VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

NÁVRH ZOBECNĚNÉHO LCD ŘADIČE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE BC. TOMÁŠ HORNUNG
AUTHOR

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

NÁVRH ZOBECNĚNÉHO LCD ŘADIČE

DESIGN OF A GENERIC LCD DRIVER

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — BC. TOMÁŠ HORNUNG

AUTHOR

VEDOUCÍ PRÁCE — ING. JOSEF STRNADEL, Ph.D.

SUPERVISOR

BRNO 2007

Abstrakt

Diplomová práce se zabývá principem řízení a činnosti LCD displejů. Jsou zde popsány různé typy LCD displejů a způsob jejich ovládání, a také obecný popis bloků řadiče LCD displeje. Jejich implementace v jazyce VHDL a popis výsledků simulace a realizace tohoto řadiče na platformě FITkit.

Klíčová slova

LCD, řadič, typy LCD, konečný automat, kódování instrukcí

Abstract

The master thesis deals with controlling and function principles of LCDs. Various LCD types are described, as well as means of control and generic description of LCD controller function blocks. They are implemented in VHDL language, simulation and simulation results are realized using FITkit platform..

Keywords

LCD, controler, LCD types, finite automation, instruction coding

Citace

Hornung Tomáš: Návrh zobecněného LCD řadiče. Brno, 2008, diplomová práce, FIT VUT v Brně.

Návrh zobecněného LCD řadiče

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením
Ing. Josefa Strnadela, Ph.D.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Tomáš Hornung
21.1.2008

Poděkování

Děkuji mému vedoucímu diplomové práce Josefu Strnadelovi za cenné rady při řešení. A také lidem z týmu FITkit za velmi dobré a cenné informace jak na webu, tak v zdrojových kódech zde řešených projektů.

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

Obsah

Obsah 1
1 Úvod 3
2 LCD 4
2.1 Typy LCD 5
2.1.1 Rozdělení podle nejmenší zobrazovací jednotky 5
2.1.2 Rozdělení podle řídící matice 5
2.1.3 Podle podsvětlení displeje 7
2.2 Napájení a typy podsvícení displeje 8
3 Řadiče LCD 10
3.1 Statické (přímé) řízení LCD 10
3.2 Multiplexované řízení LCD 11
4 Platforma FITkit 16
4.1 SPI 16
4.2 Knihovna libfitkit 18
5 Návrh zobecněného LCD řadiče 20
5.1 Konfigurace 21
5.1.1 Zápis informací o připojených pinech 21
5.1.2 Pole znaků 22
5.2 Kódování instrukcí 24
5.3 Popis hlavních částí VHDL kódu 25
5.3.1 Automat pro zpracování instrukcí 25
5.3.2 Mapování znaků do výstupní paměti 28
5.3.3 Řízení LCD 30
6 Simulace a syntéza řadičů 32
6.1 LCD displej s přímým řízením 32
6.2 LCD displej 7x1 37
6.3 LCD displej NTC/22K 42
6.4 Shrnutí výsledků syntézy 46
7 Realizace pro vybraný LCD displej 47
7.1 Propojení s FITkitem 48
7.2 Programování FPGA 49
7.3 Programování mikrokontroleru 49
7.4 Průběh komunikace PC -> LCD 49
7.4.1 Popis komunikace od příkazu až k zobrazení na LCD 50

7.4.2 Příklady použití 53
8 Závěr 56
Literatura 58
Seznam příloh 60

1 Úvod

Pro zobrazování informací je potřeba zobrazovacích prostředků. V dnešním moderním světě, kde jsou takřka všechny informace podávány elektronicky díky jejich interaktivitě, musíme využít elektronických zobrazovačů. Z pohledu dnešních dnů jsou z různých důvodů, nejvíce využívány LCD zobrazovače. Jedná se např.o velké LCD displeje nahrazující dřívější CRT obrazovky nebo o malé zobrazovače použité pro zobrazení informací, například stav tiskárny, informace o chybách, atd. Jako další aplikace můžu uvést kalkulačky, ruční měřící přístroje. Zde je s výhodami použito LCD displejů. Mezi tyto výhody patří nízká hmotnost, a podle typu LCD také nízký příkon.

Z důvodu použití grafických a znakových displejů, které používají mnoho segmentů, je u těchto displejů zavedeno multiplexování. Pomocí této metody docílíme snížení potřebných vývodů daného LCD za cenu složitějších průběhů výstupních signálů. Tyto průběhy jsou víceúrovňové, a jelikož FPGA na svých výstupech takový signál generovat nedokáží, je nutné zajistit tyto průběhy pomocí dalších elektronických součástek.

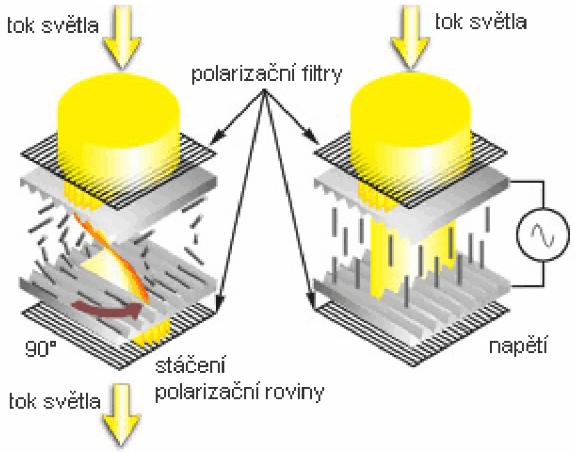
V této práci jsem se zaměřil především na řešení řadiče v FPGA, a nezabývá se řešením podpůrné elektroniky pro generování víceúrovňového výstupu pro multiplexované displeje. V navrhovaném řadiči je možnost informace o multiplexování nastavit, a podle ní řídit displej, ale z již dříve uvedených důvodů nemožnosti takový signál generovat přímo v FPGA, není tato metoda bez použití dalších elektronických obvodů použitelná přímo na platformě FITkit, pro kterou tento byl zobecněný řadič programován. Implementován je pouze jednoduchý časový multiplex, kdy dochází k přepínání řídících signálů multiplexu. Generování víceúrovňového výstupu se provádí například pomocí odporového žebříku, kde pomocí správně zvolených hodnot odporů určíme napětí jednotlivých úrovní.

Tato práce navazuje na Semestrální projekt Simulace a návrh řadiče LCD displeje, ze kterého jsou čerpány kapitoly 2, 3, a základní myšlenka zobecněného řadiče a jeho vnitřních bloků z kapitoly 5.

2 LCD

Kapalné krystaly jsou anizotropní kapaliny tj. látky s různými vlastnostmi v různých směrech [1]. Tuto odlišnost oproti běžným izotropním látkám způsobují molekuly, které se v určitém teplotním rozsahu samovolně směrově uspořádávají, ovšem pod podmínkou nekulovitého tvaru. Vlastnost kapalných krystalů využívaná v LCD displejích (uspořádání) je zachována i přesto, že kapalina zaujímá tvar nádoby (kde je umístěna), v které teče. Při používání LCD displejů je důležitý teplotní rozsah, neboť právě v tomto rozsahu existuje kapalně krystalická fáze (mezomorfní fáze), tj. fáze mezi stavem pevné látky a izotropní látkou.

V LCD displejích se bez přiloženého pole nematický kapalný krystal uspořádává tak, aby dopadající (sluneční světlo či světlo z podsvícení) polarizované světlo propouštěl. Po přiložení vnějšího pole se mezi elektrodami optická osa změní tak, že krystal polarizované světlo (světlo projde polarizátorem) nepropouští. Zobrazený znak (bod) se tedy jeví jako tmavý oproti okolí, které světlo propouští. Takový displej má malou spotřebu (dle velikosti od desítek μA) a ke změně orientace optické osy stačí malé vnější pole (přiložené napětí).



obr. 1 Průřez LCD displejem

Jak je vidět z obrázku (obr. 1), kapalný krystal otáčí optickou osu světla o 90° (TN-Twisted Nematic). Existují i LCD displeje, jejichž tekutý krystal natočí světlo i o úhel větší než je 90° (STN-Super Twisted Nematic), přičemž kontrastní poměr těchto displejů se přibližně 3x a 4x zvýší, takže čitelnost displeje je pak podstatně lepší. S rostoucím úhlem natočení světelné roviny však dochází ke zkreslení barevného podání (platí u barevných displejů). Tento nepříznivý jev lze eliminovat, jestliže světlu dáme do cesty dvě buňky, které světelnou osu otáčí v opačném směru (DSTN-Double Super Twisted Nematic).

V případě použití barev v LCD máme dva způsoby realizace:

- přidáním dichroických barev (látky, jejíchž barva je citlivá na orientaci dopadajícího světla vůči molekulám),
- pomocí barevných filtrů (používanější a levnější)

2.1 Typy LCD

V této podkapitole jsou LCD rozděleny podle různých hledisek, a to jak podle možnosti zobrazení, tak podle možnosti řízení, různých typů podsvícení.

Ucelený seznam různých typů LCD je dostupný na internetových stránkách, ze kterých jsem v této práci čerpal [2][3].

2.1.1 Rozdělení podle nejmenší zobrazovací jednotky

- **Segmentové**
  - znaky, jako jsou čísla nebo písmena, se skládají z několika segmentů, mezi tyto segmenty patří také další značky na displeji (například na hodinkách jsou písmena AM brána jako jeden segment a také se tak ovládají),
- **Znakové** (nejmenší zobrazovací jednotkou je znak),
  - - jednořádkové (1x8, 1x10, 1x12, 1x16, 1x20, 1x24, 1x40, ...),
  - - víceřádkové (2x8, 2x10, 2x16, 2x20, 2x24, 2x40, 4x16, 4x20, 4x24, 4x40, ...),
- **Grafické** (nejmenší zobrazovací jednotkou je bod),
  - kombinace čísel 64, 80, 97, 98, 120, 122, 128, 160, 192, 240, 256, 320, 640, ...

