VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY
DEPARTMENT OF CONTROL AND INSTRUMENTATION

INOVACE LABORATORNÍCH ÚLOH V BLOS
INOVATION OF LABORATORY TASKS FROM BLOS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Jakub Urban**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Radovan Holek, CSc.**
SUPERVISOR

BRNO 2019

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Kybernetika, automatizace a měření**
Ústav automatizace a měřicí techniky

Student: Bc. Jakub Urban ***ID:*** 154901
Ročník: 2 ***Akademický rok:*** 2018/19

NÁZEV TÉMATU:

Inovace laboratorních úloh v BLOS

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je rozšíření laboratorních úloh předmětu BLOS o úlohy pro využití periferií realizovaných v jazyce VHDL.

1) Prostudujte možnoti vývojového kitu DIGILENT.
2) Seznámte se s dostupnými periferiemi PMOD pro tento kit. Zvolte vhodné periferie.
3) Navrhěte sadu laboratorních úloh pro vybrané periferie. Uvažujte různé varianty řešení. Použijte jazyk VHDL.
4) Ke každé laboratorní úloze vypracujte zadávací dokumentaci pro její vyřešení.
5) Vypracujte vzorová řešení jednotlivých úloh včetně simulace.
6) Úlohy ověřte na vývojovém kitu DIGILENT.

DOPORUČENÁ LITERATURA:

www xilinx.com

www. digilent.com

Termín zadání: 4.2.2019 ***Termín odevzdání:*** 13.5.2019

Vedoucí práce: Ing. Radovan Holek, CSc.
Konzultant:

doc. Ing. Václav Jirsík, CSc.
předseda oborové rady

ABSTRAKT

Tato práce se zabývá inovací úloh bakalářského předmětu Logické obvody a systémy. Byly navrženy a odladěny celkem tři úlohy v jazyce VHDL pro vývojový kit Digilent Nexys 3 a periferie Pmod, k němu připojených. První úloha je zaměřena na čtení stisknutého tlačítka na maticové klávesnici. Druhá a třetí úloha si klade za cíl zobrazit stisknuté tlačítko na OLED displeji, liší se použitým kontrolérem pro tento displej. Pro všechny úlohy byly vytvořeny návody pro jejich řešení.

KLÍČOVÁ SLOVA

VHDL, Nexys3, Pmod, maticová klávesnice, OLED displej

ABSTRACT

This thesis deals with innovation of tasks from bachelor subject Logical systems. Three tasks in total in VHDL language were designed and tested for development kit Digilent Nexys 3 and peripheries Pmod which are connected to it. First task is focused on reading pressed key on matrix keyboard. Second and third tasks targets on displaying pressed key on OLED display, they differ by used controller for this display. Instructions for all tasks were created.

KEYWORDS

VHDL, Nexys3, Pmod, matrix keyboard, OLED display

URBAN, Jakub. *Inovace laboratorních úloh v BLOS*. Brno, 2019, 76 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky. Vedoucí práce: Ing. Radovan Holek, CSc.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Inovace laboratorních úloh v BLOS“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Radovanu Holkovi, CSc. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

podpis autora

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Předmět bakalářského studia Logické obvody a systémy (BLOS) patří mezi povinné předměty studijního oboru Automatizační a měřicí technika. Součástí předmětu jsou praktické úlohy, které mají studenti řešit samostatně. Náplň těchto úloh se liší od jednodušších logických systémů realizovaných ve schématu, přes systémy realizované i v jazyce VHDL až po složitější systémy, které jsou realizovány čistě v jazyce VHDL.

Pro tyto úlohy je využívána vývojová deska Digilent Nexys 3, která obsahuje řadu vestavěných periferií, které nabízejí zajímavé využití, ale uživatel je jimi do jisté míry limitován. Motivací pro tuto práci je rozšíření těchto základních úloh o nové úlohy, které budou využívat připojitelných periferií Pmod. Těchto periferií je celá řada, některé jsou dosti složité, proto je záměrem vybrat vhodné periferie pro realizace úloh právě v jazyce VHDL. Značnou výhodou je to, že se tyto periferie dají připojit k řadě jiným vývojovým deskám a mikrokontrolérům, takže zde vzniká perspektiva do budoucnosti.

Cílem této práce je navrhnout sadu úloh v jazyce VHDL pro již zmíněnou vývojovou desku a připojitelné periferie a vytvořit návody pro jejich řešení. Součást této práce bude také vytvoření vzorového řešení pro vzniklé úlohy, v neposlední řadě také ověření správnosti řešení na vývojové desce a také pomocí simulace.

1 Teoretická část

1.1 Vývojová deska Digilent Nexys 3

Tato vývojová deska je používán v předmětu BLOS (Logické obvody a systémy). Nexys 3 je kompletní platforma pro návrh digitálních obvodů, je založena na programovatelném hradlovém poli Spartan-6 LX16 od firmy Xilinx.

Obsahuje velké množství periferií jako je:

- Oscilátor pracující na frekvenci 100MHz
- Fyzická vrstva pro Ethernet o rychlosti 10/100 Mb/s
- 16 MB paměti RAM
- 16 MB paměti flash
- 8-bitový VGA port
- USB-UART převodník
- USB port pro připojení klávesnice či myši
- USB2 port pro programování a přenos dat
- Jeden šedesáti osmi pinový VHDC konektor
- Osm uživatelských LED diod
- Pět tlačítek
- Osm přepínačů s jezdcem
- Čtyřmístný sedmisegmentový displej
- Čtyři Pmod konektory pro přípojení rozšiřujících přídavných periferií

Programování probíhá pomocí USB kabelu. Tento kabel zajišťuje také přenos dat a napájení desky, není potřeba externího zdroje. Podporováno je několik CAD programů firmy Xilinx jako jsou EDK, Chipscope a WebPack.

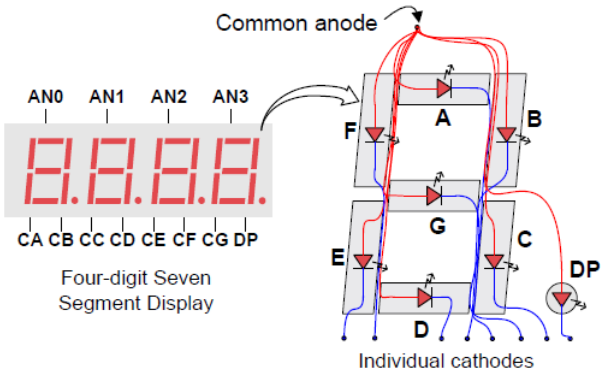
1.1.1 Vstupy a výstupy

8 uživatelských LED diod je připojeno anodami k desce skrz 380 ohm rezistory, rozsvítí se tedy při zápisu logické jedničky na příslušný pin. Mimo těchto uživatelských LED diod se na vývojové desce nacházejí další, které indikují zapnutí desky, stav programování FPGA a stav USB a Ethernet portů.

1.1.2 Sedmisegmentový displej

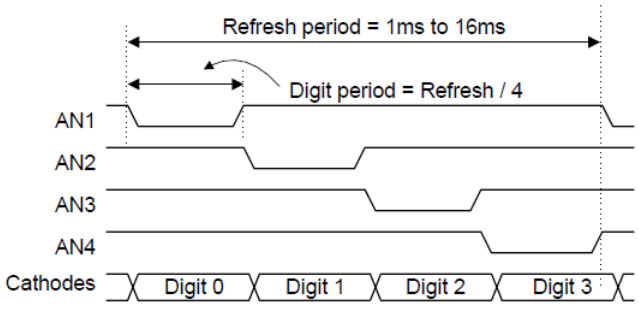
Nexys 3 obsahuje čtyři sedmisegmentové displeje. U každého displeje je kromě sedmi segmentů možno použít i desetinou tečku, každý displej potom má celkem osm LED diod. Tento displej je v zapojení se společnou anodou, anody LED diod jsou spojeny do jedné. Katody jsou z pohledu jednoho displeje odděleny, ale napříč celým

displejem jsou spojeny tak, že jsou navzájem propojeny odpovídající segmenty všech pozic. Výstupem z displeje je sedm segmentů CA-CG, desetinná tečka DP a počet anod AN, který odpovídá počtu pozic. Základním tvarem je digitální číslo osm, do kterého je možno zobrazit čísla 0-9, znaky A-F a případně i další symboly. Jednotlivé segmenty se rozsvítí při zapsaní logické nuly.



Obr. 1.1: Zapojení sedmisegmentového displeje

Pro zobrazení více různých znaků najednou se využívá nedokonalosti lidského oka, je třeba přepínat jednotlivé znaky (anody) s takovou frekvencí, aby to lidské oko nebylo schopno postřehnout. Minimální obnovovací frekvence by měla být 60Hz (což odpovídá periodě 16ms). Při použití obnovovací frekvence např. 100Hz (perioda 10ms), zbývá na jednotlivé anody $10/4 = 2{,}5$ms. [1]



Obr. 1.2: Přepínání anod sedmisegmentového displeje

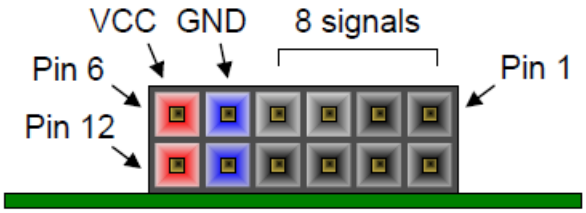
1.2 Periferní moduly Digilent Pmod

Pmod je technologický standart periferních modulů od firmy Digilent. Tyto přídavné moduly umožňují rozšířit možnosti vývojové desky o další programovatelné komponenty. Moduly jsou připojovány pomocí 6 nebo 12 pinového konektoru, buďto přímo

nebo pomocí kabelu, tyto piny přenášejí jak digitální signály tak napájecí napětí. Nejčastěji využívána rozhraní jsou SPI a I^2C, ale některé moduly používají tyto signály jako běžné vstupy a výstupy (GPIO).

Moduly jsou napájeny z hostujícího zařízení, většinou je napájecí úroveň 3,3V, ale některé moduly podporují i 5V. Při využití 12 pinového konektoru oba napájecí piny poskytují stejné napětí a mohou být spojeny na straně hostujícího zařízení nebo modulu.

Konektor na straně modulu je typu samec, zatímco na straně hostující desky to bude 12 pinový konektor typu samice, pokud se na desce nachází více konektorů, jejich středy jsou od sebe vzdáleny 0,9 palců, což dovoluje zapojit více modulů vedle sebe. Pmod konektor je zobrazen na následujícím obrázku tak, jak je umístěn na vývojové desce. Piny 12 a 6 poskytují napájecí napětí 3,3V.



Obr. 1.3: Pohled na Pmod konektor

Přiřazení jednotlivých I/O signálů je definováno následovně:

Tab. 1.1: Přiřazení I/O signálů

Pmod rozhraní typu	**Sběrnice**	**Počet pinů**
1	GPIO	6
1A	rozšířené GPIO	12
2	SPI	6
2A	rozšířené SPI	12
3	UART	6
3A	rozšířený UART	12
4	H-můstek	6
5	dualní H-můstek	6
5A	rozšířený duální H-můstek	12
6	I^2C	6
7	I^2S CODEC	12

Existuje mnoho přídavných periferních modulů jako jsou např. 3-osý MEMS akcelerometr (Pmod ACL2), H-můstek pro řízení stejnosměrných motorů (Pmod HB5), slot pro MicroSD karty (Pmod MicroSD) a další. [2]

Pro potřeby této práce byly vybrány tyto periferní moduly: maticová klávesnice(Pmod KYPD) a OLED displej(Pmod OLED).

1.2.1 Maticová klávesnice

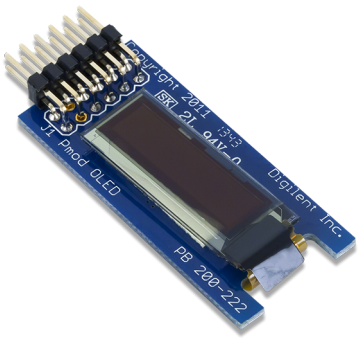
Jako maticová klávesnice je použita PmodKYPD, která má 16 tlačítek s čísly '0'-'9' a znaky 'A'-'F' uspořádaných do 4 řad a 4 sloupců. Klávesnice je zobrazena na následujícím obrázku:

Obr. 1.4: Maticová klávesnice

PmodKYPD využívá 12 pinový konektor, přičemž 2 piny jsou využity pro napájení 3.3V, 2 pro zem a zbylých 8 pinů je využito pro vstupy a výstupy do klávesnice.

1.2.2 OLED displej

PmodOLED je organický LED modul s displejem o rozlišení 128x32 pixelů. Využívá 12 pinový konektor pro SPI rozhraní.



Obr. 1.5: OLED displej

1.3 Jazyk VHDL

VHDL je zkratka pro VHSIC (Very High Speed Integrated Circuits) Hardware Description Language. Což lze přeložit jako programovací jazyk k popisu hardware pro velmi rychlé integrované obvody. Byl vyvinut pro vojenské účely na ministerstvu obranu USA s cílem vytvořit dokumentaci a popis chování integrovaných obvodů. Historie tohoto jazyka se datuje do roku 1987, kdy se objevilo první vydání. Dnes nejvýznamnější verze je z roku 1993. Na rozdíl od většiny programovacích jazyků se u VHDL používá paralelní (nikoli sekvenční) programování. To znamená, že všechny příkazy a procesy se provádějí najednou a ne postupně podle toho jak jsou zapsány. Řešení, kterého dosáhneme pomocí jazyka VHDL musí být syntetizovatelné, to znamená, že za napsaným kódem si musíme představit reálné hardwarové řešení. U kompilace kódu se nejedná o kompilaci v jazyce VHDL, nýbrž o syntézu, kterou neprovádí kompilátor, ale syntetizér, který syntetizuje obvod. Základní součástí každého VHDL kódu jsou dvě části: entita a architektura. [3]

Entita primárně určuje porty, které mohou být vstupní, výstupní a vstupně-výstupní (do kterých se dá zapisovat a zároveň z nich číst). Pod pojmem entita si lze představit celkový digitální obvod mající vstupy a výstupy (porty), přičemž entitou může být klidně jedno jednoduché logické hradlo, ale např. i složitý digitální obvod.

Architektura definuje vlastní chování a funkci entity (např. vztahy mezi porty entity). Architektura tedy definuje vnitřek entity. Každá entita musí mít alespoň jednu architekturu. [4]

VHDL se také využívá k psaní testovacích programů, tzv. testbenchů a k simulaci logických obvodů.

1.4 Vývojové prostředí Xilinx ISE Design Suite

Byl použit software ISE (Integrated Synthesis Enviroment) Design Suite ve verzi 14.3. Toto vývojové prostředí umožňuje návrh a syntézu obvodů FPGA. Návrh je možné tvořit jako schéma pro jednodušší obvody, pro složitější pak v jazyce VHDL případně Verilog. Výsledkem syntézy je soubor .bit, který je nutné nahrát do cílového zařízení buďto přímo z vývojového prostředí skrz ISE iMPACT nebo pomocí softwaru Digilent Adept, který umožňuje i konfiguraci cílového zařízení.

2 Maticové klávesnice

2.1 Teoretický úvod

PmodKYPD je maticová klávesnice s 16 tlačítky s čísly '0'-'9' a znaky 'A'-'F' uspořádaných do čtyř řad.

PmodKYPD využívá 12 pinový konektor, rozložení pinů je uvedeno v tabulce 2.1. . [5]

Tab. 2.1: Popis jednotlivých Pinů maticové klávesnice

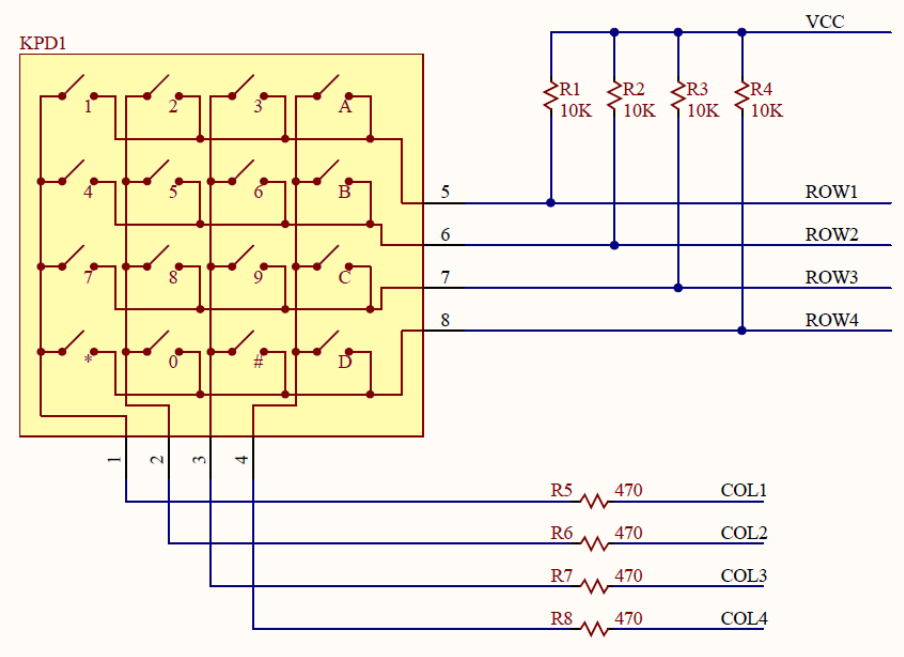
PIN	**signál**	**popis**
1	COL4	sloupec 4
2	COL3	sloupce 3
3	COL2	sloupec 2
4	COL1	sloupec 1
5	GND	zem napájecího napětí
6	VCC	napájecí napětí (3.3)
7	ROW4	řadek 4
8	ROW3	řadek 3
9	ROW2	řadek 2
10	ROW1	řadek 1
11	GND	zem napájecího napětí
12	VCC	napájecí napětí (3.3V)

2.2 Postup řešení

Při řešení této úlohy je vhodné rozdělit si práci do několika kroků, v návrhu VHDL se toto promítne jako vytvoření několika komponent, které lze podle potřeby propojit. Nejprve je třeba navrhnout komponentu, která bude číst stisknuté tlačítko na maticové klávesnici a také jeho hodnotu vhodně kódovat pro pozdější použití, nazvána může být např. Decoder. Klávesnice obsahuje 16 tlačítek, takže bude stačit 4 bitový signál pro dekódování. Kromě samotné hodnoty stisknutého tlačítka je vhodné nějakým způsobem předávat příznak, že bylo či nebylo stisknuto tlačítko a to buď přidáním pátého bitu nebo samostatné proměnné.

