s příkazy

int IDE_Wait_Status(unsigned char astatus,unsigned char nstatus,unsigned char errdetect,unsigned short count)

Funkce, která testuje stav zařízení a čeká, dokud jsou nastaveny všechny příznaky definované parametrem astatus a není nastaven žádný z příznaků nstatus. Čeká po count testovacích cyklů, pokud nejsou podmínky do té doby splněny, vrací IDE_RET_COUNT. Pokud je errdetect nula, netestuje stav ERR, v jiném případě vrací IDE_RET_ERROR. Při závažné chybě vrací IDE_RET_DF. Nastanou-li žádané podmínky, vrací IDE_RET_OK.

int IDE_CMD(unsigned char cmd)

Pokusí se spustit příkaz, pokud příkaz uspěl, vrací IDE_RET_OK. Využívá funkci IDE_Wait_Status a vrací stejné hodnoty. Počet cyklů čekání je nastaven v globální proměnné IDE_Wait_Count.

6.1.4 Nastavení adresy sektoru/aktivního disku

void IDE_Sector(unsigned long addr)

Nastaví číslo sektoru, adresovaného pomocí LBA, na aktivním disku. Je použito pouze 28 nejnižších bitů (LBA28).

void IDE_Change_Disc(int disc)

Vybere aktivní disk, 0 = disk 0, jiná hodnota = disk 1.

6.1.5 Přenos dat

int IDE_Read_Data(unsigned char *buffer,unsigned short count)
int IDE_Write_Data(unsigned char *buffer,unsigned short count)

Tyto funkce přečtou/zapíší určitý počet **slov** daný hodnotou count přes datový registr. Před zahájením přenosu testují stav zařízení a vrací případné chyby (viz funkce IDE_Wait_Status), jinak vratí IDE_RET_OK.

Funkce provádějí ošetření případu, kdy adresa buffer není zarovnána na sudé adresy (platforma FITKit používá 16-bitový procesor).

int IDE_Read_Sector(unsigned long addr,unsigned char *buff)
int IDE_Write_Sector(unsigned long addr,unsigned char *buff)

Nastaví adresu sektoru na aktivním disku a přečte/zapíše jeden sektor ze zařízení. Přenáší 512Bytů, návratové hodnoty jsou stejné, jako u předchozích funkcí.

6.1.6 Práce s cache funkcemi

Slouží pro abstraktnější přístup k diskové paměti. Umožňují nastavit 512B paměť cache, která je prakticky nutností při zápisu na disk (není potřebná, pokud zapisujeme celý sektor najednou). Při čtení nutností není. Všechny funkce (až na IDE_Cache_Set) vracejí IDE_C_OK (vše v pořádku), případně IDE_C_ERR (nastala chyba).

void IDE_Cache_Set(unsigned char *buff)

Nastaví paměť pro cache, pokud buff = NULL, pak je cache paměť vypnuta.

int IDE_Cache_Flush()

Ukončí všechny operace cache funkcí, popř. uloží změněná data na disk. Je nutné ji zavolat před přístupem na disk jinak, než pomocí funkcí cache.

int IDE_Cache_Operating()

Vrací IDE_C_OK, pokud právě neprobíhá žádná operace cache funkcí (lze přistupovat na disk).

int IDE_Set_Sector(unsigned long sector,unsigned char disk)

Nastaví přístup cache funkcí na zadanou adresu sektoru a číslo disku.

int IDE_Read(unsigned char offset,unsigned short count,unsigned char *data)
int IDE_Write(unsigned char offset,unsigned char count,unsigned char *data)

Čtení/zápis dat v rámci právě zvoleného sektoru, jsou přečtena/zapsána data, hodnota count udává počet 16-bitových slov. Hodnota offset udává polohu v sektoru po 16-bitových slovech (offset i count počítají 16-bitová slova, nikoliv znaky). Vrací počet načtených slov, nebo IDE_C_ERR.

Je-li nastavena paměť pro cache, jsou data před zahájením operace do ní načtena (i pro zápis) a jsou zapisována/čtena do/z ní. Na disk se zapisuje až při změně sektoru/disku (pokud jsou v cache pozměněné hodnoty) a nebo použitím funkce IDE_Cache_Flush. Umožňují tedy náhodný přístup v rámci jednoho sektoru.