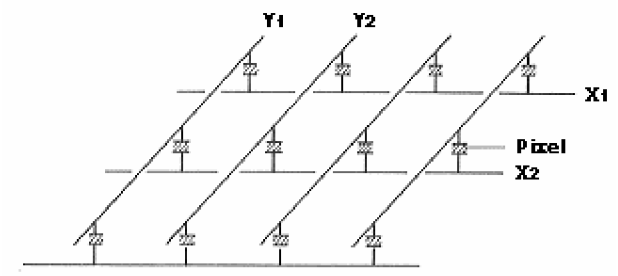
2.1.2 Rozdělení podle řídící matice

Na prvopočátku, kdy se LCD displeje používaly např. v digitálních hodinkách, byly jednotlivé segmenty řízeny samostatně. Takový LCD displej obsahoval dle typu okolo 40 zobrazovacích segmentů. Na jejich řízení proto stačilo 40 vodičů (plus napájení).

V dnešní době, kdy LCD displeje jsou běžně schopny zobrazení 320x240 bodů, ovšem toto řízení vzhledem k rozměrům není možné (bylo by nutno 76800 vodičů). Proto se používá maticové řízení displeje, kdy je zapotřebí při rozlišení 320x240 jen 760 vodičů.

2.1.2.1 Pasivní řídící matice

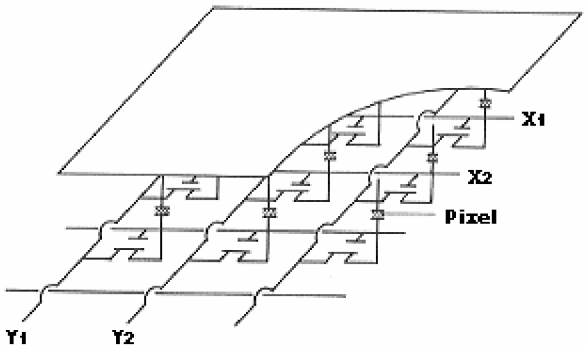
Při maticovém řízení (obr. 2) pomocí pasivní matice mají elektrody v LCD tvar průhledných řádkových a sloupcových elektrod a aktivní je ten zobrazovací prvek, který leží v průsečíku těch elektrod, na něž je přiloženo aktivní řídicí napětí. U tohoto typu řízení displeje je možno najednou ovládat nejen jeden bod, ale celý řádek či sloupek. Při zobrazování obrazu na LCD displeji je zapotřebí obraz obnovovat, přičemž setrvačnost zobrazovaných bodů je 180 – 350ms.



obr. 2 Pasivní řídící matice

2.1.2.2 Aktivní řídící matice

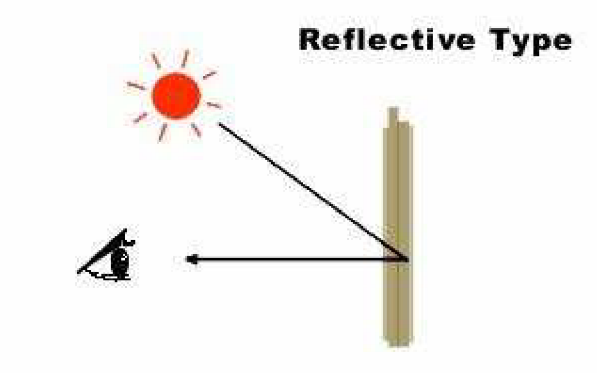
Problémy s pasivní řídící maticí nastanou, jestliže chceme použít k rozlišení většího počtu úrovní například šedi. Řídící napětí se musí měnit ve velkém rozsahu a velké změny v jednom bodě způsobí rušení okolních bodů. V takovémto případě je vhodnější použít řízení pomocí aktivní matice (obr. 3), kdy řízení jasu jednotlivých bodů zajišťuje aktivní prvek (tranzistor). Řídící napětí (jasu) je zapotřebí podstatně nižší a proměnné elektrické pole, řídící jas, se vytváří až na jeho elektrodách - neovlivňuje okolní body. Výroba takovýchto LCD je ovšem komplikovanější, kdy se např. jednotlivé tranzistory vytvářejí litografický, tj. na amorfním křemíku naneseném na skleněném podkladu, takže vlastně vzniká jakýsi čip na skle. Setrvačnost bodu displeje s aktivní řídící maticí je okolo 30ms.



obr. 3Aaktivní řídící matice

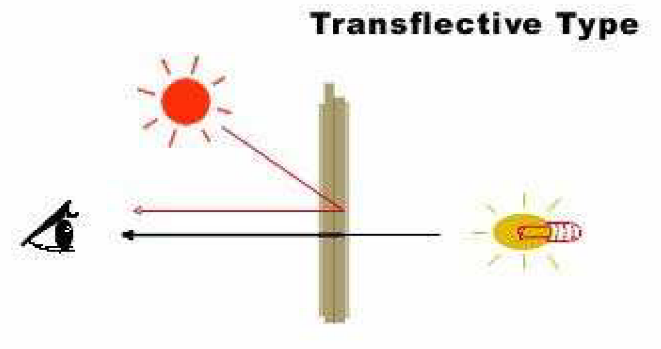
2.1.3 Podle podsvětlení displeje

- **Reflektivní:** Takový displej obsahuje difuzor. Tato vrstva odráží světlo, které dopadá na přední stranu displeje. Reflektivní displej potřebuje okolní osvětlení jako světelný zdroj, protože nemá žádné vlastní podsvětlení (obr. 4),



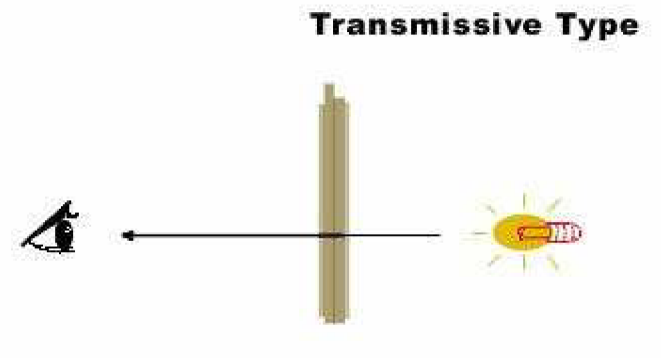
obr. 4 Reflektivní podsvětlení

- **Transflektivní:** Typ, který je vázán na zadní polarizátor. Umožňuje světlu projít jak skrz, tak odrazit se od zadní strany displeje (obr. 5),



obr.5 Transflektvní podsvětlení

- **Transmitivní:** Typ, který nemá reflektor nebo transflektor laminovaný na zadní polarizátor. Tento typ musí používat podsvětlení. Nejčastější je transmitivní negativní obraz (obr. 6).



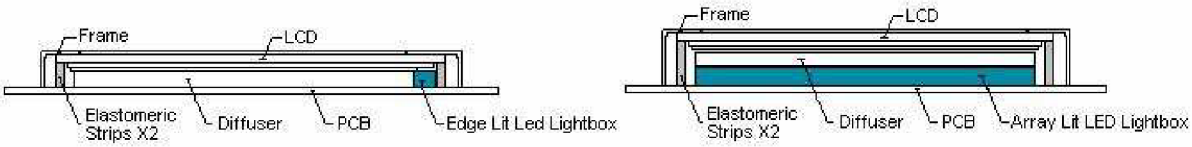
obr. 6 Transmitivní podsvětlení

Barevné displeje bývají transmitivní, jednobarevné jsou nejčastěji reflexní.

2.2 Napájení a typy podsvícení displeje

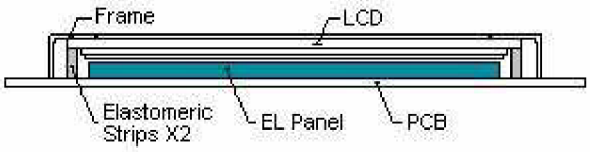
Dále dle [4] je možné rozlišit následující typy podsvíceni LCD:

- **Podsvíceni pomocí LED** – jako podsvicovací prvek slouží LED, jejichž světlo se rozptyluje na celou plochu displeje pomocí speciální vrstvy (obr. 7). Tato vrstva mívá tloušťku několika milimetrů, čímž se zvětšuje celková výška displeje. Na napájení podsvícení tohoto typu displeje ovšem vystačíme s napětím obdobným (standardně 4,2V) napájecímu napětí displeje. Životnost tohoto druhu podsvícení se běžně pohybuje nad 50 000 až 120 000 provozních hodin (dle intenzity),



obr. 7 Led podsvícení

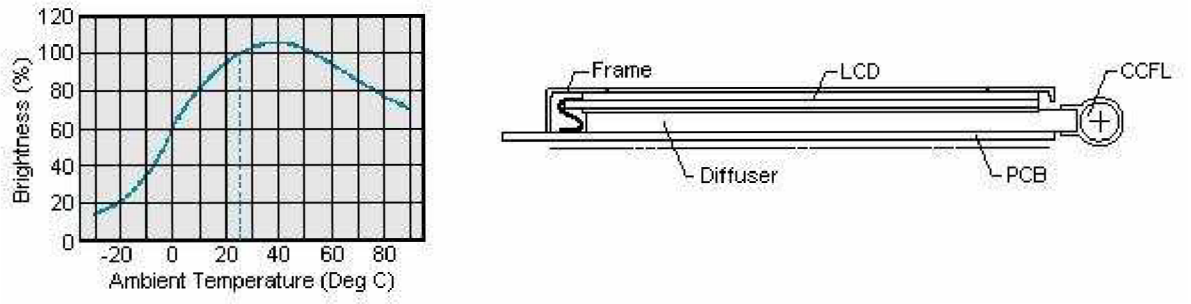
- **Podsvícení pomocí fólie (EL)** – při tomto tytu podsvícení odpadá použití rozptylovací vrstvy, čímž se sníží výška displeje (obr. 8). Toto podsvícení má dvě základní nevýhody,
  - napájecí napětí – tuto fólii musíme napájet střídavým napětím o frekvencích od desítek Hz až do 1 kHz, přičemž frekvence má vliv na stabilitu intenzity podsvícení (pozor na blikání displeje při nízkých frekvencích). Velikost napájecího střídavého (efektivního) napětí je dle typu od 60V do 150V (standardně 100V 400Hz), přičemž spotřeba proudu je přibližně 0,15mA/cm^2 (dle frekvence a napětí napájení),
  - životnost je podstatně menší než u podsvícení LED, přičemž normálně dosahuje 2000 provozních hodin, u displejů s prodlouženou dobou životnosti až 8 000 provozních hodin,



obr. 8 Podsvětlení pomocí EL folie

- **Podsvícení pomocí fluorescenční lampy se studenou katodou (CCFL)** – tento druh podsvícení je vhodný především pro barevné maticové displeje, přičemž barva podsvícení je jasně bílá, tudíž nezkresluje barvy na displeji. Životnost tohoto druhu podsvícení je okolo 12 000 (max 30 000) provozních hodin. Nevýhodou tohoto druhu podsvícení je napájení, které

vyžaduje řádově stovky (až 1000 V) voltů s frekvencí desítek kHz (20 kHz až 45 kHz, standartně 800 V, 3-6 mA) (obr. 9).