Čtení stisknutého tlačítka na klávesnici je založeno na principu postupující nuly. Postupně je zapisována do jednotlivých sloupců logická nula. Na signály jsou připojeny pull-up rezistory, a proto v klidovém stavu (není-li stisknuto tlačítko) je na výstupu těchto signálů logická jednička. Při stisknutí tlačítka dojde k uzemnění signálu a výstupní signál příslušného řádku se změní na logickou nulu. Znalostí, do kterého sloupce byla zapsána logická nula a ve kterém řádku se objevila logická nula určíme, které tlačítko bylo stisknuto.

Pull-up rezistor se používá na udržení napětí na vstupu na úrovni napájecího napětí (což odpovídá logické 1). Řádky a sloupce jsou izolované, je možno detekovat stisknutí více tlačítek najednou, výjimku tvoří tlačítka nacházející se ve stejné řadě.



Obr. 2.1: Čtení stisknutého tlačítka z maticové klávesnice

Při čtení stisknutého tlačítko je třeba dodržet zpoždění mezi zapsáním hodnoty do jednotlivých sloupců, taková prodleva by měla být zhruba 1ms. Řádově kratší prodleva se uplatňuje i mezi zapsáním hodnoty do sloupce a vyčtením hodnoty z řádku.

Při vytváření .ufc souboru je třeba deklarovat příslušný konektor JX, ke kterému je připojena maticová klávesnice. Dále v programu je třeba definovat horní část portu JX jako výstupní a jeho dolní část jako vstupní. To je patrné z tabulky jednotlivých pinů a z toho, že je potřeba zapisovat hodnoty do sloupců a číst je z řádků.

Dalším krokem je zobrazení dekódovaného znaku na sedmisegmentovém displeji. Tato periferie je dobře známa ze standartních úloh. Pro každý zakódovaný znak ve čtyř bitovém kódu je třeba rozsvítit příslušné segmenty displeje. Pokud chceme

zobrazit pouze právě stisknutý znak klávesnice, stačí aktivovat příslušnou anodu.

Pro zobrazení historie stisknutých kláves (nejpozději stisknutá klávesa se zobrazí na pozici úplně vpravo na sedmisegmentovém displeji, předchozí znaky se posunou směrem doleva) je třeba vytvořit 16 bitový posuvný registr, do kterého se budou načítat nově příchozí data zprava. Pro tento úkon můžeme využít operátoru sjednocení, jak je ukázáno na následujícím příkladu:

```
signal shift_registr : STD_LOGIC_VECTOR (15 downto 0);
shift_registr <= shift_registr (11 downto 0) & data;
```

Pro zobrazení více různých čísel najednou se využívá nedokonalosti lidského oka, je třeba přepínat jednotlivé číslice (anody) s takovou frekvencí, aby to lidské oko nebylo schopno postřehnout. Minimální obnovovací frekvence by měla být 60Hz (což odpovídá periodě 16ms).

Zde můžeme využít 20 bitový counter, který nám základní hodinový signál vydělí na frekvenci 95Hz, což přibližně odpovídá periodě 10,5ms. To splňuje podmínku přepínání jednotlivých číslic. Ideální je vytvořit multiplexer z horních dvou bitů našeho counteru. V každém ze čtyř kroků rozsvítíme jednu anodu a do proměnné, která představuje zobrazovaný znak na sedmisegmentovém displeji zapíšeme část shift registru, který reprezentuje znak na pozici rozsvícené anody.

3 OLED displej

3.1 Teoretický úvod

Pmod OLED má tyto rozměry: 16 řádků a 4 sloupce.

Grafický panel využívá řadič displeje Solomon Systech SSD1306. Pmod OLED komunikuje s vývojovým kitem pomocí SPI rozhraní. Zápisem a ponecháním signálu Chip Select (CS) na úrovní logické nuly je uživatel schopen posílat příkazy i data na řadič displeje v závislosti na stavu Data/Command (D/C) pinu.

RES pin je využíván k resetu řadiče displeje SSD1306

Uživatel může rozsvítit kterýkoliv pixel, zobrazit předem definovaný znak nebo dokonce nahrát bitmapový obrázek na obrazovku displeje.

Dva zdroje napájení displeje jsou řízeny dvěma FET tranzistory. VDDC řídí napájení do logiky displeje a VBATC řídí napájení displeje jako takového.

Pro správné fungovaní je třeba displej nejprve inicializovat.

OLED displej má speciální sekvence zapnutí a vypnutí pro prodloužení životnosti.

Sekvence zapnutí:

1. Zapnout napájení VDD.
2. Odeslat příkaz Vypnout zobrazení.
3. Inicializovat displej na požadovaný provozní režim.
4. Vymazat obrazovku.
5. Zapnout napájení VBAT.
6. Zpoždění 100ms.
7. Odeslat příkaz Zobrazit

Sekvence vypnutí:

1. Odeslat příkaz Vypnout zobrazení.
2. Vypnout napájení VBAT.
3. Zpoždění 100 ms.
4. Vypnout napájení VDD.

[6]

Tabulka 3.1 zobrazuje popis jednotlivých pinů.

Tab. 3.1: Popis jednotlivých pinů OLED displeje

PIN	**signál**	**popis**
1	CS	Chip Select (výběr chipu)
2	MOSI	Master Out,Slave In (master výstup, slave vstup)
3	NC	Not Connected (není připojen)
4	SCK	Serial Clock (sériový hodinový signál)
5	GND	Power Supply Ground (zem napájecího napětí)
6	VCC	Power Supply (napájecí napětí)(3.3V)
7	D/C	Data/Command control (data/příkaz)
8	RES	Power Reset (reset)
9	VBATC	Vbat Battery Voltage Control (řízení napájení Vbat)
10	VDDC	Vdd Logic Voltage Control (řízení napájení Vdd)
11	GND	Power Supply Ground (zem napájecího napětí)
12	VCC	Power Supply (napájecí napětí)(3.3V)

3.2 OLED kontrolér na bázi jádra mikrokontroléru

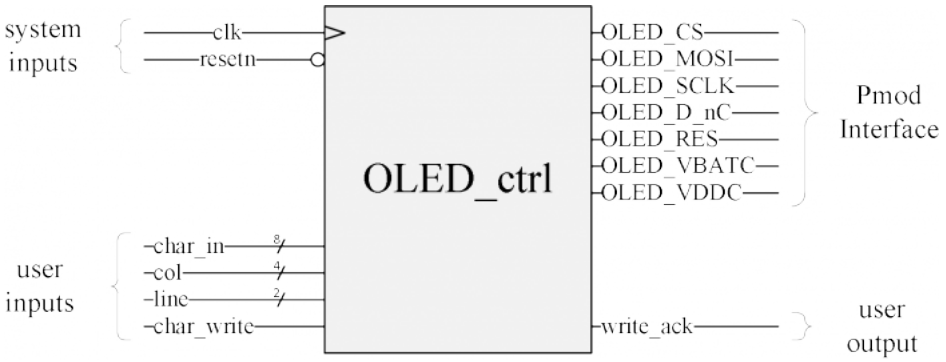
Blokový diagram tohoto kontroléru se nachází v příloze. Tento modul zobrazuje text zakódovaný v ASCII znacích na OLED displeji. Dokáže zobrazit znak na jakékoli pozici na displeji, pomocí uživatelských proměnných char_in (vstupní znak), line (řádek) a col (sloupec). Jsou podporovány pouze standartní znaky (ASCII hodnoty od 0 do 127).

Zápis znaku probíhá tak, že pomocí osmi přepínačů char_in je zvolen konkrétní znak, dvoubitová proměnná line určuje, na kterém řádku (0-4) OLED displeje bude zvolený znak zobrazen a čtyřbitová proměnná col zase, ve kterém sloupci (0-15). K potvrzení zápisu znaku slouží proměnná char_write, během doby, kdy je tato proměnná v logické jedničce se nesmí měnit hodnota ASCII znaku ani pozice. Také musí proměnná zůstat v logické jedničce dokud nepřijde potvrzení v podobě signálu write_ack, reakční doba tohoto kontroléru se pohybuje mezi 100μs a 1ms.

Dalšími vstupními signály je hodinový signál clk a resetn, který je aktivní při logické nule. Dalších výstupních signálů je sedm a jsou to OLED_CS, OLED_MOSI, OLED_SCLK, OLED_D_nC, OLED_RES, OLED_VBATC a OLED_VDDC.

Komunikačním protokolem nad tímto displejem je rozhraní SPI.

Kontrolér se skládá z těchto modul komponent: OLED_CPU, OLED_SPI, Coderam_mgr a RAM_prog.

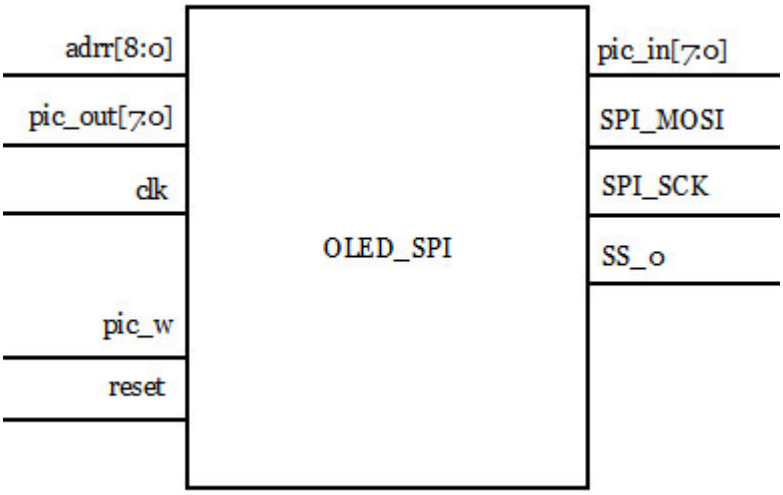


Obr. 3.1: Vstupy a výstupy OLED kontroléru

3.2.1 Komponenty pro RAM paměť

V komponentě RAM_prog se nachází ASCII znak pro každou adresu. Definicí prvních 64 adres lze definovat, co se na displeji defaultně zobrazí. Komponenta Coderam_mgr zajišťuje, zda se na konkrétní pozici displeje zobrazí znak uložený v RAM paměti nebo se načte znak zadaný uživatelskou vstupní hodnotou.

3.2.2 Komponenta pro SPI rozhraní



Obr. 3.2: Blokové schéma komponenty OLED_SPI

Blokový diagram této komponenty se nachází v příloze. Signály, které se využívají při komunikaci mezi OLED displejem a vývojovým kitem jsou následující:

1. $\overline{CS}$: Tento signál je pro Chip Select(výběr čipu), pokud se nachází v logické nule, OLED displej přijímá sériové příkazy, naopak pokud je v logické jedničce, nepřijímá další příkazy.

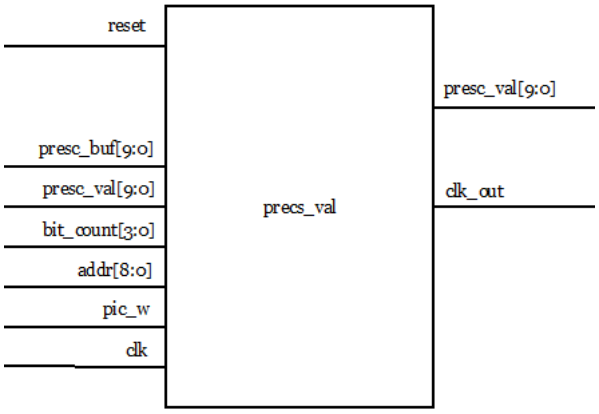
2. SCK: Toto je hodinový signál SPI, stoupající hrana tohoto signálu určuje platnost posílaných dat.
3. MOSI: Tento signál je pro posílání dat od Masteru ke Slavu.

Běžně je využíván také signál MISO pro přenos dat od Slavu k Masteru, ale v našem případě není použit. Přenášena SPI zpráva má délku 8 bitů.

Dále zde budou popsány jednotlivé procesy, které vytvářejí tento SPI modul.

3.2.2.1 Proces hodnoty předděličky

První je proces hodnoty předděličky, tento proces má synchronní reset. Desetibitová předdělička (prescaler) určuje frekvenci sériového hodinového kmitočtu(SPI_SCK), pracuje jako čítač směrem dolů, pokud je její hodnota nulová a zároveň nedošlo k odeslání všech osmi bitů, dochází ke změně úrovně sériového hodinového signálu z logické jedničky na logickou nuly nebo naopak. Jinak je úroveň sériového hodinového signálu nulová.



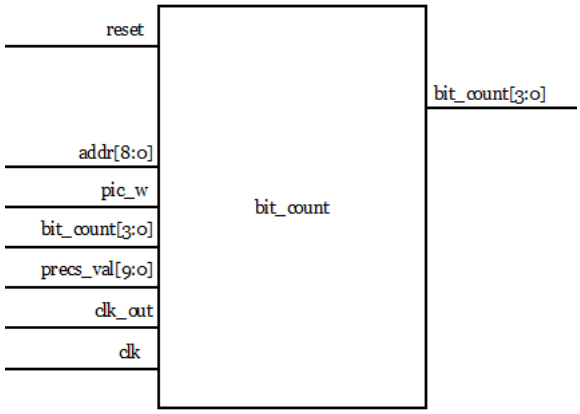
Obr. 3.3: Proces hodnoty předděličky

```
-- generates serial clock when prescaler finishes
   counting and sets prescaler value to presc_buf
precs_val_process: process(clk)
begin
  if clk'event and clk='1' then
     if reset='1' then
      clk_out   <= '0';
      presc_val <= "0000000000";
    elsif presc_val = "0000000000" then
        if bit_count/= "0000" then
          clk_out   <= not clk_out;
```

```
            presc_val <= presc_buf;
          else
            clk_out   <= '0';
            if pic_w='1' and addr=RXTX_ADDR then
              Presc_val <= presc_buf;
            end if;
          end if;
      else
          presc_val <= presc_val - 1;
      end if;
    end if;
end process;
```

3.2.2.2 Proces počtu odeslaných bitů

Tento proces má také synchronní reset. Při náběžné hraně základního hodinového signálu je kontrolována podmínka: není-li odesláno všeh osm bitů a zároveň dočítala předdělička a úroveň sériového hodinového kmitočtu je logická jednička, pak se mění počet odeslaných bitů o jedna směrem dolů.



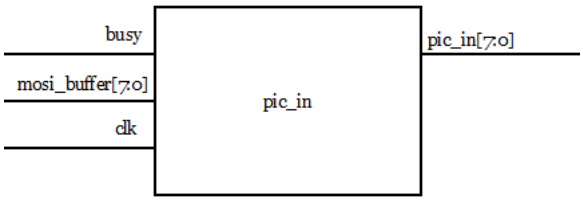
Obr. 3.4: Proces bitového čítače

```
-- counting the number of sent bits through SPI
bit_count_process: process(clk)
begin
  if clk'event and clk='1' then
    if reset = '1' then
      bit_count <= "0000";
    elsif bit_count/="0000" then
```

```
      if presc_val = "0000000000" and clk_out = '1' then
        bit_count <= bit_count - 1;
      end if;
    elsif pic_w='1' and addr=RXTX_ADDR then
      bit_count <= "1000";
    end if;
  end if;
end process;
```

3.2.2.3 Proces výstupního statusového registru

Jedná se o kombinační obvod, pokud je adresou standartní adresa, pak je posledním bitem statusového registru příznak busy. Pokud se jedná o adresu pro zápis a čtení, pak je obsahem tohoto registru mosi_buffer.

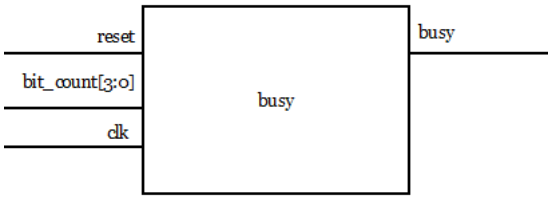


Obr. 3.5: Proces výstupního statusového registru

```
-- sets last bit of output status register to busy flag
pic_in_process: process(addr, busy, mosi_buffer)
begin
   if addr=ST_ADDR  then
    pic_in <= "0000000" & busy;
  elsif addr=RXTX_ADDR then
    pic_in <= mosi_buffer;
  else
    pic_in <= "00000000";
  end if;
end process;
```

3.2.2.4 Proces příznaku busy

Příznak busy (zaneprázdněno) není nastaven v případě synchronního resetu nebo v případě, kdy došlo k odeslání všech osmi bitů (je hotový přenos). Jinak je tento příznak nastaven na logickou jedničku.