Pokud však není nastavena paměť, jsou data při čtení načítána postupně (v této době se nesmí přistupovat k disku – není dokončena operace čtení), avšak nepoužité hodnoty jsou ignorovány (nastavení hodnoty offset za aktuální pozici v sektoru). Chceme-li číst hodnoty z dřívějších ignorovaných dat, je sektor znovu načten a proces se opakuje. Dosáhne se tak také náhodného přístupu k sektoru. Zápis je ovšem problematický, přeskočená data jsou zapsána jako hodnota nla (bez použití cache paměti neznáme předchozí hodnoty). Proto, pokud je vyžadován náhodný přístup i při zápisu, je nutné použít cache paměť.

Mnohé nedostatky s náhodným přístupem a spotřebou paměti (512B cache paměť je čtvrtina celkové paměti procesoru) by měly být odstraněny použitím cache na řadiči a přenosu pomocí DMA, což je plánováno u pokročilejšího řadiče rozhraní.

Celkově práce s cache funkcemi vypadá následovně:

1) funkcí IDE_Set_Sector je nastaven sektor/disk, se kterým chceme pracovat (pokud se nezměnil, funkce nic nedělá).
1) pomocí IDE_Read/IDE_Write se přistupuje k datům
2) jiné funkce se nepoužívají, pouye pokud chceme přistupovat k disku jinými funkcemi, nebo je potřeba uložit pozměněná data (např. při ukončení práce) pomocí funkce IDE_Cache_Flush()

6.1.7 Informace o discích

V globální proměnné IDE_HDD jsou dostupné informace o detekovaných discích funkcí IDE_HDD_Setup(). Je zde uložen celkový počet sektorů disku (sectors) a jeho stav (status). Pro stav platí následující:

- 0 = neznámý (před detekcí)
- 1 = nenalezen
- 2 = chybný
- 128 = ATA
- 129 = ATAPI
- +64 = zapnuta funkce S.M.A.R.T. (zatím není plně funkční)

6.2 Knihovna fat.h (práce s FAT32)

Knihovna je zaměřena pro práci s oddíly disku a se souborovým systémem FAT (zatím pouze FAT32, systémy FAT16 a FAT12 mají jen minimální podporu a není potřeba je podporovat). Umožňuje práci i s více FAT systémy na obou discích.

Knihovna používá funkce cache z knihovny ide.h a je tedy nutné po použití funkce z knihovny volat funkci IDE_Cache_Flush() před přístupem na disk jinými funkcemi, než z knihovny fat.h.

Knihovna je rozdělena na tři části. První část popisuje funkce pro načtení FAT oddílů z disku, druhá část popisuje funkce pro práci s adresářovou strukturou a třetí část obsahuje funkce pro práci se soubory.

6.2.1 Nastavení FAT souborového systému

Základem je struktura tFAT, která nese informace o jednom FAT systému. Je naplněna následujícími funkcemi:

int FAT_init_Part(unsigned char part,tFAT *fat,char *volume,unsigned char disc)

Pokud parametr fat je NULL, vrací počet nalezených podporovaných FAT oddílů na zvoleném disku, v ostatních případech se pokusí načíst zvolený oddíl part (index podporovaného oddílu, začínající indexem 0) do struktury fat. Jestliže parametr volume nemá hodnotu NULL, je zde, v případě úspěchu, uloženo jméno svazku. Po úspěšném načtení vrací hodnotu FAT_OK, případně FAT_ERR při neúspěchu.

Funkce zatím nepodporuje rozšířené oblasti.

int FAT_Init(tFAT *f,unsigned long offset,unsigned char disk)

Načte FAT systém na zadaném disku, který začíná na sektoru daným parametrem offset. Pokud je systém rozpoznán a načten, vrací FAT_OK jinak FAT_ERR.

6.2.2 Práce s adresářovou strukturou

Základem práce se soubory/adresáři je struktura tFile, která je pro ně společná.

int FAT_Root_Dir(tFAT *f,tFile *dir)

Zinicializuje strukturu dir (typu tFile) tak, aby ukazovala na kořenový adresář FAT systému.

void FAT_Set_Find(tFile *dir,char *name)

Nastaví adresář na jeho prohledávání, jméno hledané položky může obsahovat i zástupné znaky (? a *), pokud je rovna NULL, znamená „*.*“.

int FAT_Find_Next(tFile *dir,char *Fname,unsigned char *ext,unsigned char *attr,tFile *file)

Nalezne další položku v adresáři, která vyhovuje hledanému jménu. Do polí Fname a attr uloží jméno souboru a jeho atribut, pokud neobsahují hodnotu NULL, hodnota ext (není-li nastavena na NULL) obsahuje index prvního znaku koncovky jména. Je-li položka file nastavena, je nalezená položka zkopírována do této struktury.