obr. 9 Podsvětlení CCFL

Slovníček k obrázkům 7 až 9 :

Frame - rám

Elastomeric Strips X2 – dva pružné pásky

Diffuser – difuzor (rozptýlení světla)

PCB – deska plošných spojů

Edge Lit LED Lightbox – Led podsvětlení s diodou na kraji displeje

Array Lit LED Lightbox – Led podsvětlení s polem diod pod displejem

EL Panel – elektroluminiscenční panel

CCFL - fluorescenční lampa se studenou katodou

3 Řadiče LCD

V této kapitole je rozebrány způsoby řízeni LCD displejů, kdy pro jednoduché displeje s malým počtem segmentů se využívá statické řízení.

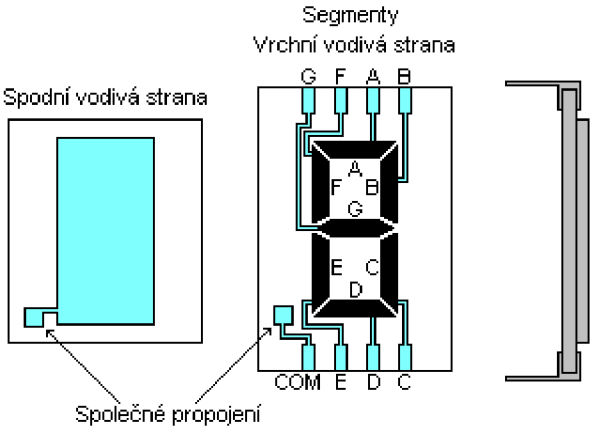
Každý LCD segment má dva vývody:

1. vývod segmentu spojený s BP (COM) displeje pomocí vodivé průhledné folie (angl. tzv. backplane) umístěné na zadní destičce bývá označován BPn (COMn) a
2. zbylý – druhý – vývod (angl. tzv. frontplane) bývá označován FPn (SEGn).

V případě většího počtu segmentů se z důvodu snížení počtu vývodů používá multiplexované řízení. Jelikož řízení složitějších displejů není jednoduché, existují pro tyto displeje řadiče buď v podobě externího čipu, nebo jsou takové řadiče integrovány v mikrokontrolérech. Je-li řadič integrován v LCD, pak hovoříme o LCD modul. Oproti LCD displeji je ovládání LCD modulu jednodušší, ovšem jeho cena je díky integrovanému řadiči vyšší.

3.1 Statické (přímé) řízení LCD

Tato metoda řízení je určena pro ovládání jednotlivých segmentů displeje [5]. Tzn. že každý segment má vyveden vodič na kontakty LCD displeje (obr. 10). Tato metoda používá velké množství propojek, proto pomocí tohoto řadiče nelze ovládat složité displeje. Ale pomocí vytvoření různě vypadajících segmentů lze vytvořit na pohled pěkný displej.



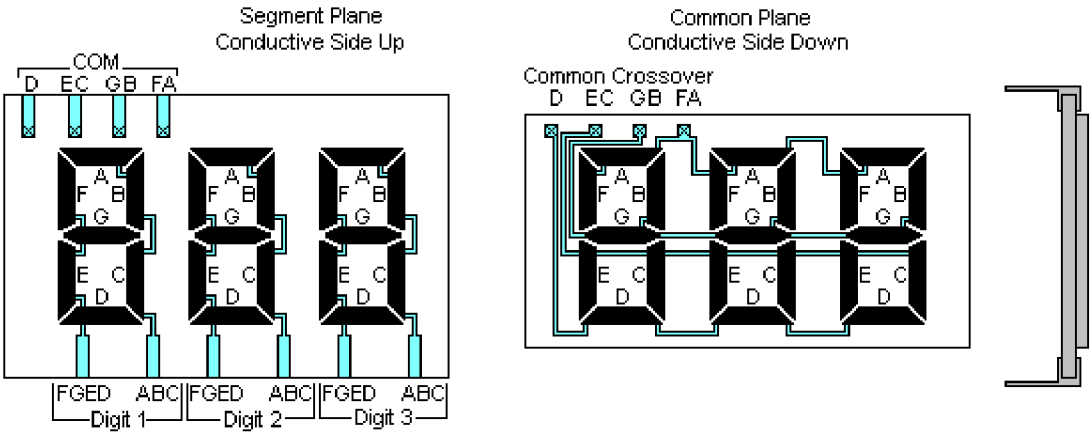
obr. 10 Segment, s přímým řízením

Jednotlivé segmenty jsou řízeny velikostí střídavého napětí přes LCD segment, ale vždy musí být použito střídavé napětí přes všechny segmenty. Pokud použijeme stejnosměrné napětí, můžeme

způsobit elektrochemické reakce uvnitř segmentu, které výrazně zkracují životnost. Jako indikace příliš velkého stejnosměrného napětí na segmentu je jev, kdy se jednotlivé okraje segmentu ztrácí.
TN LCD displeje reagují na efektivní napětí, efektivní hodnotou svorkového napětí mezi COM a daným segmentem určujeme kontrast daného segmentu. Frekvence pro řízení takového LCD je obvykle mezi 30 Hz a 100 Hz.

3.2 Multiplexované řízení LCD

V tomto uspořádání může každá svorka spojovat více segmentu displeje (obr. 11) a podle kombinace spojení svorek je zobrazen určitý segment [6]. Tato metoda minimalizuje počet propojení. Používá se pro komplexní displeje, na kterých není místo na všechny svorky, nebo z důvodu snížení počtu budících obvodů. Toto snížení externích spojů zvyšuje spolehlivost a také zvyšuje potenciální hustotu displeje.



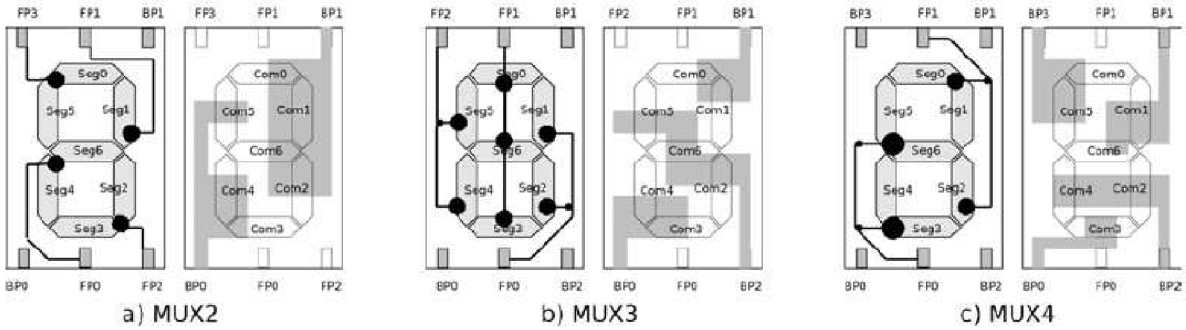
obr. 11 LCD s multiplexovaným řízením

Metoda multiplexování je v podstatě časový multiplex s vícenásobným dělením, který je roven dvojnásobku společných vrstev použitých pro daný displej. Za účelem předejití nevratné elektrochemické reakci ničící displej musíme napětí všech segmentů periodicky obracet, tak aby stejnosměrná složka měla nulovou hodnotu. To je důvod pro zapojeni s časovým multiplexem. Frekvence pro tento typ ovládání je mezi 60 Hz a 90 Hz.

Cílem těchto metod je snížit počet vývodů nutných k řízení LCD displeje za cenu generování komplikovanějších řídících signálů [7]. Princip snížení počtu vývodů nutných k řízení LCD displeje tvořeného n segmenty je následující: jeden FP-vývod displeje bude sdílen k segmenty ($1 < k < n$) zatímco každý z těchto segmentů bude napojen na jiný z k BP-vývodů. Následující tabulka ukazuje vztah mezi $1 \leq k \leq 4$ a počtem vývodů nutných k řízení LCD displeje s n segmenty. Tato tabulka ukazuje, že v případě zvolení komplikovanějších řídících signálů, jsme schopni snížit potřebný počet vývodů LCD displeje.