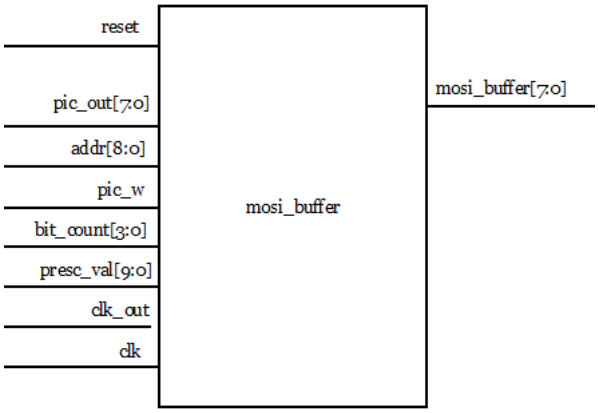
Obr. 3.6: Proces příznaku busy

```
-- busy flag is not set when the synchronous reset is
   acitve or when all eight bits were sent, otherwise
   busy flag is set
busy_process: process(clk)
begin
  if clk'event and clk='1' then
    if reset = '1' or bit_count="0000" then
      busy <= '0';
    else
        busy <= '1';
    end if;
  end if;
end process;
```

3.2.2.5 Proces mosi bufferu

Při synchronním resetu je buffer vynulován, při splnění podmínky, kdy nebylo odesláno všechn osm bitů a úroveň sériového hodinového signálu je logická jednička dochází k bitovému posunu směrem doleva.

```
-- when not all eight bits were sent and the level of
   serial clock is high mosi_buffer is shifted left
mosi_buffer_process: process(clk)
begin
  if clk'event and clk='1' then
    if reset = '1' then
      mosi_buffer <= "00000000";
    elsif bit_count/="0000" and clk_out='1' and presc_buf
        = presc_val then
      mosi_buffer <= mosi_buffer(6 downto 0) & '0';
    elsif bit_count="0000" and pic_w='1' and addr=
       RXTX_ADDR then
```

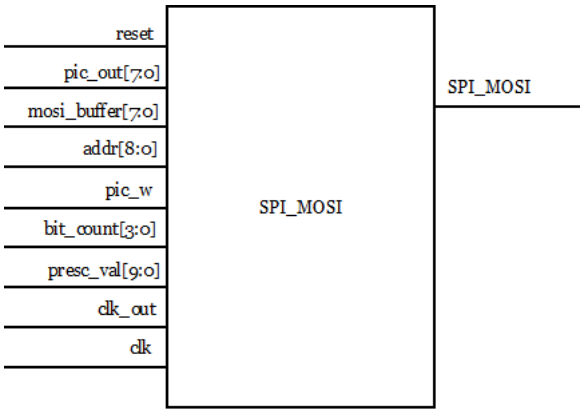


Obr. 3.7: Proces mosi bufferu

```
        mosi_buffer <= pic_out;
    end if;
  end if;
end process;
```

3.2.2.6 Proces signálu SPI_MOSI

Při synchronním resetu je hodnota signálu SPI_MOSI nastavena na logickou jedničku. Pokud je splněna podmínka: úroveň sériového hodinového signálu je logická jednička, dočítala předdělička a nebylo odesláno všech osm bitů, pak je hodnota signálu nastavena na poslední bit mosi_bufferu. Pokud je splněná jiná podmínka, kdy bylo odesláno všech osm bitů, je povolen zápis a adresa je nastavená pro čtení a zápis, pak je hodnota signálu nastavena na poslední bit vstupního statusového registru(pic_out).

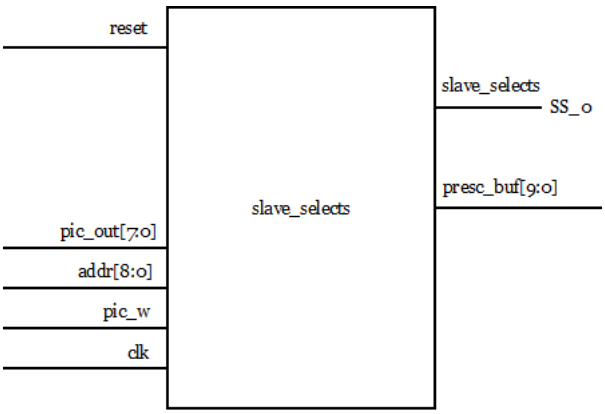


Obr. 3.8: Proces signálu SPI_MOSI

```
-- when the level of serial clock is high, prescaler
   finished counting and not all eight bits were sent,
   one bit is sent through SPI
SPI_MOSI_process: process(clk)
begin
  if clk'event and clk='1' then
    if reset = '1' then
      SPI_MOSI <= '1';
    elsif clk_out='1' and presc_val = "0000000000" and
       bit_count/="0000" then
      SPI_MOSI <= mosi_buffer(7);
    elsif bit_count="0000" and pic_w='1' and addr=
       RXTX_ADDR then
        SPI_MOSI <= pic_out(7);
    end if;
  end if;
end process;
```

3.2.2.7 Proces výběru slavu

Tento proces slouží jednak pro aktivaci slave zařízení a také pro nastavení hodnoty bufferu předděličky. Při synchronním resetu je buffer vynulován a slave zařízení není aktivní. Pokud je splněna podmínka povolení zápisu a adresa je nastavena na standartní, dochází k aktivaci slave zařízení a zároveň jsou do bufferu předděličky zapisovány vrchní tři bity vektoru pic_out a následně posunovány vlevo v závislosti na pic_out(4 downto 2).

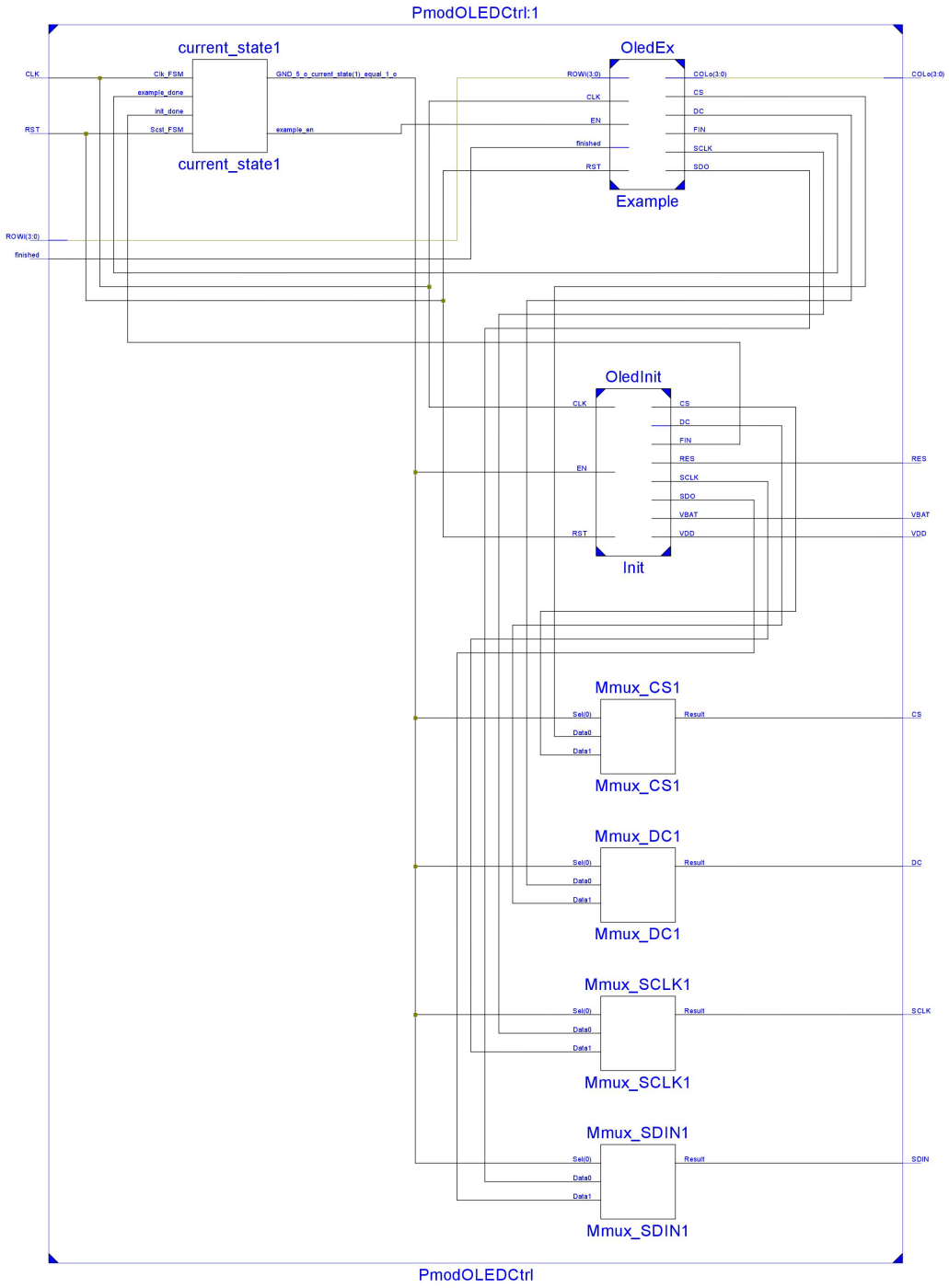


Obr. 3.9: Proces výběru slavu

```
-- selects a slave device and generating value of
   prescaler buffer
slave_selects_process: process(clk)
begin
  if clk'event and clk='1' then
    if reset = '1' then
      presc_buf     <= "0000000000";
      slave_selects <= "0";
    elsif pic_w='1' and addr=ST_ADDR then
         case pic_out(4 downto 2) is
            when "000" => presc_buf <= "0000000" &
               pic_out(7 downto 5);
            when "001" => presc_buf <= "000000" & pic_out
               (7 downto 5) & "1";
            when "010" => presc_buf <= "00000" & pic_out
               (7 downto 5) & "1" & pic_out(7);
            when "011" => presc_buf <= "0000" & pic_out(7
                downto 5) & "1" & pic_out(7 downto 6);
            when "100" => presc_buf <= "000" & pic_out(7
               downto 5) & "1" & pic_out(7 downto 5);
            when "101" => presc_buf <= "00" & pic_out(7
               downto 5) & "1" & pic_out(7 downto 5) &
               "1";
            when "110" => presc_buf <= "0" & pic_out(7
               downto 5) & "1" & pic_out(7 downto 5) &
               "1" & pic_out(7);
            when others => presc_buf <=     pic_out(7
               downto 5) & "1" & pic_out(7 downto 5) &
               "1" & pic_out(7 downto 6);
         end case;
      slave_selects <= pic_out(0 downto 0);
    end if;
  end if;
end process;
```

3.3 OLED kontrolér na bázi stavového automatu

3.3.1 Přehled

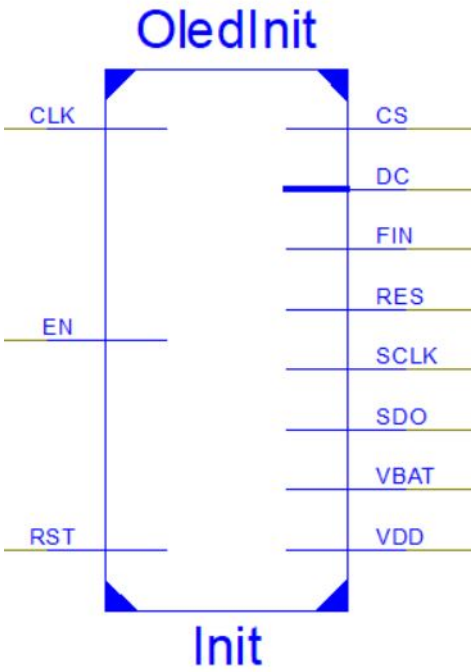


Obr. 3.10: Schéma PmodOLEDctrl

Displej mám velikost 128x32 pixelů, každá znak má velikost 8x8 pixelů, z toho vyplývá že na displeji je možné zobrazit 64 znaků najednou. Je rozdělen na 4 řádky/4 stránky, na každý se vejde 16 znaků. Nejprve je provedena inicializace displeje podle postupu doporučeného výrobcem, poté je zobrazena anglická abeceda a čísla 0-9, dále program několik sekund čeká a následně je zobrazen text: „This is Digilent's PmodOLED". Program implementuje SPI rozhraní pro komunikaci s OLED displejem. PmodOLEDctrl se skládá z těchto bloků: OledInit a OledExample.

3.3.2 Blok OledInit

Vstupy do tohoto bloku jsou signály CLK (systémový hodinový signál o frekvenci 100MHz), EN (povolující tento blok) a RST (globální reset). Výstupy z tohoto bloku jsou signály CS (SPI výběr chipu), DC (data/příkaz), FIN (udávající, že tento blok dokončil svou činnost), RES (PmodOLED RES), SCLK (sériový hodinový signál), SDO (SPI sériový datový výstup), VBAT (povolení VBAT) a VDD (povolení VDD).



Obr. 3.11: Blokové schéma OledInit

Blok OledInit posílá příkazy pro inicializaci displeje a dodržuje mezi nimi určitý interval podle specifikace výrobce. Ve VHDL je posloupnost příkazů realizována jako stavový automat. Výchozím stavem je stav Idle. Pro stavový automat existují dva typy stavů current_state a after_state. Pokud mezi stavy není Transition, přechází se do další stavu v podstatě ihned, pokud tam Transition je, proběhne přechod (SPI

přenos nebo čekaní v bloku Delay) a až poté dojde k přechodu do dalšího stavu. Tento blok využívá bloky SpiCtrl a Delay.

3.3.2.1 Přechodové stavy

Přechodové stavy jsou Transition1, Transition2, Transition3, Transition4 a Transition5. První dva jsou SPI přechodové stavy, ve kterých se povolí SPI přenos a čeká se až proběhne, poté se přechází do stavu Transition 5. Stavy Transition3 a Transition4 jsou čekací stavy, ve kterých se povolí start bloku Delay a čeká se až uběhne daný počet milisekund, poté se přechází do stavu Transition5. Ve stavu Transition5 dochází k zakázání SPI přenosu a bloku Delay. Stavový automat přechází do následujícího stavu (který je zapsán v after_state).

3.3.2.2 SPI stavy

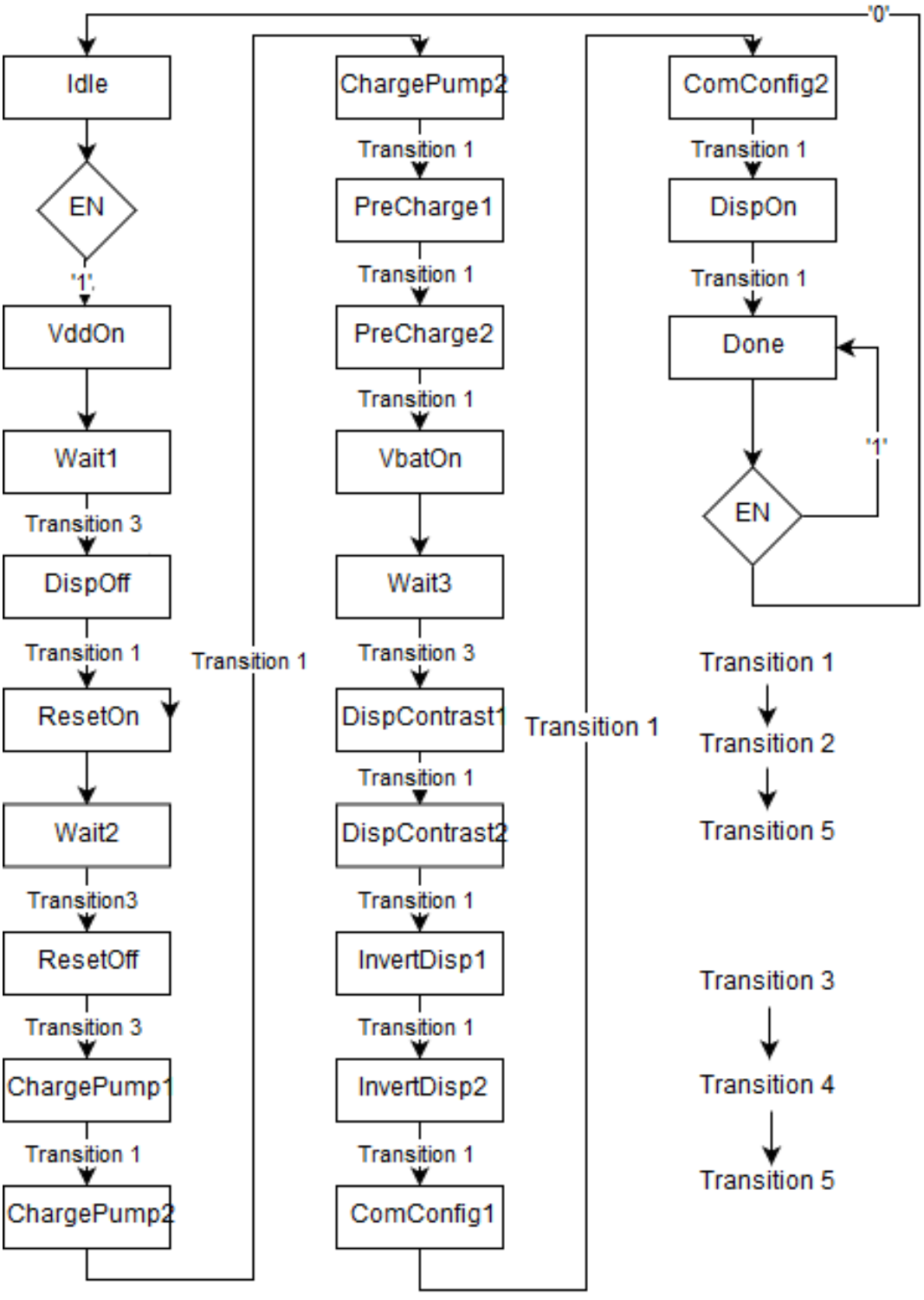
Jedná se o stavy DispOff, ChargePump1, ChargePump2, PreCharge1, PreCharge2, DispContrast1, DispContrast2, ComCongif1, ComConfig2 a DispOn. V těchto stavech se nastaví příkaz pro řadič displeje, který je následně poslán do bloku SPI, poté se čeká na dokončení SPI přenosu a přechází se na další stav.

3.3.2.3 Čekací stavy

Jedná se o stavy Wait1, Wait2 a Wait3, v těchto stavech se pouze čeká daný počet milisekund než proběhne blok Delay.

3.3.2.4 Ostatní stavy

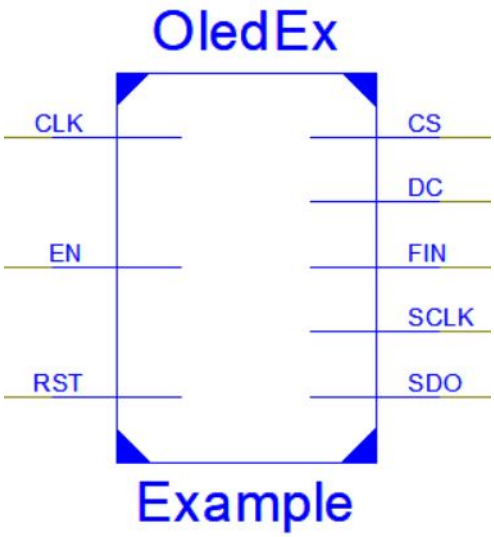
Jedná se o stavy Idle, VddOn, ResetOn, ResetOff, VbatOn a Done. V těchto stavech se nastaví příslušné signály na požadovanou hodnotu.