Soubory a adresáře jsou otevírány právě touto funkcí, lze je ihned procházet pomocí FAT_Set_Find a FAT_Find_Next (adresáře), nebo FAT_Read/FAT_Write.

6.2.3 Práce se soubory

int FAT_Read(tFile *f,unsigned short length,unsigned char *buff)

Čte data ze souboru f, čte length **bytů** do pole buff. Vrací počet úspěšně načtených bytů dat, nebo FAT_EOF, je-li již na konci souboru (nelze načíst žádná data). Při chybě vrací FAT_ERR.

int FAT_Write(tFile *f,unsigned short length,unsigned char *buff)

Zapíše data do souboru, pokud nestačí jeho kapacita, zvětší jeho velikost. Není-li již volné místo, vrací FAT_FULL. Pro správnou funkci je nutné mít přiřazenou paměť pro cache.

int FAT_Close_File(tFile *f)

Uzavře soubor a uloží neuložená data. Je nutné ji volat pouze po zápisu na disk. Funkce volá funkci IDE_Cache_Flush(), lze tedy poté přistupovat přímo k disku.

int FAT_Seek(tFile *f,unsigned long pos)

Nastaví polohu v souboru, povolené hodnoty jsou < 0; f->FileSize >.

int FAT_Clear(tFile *f)

Nastaví nulovou délku souboru a uvolní jím alokovanou paměť.

int FAT_Create_File(tFile *dir,char *name,unsigned char attr,tFile *f)

Vytvoří soubor nulové délky v zadaném adresáři se jménem name a atributy attr. Pokud se parametr f nerovná NULL, bude připraven k použití.

7 Možnosti rozšíření práce s diskem

Navržený řadič IDE rozhraní podporuje jen základní operace a řadu jeho funkcí je možné implementovat efektivněji. Zvýšit můžeme nejen rychlost přenosu, ale i přístupnost dat. Následující část popisuje možnosti jeho rozšíření.

7.1 Použití vyrovnávací paměti

Velmi omezující je dostupná paměťová kapacita procesoru platformy FITKit. Pokud je zapotřebí zápisu na disk, musí být použita vyrovnávací paměť o kapacitě minimálně 512B, což je ale čtvrtina celkové kapacity. Společně s dalšími paměťovými požadavky nelze použít tuto vyrovnávací paměť s např. současným přenosem souborů pomocí protokolu Xmodem 1K.

Navíc v případě, kdy potřebujeme náhodný přístup k datům, je při použití paměti cache přenášen celý sektor, což je časově velmi náročná operace (vzhledem k max. přenosové rychlosti rozhraní SPI). Pokud není paměť použita a data jsou čtena, je náhodný přístup pouze simulován a taktéž dochází k přenosu celého sektoru (a to i opakovaně).

Také práce s tabulkami FAT (hlavně při zápisu) by se velmi zrychlila použitím paměti pro více sektorů (zápis do FAT tabulky paralelně se zápisem souboru, popř. práce s více soubory současně).

Z těchto hledisek se jeví jako velmi výhodné vytvořit vyrovnávací paměť přímo na řadiči uvnitř FPGA. K tomuto účelu lze s výhodou použít některou ze čtyř 2KB rychlých pamětí (BRAM), nacházejících se na jednotce FPGA. Tato paměť je dvou-portová, zjednodušující řešení současného přístupu disku i rozhraní SPI k uloženým datům, a navíc má dostatečnou kapacitu na uložení čtyř sektorů.

Další nesporná výhoda je možnost přístupu k datům ostatních obvodů na jednotce FPGA. Těmito obvody mohou být např. obvody pro práci s 8MB dynamickou pamětí, práce s grafickým rozhraním a jiné obvody pro práci s velkým objemem dat. Ve vyrovnávací paměti může být tedy zároveň uložen poslední čtený/zapisovaný sektor tabulky FAT, aktuálně zpracovávaný sektor souboru pomocí rozhraní SPI, aktuálně přenášený sektor mezi pamětí a diskem a sektor dat pro další zařízení.