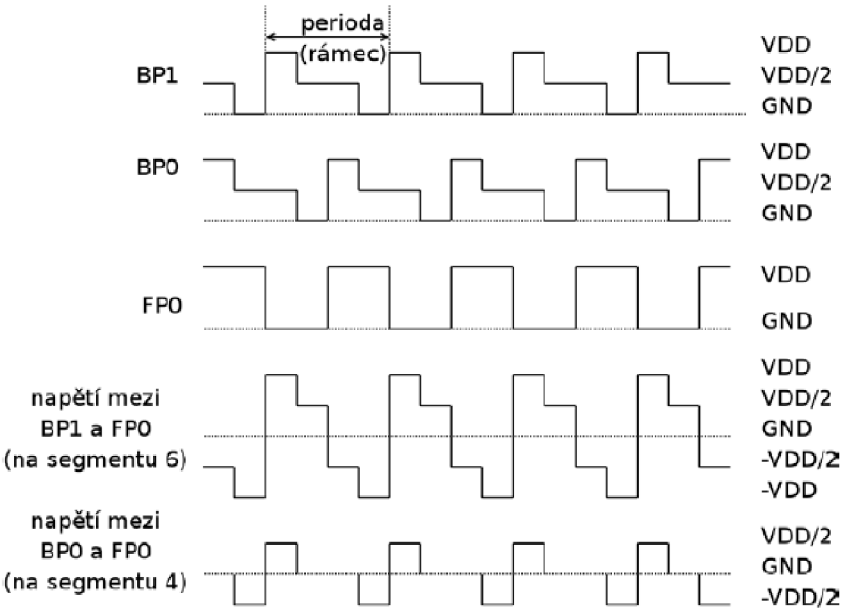
Medoda (k)	Počet vývodů pro řízení n segmentů	Př.: počet vývodů pro n=80
Statická (1)	$1+n$	81
MUX2 (2)	$\lceil k+(n/k) \rceil$	42
MUX3 (3)		30
MUX4 (4)		24

Příklad zapojení 7 segmentů v LCD segmentovkách sdílejícími 2, 3 resp. 4 vývody FP je uveden na obr. 12 a, b, resp. c.



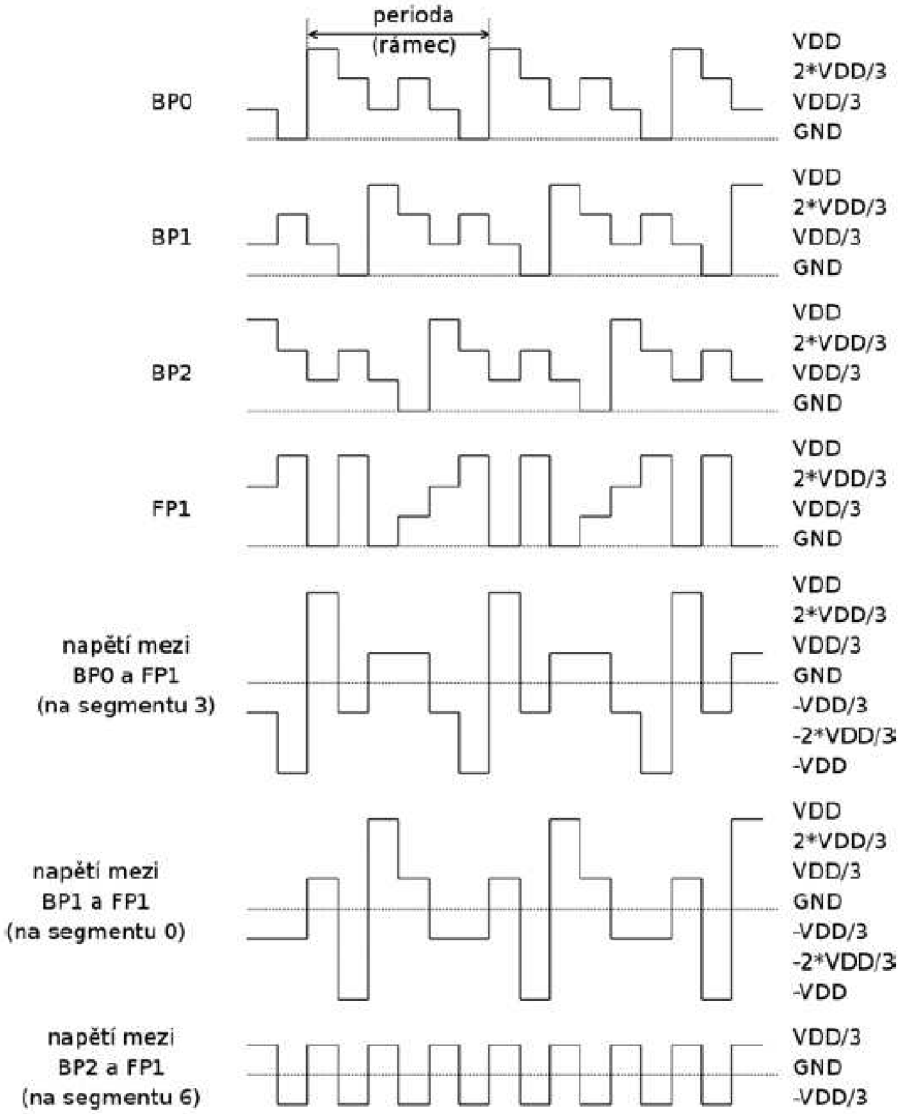
obr. 12 Multiplexované řízení 7 segmentovky

Jako příklad k metodě MUX2 jsou na obr. 13 vyobrazeny průběhy řídících napěťových signálů potřebných pro zapnutí segmentu 6 a vypnutí segmentu 4. Tyto 2 segmenty sdílejí řídící signál FP0, avšak jsou připojeny k odlišným backplane – segment 6 k BP1, segment 4 k BP0. Všimněme si, že průběhy BP-signálů jsou neměnné a že BP-signály jsou generovány tak, aby umožnily zapnutí resp. vypnutí daných segmentů příslušným FP-signálem. V našem případě je jeho průběh takový, že v jistém časovém úseku (úsek, kdy signál na BP1 přechází z VDD na VDD/2, tedy úsek 1. poloviny periody BP-signálů) slouží k řízení segmentu 6 a v jistém časovém úseku (úsek, kdy signál na BP0 přechází z VDD na VDD/2, tedy úsek 2. poloviny periody BP-signálů) k řízení segmentu 4. Na rozdíl od statické metody, která si vystačila se 2 napěťovými úrovněmi pro řízení, budeme v tomto případě potřebovat 3 napěťové úrovně (VDD, VDD/2 a GND).



obr. 13 Řídící napěťové signály pro zapojení MUX2

Jako příklad k metodě MUX3 jsou na obr. 14 vyobrazeny průběhy řídících napěťových signálů potřebných pro vypnutí segmentu 6 a zapnutí segmentů 0 a 3. Tyto 3 segmenty sdílejí řídící signál FP1, avšak jsou připojeny k odlišným backplane signálům – segment 3 k BP0, segment 0 k BP1 a segment 6 k BP2. Opět si všimněme neměnných průběhů BP-signálů. Zapnutí resp. vypnutí segmentů z tohoto příkladu je řízeno signálem FP1, který v 1. třetině periody BP-signálů zapíná segment 3, ve druhé třetině periody BP-signálů zapíná segment 0 a ve třetí třetině periody BP-signálů vypíná segment 6. Narozdíl od předchozího příkladu budeme pro řízení potřebovat 4 napěťové úrovně (VDD, 2*VDD/3, VDD/3 a GND).



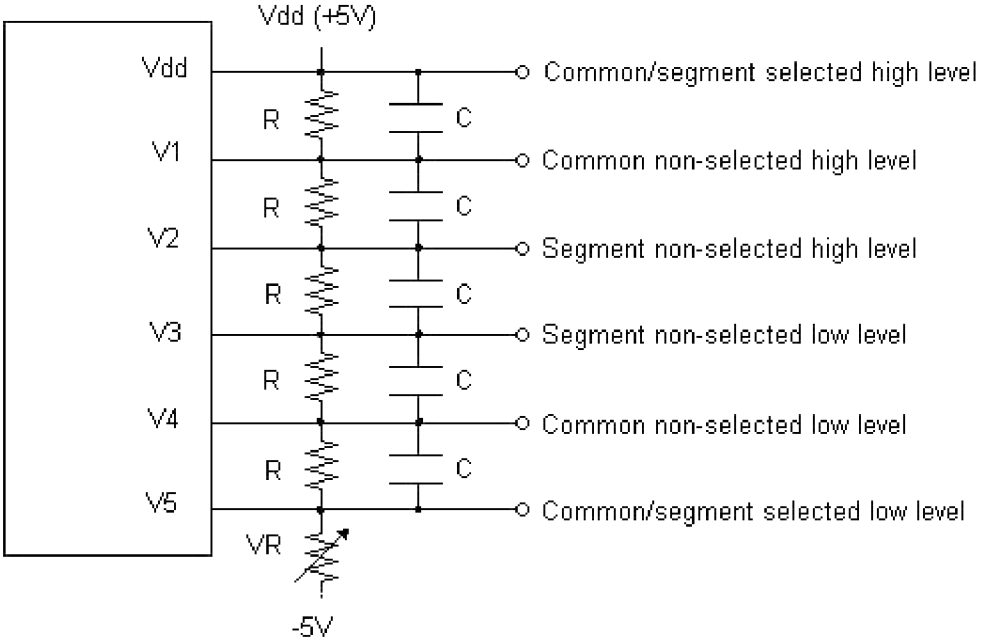
obr. 14 Řídící napěťové signály pro zapojení MUX3

Z výše uvedených příkladů je patrné, že parametry řídících signálů jsou závislé na počtu BP a FP-vývodů v rozhraní LCD. Zatímco v případě statické metody byl jedním FP-signálem řízen právě jeden segment (k = 1), LCD obsahovalo pouze jednu BP plochu na zadní destičce a k řízení postačovaly 2 napěťové úrovně, pak případě metody MUXk je jedním FP-signálem řízeno k segmentů, LCD obsahuje na zadní destičce k galvanicky oddělených BP ploch a k řízení je třeba více napěťových úrovní. Rovněž platí, že s rostoucím k se zkracuje doba vyhrazená pro řízení konkrétního segmentu. Parametr LCD, udávající jaká část doby trvání rámce je vyhrazena řízení jednoho segmentu, je označován pojmem duty cycle (ratio). Duty cycle = 1/k. Další parametr (bias) určuje poměr velikosti nejmenší možné změny řídícího napěťového signálu ku VDD. Bias = 1/(počet napěťových úrovní pro řízení - 1). Hodnoty LCD parametrů pro vybraná k, jsou k dispozici v následující tabulce.

k	1	2	3	4	6	7	11	12
Duty ratio	1	1/2	1/3	1/4	1/6	1/7	1/11	1/12
Počet napěťových úrovní	2	3	4	4	4	5	5	6
Bias	1	1/2	1/3	1/3	1/3	1/4	1/4	1/5

Tato tabulka ukazuje závislosti základních parametru LCD, při multiplexovaném řízení.