Obr. 3.12: Vývojový diagram bloku OledInit

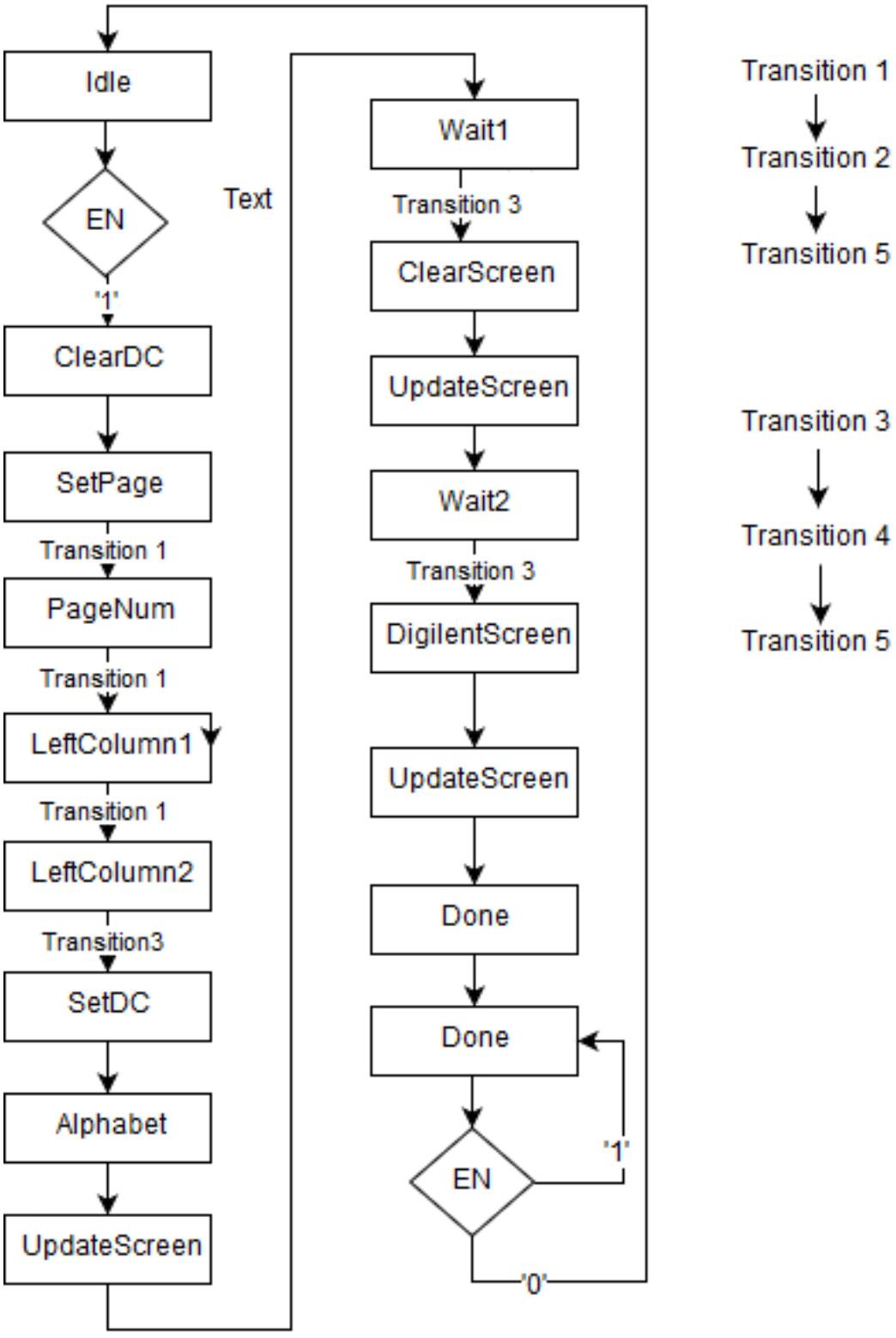
3.3.3 Blok OledExample

Vstupy do tohoto bloku jsou signály CLK (systémový hodinový signál o frekvenci 100MHz), EN (povolující tento blok) a RST (globální reset). Výstupy z tohoto bloku jsou signály CS (SPI výběr chipu), DC (data/příkaz), FIN (udávající, že tento blok dokončil svou činnost), SCLK (sériový hodinový signál) a SDO (SPI sériový datový výstup).



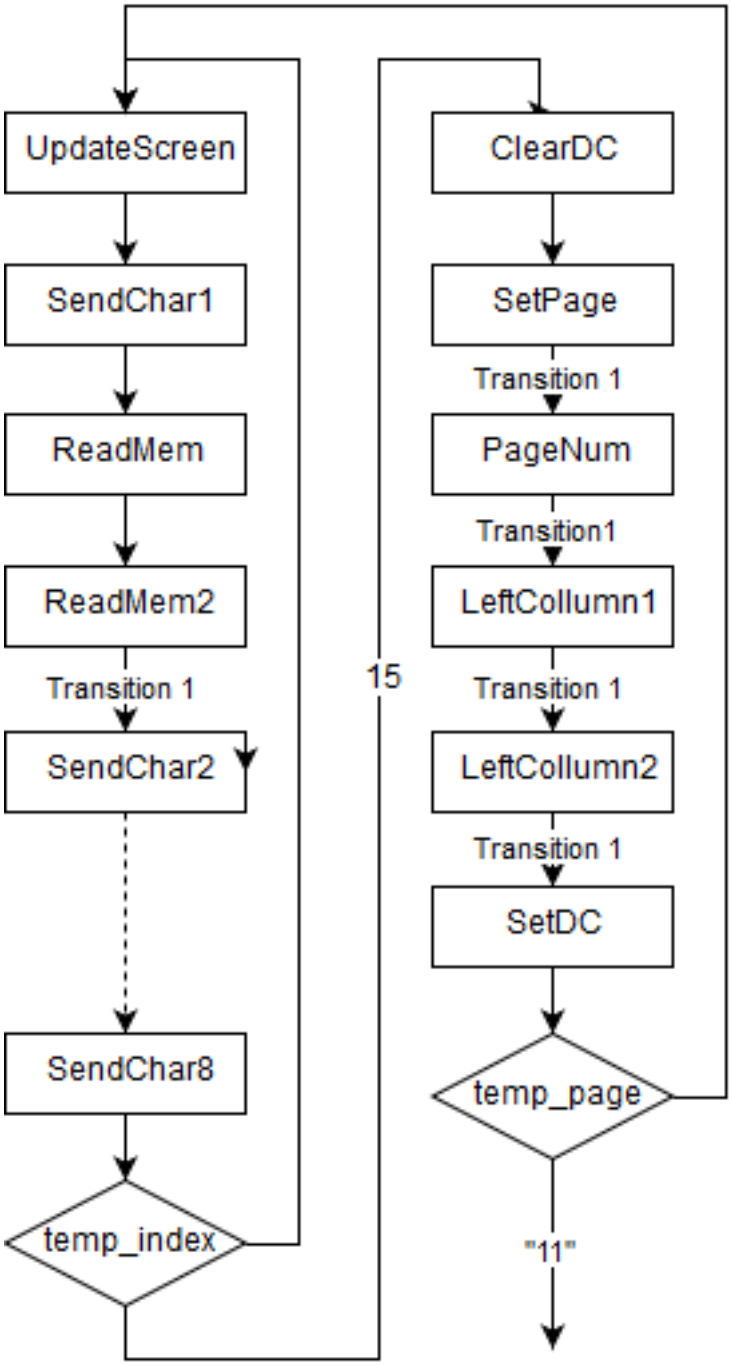
Obr. 3.13: Blokové schéma OledExample

Blok OledExample čeká do té doby, než je signál EN nastaven do logiké jedničky a poté přechází do stavu ClearDC. Ze stavu ClearDC se dostává do stavů SetPage, PageNum, LeftColumn1, LeftColumn2 a SetDC, který nastaví počet stran PmodOlED na 0 předtím než dojde k přechodu do stavu Alphabet. Signál current_screen je nastaven na abecedu a poté se přechází na stav UpdateScreen, který aktualizuje obrazovku PmodOLED na ASCII, které se nacázejí v current_screen. Poté čeká 4 sekundy vymaže obrazovku, čeká 1 sekundu a poté se na obrazovku vypíše „This is Digilent's PmodOLED".



Obr. 3.14: Vývojový diagram OledExample

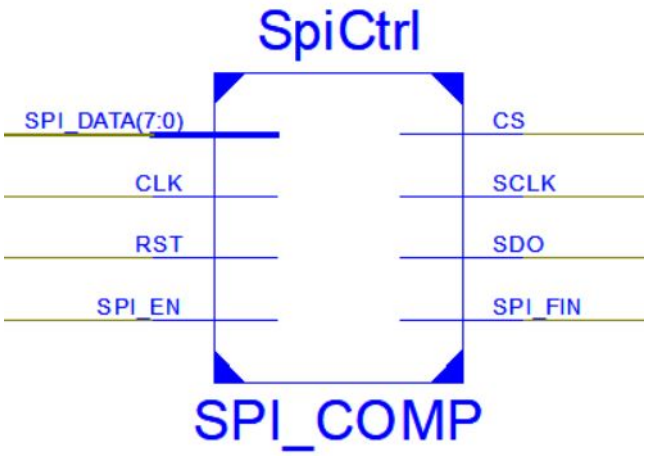
Průběh stavu UpdateScreen: Postupně je odesíláno 8 částí ASCII znaku, postupuje se na další znak a přehází se zpět do stavu Updatescreen do té doby než se dojde na konec řádku (resp. stránky, kdy temp_char = 15, jedná se o 16.znak). Poté se znovu prochází stavy ClearDc, SetPage, PageNum, LeftCollumn1. LeftCollumn2 a SetDC. Celý proces se opakuje do té doby než se dojde na konec displeje (kdy temp_page = „11“ a temp_char = 15, jedná se o 16.znak na 4.řádku, resp. stránce).



Obr. 3.15: Vývojový diagram pro UpdateScreen

3.3.4 Blok SpiCtrl

Vstupy do tohoto bloku jsou signály CLK (systémový hodinový signál o frekvenci 100MHz), RST (synchronní reset), SPI_EN (povolující tento blok) a SPI_DATA (obsahující byte, který se má odeslat). Výstupy z tohoto bloku jsou signály CS (výběr chipu), SDO (samotný výstup SPI rozhraní), SCLK (sériový hodinový signál o frekvenci 3,125MHz) a SPI_FIN (udávající, že blok dokončil svou činnost).



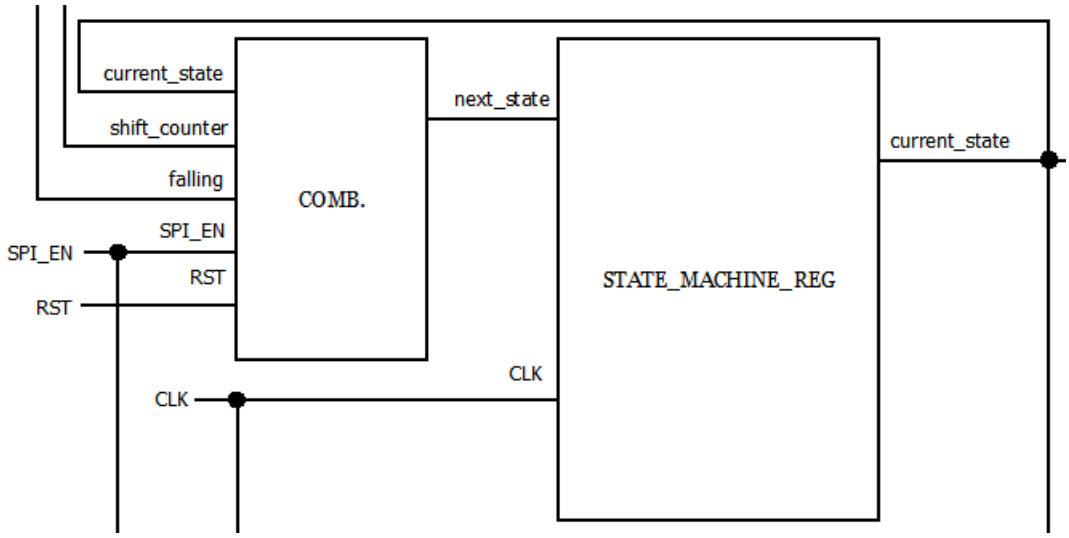
Obr. 3.16: Blokové schéma SpiCtrl

Jádrem tohoto bloku je stavový automat se stavy Idle, Send, Hold1, Hold2, Hold3, Hold4 a Done.

Blok SpiCtrl vytváří ze základního hodinového signálu o frekvenci 100MHz sériový hodinový signál (SCLK) o frekvenci 3,125MHz, který je využíván pro SPI komunikaci. Tento blok čeká dokud není zapsána do SPI_EN logická jednička a poté přechází do odesílacího stavu (Send). Kontrolér nastaví Chip Select (CS) do nuly a začne posouvat byte, který se právě nachází v SPI_DATA postupně po jednom bitu do Sériového datového výstupu (SDO) při každé klesající hraně sériového hodinového signálu (SCLK). Poté co je přeneseno všech osm bitů, kontrolér čeká několik hodinových cyklů, tak jak je předepsáno v datasheetu.

Kontrolér poté přechází do stavu hotovo (Done), ve kterém je signál SPI_FIN nastaven do logické jedničky. Kontrolér čeká do té doby než je signál SPI_EN nastaven do logické jedničky a ve stejném čase přechází do stavu Idle a signál SPI_FIN je nastaven do logické nuly.

3.3.4.1 Procesy stavového automatu



Obr. 3.17: Proces statového automatu

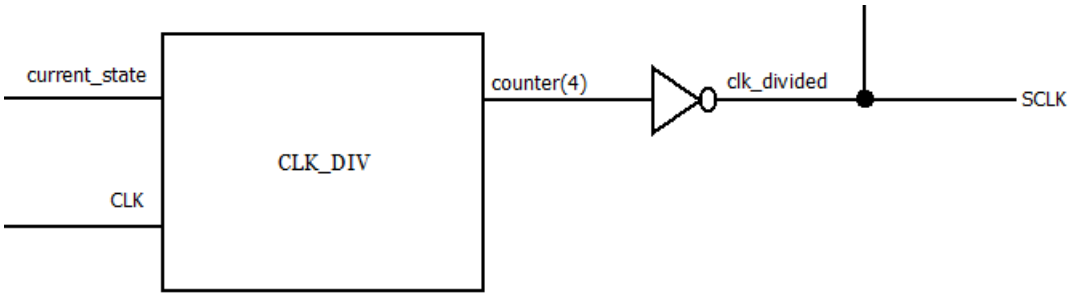
Tento proces popisuje stavový automat. Pokud je aktivován globální reset, stavový automat přechází do stavu Idle. Stav Idle je zároveň výchozí stav, z něj se přechází do stavu Send pokud je splněna podmínka aktivace SPI přenosu. Ze stavu Send se přechází do stavu Hold1 v momentě, kdy je odeslán celý byte a přišla sestupná hrana sériového hodinového signálu. Ze stavu Hold1 se postupně přechází do stavů Hold2, Hold3, Hold4 a Done. Ze stavu Done se přechází zpět do stavu Idle pokud je povolující signál bloku SpiCtrl znovu nastaven do logické nuly.

```
STATE_MACHINE_REG : process (CLK)
begin
  if(rising_edge(CLK)) then
    current_state <= next_state;
  end if;
end process;

STATE_MACHINE_COMB : process(RST, current_state, SPI_EN
   , shift_counter, falling)
begin
  if(RST = '1') then --Synchronous RST
    next_state <= Idle;
  else
    case (current_state) is
      when Idle => --Wait for SPI_EN to go high
```

```
        if(SPI_EN = '1') then
          next_state <= Send;
        else
          next_state <= Idle;
        end if;
      when Send => --Start sending bits, transition out
          when all bits are sent and SCLK is high
        if(shift_counter = "1000" and falling = '0')
           then
          next_state <= Hold1;
        else
          next_state <= Send;
        end if;
      when Hold1 => --Hold CS low for a bit
        next_state <= Hold2;
      when Hold2 => --Hold CS low for a bit
        next_state <= Hold3;
      when Hold3 => --Hold CS low for a bit
        next_state <= Hold4;
      when Hold4 => --Hold CS low for a bit
        next_state <= Done;
      when Done => --Finish SPI transimission wait for
         SPI_EN to go low
        if(SPI_EN = '0') then
          next_state <= Idle;
        else
          next_state <= Done;
        end if;
      when others =>
        next_state <= Idle;
    end case;
  end if;
end process;
```

3.3.4.2 Proces děličky kmitočtu



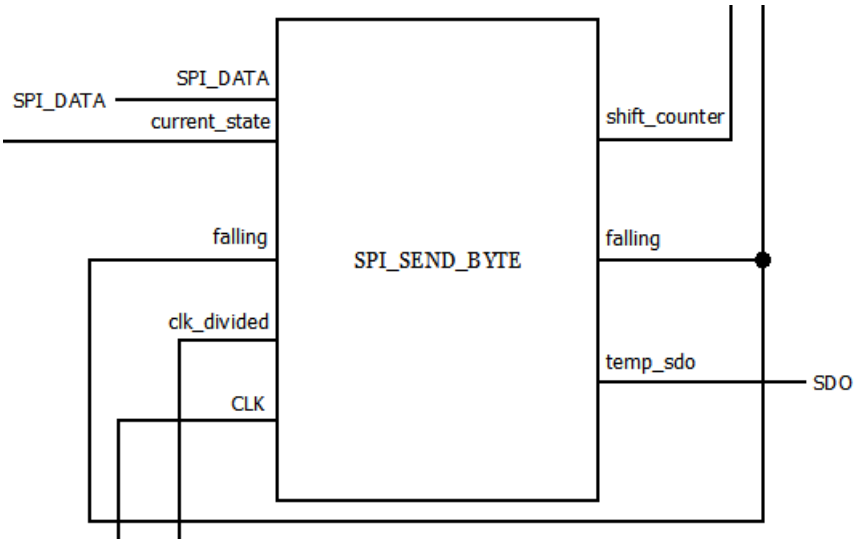
Obr. 3.18: Proces děličky kmitočtu

Tento proces slouží pro vydělení systémového hodinového kmitočtu (100MHz) hodnotou 32, čím se dostáváme na frekvenci 3,125 MHz. Je inkrementován čítač při každém hodinovém cyklu pokud se stavový automat nachází ve stavu Send. Negováním čtverého bitu tohoto čítače dostáváme signál clk_divided, který je vyžit v dalším procesu.

```
clk_divided <= not counter(4); --SCLK = CLK / 32
SCLK <= clk_divided;

CLK_DIV : process (CLK)
begin
  if(rising_edge(CLK)) then
    if (current_state = Send) then --start clock
       counter when in send state
      counter <= counter + 1;
    else --reset clock counter when not in send state
      counter <= (others => '0');
    end if;
  end if;
end process;
```