Zvýšit lze též propustnost rozhraní SPI při čtení z disku, neboť dosavadní řadič IDE rozhraní zpomaluje čtení z disku o čekání na odpověď žádosti o čtení. Rozšířený řadič pak může sektor načíst/uložit sám a přenášet data bez zdržení, případně je dát přímo k dispozici obvodům uvnitř jednotky FPGA.

7.2 Přenos pomocí DMA

Dalším plánovaným rozšířením je možnost přenášet data v DMA módech. Zatímco vzhledem k propustnosti SPI rozhraní stačí i nejpomalejší PIO mód, pro obvody na jednotce FPGA může být (a zřejmě i bude) jeho přenosová rychlost nedostatečná.

Přenos v DMA módech je podmíněn právě použitím vyrovnávací paměti na řadiči. Případný řadič, který by měl na starost přenos celého sektoru by mohl být jednodušší, díky řízení ze strany disku.

7.3 Nové verze knihoven funkcí

Knihovny funkcí jsou stavěny na stávajícím typu řadiče. Počítají s vyrovnávací pamětí na straně procesoru. Bude nutné vytvořit nové ovládání vyrovnávací paměti a nových obvodů na straně řadiče. Dosud se pracovalo se šestnácti-bitovými slovy, ale nový řadič by mohl umožnit náhodný přístup i k jednotlivým znakům.

Taktéž je možné vytvořit několik verzí specializovaných knihoven. Užitečná by mohla být např. odlehčená verze knihovny, která bude poskytovat jen základní funkce pro práci s diskem (předpokládala by přítomnost pouze jednoho disku a jednoho oddílu s FAT32), případně pouze s funkcemi čtení z disku (není-li zápis potřeba). Nebo naopak verze, nabízející velkou sadu funkcí. Nabízí se i verze, specializované pro práci s obvody na jednotce FPGA.

Dále pak mohou vzniknout knihovny funkcí pro práci s jinými souborovými systémy (FAT 16 a FAT 12), případně pracující s rozhraním ATAPI.

8 Závěr

Cílem této bakalářské práce bylo dodat platformě jednoduché a kompatibilní velkokapacitní úložiště dat. To bylo také získáno v podobě disku s IDE rozhraním, který je navíc v dnešní době poměrně levný (disky s malou kapacitou kolem několika GB lze někdy získat i zadarmo) a rychlý, implementovaný řadič je také jednoduchý a nespotřebovává mnoho místa na jednotce FPGA.

Připojení IDE disku k platformě poskytuje jiným projektům přístup k velkokapacitnímu zařízení, na kterém lze skladovat data velkých rozměrů a přenášet je i dále použitím souborového systému, který je velmi rozšířen a podporován v mnoha operačních systémech a aplikacích. Projekt by měl být i nadále rozšiřován, aby se dosáhlo větších výkonů a snadnější manipulace s daty.

Navržené řešení může být s výhodou použito u dalších prací, např. pro MP3 přehrávač, operační systémy a rozhraní Ethernet (přenos souborů). Další uplatnění může nalézt v projektech, pro sběr dat (různá měření teplot, napětí atd.) nebo práce s grafickým rozhraním (přehrávání videí).

Literatura

[1] *AT Attachment Interface with Extensions (ATA-2)*. Revize 4c [online], poslední aktualizace 18. 3. 1996. Dostupné z WWW: <http://www.t13.org/Documents/UploadedDocuments/project/d0948r4c-ATA-2.pdf>

[2] Microsoft Corporation. *Microsoft Extensible Firmware Initiative FAT32 File System Specification* Verze 1.03 [online]. c 2000, poslední aktualizace 6. 12. 2000. Dostupné z WWW: <http://www.microsoft.com/whdc/system/platform/firmware/fatgen.mspx>

[3] Dobiash, Jack. *FAT32 Structure Information* [online], poslední aktualizace 4. 12. 2005. Dostupné z WWW: <http://home.teleport.com/~brainy/fat32.htm>

[4] Fučík, Otto. Platforma FITKit [online], [cit. 10. 5. 2007]. Dostupné z WWW: <http://merlin.fit.vutbr.cz/FITkit/>

Seznam příloh