Napěťové úrovně lze získat pomocí odporového žebříku [6]. Na obr. 15 je ukázka zapojení pro Bias rovno 1/5. Pomocí jednotlivých odporů je možno nastavit jednotlivé úrovně napětí.



obr.15 Odporový žebřík pro Bias = 1/5

4 Platforma FITkit

V této kapitole budou krátce shrnuty informace o přípravku, na kterém je provedena hardwarová realizace zobecněného LCD řadiče.

Tento kit obsahuje následující komponenty [8]:

- FPGA Spartan 3 XC3S50-4PQ208C (Xilinx)
- mikrokontrolér MSP430F168 (Texas Instruments)
- USB-UART převodník FT2232C
- zvukový vstup/výstup
- konektory PS2
- konektor VGA
- konektor RS232
- DRAM 8x8mbit
- Klávesnice
- Řádkový LCD displej
- vývody k obecnému použití

Pro realizaci řadiče LCD využiji FPGA a mikrokontrolér, mezi kterými probíhá komunikace pomocí SPI. Displej, na kterém bude realizace zobecněného řadiče vyzkoušena, bude připojen na vývody, které jsou označený jako JP10.

Dále popíšu komunikaci a základní programovou knihovnu tohoto kitu.

Podrobnější informace o FITkitu jsou dostupné na webu tohoto projektu [9].

4.1 SPI

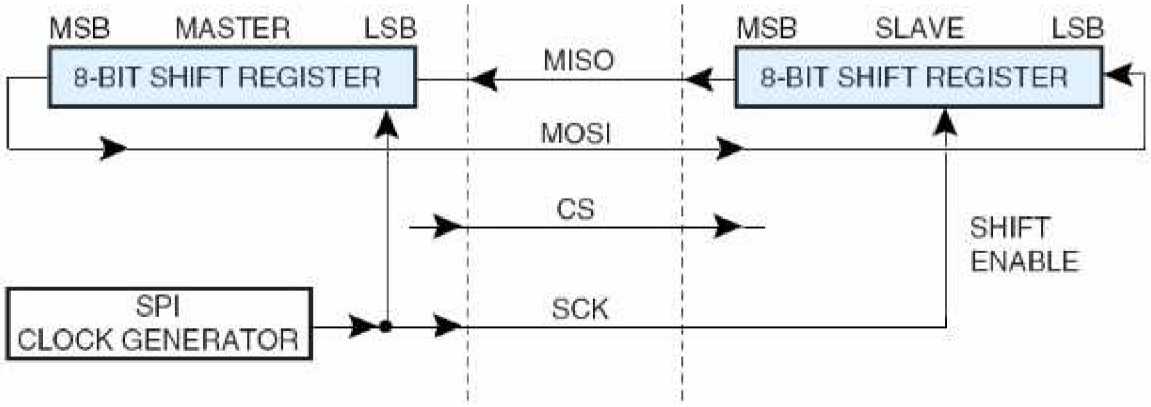
Rozhraní SPI (*Synchronous Peripheral interface*) je sériové vysokorychlostní rozhraní typu master-slave [10]. Rozhraní umožňuje obousměrnou komunikaci řízenou tzv. master (nadřazeným) zařízením s libovolným počtem slave zařízení, které jsou k této sběrnici připojeny. Rozhraní tvoří následující signály:

- **CS** (*chip select*) - signál vymezující datový rámec, aktivní v logické 0 (při přenosu); signál generuje master
- **SCK** (*clock*) - hodinový signál, který v závislosti na zvoleném režimu určuje okamžik, kdy dochází k vzorkování dat; generuje master
- **MOSI** (*master out slave in*) - data vystupující z master zařízení (data out); generuje master
- **MISO** (*master in slave out*) - data vstupující do master zařízení (data in); generuje slave

Připojení jednotlivých zařízení k jednomu SPI rozhraní je možno realizovat několika způsoby. Jednotlivá zařízení mohou sdílet veškeré signály, pak je nutné rozlišovat adresované zařízení na

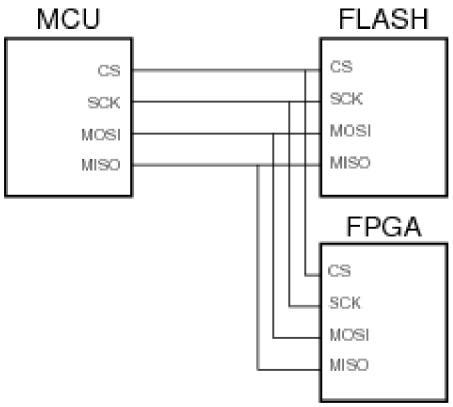
úrovni přenášených bitů a signály MISO musí být realizovány pomocí třístavových budičů. Další možností je propojit jednotlivá zařízení do řetězce, čímž se prodlouží datový rámec na k-násobek. Poslední možností je sdílet veškeré signály kromě signálu CS, který je rozveden individuálně ke každému zařízení na sběrnici.

Komunikace prostřednictvím SPI spočívá ve výměně obsahu dvou osmibitových registrů - jednoho umístěného v master a druhého v slave zařízení (obr. 16). Přenos probíhá tak, že se nejprve zapíše osmibitová hodnota do posuvného registru a tím se automaticky nastartuje přenos dat. Signál CS indikuje zahájení transakce přechodem do log 0. Master generuje hodinový signál SCK, jehož příslušná hrana generuje signál enable oběma posuvným registrům. Po osmi hodinových taktech dojde k výměně obsahu registrů v master a slave zařízení, čímž dojde současně k zápis i čtení do/z slave zařízení. Přenos z master zařízení probíhá od nejvýznamnějšího (MSB) po nejméně významný bit (LSB). Čtení ze slave zařízení postupuje opačně.



obr. 16 Komunikace pomocí SPI

SPI na FITkitu umožňuje procesoru, který je nakonfigurován v master režimu, kromě komunikace s pamětí FLASH uchovávající program pro MCU a konfiguraci FPGA, komunikovat i s periferiemi realizovanými uvnitř FPGA. Typicky se SPI používá ke komunikaci s řadiči periferií, pamětí RAM, apod. Slave zařízení FLASH a FPGA sdílejí veškeré signály (obr.17) a rozlišení adresovaného zařízení se provádí na úrovní přenášených bitů. Aby byla zajištěna bezchybná komunikace, musí mít FPGA i FLASH vzájemně disjunktní množinu operačních kódů.



obr. 17 : SPI na FITkitu

Rozhraní SPI je nakonfigurováno tak, že k vzorkování signálů MOSI a MISO dochází při vzestupné hraně hodinového signálu SCK, přičemž neaktivní úroveň SCK je logická 1. Konfiguraci SPI a jednotlivé funkce je možné nalézt v souboru SPI.c knihovny libfitkit. Frekvence hodinového signálu SCK je určena hodnotou uloženou v registrovém páru U1BR1a U1BR0. Tyto registry specifikují dělitel hodinového signálu SMCLK, který má kmitočet 7.3728MHz (generován z krystalu X2). Protože nejmenší možný dělitel je roven 2, je maximální možná frekvence SCK rovna 3.6864MHz, což odpovídá přenosové rychlosti 460.8 kB/s.

4.2 Knihovna libfitkit

Knihovna libfitkit zastřešuje základní funkce pro použití MCU a komunikaci MCU s komponentami na FITKitu nebo PC [11].

Dále knihovna poskytuje základní platformu pro použití FITKitu a poskytuje následující základní činnosti:

- **Komunikaci s terminálem na PC přes USB a UART MCU** - V knihovně je implementovaná komunikace přes sériové rozhraní, které je připojeno přímo na port B čipu FT2232C (COM s vyšší hodnotou). Nad sériovou komunikací je implementován jednoduchý příkazový interpret, do kterého lze zadávat příkazy pro MCU, které knihovna zpracuje. Pokud se jedná o příkaz knihovny, knihovna daný příkaz vykoná. V opačném případě se příkaz zadaný do příkazového řádku pošle funkci USER_CMD_Decode, kterou si uživatel musí nadefinovat. Dále je možné pomocí sériové linky přenášet soubory mezi terminálem a MCU pomocí protokolu xmodem.
- **Konfigurace FPGA** - Základní úkol knihovny je nahrávání konfigurace do FPGA. Po inicializaci knihovny se automaticky nahraje konfigurace pro FPGA uložená ve FLASH paměti do FPGA. Knihovna dále umožňuje pomocí příkazů nahrávat

konfiguraci pro FPGA z PC přímo do FPGA, z PC do FLASH paměti a z FLASH paměti do FPGA.

- **Podpora SPI rozhraní** pro komunikaci s FLASH pamětí a FPGA

5 Návrh zobecněného LCD řadiče

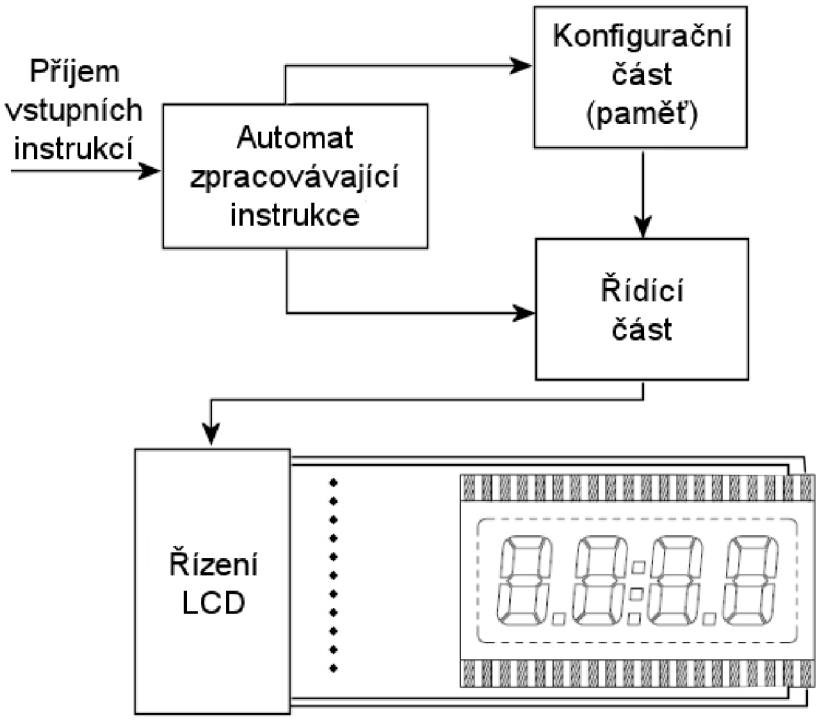
V této kapitole se věnuji návrhu zobecněného řadiče. Který je implementován v jazyce VHDL.