3.3.4.3 Proces SPI odeslání bytu



Obr. 3.19: Proces SPI odeslání bytu

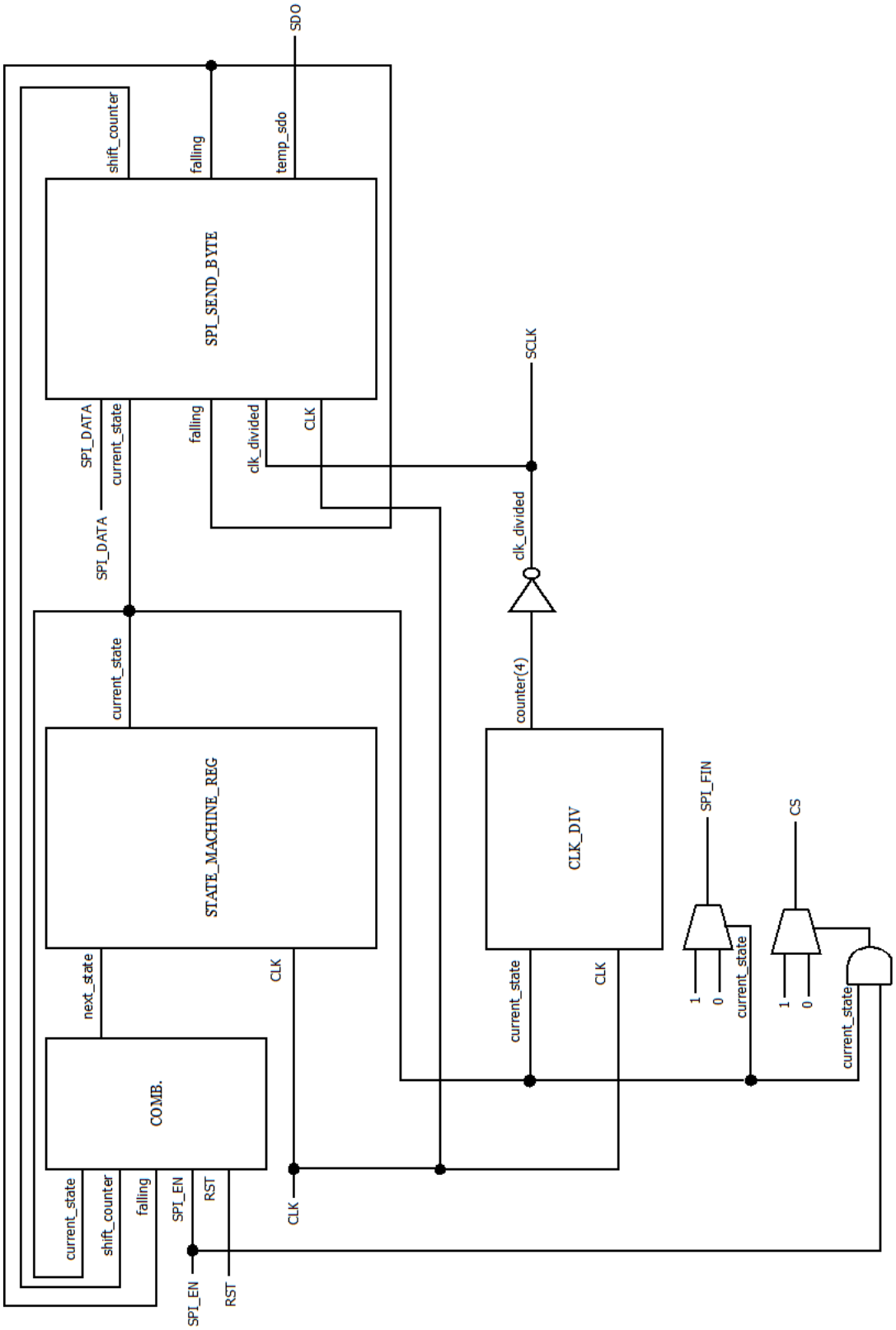
Tento proces odesílá byte přes SPI rozhraní, pokud je stavový automat ve stavu Idle, probíhá vkládání aktuálních paralelních dat do shift registru. Když je stavový automat ve stavu Send při každé klesající hraně sériového hodinového signálu dochází k posunutí po jednom bitu ze shift registru do sériového datového výstupu (SDO). To se děje dokud není odesláno všech 8 bitů.

```
SDO <= temp_sdo;

SPI_SEND_BYTE : process (CLK) --sends SPI data
   formatted SCLK active low with SDO changing on the
   falling edge
begin
  if(CLK'event and CLK = '1') then
    if(current_state = Idle) then
      shift_counter <= (others => '0');
      shift_register <= SPI_DATA; --keeps placing
         SPI_DATA into shift_register so that when
         state goes to send it has the latest SPI_DATA
      temp_sdo <= '1';
    elsif(current_state = Send) then
      if( clk_divided = '0' and falling = '0') then --
         if on the falling edge of Clk_divided
```

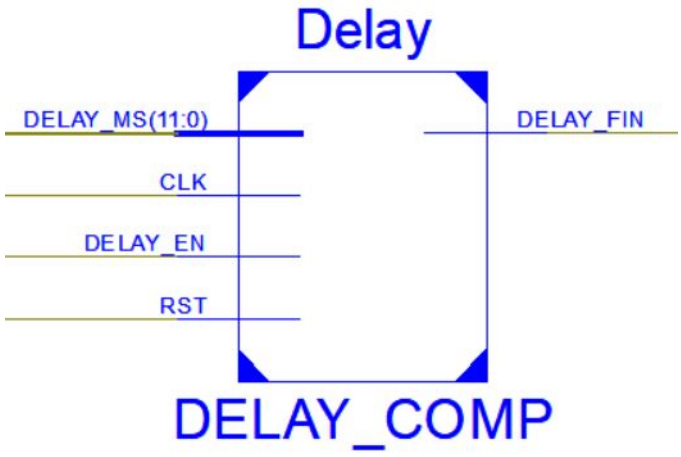
```
            falling <= '1'; --Indicate that it is passed
               the falling edge
            temp_sdo <= shift_register(7); --send out the
               MSB
            shift_register <= shift_register(6 downto 0) &
               '0'; --Shift through SPI_DATA
            shift_counter <= shift_counter + 1; --Keep
               track of what bit it is on
          elsif(clk_divided = '1') then --on SCLK high
             reset the falling flag
            falling <= '0';
          end if;
        end if;
      end if;
    end process;
```

3.3.4.4 Blokový diagram



3.3.5 Blok Delay

Vstupy do tohoto bloku jsou signály DELAY_MS (zpoždění o daném počtu milisekund), CLK (systémový hodinový signál o frekvenci 100MHz), DELAY_EN (povolující tento blok) a RST (globální reset). Výstupem z tohoto bloku je signál DELAY_FIN (udávající, že blok dokončil svou činnost).



Obr. 3.20: Blokové schéma Delay

Blok Delay využívá systémový hodinový signál ke generování 1kHz čítače, který slouží k počítaní uběhlých milisekund. Tento blok čeká do té doby, než je do signálu DELAY_EN zapsána logická jednička a poté přechází do stavu čekání (Hold). 1kHz čítač čítá dokud jeho hodnota není shodná s vektorem DELAY_MS. Kontrolér poté přechází do stavu hotovo (Done) a signál DELAY_FIN je nastaven do logické jedničky. Kontrolér čeká, až je signál SPI_EN znovu nastaven do logické jedničky a poté přechází do nečinného stavu (Idle), signál SPI_FIN je nastaven do logické nuly.

3.3.6 Paměťový blok charLib

Tento paměťový blok obsahuje bytové mapy pro jednotlivé znaky. Znaky jsou obsaženy v 8 bytových částech. 11 Bitové adresování je následující: ASCII hodnota & XXX, kde poslední 3 bity představují 8 částí každého znaku. Např. adresa "01000001001"představuje druhou část znaku 'A'. Bytová mapa se nachází v souboru charlib.coe, tento soubor má 3 řádkový offset, takže např. znak 'A' (ASCII hodnota 65 v desítkové soustavě) se nachází na řádku 68. Každý byte představuje jeden sloupec v 8x8 pixelovém rastru. Nejvýznamnější bit (MSB) se nachází úplně dole sloupce. Jednotlivé znaky jsou odděleny mezerou dole a vpravo o šířce jeden pixel (samé nuly v bitové mapě).

```
64    00,14,14,14,14,14,14,00,
65    00,00,22,14,14,08,00,00,
66    00,02,01,59,05,02,00,00,
67    3e,41,5d,55,4d,51,2e,00,
68    40,7c,4a,09,4a,7c,40,00,
69    41,7f,49,49,49,49,36,00,
70    1c,22,41,41,41,41,22,00,
71    41,7f,41,41,41,22,1c,00,
```

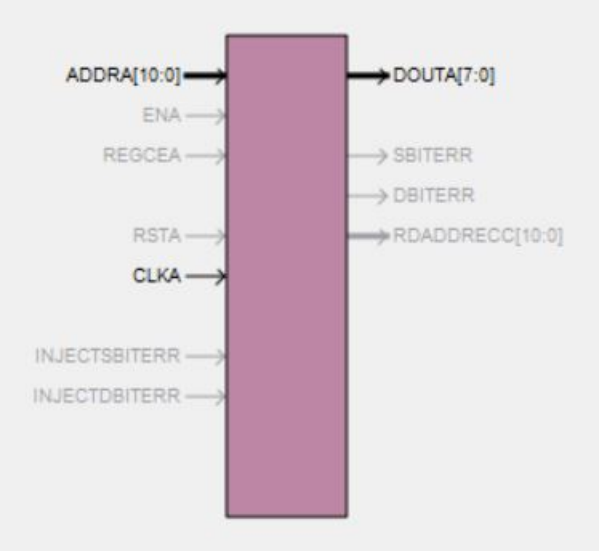
Obr. 3.21: Soubor charlib.coe

Znak 'A' v bitové mapě vypadá následovně (1 znamená rozsvícený pixel):

```
00010000
00101000
01000100
01111100
01000100
01000100
11101110
00000000
```

Obr. 3.22: Bitová mapa znaku 'A'

Tento blok byl vytvořen pomocí Xilinx Block Memory Generátoru. Jeho schéma vypadá následovně.



Obr. 3.23: Schéma paměťového bloku

Ve VHDL popisu je zapsán takto.

```
COMPONENT charLib
  PORT (
    clka : IN STD_LOGIC; --Attach System Clock to it
    addra : IN STD_LOGIC_VECTOR(10 DOWNTO 0); --First 8
       bits is the ASCII value of the character the last
       3 bits are the parts of the char
    douta : OUT STD_LOGIC_VECTOR(7 DOWNTO 0) --Data byte
       out
  );
END COMPONENT;
```

Vstupem je tedy 11 bitová adresa a systémový hodinový signál, výstupem je příslušný byte (odpovídající sloupci v bitové mapě) pro adresovanou část znaku.

ASCII hodnoty pro znaky, které se nacházejí na maticové klávesnici jsou v tabulce 3.2.

Tab. 3.2: ASCII hodnoty

znak	**Binární**	**Hexadecimální**	**Decimální**
0	00110000	30	48
1	00110001	31	49
2	00110010	32	50
3	00110011	33	51
4	00110100	34	52
5	00110101	35	53
6	00110110	36	54
7	00110111	37	55
8	00111000	38	56
9	00111001	39	57
A	01000001	41	65
B	01000010	42	66
C	01000011	43	67
D	01000100	44	68
E	01000101	45	69
F	01000110	46	70

3.4 Porovnání obou kontrolérů

Největší rozdíl spočívá v tom, že první kontrolér využívá komponentu pro jádro mikrokontroléru, která zahrnuje inicializaci, zatímco druhý kontrolér pracuje na bázi stavového automatu a je rozdělen na inicializační a aplikační část. První kontrolér má větší počet řádků kódu než kontrolér druhý. Co se týká hardwarových prostředků, první kontrolér spotřebovává 165 sliců, zatímco druhý kontrolér 97. Rozdílný je i přístup k paměti, první kontrolér využívá samostatné komponenty pro přístup a adresování paměti, druhý kontrolér obsahuje paměťový blok, ve kterém jsou zamapovány ASCII znaky.

Oba kontroléry využívají SPI rozhraní pro komunikaci s displejem. Liší se tím, kdy je čtena hodnota, u prvního kontroléru to je při náběžné hraně sériového hodinového signálu, u druhého kontroléru při sestupné. Kód SPI komponenty druhého kontroléru je o poznání lépe čitelný a pochopitelný než je tomu u prvního kontroléru.

3.5 Řadič displeje

Spolu se samotným OLED displejem se na použitém Pmodu nachází i řadič displeje SSD1306 od firmy Solomon Systech. Originálně slouží pro displej o rozlišení 128x64 pixelů, v našem případě je tedy spodní polovina kopií vrchní části, aby nedocházelo k problémům. Mezi jeho další vlastnosti patří nastavitelný kontrast a svítivost, vestavěný oscilátor a možnost vertikálního a horizontální scrolování.

Pro komunikaci mezi mikroprocesorem a řadičem lze využít 6800/8000 kompatibilní paralelní rozhraní, tří nebo čtyř drátové SPI rozhraní nebo rozhraní I^2C.

3.5.1 Popis pinů

D/C# je vstupní pin pro data/příkaz, pokud je nastaven do logické jedničky, obsah D[7:0] je považován za data. Pokud je nastaven do logické nuly, obsah D[7:0] je převeden do příkazového registru.

R/W# je vstupní pin, který slouží pro určení, zda se jedná o čtení (logická jednička) nebo o zápis (logická nula).

D[7:0] je obousměrná datová sběrnice. Pokud je zvolený SPI přenos a jedná se o zápis dat, D0 slouží jako sériový hodinový signál (SCLK), D1 jako sériový vstup pro data (SDIN) a D2 by neměl být připojen. [7]

3.5.2 Použité příkazy

Pokud chceme posílat příkazy pro řadič displeje musíme zapsat D/C# = 0, R/W# = 0 a D[7:0] podle nastudující tabulky.

Tab. 3.3: Použité příkazy

Hex	D7	D6	D5	D4	D3	D2	D1	D0	Příkaz
AE	1	0	1	0	1	1	1	0	Vypnout displej
AF	1	0	1	0	1	1	1	1	Zapnout displej
8D	1	0	0	0	1	1	0	1	Nabití pumpy
14	0	0	0	1	0	0	1	1	Nabití pumpy
D9	1	1	0	1	1	0	0	1	Přednabití
F1	1	1	1	1	0	0	0	1	Přednabití 2
81	1	0	0	0	0	0	0	1	Nastavení kontrastu
0F	0	0	0	0	1	1	1	1	Nastavení kontrastu 2
A1	1	0	1	0	0	0	0	1	Nastavit segment re-map
C8	1	1	0	0	1	0	0	0	Invertovaný displej
DA	1	1	0	1	1	0	1	0	Nastavit COM konfiguraci
20	0	0	1	0	0	0	0	0	Nastavit mód adresování paměti
A5	1	0	1	0	0	1	0	1	Zapnout celý displej
00	0	0	0	0	0	0	0	0	Nastavení levého sloupce
10	0	0	0	1	0	0	0	0	Nastavení levého sloupce
22	0	0	1	0	0	0	1	0	Nastavení adresy stránky

4 Simulace

Simulace byla provedena pouze pro úlohu s maticovou klávesnicí, pro úlohy s OLED displejem kvůli jejich složitosti provedena nebyla.

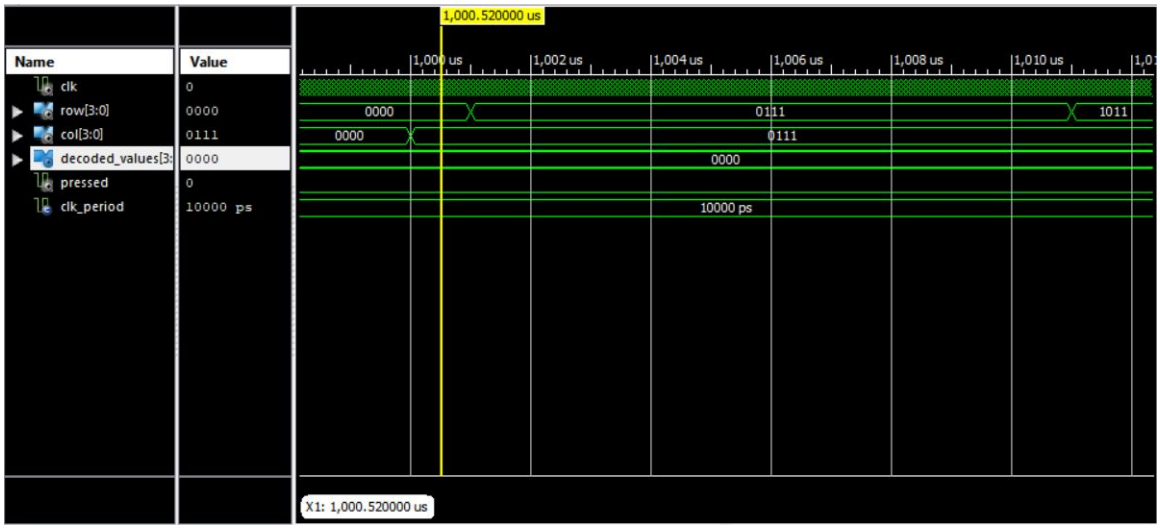
4.1 Maticová klávesnice

Pro ověření správné funkčnosti úlohy byla provedena simulace. Jako simulátor byl použit Xilinx ISim.