Jelikož se jedná o zobecněný řadič, musíme parametry pro konkrétní vybraný displej zapsat do jeho konfigurace. Podle doporučení vedoucího tyto parametry zadávám do přímo do VHDL souboru, ve kterém je řešen celý LCD řadič. Díky těmto parametrům, které je nutno zadat pro konkrétní typ LCD, můžeme po syntéze tento řadič použít.

Mezi nastavitelné parametry patří např. frekvence, kterou je LCD displej řízen, počet pinů pouzdra, počet jednotlivých segmentů v zobrazitelném znaku, počet znaků na displeji.
Mezi hlavní konfigurační údaje patří pole pinů. Toto pole udává, který pin pouzdra je spojen s daným znakem. A dále zde máme pole (paměť), kde máme uloženy informace o jednotlivých znacích.

Celý řadič se tedy skládá z těchto částí (obr. 18):

- **automatu**, který dekóduje vstupní instrukce a podle daného stavu automatu bude prováděná daná akce. Jako např. zápis do konfigurační paměti, posun kurzoru a další instrukce
- **konfigurační část**, zde je uložena konfigurace pro připojený LCD displej
- **řídící část**, ta má na starosti počítat aktuální pozici kurzoru a ovládání paměti, ve které je uložen aktuální stav zobrazovaných znaků na displeji
- jako poslední část je část **řízení LCD**, který se stará o střídavé napětí na vstupech LCD displeje



obr. 18 Blokové schéma zobecněného LCD řadiče

5.1 Konfigurace

V této podkapitole se zabývám konfigurací parametrů řadiče. Implementace je provedena v jazyce VHDL. Tuto část kódu lze nalézt na přiloženém CD, cesta k souboru je X:\zdrojove_kody\top\display_mag.vhd.

5.1.1 Zápis informací o připojených pinech

K zápisu těchto informací je použito pole s označením pin.

```
type pin_konfig is array(0 to (pocet_segznak*pocet_seg)) of
std_logic_vector(7 downto 0);
signal pin : pin_konfig := (
X"05",X"06",X"07",X"08",X"22",X"23",X"24",X"25",
X"09",X"0A",X"0B",X"0C",X"1D",X"1E",X"1F",X"20",
X"0D",X"0E",X"0F",X"10",X"18",X"19",X"1A",X"1B",
X"11",X"12",X"13",X"1C",X"14",X"15",X"16",X"17",
others => "00000000"
);
```

Počet položek v tomto poli je vypočítáno jako součin počtu pozic pro znaky (pocet_segznak) a počtu segmentů pro daný znak (pocet_seg). Pokud se jedná o displej bez multiplexovaného řízení je zde, v pořadí, které si určí upravovatel kódu, zapsána přímo hodnota daného pinu. Toto je i můj případ u realizace. Pokud má displej multiplexované řízení pak je hodnota těchto pinů uvedena na prvním místě (na uvedeném příkladě se jedná o piny 1, 2, 3 zapsane v hexadecimálním tvaru).

Př.:

```
type pin_konfig is array(0 to (pocet_segznak*pocet_seg)) of
std_logic_vector(7 downto 0);
signal pin : pin_konfig := (
X"01",X"04",X"05",X"06",
X"02",X"04",X"05",X"06",
X"03",X"04",X"05",X"06",
others => "00000000"
```

Tyto hodnoty se pak využijí pro zápis hodnot na správné místo v paměti zobrazující aktuální stav paměti výstupných pinů. Pro každý řídící multiplexovaný signál se pak v paměti uloží aktuální stav na výstupních pinech. Při výsledném zobrazování se pak v cyklu přepíná mezi jednotlivými řídícími signály. Přesný popis zadávání hodnot do tohoto pole je v kapitole 6.

5.1.2 Pole znaků

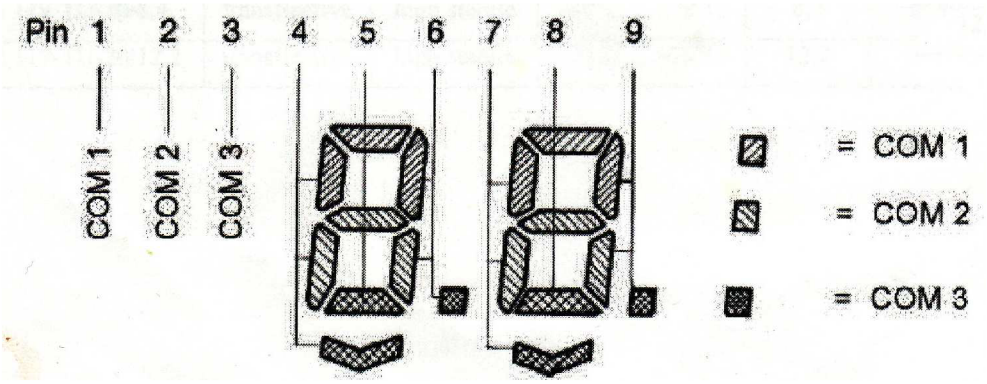
Toto pole obsahuje informace, které segmenty jsou pro daný vstup aktivní. Velikost pole se počítá jako součin počtu segmentů ve znaku a počtem řídících multiplexovaných signálů. Jelikož každá hodnota reprezentuje nastavení pinů pouze pro jeden multiplex, je jeden znak charakterizován počtem hodnot shodným s počtem použitých multiplexů.

```
type CHAR_RAM is array(0 to (pocet_znakseg*pocet_mux)-1) of
     std_logic_vector(0 to pocet_segznak-1);
signal pamet_znaku : CHAR_RAM := (
"11101110", -- znak 0
"00101000", -- znak 1
"11001101", -- 2
"01101101", -- 3
"00101011", -- 4
"01100111", -- 5
"11100111", -- 6
"00101100", -- 7
"11101111", -- 8
"01101111", -- 9
"00010000", -- DP, v pripade 4 znaku COL
others => "00000000");
```

V případě, že máme displej s přímým řízením, zapisujeme přímo bitově, které segmenty jsou pro daný znak aktivní.

V případě multiplexovaného řízení jsou zde vždy informace, kterého řídícího signálu se tyto bitové hodnoty týkají, a dále označené segmenty, které jsou v dané chvíli aktivní. Pro přesnější vysvětlení uvedu příklad.

Máme displej se dvěma znaky, a je použito triplex řízení (obr. 19).



obr. 19 LCD s triplex řízením

V tabulce o připojených pinech nastavím celkem 12 hodnot v pořadí řídícího signálu (COMx) a piny patřící k danému znaku. Takže hodnoty pro jeden znak pak budou vypadat takto:

```
1,4,5,6,
2,4,5,6,
3,4,5,6,
```

Hodnota 1 říká, že ovládáme pin 1, a jelikož je na prvním místěn a máme multiplexované řízení (pocet_mux > 1) víme, že tento pin je typu COM. Následující 3 hodnoty (podle proměnné pocet_mux) odpovídají pinům ovládající první znak (jde o hodnoty 4, 5, 6). V dalších dvou řádcích jsou zapsány hodnoty pinů pro 2 zbývající multiplexy, a to stejným způsobem jako na předchozí řádku konfigurace pinů.

Hodnoty v následující tabulce představují aktivitu pinu pro znak čísla 7:

```
0011
0001
0000
```

Jelikož se jedná o multiplexované řízení (pocet_mux > 1) první hodnota bude vždy nula, jedná se totiž o signál COM, který je aktivní právě v hodnotě nula. Následující tri hodnoty ukazují výběr segmentů v daném multiplexu (pro zápis tohoto znaku na pozici 0 to znamená, že: 0 – je pro segmenty ovládané pinem 4, 1 – je pro segmenty ovládané pinem 5, 1 – je pro segmenty ovládané pinem 6). Další dvě hodnoty (0001 a 0000) jsou pro další dva mutliplexy, způsob vzniku těchto hodnot je stejný jako u předcházející hodnoty.

Výsledný VHDL kód konfigurace pinů a paměti znaků pak vypadají následovně:

```
type pin_konfig is array(0 to (pocet_segznak*pocet_seg)) of
std_logic_vector(7 downto 0);
signal pin : pin_konfig := (
X"01",X"04",X"05",X"06",
X"02",X"04",X"05",X"06",
X"03",X"04",X"05",X"06",
others => "00000000"
);
type CHAR_RAM is array(0 to (pocet_znakseg*pocet_mux)-1) of
     std_logic_vector(0 to pocet_segznak-1);
signal pamet_znaku : CHAR_RAM := (
"0011", -- znak 0
```

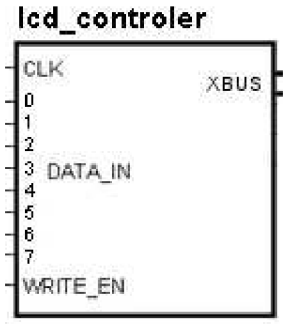
```
"0001", -- znak 1
"0000", -- 2
others => "1000");
```

Takto nastavené tabulky se pak použijí pro výpočet (procedura výstup), které piny jsou pro daný řídící signál aktivní. Tento výpočet mapuje požadované znaky do pole reprezentující nastavení pinů pro daný LCD displej.

Další možnosti nastavení pro další displeje jsou uvedeny v kapitole 6, kde se zabývám simulací a syntézou zobecněného řadiče.