Nejprve byl simulován dekodér pro ověření čtení stisknutého tlačítka na maticové klávesnici. Celkový čas simulace byl nastaven na 8ms, protože dekodér čte jednotlivé klávesy s periodou 1ms. V čase 1ms byl signál ROW ve stimulu nastaven na hodnotu "0111"(což představuje stisknuté tlačítko v první řadě), signál COL byl nastaven na tutéž hodnotu (stisknuté tlačítko v prvním sloupci) automaticky. Vidíme, že signály PRESSED a DECODED_VALUES změnily svou hodnotu s velmi malým zpožděním (konkrétně 0.09us), což je očekávané chování. Hodnota signálu DECODED_VALUES je "0001", což odpovídá znaku '1'. Tento znak se opravdu nachází v prvním sloupci na prvním řádku, takže můžeme tuto detekci prohlásit za správnou.

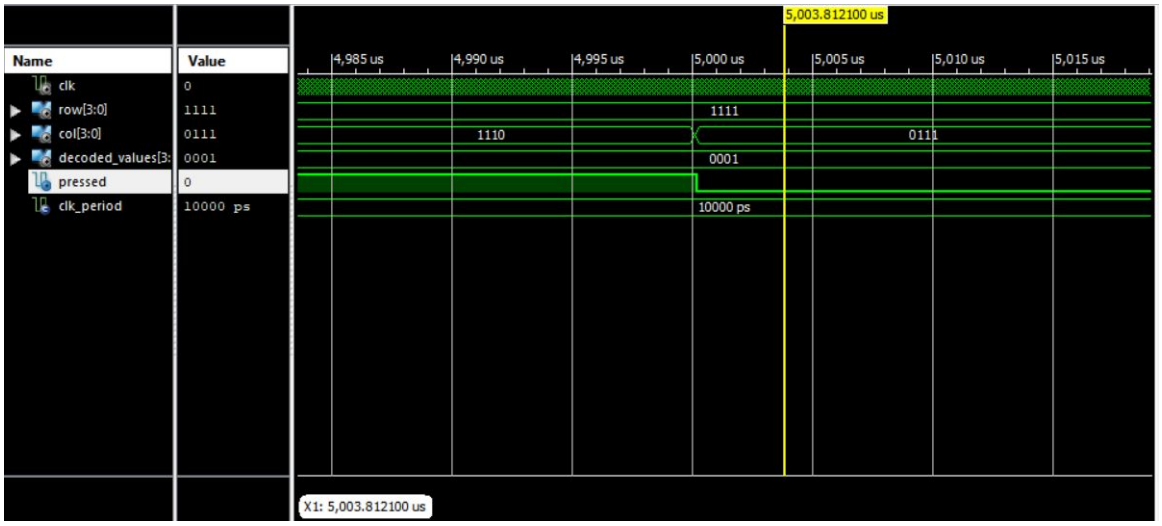
Obr. 4.1: Simulace dekodéru

V teoretickém případě kdy je klávesa stisknuta (změní se hodnota signálu ROW) v čase mezi 1ms a 2ms nedojde k přečtení znaku. Což je také očekávané chování. Reálně člověk nedokáže tak rychle stisknout tlačítko.



Obr. 4.2: Simulace dekodéru 2

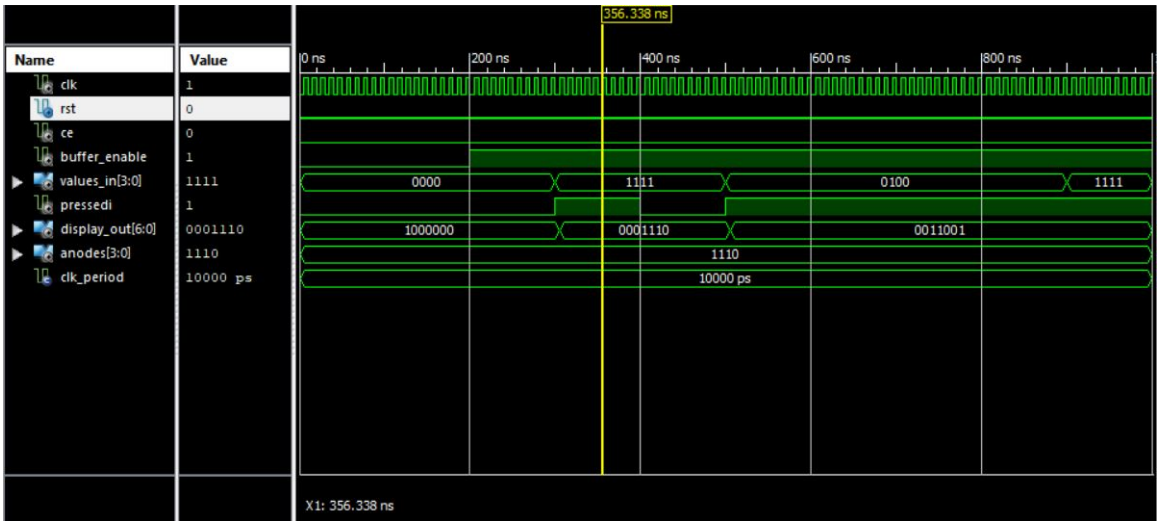
Ve druhém případě v čase 5ms se mění hodnota signálu PRESSED na logickou nulu, jelikož za celou dobu čtení nebylo stisknuto žádné tlačítko (signál ROW má hodnotu "1111").



Obr. 4.3: Simulace dekodéru 3

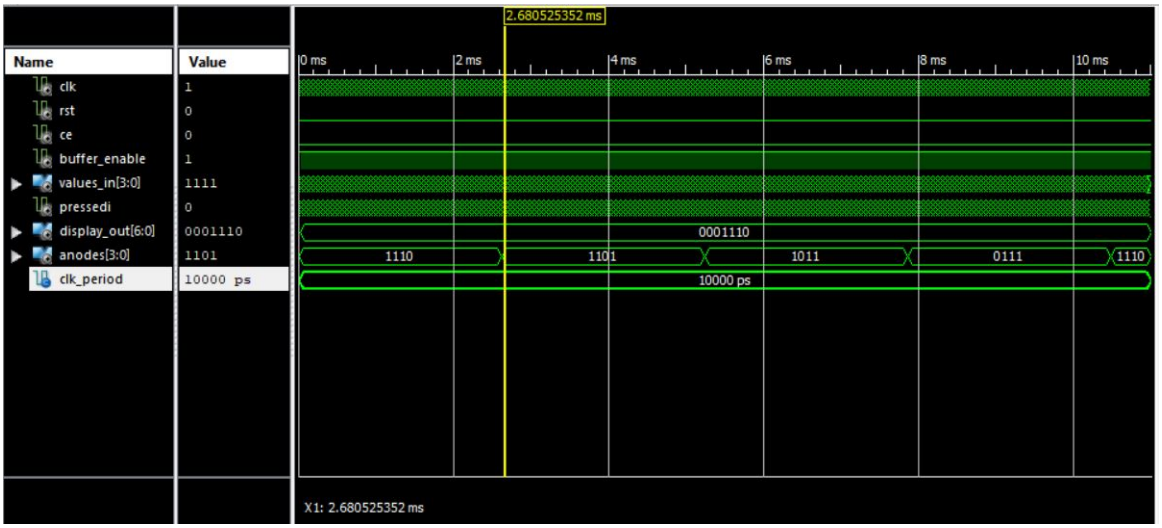
Dále byl simulován DisplayController pro ověření zobrazování znaků a přepínání jednotlivých anod na sedmisegmentovém displeji. V čase 200ns byl signál BUFFER_ENABLE nastaven do logické jedničky. Vidíme, že DISPLAY_OUT se mění pouze při změně signálu VALUES_IN a změně z logické nuly na logickou jedničku

signálu PRESSEDi. Hodnota signálu DISPLAY_OUT v čase 400ns opravdu odpovídá znaku 'F' (hodnota "1111"). Což je chování, které od této komponenty očekáváme.



Obr. 4.4: Simulace DisplayControlleru

Pro ověření přepínání anod sedmisegmentového displeje byl čas simulace nastaven na 11ms. Obnovovací perioda pro všechny anody je 10,5ms, pro jednu anodu tedy připadá zhruba 2,625ms. Vidíme, že opravdu v čase 2.68ms dochází ke změně hodnoty signálu ANODES.



Obr. 4.5: Simulace DisplayControlleru 2

5 Závěr

Výsledkem diplomové práce je návrh a odladění tří úloh v jazyku VHDL pro vývojový kit Digilent Nexys 3 a k němu připojených periferií Pmod.

První úloha je zaměřená na práci s připojenou maticovou klávesnicí, konkrétně zobrazování stisknutých kláves na vnitřní sedmisegmentový displej. Druhá úloha využívá část první úlohy, kdy je požadováno stisknutou klávesu zobrazit na připojený OLED displej, je použit OLED kontrolér na bázi jádra mikrokontroléru. Třetí úloha je prakticky stejná jako druhá úloha s tím rozdílem, že je využit OLED kontrolér na bázi stavového automatu a také je zde požadavek na grafické zobrazení na displeji. Každá z úloh obsahuje řadu variací.

Teoretická část semestrální práce se věnuje popisu možností vývojového kitu a přídavných periferií a mimo jiné také stručnému popisu jazyka VHDL a vývojového prostředí, které bylo použito pro řešení navržených úloh.

Kapitola Maticová klávesnice popisuje vlastnosti této klávesnici a také uvádí doporučený postup řešení úlohy č.1, která je na ni zaměřena.

V kapitole OLED displej jsou popsány vlastnosti toho displeje a také dva kontroléry, které jsou rozdílné, ale jsou využity pro ovládání stejné periferie. Jsou zde popsány jednotlivé části těchto kontrolérů, důraz je kladen na realizaci SPI rozhraní, tyto kontroléry jsou pak také porovnány.

Samostatná kapitola se věnuje simulaci navržených úloh pro ověření jejich správného fungování. Zde z důvodu složitosti OLED kontrolérů byla provedena simulace pouze u první úlohy s maticovou klávesnicí.

Návody k jednotlivým úlohám se nacházejí v příloze.

Literatura

[1] Digilent: *Nexys3^{TM} Board Reference Manual*. April 2013.
URL `https://www.xilinx.com/support/documentation/university/XUP%20Boards/XUPNexys3/documentation/Nexys3_rm.pdf`

[2] Digilent: *Digilent Pmod^{TM} Interface Specification*. November 2011.
URL `https://www.digilentinc.com/Pmods/Digilent-Pmod_%20Interface_Specification.pdf`

[3] PINKER, J.; POUPA, M.: *Číslicové zpracování signálů*. Praha: BEN - technická literatura, 2006, ISBN 80-7300-198-5.

[4] KRÁL, J.: *Řešené příklady ve VHDL : hradlová pole FPGA pro začátečníky*. Praha: BEN - technická literatura, 2010, ISBN 978-80-7300-257-2, 127 s.

[5] Digilent: *PmodKYPD^{TM} Reference Manual*. April 2016.
URL `https://reference.digilentinc.com/_media/pmod:pmod:pmodKYPD_rm.pdf`

[6] Digilent: *PmodOLED^{TM} Reference Manual*. May 2016.
URL `https://reference.digilentinc.com/_media/reference/pmod/pmodoled/pmodoled_rm.pdf`

[7] Solomon Systech: *SSD1306 Datasheet*. April 2008.
URL `https://cdn-shop.adafruit.com/datasheets/SSD1306.pdf`

Seznam symbolů, veličin a zkratek

BLOS	Logické obvody a systémy
FPGA	Programovatelné hradlové pole – Field programmable gate array
Pmod	Periferení moduly – Peripheral module
SPI	Serial Peripheral interface
GPIO	Obecné vstupy a výstupy – General purpose input/output
OLED	Organic light-emitting diode
VHDL	VHSIC hardware desription language

Seznam příloh

A Obsah přiloženého CD

Všechny projektové soubory byly testovány v softwaru Xilinx ISE ve verzi 14.3.

- `/` ..... kořenový adresář přiloženého CD
  - `dokumentace` ..... použité dokumenty a datasheety
    - `Digilent-Pmod_Interface_Specification.pdf`
    - `Nexys3_OLED_Interface_RefProj.pdf`
    - `PmodKYPD_Demo_Nexys3.pdf`
    - `pmodkypd_rm.pdf`
    - `pmodoled_rm.pdf`
    - `SSD1306.pdf`
  - `uloha1` ..... zadání a řešení úlohy č.1
    - `reseni`
      - `PmodKYPD`
    - `soubory`
      - `PmodKYPD`
    - `uloha1B.pdf`
  - `uloha2` ..... zadání a řešení úlohy č.2
    - `reseni`
      - `OLED_ctrl_V1_0`
      - `OLED_ctrl_V1_0_shifting`
    - `soubory`
      - `OLED_ctrl_V1_0`
    - `OLED_SPI.pdf` ..... popis komponenty pro SPI přenos
    - `uloha2B.pdf`
  - `uloha3` ..... zadání a řešení úlohy č.3
    - `reseni`
      - `PmodOLED`
      - `PmodOLED_graphic_v2`
    - `soubory`
      - `PmodOLED`
    - `uloha3B.pdf`
    - `SpiCtrl.pdf` ..... popis bloku SpiCtrl
  - `blokove_diagramy.pdf` ..... blokové diagramy pro OLED kontrolér č.1

B Úloha č.1 - čtení maticové klávesnice

B.1 Cíle

- Připojit maticovou klávesnici pomocí Pmod konektoru (dbát na správné připojení napájecích pinů!)
- Vytvořit algoritmus v jazyce VHDL pro čtení stisknuté klávesy
- Zobrazit stisknutou klávesu na sedmisegmentovém displeji
- Zobrazit historii stisknutých kláves (nejnovější znak bude na pozici úplně vpravo)
- Opakovaně zobrazit znak s volitelnou periodou při držení klávesy
- Implementovat návrh do cílového FPGA obvodu Spartan6 na vývojové desce

B.2 Teoretický úvod

Pmod^{TM} je technologický standart periferních modulů od firmy Digilent. Tyto přídavné moduly umožňují rozšířit možnosti vývojové desky o další programovatelné komponenty. Moduly jsou připojovány pomocí 6 nebo 12 pinnového konektoru, buďto přímo nebo pomocí kabelu, a tyto piny přenášejí jak digitální signály tak napájecí napětí. Nejčastěji využívána rozhraní jsou SPI a I^2C, ale některé moduly používají tyto signály pro běžné vstupy a výstupy (GPIO).

PmodKYPD je maticová klávesnice s 16 tlačítky s čísly '0'-'9' a znaky 'A'-'F' uspořádaných do 4 řad. Klávesnice je zobrazena na následujícím obrázku:

Obr. B.1: Maticová klávesnice

PmodKYPD využívá 12 pinový konektor, rozložení pinů je uvedeno v tabulce B.1.

Při řešení této úlohy je vhodné rozdělit si práci do několika kroků, v návrhu VHDL se toto promítne jako vytvoření několika komponent, které lze podle potřeby propojit. Nejprve je třeba navrhnout komponentu, která bude číst stisknuté tlačítko

Tab. B.1: Popis jednotlivých PINů

PIN	**signál**	**popis**
1	COL4	sloupec 4
2	COL3	sloupce 3
3	COL2	sloupec 2
4	COL1	sloupec 1
5	GND	zem napájecího napětí
6	VCC	napájecí napětí (3.3V/5V)
7	ROW4	řadek 4
8	ROW3	řadek 3
9	ROW2	řadek 2
10	ROW1	řadek 1
11	GND	zem napájecího napětí
12	VCC	napájecí napětí (3.3V/5V)

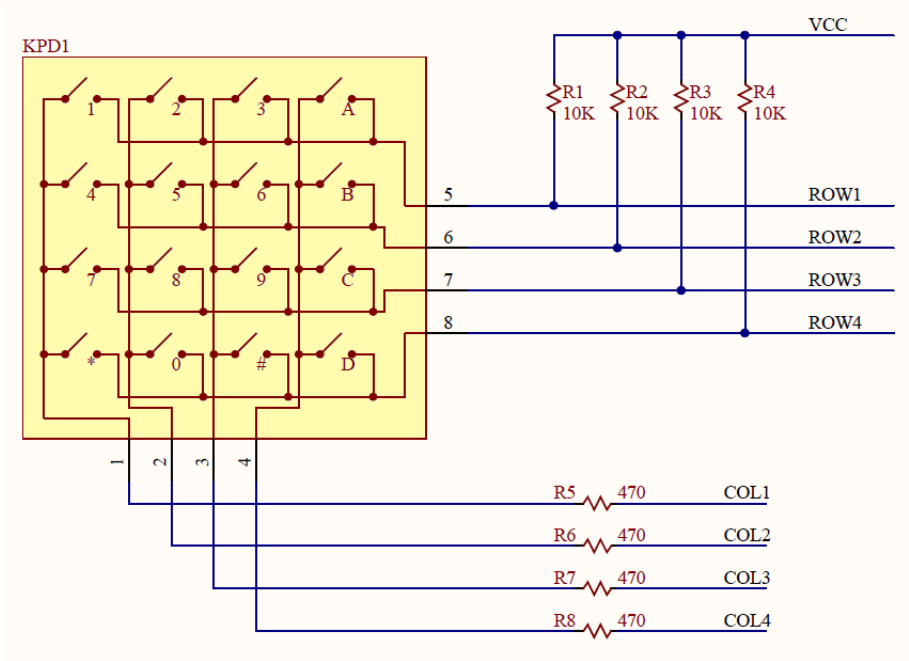
na maticové klávesnici a také jeho hodnotu vhodně kódovat pro pozdější použití, nazvána může být např. Decoder. Klávesnice obsahuje 16 tlačítek, takže bude stačit 4 bitový signál pro dekódování. Kromě samotné hodnoty stisknutého tlačítka je vhodné nějakým způsobem předávat příznak, že bylo či nebylo stisknuto tlačítko a to buď přidáním pátého bitu nebo samostatné proměnné.

Čtení stisknutého tlačítka na klávesnici je založeno na principu postupující nuly. Postupně je zapisována do jednotlivých sloupců logická nula. Na signály jsou připojeny pull-up rezistory, a proto v klidovém stavu (není stisknuto tlačítko) je na výstupu těchto signálů logická jednička. Při stisknutí tlačítka dojde k uzemnění signálu a výstupní signál příslušného řádku se změní na logickou nulu. Znalostí, do kterého sloupce byla zapsána logická nula a ve kterém řádku se objevila logická nula určíme, které tlačítko bylo stisknuto.

Pull-up rezistor se používá na udržení napětí na vstupu na úrovni napájecího napětí (což odpovídá logické 1). Řádky a sloupce jsou izolované, je možno detekovat stisknutí více tlačítek najednou, výjimku tvoří tlačítka nacházející se ve stejné řadě.

Při čtení stisknutého tlačítko je třeba dodržet zpoždění mezi zapsáním hodnoty do jednotlivých sloupců, taková prodleva by měla být zhruba 1ms. Řádově kratší prodleva se uplatňuje i mezi zapsáním hodnoty do sloupce a vyčtením hodnoty z řádku.

Při vytváření ucf souboru je třeba deklarovat příslušný konektor JX, ke kterému je připojena maticová klávesnice. Dále v programu je třeba definovat horní část portu



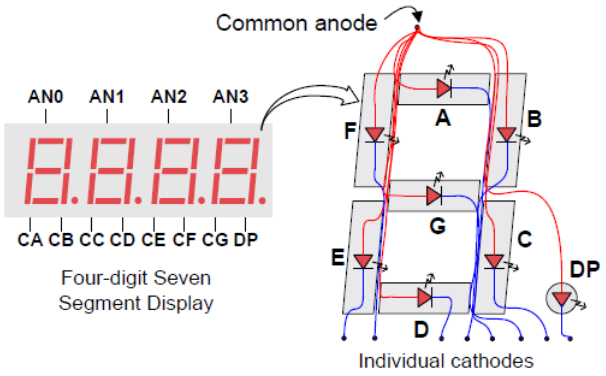
Obr. B.2: Čtení stisknutého tlačítka z maticové klávesnice

JX jako výstupní a jeho dolní část jako vstupní. To je patrné z tabulky jednotlivých pinů a z toho, že je potřeba zapisovat hodnoty do sloupců a číst je z řádků.

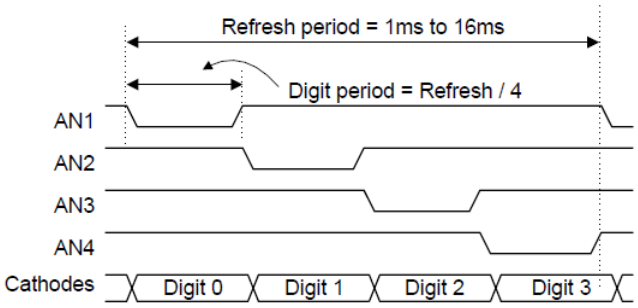
Dalším krokem je vytvoření komponenty pro zobrazení dekódovaného znaku na sedmisegmentovém displeji. Tato periferie je dobře známa ze standartních úloh předmětu BLOS. Tento displej je v zapojení se společnou anodou. Katody jsou spojeny napříč celým displejem tak, že jsou navzájem propojeny odpovídající segmenty všech pozic. Výstupem z displeje je sedm segmentů , desetinná tečka a počet anod, který odpovídá počtu pozic, v našem případě tedy čtyři. Pokud chceme zobrazit pouze jeden znak na zvolené pozici stačí aktivovat příslušnou anodu a do katod zapsat hodnoty, které odpovídají jednotlivým znakům.

Při zobrazování všech čtyř znaků najednou musíme cyklicky přepínat jednotlivé anody s takovou frekvencí, aby pro lidské oko působily zapnuté všechny najednou. To znamená, že tato frekvence nesmí být příliš vysoká, protože by se anody nestíhaly přepnout a displej by se tak stal prakticky nečitelným. Na druhou stranu nesmí být ani příliš nízká, to by potom byly vidět jednotlivé znaky samostatně. Z toho vyplývá, že by tato frekvence měla být v rozmezí od 60Hz do 1KHz, čemuž odpovídá perioda 1ms - 16ms.

Dále pro zobrazení historie stisknutých kláves (nejpozději stisknutá klávesa se zobrazí na pozici úplně vpravo na sedmisegmentovém displeji, předchozí znaky se posunou směrem doleva) je třeba vytvořit 16 bitový posuvný registr, do kterého se budou načítat nově příchozí data zprava. Pro tento úkon můžeme využít operátoru



Obr. B.3: Zapojení sedmisegmentového displeje



Obr. B.4: Přepínání sedmisegmentového displeje

sjednocení, jak je ukázáno na následujícím příkladu:

```
signal shift_registr : STD_LOGIC_VECTOR (15 downto 0);

shift_registr <= shift_registr (11 downto 0) & data;
```

Aby se do posuvného registru načítala data pouze při stisknutém tlačítku klávesnice, je třeba vhodně reagovat na hranu signálu (příznaku), který nese hodnotu 0 nebo 1 (bylo nebo nebylo stisknuto tlačítko).

B.3 Vypracování laboratorní úlohy

B.3.1 Úkol č.1

Do tabulky doplňte, které tlačítko bude stisknuto.

Tab. B.2: Hodnoty pro stisknuté tlačítko

COL1	COL2	COL3	COL4	ROW1	ROW2	ROW3	ROW4	tlačítko
1	1	1	1	1	1	1	1	žádné
0	1	1	1	0	1	1	1	’1’
0	1	1	1	1	0	1	1	
0	1	1	1	1	1	0	1	
0	1	1	1	1	1	1	0	
1	0	1	1	0	1	1	1	
1	0	1	1	1	0	1	1	
1	0	1	1	1	1	0	1	
1	0	1	1	1	1	1	0	’F’
1	1	0	1	0	1	1	1	
1	1	0	1	1	0	1	1	
1	1	0	1	1	1	0	1	
1	1	0	1	1	1	1	0	
1	1	1	0	0	1	1	1	
1	1	1	0	1	0	1	1	
1	1	1	0	1	1	0	1	
1	1	1	0	1	1	1	0	’D’

B.3.2 Úkol č.2

Vytvořte projekt uloha1b. Doplňte prázdná místa ve VHDL popisu. Vytvořte program ve VHDL na PC tak, aby byla přečtena stisknutá klávesa na maticové klávesnice. Přečtenou klávesu zobrazte na poslední (pravé) anodě sedmisegmentového displeje. Implementujte výsledný návrh do paměti PROM. Výslednou funkci ověřte pomocí vývojové desky.

```
process(clk)
begin
if clk'event and clk = '1' then
  --zpozdeni 1ms
  if sclk = "00011000011010100000" then
    --zapsani do sloupce 1
    Col <= "0111";
    sclk <= sclk+1;
  --kontrola, ve kterem radku je zapsana 0
```

```
elsif sclk = "00011000011010101000" then
  -- radek1
  if Row = "0111" then
    DecodeOut <= "0001"; -- klavesa1
  -- radek2
  elsif Row = "1011" then
    DecodeOut <= "    "; -- klavesa4
  -- radek3
  elsif Row = "    " then
    DecodeOut <= "    "; -- klavesa7
  -- radek4
  elsif Row = "    " then
    DecodeOut <= "0000"; -- klavesa0
  end if;
  sclk <= sclk+1;
```

B.3.3 Úkol č.3

Na 7-segmentovém displeji vytvořte historii stisknutých kláves, nejdříve stisknutá klávesa se zobrazí na displeji vpravo, po stisknutí další klávesy se původní znak posune doleva a stávající se zobrazí na jeho místě atd.

B.3.4 Bonusový úkol

Na 7-segmentovém displeji opakovaně zobrazujte znak s volitelnou periodou pokud je příslušná klávesa držena.

C Úloha č.2 - komunikace s OLED displejem 1

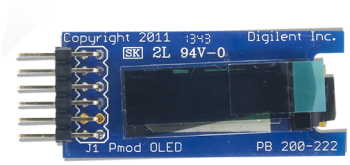
C.1 Cíle

- Připojit OLED displej a maticovou klávesnici pomocí Pmod konektoru (dbát na správné připojení napájecích pinů!)
- Zprovoznit program pro výpis textu na tento displej
- Zobrazit stisknutou klávesu na klávesnici na zvolené pozici na displeji
- Implementovat návrh do cílového FPGA obvodu Spartan6 na vývojové desce

C.2 Teoretický úvod

Pmod^{TM} je technologický standart periferních modulů od firmy Digilent. Tyto přídavné moduly umožňují rozšířit možnosti vývojové desky o další programovatelné komponenty. Moduly jsou připojovány pomocí 6 nebo 12 pinnového konektoru, buďto přímo nebo pomocí kabelu, a tyto piny přenášejí jak digitální signály tak napájecí napětí. Nejčastěji využívána rozhraní jsou SPI a I^2C, ale některé moduly používají tyto signály pro běžné vstupy a výstupy(GPIO).

PmodOLED je monochromatický OLED displej velikosti 128x32 pixelů. Displej je zobrazen na následujícím obrázku:



Obr. C.1: OLED displej

Pmod OLED má tyto rozměry: 16 řádků a 4 sloupce.

Grafický panel využívá řadič displeje Solomon Systech SSD1306. Pmod OLED komunikuje s vývojovým kitem pomocí SPI rozhraní. Zápisem a ponecháním signálu Chip Select(CS) na úrovní logické nuly je uživatel schopen posílat příkazy i data na řadič displeje v závislosti na stavu Data/Command(D/C) pinu.

RES pin je využíván k resetu řadiče displeje SSD1306

Uživatel může rozsvítit kterýkoliv pixel, zobrazit předem definovaný znak nebo dokonce nahrát bitmapový obrázek na obrazovku displeje.

Dva zdroje napájení displeje jsou řízeny dvěma FET tranzistory. VDDC řídí napájení do logiky displeje a VBATC řídí napájení displeje jako takového.

Pro správné fungovaní je třeba displej nejprve inicializovat.

PmodOLED využívá 12 pinový konektor se sériovým periferním rozhraním (SPI), rozložení pinů je uvedeno v tabulce.

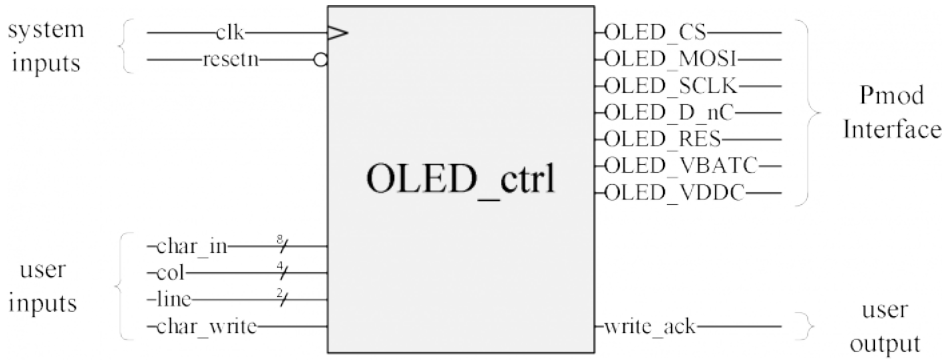
Tab. C.1: Popis jednotlivých pinů OLED displeje

PIN	**signál**	**popis**
1	CS	Chip Select (výběr chipu)
2	MOSI	Master Out,Slave In (master výstup, slave vstup)
3	NC	Not Connected (není připojen)
4	SCK	Serial Clock (sériový hodinový signál)
5	GND	Power Supply Ground (zem napájecího napětí)
6	VCC	Power Supply (napájecí napětí)(3.3V)
7	D/C	Data/Command control (data/příkaz)
8	RES	Power Reset (reset)
9	VBATC	Vbat Battery Voltage Control (řízení napájení Vbat)
10	VDDC	Vdd Logic Voltage Control (řízení napájení Vdd)
11	GND	Power Supply Ground (zem napájecího napětí)
12	VCC	Power Supply (napájecí napětí)(3.3V)

C.2.1 OLED kontrolér

Tento modul zobrazuje text zakódovaný v ASCII znacích na OLED displeji. Dokáže zobrazit znak na jakékoli pozici na displeji, pomocí uživatelských proměnných char_in(vstupní znak), line(řádek) a col(sloupec). Jsou podporovány pouze standartní znaky(ASCII hodnoty od 0 do 127).

Zápis znaku probíhá tak, že pomocí osmi přepínačů je zvolen konkrétní znak, dvoubitová proměnná line určuje, na kterém řádku(0-4) OLED displeje bude zvolený znak zobrazen a čtyřbitová proměnná col zase, ve kterém sloupci(0-15). K potvrzení zápisu znaku slouží proměnná char_write, během doby, kdy je tato proměnná v logické jedničce se nesmí měnit hodnota ASCII znaku ani pozice. Také musí proměnná zůstat v logické jedničce dokud nepřijde potvrzení v podobě signálu write_ack, reakční doba tohoto kontroléru se pohybuje mezi 100μs a 1ms.



Obr. C.2: Vstupy a výstupy OLED kontroléru

Vstupními signály je hodinový signál clk a resetn, který je aktivní při logické nule. Výstupních signálu je sedm a jsou to OLED_CS, OLED_MOSI, OLED_SCLK, OLED_D_nC, OLED_RES, OLED_VBATC a OLED_VDDC.

Komunikačním protokolem nad displejem je rozhraní SPI.

Kontrolér se skládá z těchto modul komponent: OLED_CPU, OLED_SPI, Coderam_mgr a RAM_prog.

C.2.2 Komponenty pro RAM paměť

V komponentě RAM_prog se nachází ASCII znak pro každou adresu. Definicí prvních 64 adres lze definovat, co se na displeji defaultně zobrazí. Komponenta Coderam_mgr zajišťuje, zda se na konkrétní pozici displeje zobrazí znak uložený v RAM paměti nebo se načte znak zadaný uživatelskou vstupní hodnotou.

C.2.3 Komponenta pro SPI rozhraní

Popis této komponenty se nachází v příloze.

C.3 Vypracování laboratorní úlohy

Prostudujte jednotlivé komponenty OLED_ctrl, zaměřte se na top komponentu, ve které jsou všechny ostatní komponenty zamapovány. Všimněte si, že jsou zamapovány přímo bez samostatného deklarování entity.

C.3.1 Úkol č.1

Do připraveného projektu vytvořte nový ucf soubor, vkopírujte do něj uzly pro jeden z 12 pinových konektorů (JA-JD) ze souboru master.ucf, důležité jsou položky

„Sch name“. Zapište jména výstupních signálů, které najdete v top komponentě OLED_ctrl do ucf souboru s pomocí tabulky rozložení pinů.

Dále je potřeba deklarovat přepínače pro vstupní signály resetn, col, line a char_write. Signál resetn slouží pro invertovaný reset, pro spuštění programu musí tedy být přepínač v hodnotě logická jednička! Signály col a line určují sloupec a řádek do kterého má být zapsán znak. Signál char_write slouží pro potvrzení, že byl zadán znak a displej má být aktualizován.

Bohužel vývojová deska Nexys 3 neobsahuje dostatečný počet přepínačů pro nastavení hodnoty ASCII znaku (signál char_in), proto je třeba v deklaraci entity tento signál zakomentovat. Také je třeba zjistit jak je tento signál využíván v programu a místo něj použít pevnou ASCII hodnotu, inspirovat se můžete touto tabulkou.

Tab. C.2: ASCII hodnoty

znak	**Binární**	**Hexadecimální**	**Decimální**
0	00110000	30	48
1	00110001	31	49
2	00110010	32	50
3	00110011	33	51
4	00110100	34	52
5	00110101	35	53
6	00110110	36	54
7	00110111	37	55
8	00111000	38	56
9	00111001	39	57
A	01000001	41	65
B	01000010	42	66
C	01000011	43	67
D	01000100	44	68
E	01000101	45	69
F	01000110	46	70

Ověřte, že program funguje (na displeji se objeví text). Zkuste měnit přepínače col a line, poté přepněte přepínač char_write do logické jedničky a ověřte, že byl zapsán znak na vámi zvolenou pozici.

C.3.2 Úkol č.2

Do ucf souboru vkopírujte uzly z úlohy č.1 pro další 12 pinový konektor, do kterého je připojena maticová klávesnice. Deklarujte entitu pro komponentu Dekoder a namapujte ji. Nezapomeňte tuto komponentu přidat do projektu.

Je třeba vytvořit převod mezi dekódovanou 4 bitovou hodnotou a ASCII hodnotou, využijte tabulku výše. Dále je nutné dříve pevně nastavenou ASCII hodnotu (místo char_write) změnit na tuto převedenou dekódovanou hodnotu stisknutého znaku na klávesnici.

Zajistěte, aby při stisknutí klávesy došlo automaticky k aktualizaci displeje.

C.3.3 Bonusový úkol

Upravte program tak, aby docházelo k automatickému posunu zapisované pozice o jednu doprava a vznikl tak „souvislý“ text. Pokud se dojde na poslední pozici na displeji (15.sloupec na 4.řádku), vrátí se zpět na začátek.

D Úloha č.3 - komunikace s OLED displejem 2

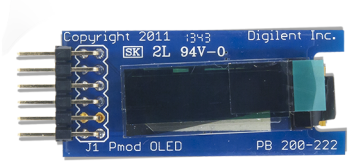
D.1 Cíle

- Připojit OLED displej a maticovou klávesnici pomocí Pmod konektoru (dbát na správné připojení napájecích pinů!)
- Otestovat program pro výpis textu na tento displej
- Zobrazit stisknuté klávesy na klávesnici na displeji jako souvislý text
- Zobrazit sloupcový graf na výšku v závislosti na stisknuté klávese na klávesnici
- Implementovat návrh do cílového FPGA obvodu Spartan6 na vývojové desce

D.2 Teoretický úvod

Pmod^{TM} je technologický standart periferních modulů od firmy Digilent. Tyto přídavné moduly umožňují rozšířit možnosti vývojové desky o další programovatelné komponenty. Moduly jsou připojovány pomocí 6 nebo 12 pinnového konektoru, buďto přímo nebo pomocí kabelu, a tyto piny přenášejí jak digitální signály tak napájecí napětí. Nejčastěji využívána rozhraní jsou SPI a I^{2}C, ale některé moduly používají tyto signály pro běžné vstupy a výstupy(GPIO).