5.2 Kódování instrukcí

Pro řadič jsem zvolil vstupní datovou sběrnici o šířce 8 bitů (obr. 20), nejvyšší bit je použit na řízení, zda jde zobrazení znaku na displeji nebo je na datové sběrnici příkaz. To znamená, že máme možnost zobrazovat až 128 různých znaků.



obr. 20 Blokové schéma lcd řadiče

V případě, že jde o příkaz, je zvolen nějaký kód, jelikož se nejedná o složité instrukce, ve kterých by byla obsažena ještě další informace, nejedna se o složité kódování. V případě, že potřebujeme příkaz, který má další údaje, je to řešeno pomocí stavu automatu, kdy se příkazem dostaneme do stavu, kde očekáváme další data na vstupní datové sběrnici. Jde například o instrukci zadávání nového znaku do paměti znaků, kdy jde prvně instrukce a za ní další dvě 8 bitové hodnoty představující adresu do paměti znaků a konfiguraci segmentů pro daný znak. V následující tabulce je ukázka instrukcí:

0xxxxxxx	Datová instrukce - zápis znaku na displej, znak určen 7 bity
11000001	Řídící instrukce – posun kurzoru o pozici doprava
11000010	Řídící instrukce – posun kurzoru o pozici doleva
11000000	Řídící instrukce – smazání displeje
11000011	Řídící instrukce – zápis nového znaku do paměti znaků

Pokud bychom potřebovali přenášet hodnoty větší než 128, musíme zvětšit datovou sběrnici. Jelikož implementace počítá s propojovacím systémem FITkitu, který používá ke komunikaci mezi mikrokontrolérem a FPGA sériový přenos SPI, lze úpravou parametrů entity SPI_adc docílit širší sběrnice. Nastaveni této entity se provádí v souboru top_level.vhd, který se stará o propojení všech potřebných komponent.

5.3 Popis hlavních částí VHDL kódu

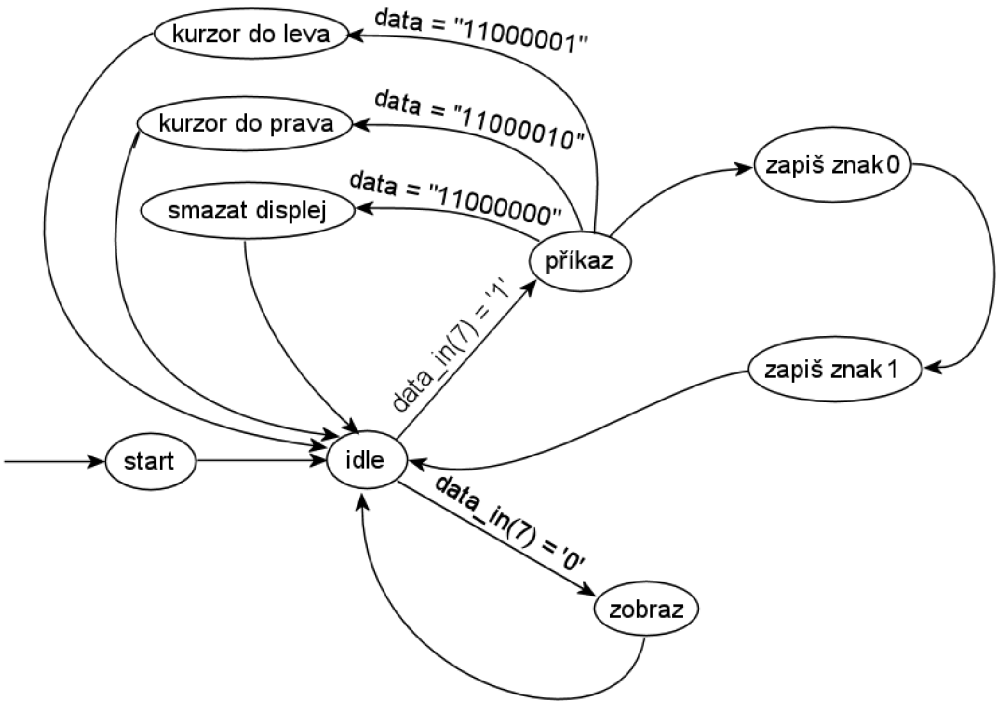
V této podkapitole se rozepisuji o implementaci jednotlivých bloků LCD řadiče. Kapitola 5.3.1 popisuje implementaci konečného automatu, který má na starosti vnitřní řízení řadiče. A to především vyhodnocení dat na vstupním portu, pomocí kterých jsou zadávány jednotlivé příkazy řadiči.

V následující kapitole je pak popsán způsob mapování znaků do paměti obsahující konfiguraci výstupních pinů.

Dále je zde kapitola 5.3.3 popisující blok zajišťují poslání konfigurace výstupních pinů z paměti řadiče na výstupní piny označené jako XBUS. XBUS je výstupní sběrnice entity Display_base2 (soubor display_mag.vhd).

5.3.1 Automat pro zpracování instrukcí

Na obr. 21 je zobrazen zde popisovaný automat, pro zvýšení přehlednosti byly vynechány čekací stavy.



obr. 21 Automat pro zpracování instrukcí

V jednotlivých stavech jsou, jak je uvedeno dále v popisu kódu, vykonávány operace modifikující řídící signály.

Hlavní řízení se skládá z 8 stavů, z nich některé jsou pouze tzv. čekací, tzn. že je zde zavedeno časové zpoždění pro správné nastavení vstupních signálů (jde o stavy idle0,idle1, zapis_znak0). V VHDL je automat implementován následovně:

```
process(clk)
   begin
    if clk'event and clk = '1' then
      case d_state is
```

V případě že jsme spustili běh řadiče, dojde k smazání signálu data. Jelikož se jedná o univerzální řadič, a není dopředu známa datová šířka daného signálu, je využíváno funkce XOR k vynulování daného signálu.

```
when start => d_state <= idle0;
              data <= data xor data;
when idle0 => d_state <= idle1;
when idle1 => d_state <= idle2;
```

V tomto stavu kontroluji, zda jde o instrukci pro zápis znaku, nebo o instrukce řídící LCD. Tyto dva stavy jsou rozlišeny podle nejvyššího bitu vstupního signálu data_in.

```
when idle2 =>
       if WRITE_EN = '1' then
             if data_in(7) = '1' then d_state <= prikaz;
                   else d_state <= zobraz;
             end if;
             data <= data_in;
       end if;
```

Ve stavu příkaz kontroluji, o jaký řídící příkaz se jedná.

```
when prikaz =>
```

Signál data s hodnotou “11000001“ říká, že se jedná o příkaz posun kurzoru doprava.

```
if data = "11000001" then
   kurzor <= kurzor + 1;
```

```
      d_state <= idle0;
   end if;
```

Signál data s hodnotou “11000010“ říká, že se jedná o příkaz posun kurzoru doleva.

```
   if data = "11000010" then
      kurzor <= kurzor - 1;
      d_state <= idle0;
   end if;
```

Signál data s hodnotou “11000000“ zajistí smazání výstupního pole lcd, pomocí cyklu.

```
   if data = "11000000" then
       for ii in 0 to pocet_mux-1 loop
          lcd(ii) <= lcd(ii) xor lcd(ii);
       end loop;
       kurzor <= kurzor xor kurzor;
       d_state <= idle0;
   end if;
```

Signál data s hodnotou “11000011“ zajišťuje zápis nového znaku do paměti znaků.

```
   if data = "11000011" then
       d_state <= zapis_znak0;
   end if;
```

Stav zobraz zajistí zobrazení znaku na LCD pomocí procedury vystup. Jelikož hodnota znaku je po SPI poslána podle ASCII tabulky je nutno tuto hodnotu posunout o číslo 48, které v ASCII vyjadřuje znak 1.

```
  when zobraz =>
    vystup(data - 48);
    kurzor <= kurzor + 1;
    d_state <= idle0;
```

Načtení čísla pozice pro paměť znaků nového znaku

```
            when zapis_znak0 =>
              if WRITE_EN = '1' then
                  data <= data_in;
                  d_state <= zapis_znak1;
              end if;
```

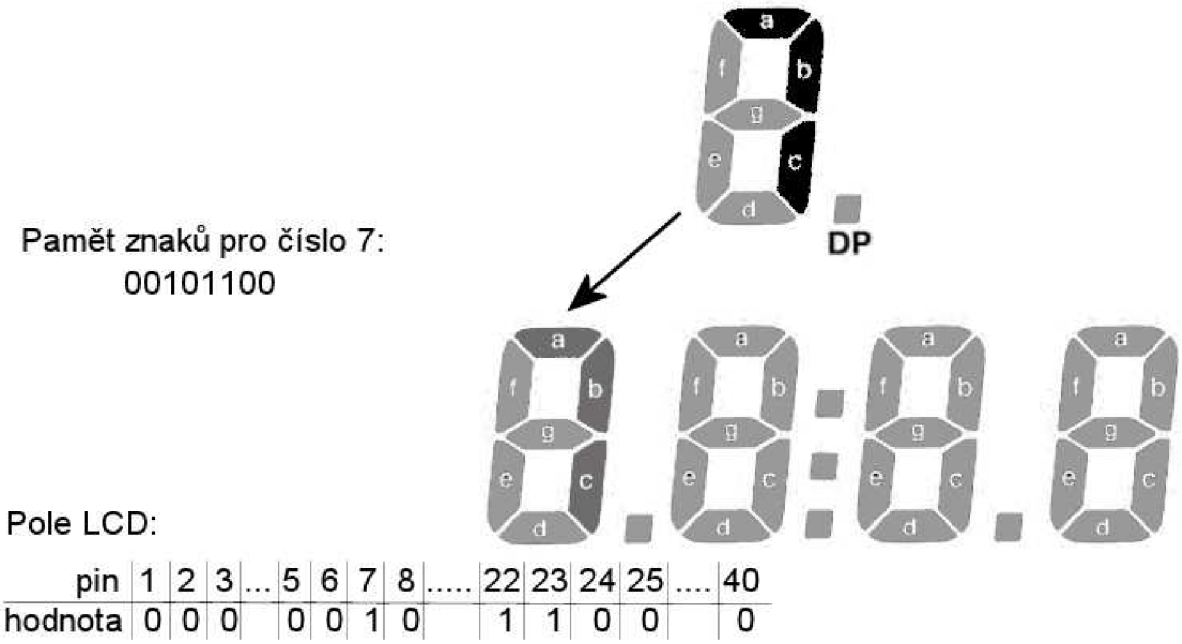
Zápis nového znaku, na předem načtenou pozici, s hodnotou načtenou v signálu data_in.

```
            when zapis_znak1 =>
              if WRITE_EN = '1' then
                  pamet_znaku(conv_integer(data))<= data_in;
                  d_state <= idle0;
              end if;
            end case;
        end if;
       end process;
```

V případě potřeby více znaků než dovoluje ASCII, lze napsat novou funkci pro mikrokontroler, a to takovou aby posílal pouze hodnotu ukazující do pole znaků. Velikost této hodnoty pak bude závislá na velikosti vstupní sběrnice. Dále bude nutno odstranit konstantu 48 ve volaní procedury výpis, která je obsažena v VHDL kódu LCD řadiče.