PmodOLED je monochromatický OLED displej velikosti 128x32 pixelů. Displej je zobrazen na následujícím obrázku:



Obr. D.1: OLED displej

Pmod OLED má tyto rozměry: 16 řádků a 4 sloupce.

Grafický panel využívá řadič displeje Solomon Systech SSD1306. Pmod OLED komunikuje s vývojovým kitem pomocí SPI rozhraní. Zápisem a ponecháním signálu Chip Select(CS) na úrovní logické nuly je uživatel schopen posílat příkazy i data na řadič displeje v závislosti na stavu Data/Command(D/C) pinu.

RES pin je využíván k resetu řadiče displeje SSD1306

Uživatel může rozsvítit kterýkoliv pixel, zobrazit předem definovaný znak nebo dokonce nahrát bitmapový obrázek na obrazovku displeje.

Dva zdroje napájení displeje jsou řízeny dvěma FET tranzistory. VDDC řídí napájení do logiky displeje a VBATC řídí napájení displeje jako takového.

Pro správné fungovaní je třeba displej nejprve inicializovat.

PmodOLED využívá 12 pinový konektor se sériovým periferním rozhraním (SPI), rozložení pinů je uvedeno v tabulce.

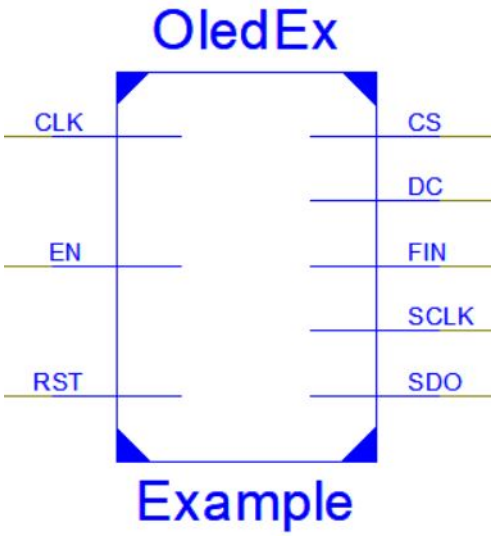
Tab. D.1: Popis jednotlivých pinů OLED displeje

PIN	**signál**	**popis**
1	CS	Chip Select (výběr chipu)
2	MOSI	Master Out,Slave In (master výstup, slave vstup)
3	NC	Not Connected (není připojen)
4	SCK	Serial Clock (sériový hodinový signál)
5	GND	Power Supply Ground (zem napájecího napětí)
6	VCC	Power Supply (napájecí napětí)(3.3V)
7	D/C	Data/Command control (data/příkaz)
8	RES	Power Reset (reset)
9	VBATC	Vbat Battery Voltage Control (řízení napájení Vbat)
10	VDDC	Vdd Logic Voltage Control (řízení napájení Vdd)
11	GND	Power Supply Ground (zem napájecího napětí)
12	VCC	Power Supply (napájecí napětí)(3.3V)

OLED kontrolér obsahuje hlavní bloky OledInit a OledEx. OledInit provádí incializaci displeje podle doporučeného postupu výrobcem.

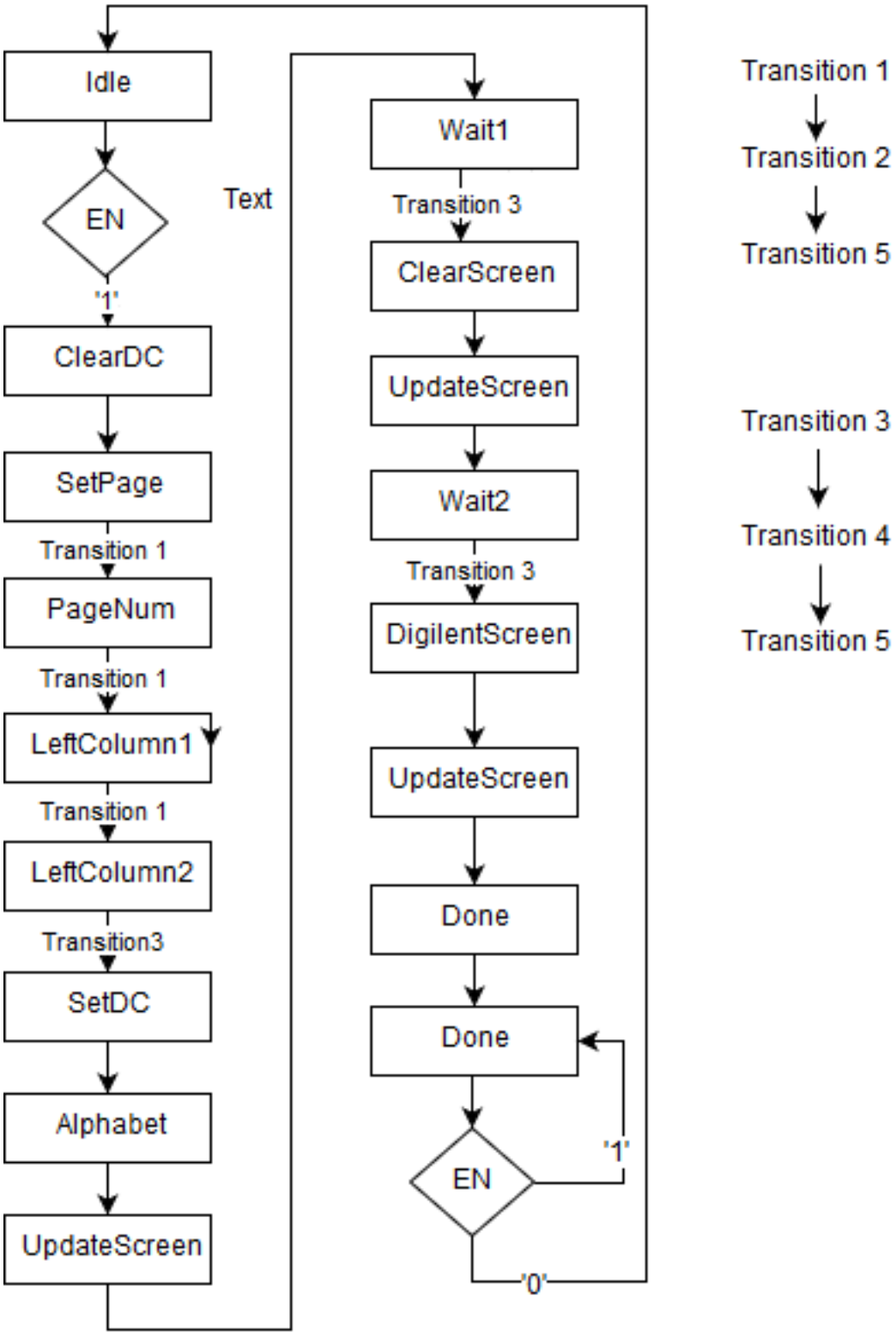
D.2.1 Blok OledExample

Vstupy do tohoto bloku jsou signály CLK (systémový hodinový signál o freknenci 100MHz), EN (povolujicí tento blok) a RST (globální reset). Výstupy z tohoto bloku jsou signály CS (SPI výběr chipu), DC (data/příkaz), FIN (udávající, že tento blok dokončil svou činnost), SCLK (sériový hodinový signál) a SDO (SPI sériový datový výstup).



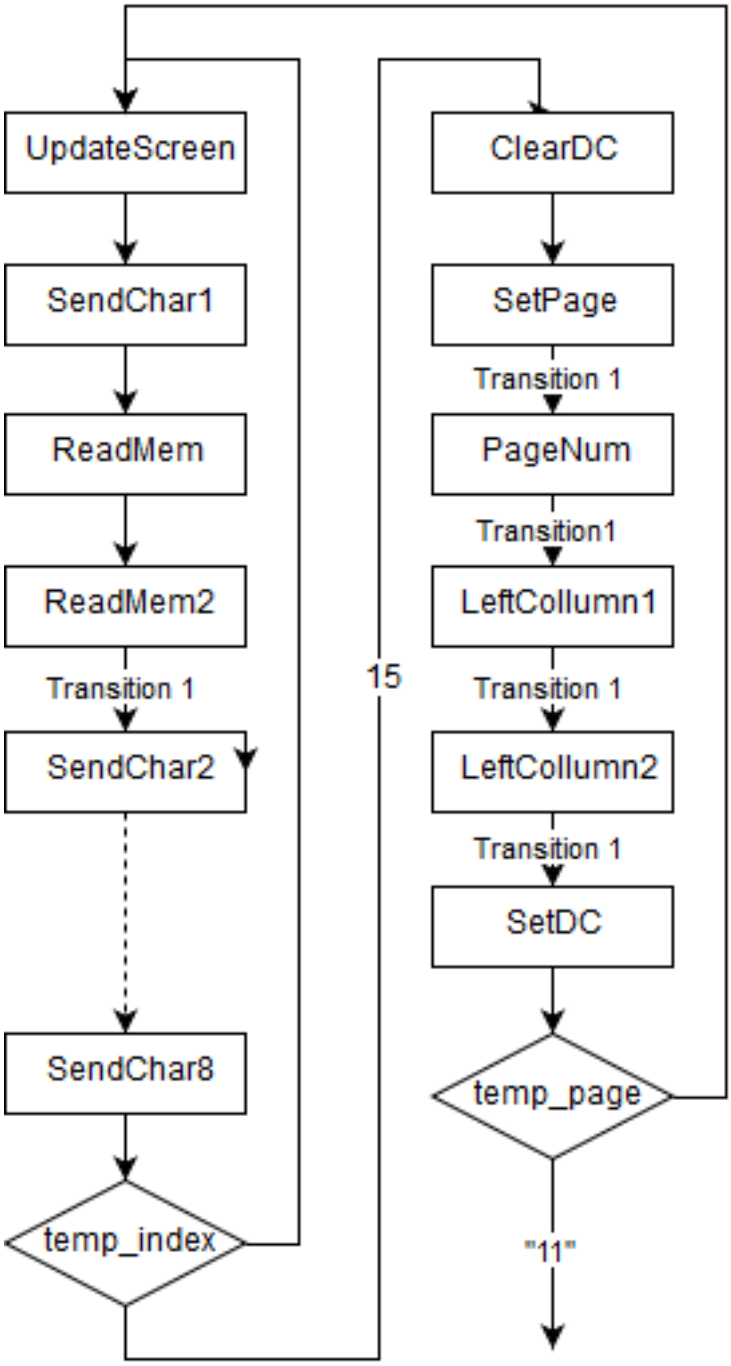
Obr. D.2: Blokové schéma OledExample

Blok OledExample čeká do té doby, než je signál EN nastaven do logiké jedničky a poté přechází do stavu ClearDC. Ze stavu ClearDC se dostává do stavů SetPage, PageNum, LeftColumn1, LeftColumn2 a SetDC, který nastaví počet stran PmodOlED na 0 předtím než dojde k přechodu do stavu Alphabet. Signál current_screen je nastaven na abecedu a poté se přechází na stav UpdateScreen, který aktualizuje obrazovku PmodOLED na ASCII znaky uložené v current_screen. Poté čeká 4 sekundy vymaže obrazovku, čeká 1 sekundu a poté se na obrazovku vypíše „This is Digilent's PmodOLED".



Obr. D.3: Vývojový diagram OledExample

Průběh stavu UpdateScreen: Postupně je odesíláno 8 částí ASCII znaku, postupuje se na další znak a přehází se zpět do stavu Updatescreen do té doby než se dojde na konec řádku (resp. stránky, kdy temp_char = 15, jedná se o 16.znak). Poté se znovu prochází stavy ClearDc, SetPage, PageNum, LeftCollumn1. LeftCollumn2 a SetDC. Celý proces se opakuje do té doby než se dojde na konec displeje (kdy temp_page = "11"a temp_char = 15, jedná se o 16.znak na 4.řádku, resp. stránce).



Obr. D.4: Vývojový diagram pro UpdateScreen

D.2.2 Paměťový blok charLib

Tento paměťový blok obsahuje bytové mapy pro jednotlivé znaky. Znaky jsou obsaženy v 8 bytových částech. 11 Bitové adresování je následující: ASCII hodnota & XXX, kde poslední 3 bity představují 8 částí každého znaku. Např. adresa „01000001001“ představuje druhou část znaku 'A'. Bytová mapa se nachází v souboru charlib.coe, tento soubor má 3 řádkový offset, takže např. znak 'A' (ASCII hodnota 65 v desítkové soustavě) se nachází na řádku 68. Každý byte představuje jeden sloupec v 8x8 pixelovém rastru. Nejvýznamnější bit (MSB) se nachází úplně dole sloupce. Jednotlivé znaky jsou odděleny mezerou dole a vpravo o šířce jeden pixel (samé nuly v bitové mapě).

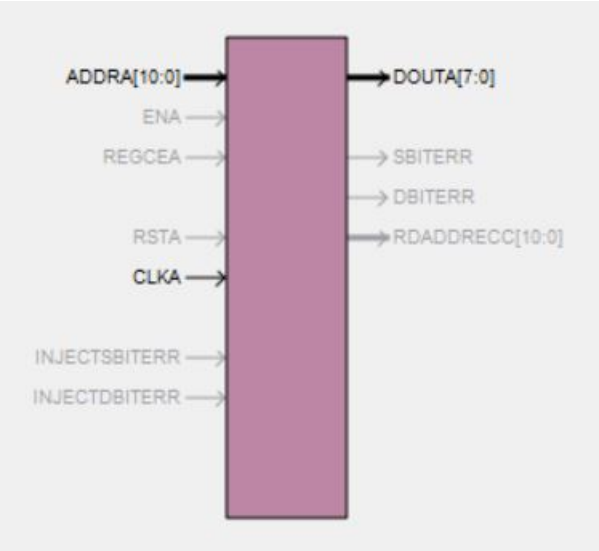
```
64    00,14,14,14,14,14,14,00,
65    00,00,22,14,14,08,00,00,
66    00,02,01,59,05,02,00,00,
67    3e,41,5d,55,4d,51,2e,00,
68    40,7c,4a,09,4a,7c,40,00,
69    41,7f,49,49,49,49,36,00,
70    1c,22,41,41,41,41,22,00,
71    41,7f,41,41,41,22,1c,00,
```

Obr. D.5: Soubor charlib.coe

Znak 'A' v bitové mapě vypadá následovně (1 znamená rozsvícený pixel):

```
00010000
00101000
01000100
01111100
01000100
01000100
11101110
00000000
```

Obr. D.6: Bitová mapa znaku 'A'



Obr. D.7: Schéma paměťového bloku

Tento blok byl vytvořen pomocí Xilinx Block Memory Generátoru. Jeho schéma je zobrazeno na obrázku D.7.

Ve VHDL popisu je zapsán takto.

```
COMPONENT charLib
  PORT (
    clka : IN STD_LOGIC; --Attach System Clock to it
    addra : IN STD_LOGIC_VECTOR(10 DOWNTO 0); --First 8
       bits is the ASCII value of the character the last
       3 bits are the parts of the char
    douta : OUT STD_LOGIC_VECTOR(7 DOWNTO 0) --Data byte
       out
  );
END COMPONENT;
```

Vstupem je tedy 11 bitová adresa a systémový hodinový signál, výstupem je příslušný byte (odpovídající sloupci v bitové mapě) pro adresovanou část znaku.

D.2.3 Blok SpiCtrl

Popis bloku SpiCtrl se nachází v příloze SpiCtrl.pdf

D.2.4 Nastavení cesty k souboru charlib.coe

Pro změnu vykreslování ASCII znaků je třeba editovat soubor charlib.coe, který se nachází ve složce projektu. Po editaci je třeba znovu generovat paměťový blok a to

následovně: V prostředí Xilinx ISE kliknout na komponentu CHAR_LIB_COMP a zvolit Manage Cores. Otevře se nové okno Xilinx CORE Generator, v něm v levém dolním rohu kliknout na charLib, v pravém okně v nabídce Actions kliknout na Recustomize and Generate (Under Current Project Settings). Otevře se další okno Block Memory Generator, v něm klikat na tlačíkto Next, neměnit žádné nastavení, až na straně 4 v sekci Memory Initialization kliknout na tlačítko Browse a zvolit cestu k upravenému souboru charlib.coe, poté kliknout na Generate. Potvrdit přepsání. Zavřít Block Memory Generator. Nyní po syntéze bude použit upravený soubor.

D.3 Vypracování laboratorní úlohy

Otestujte funkčnost přiloženého programu, na displeji by se měl zobrazit text: „This is Digilent's PmodOLED". Prostudujte jednotlivé komponenty. Zaměřte se na blok OledEx a stavový automat, který je v něm obsažen.

D.3.1 Úkol č.1

Připojte do projektu komponentu pro dekódování stisknutého tlačítka na klávesnici. Nezapomeňte upravit ucf soubor. Upravte blok OledEx tak, aby zobrazoval stisknuté klávesy na displeji za sebou, aby vznikl „souvislý" text.

Ve standartu VHDL93 nelze rozdělit více rozměrné pole Digilent_screen, ve kterém jsou uloženy ASCII znaky. Je třeba vytvořit funkce pro rozdělení 2D pole na více 1D polí a jejich opětovné spojení ve 2D pole. Pro převod mezi dekódovanou hodnotou tlačítka a ASCII hodnotou využijte následující tabulku.

Tab. D.2: ASCII hodnoty

znak	Binární	Hexadecimální	Decimální
0	00110000	30	48
1	00110001	31	49
2	00110010	32	50
3	00110011	33	51
4	00110100	34	52
5	00110101	35	53
6	00110110	36	54
7	00110111	37	55
8	00111000	38	56
9	00111001	39	57
A	01000001	41	65
B	01000010	42	66
C	01000011	43	67
D	01000100	44	68
E	01000101	45	69
F	01000110	46	70

D.3.2 Úkol č.2

Graficky zobrazte na displeji sloupcový graf na výšku. Jeden znak bude reprezentovat 2-32 rozsvícených pixelů na výšku se šířkou 8 pixelů. 2 rozsvícené pixely odpovídají stisknuté klávese '0' a 32 rozsvícených pixelů klávese 'F'. Nejpozději stisknutý znak se bude zobrazovat vždy vlevo na displeji, ostatní znaky se budou posouvat směrem doprava.

Obr. D.8: Ukázka grafického zobrazení