5.3.2 Mapování znaků do výstupní paměti

Procedura zajištující mapování znaků do výstupní paměti se jmenuje vypis a najdeme ji v souboru display_mag.vhd. Tato procedura zajišťuje mapování požadovaného znaku z paměti znaku do pole reprezentující výstupní nastavení pinů, kdy z pole pinů je načtena hodnota pinu, která náleží dané pozici znaku na displeji a aktuálního segmentu a z pole znaků je načtena logická hodnota odpovídající právě zpracovávaného segmentu daného znaku (viz. obr. 22).



obr. 22 Ukázka mapování znaku do výstupního pole

```
procedure vystup (znak : std_logic_vector(7 downto 0)) IS
     variable b : integer;
     variable c : integer;
     variable aa : integer;
     variable prvni_mux : integer;
  begin
```

V cyklu projdeme mapování znaku přes všechny multiplexované signály.

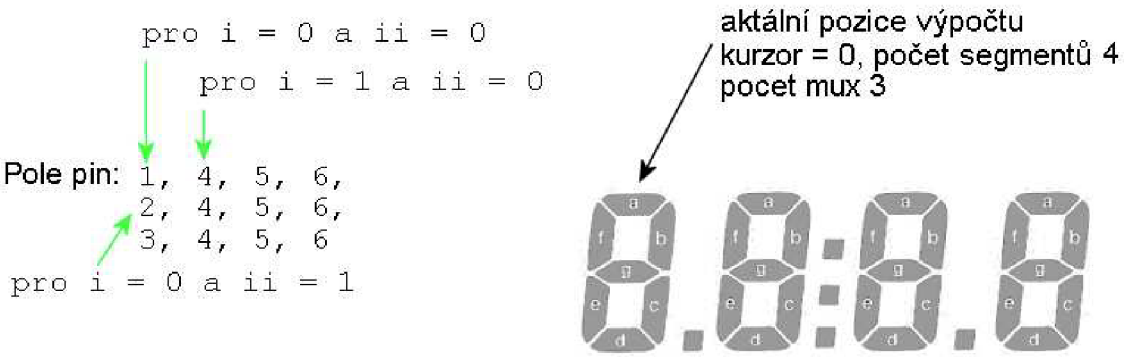
```
     for ii in 0 to pocet_mux-1 loop
```

A všechny piny.

```
       for i in 0 to pocet_segznak-1 loop
```

Pomocný výpočet určující pozici v poli pin (obr. 23).

```
        aa := (conv_integer(kurzor)*pocet_segznak)+i+
              pocet_segznak*ii;
```



obr. 23 Ilustrace průběhu výpočtu pozice v poli pin

Pomocný výpočet určení aktuální pozice v poli pamet_znaku.

```
b := (conv_integer(znak)*pocet_mux)+ii;
```

Načtení hodnoty pinu z pole pin, který v tuto chvíli modifikujeme.

```
c := conv_integer(pin(aa))-1;
```

V případě, že chceme nové segmenty pouze přidat (nikoliv přepsat znak na dané pozici) použijeme logickou funkci OR.

```
if soucet then lcd(ii)(c) <= lcd(ii)(c) or
                              pamet_znaku(b)(i);
```

Zápis nového znaku do výstupního pole.

```
else lcd(ii)(c) <= pamet_znaku(b)(i);
end if;
  end loop;
 end loop;
end procedure;
```

5.3.3 Řízení LCD

Proces zajišťující dodání nastavení výstupních pinů se správnou frekvenci a pro správný multiplex. V případě multiplexovaného řízení je ještě nutné, pomocí další elektroniky, zajistit aby aktuální multiplex měl nejvyšší úroveň napětí a ostatní multiplexy nižší, takovou, která jim přísluší.

```
process(lcd, cnt(frekvence))
   begin
      if (cnt(frekvence)='1') then
```

```
        XBUS(pocet_pinu-1 downto 0) <=
            lcd(conv_integer(aktual_mux));
    else XBUS(pocet_pinu-1 downto 0) <= resetlcd;
    end if;
end process;
```

6 Simulace a syntéza řadičů

Tato kapitola je věnována simulacemi a výsledky syntézy pro některé displeje. U simulace jsou ukázány změny hodnot signálů navrhovaného LCD řadiče. A výsledek syntézy ukazuje využití bloků FPGA Spartan 3, který je využit v platformě FITkit. Pro simulaci jsou vybrány instrukce zobrazení znaku, případně posuny kurzoru.

6.1 LCD displej s přímým řízením

V tomto bodě jsem syntetizoval a simuloval s nastavením, které je použito v hardwarové realizaci. Jedná se o 7. segmentový displej s přímím řízením [12] (obr. 24).

obr. 24 - 7. segmentový displej

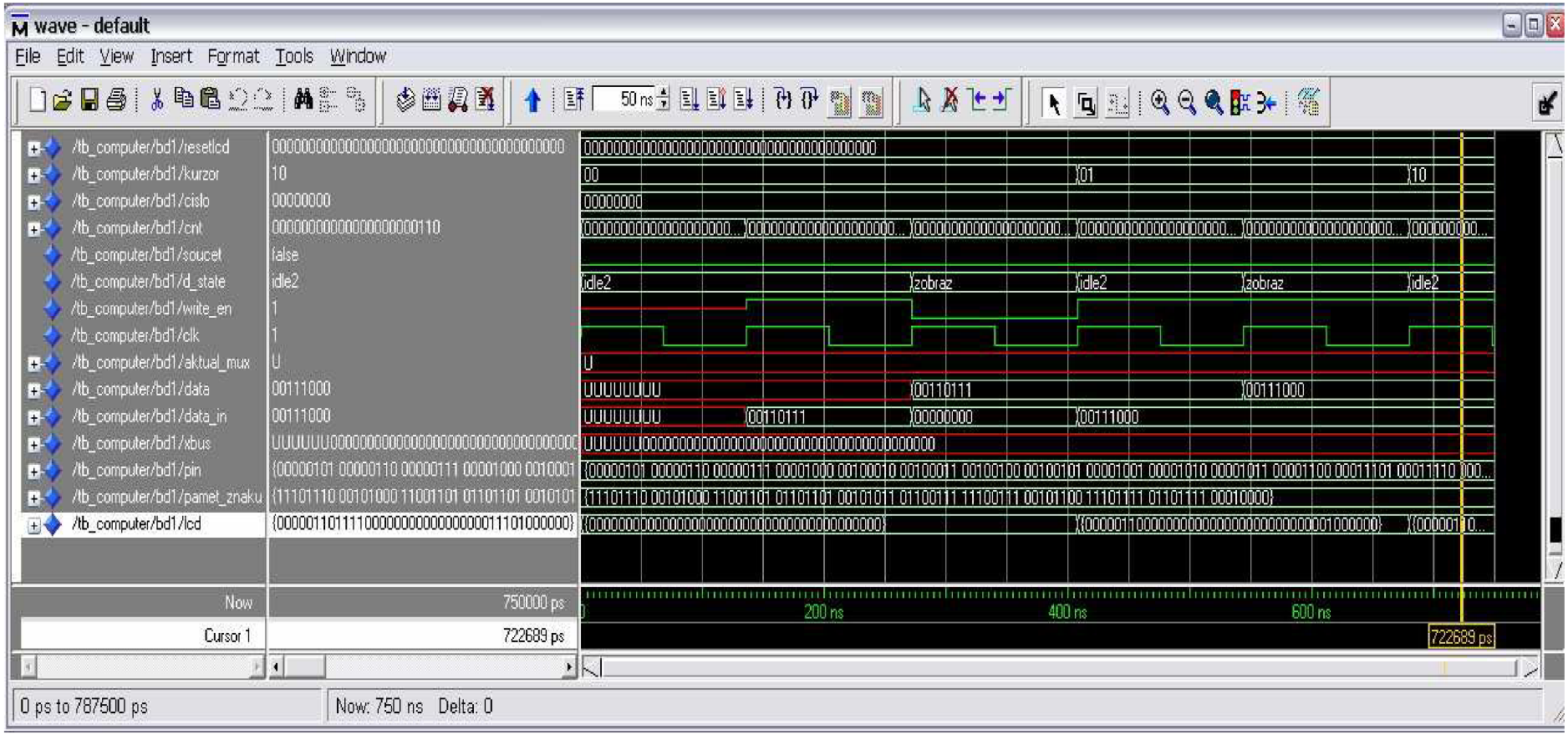
Nastavení pro tento displej je následující:

```
constant frekvence : integer := 16; -- pocet bitu pro deleni
                zakladni frekvence SMCLK
constant pocet_pinu : integer := 40; -- pocet pinu displeje
constant pocet_bitu : integer := 6; -- pocet potrebnych bitu pro
                zapis ktery pin je aktivni
constant pocet_segznak : integer := 8; -- pocet segmentu urcujici
                znak
constant pocet_seg : integer := 4; -- pocet segmentu
constant bitu_seg : integer := 2; -- pocet bitu urcujici segment
constant pocet_znakseg : integer := 11; -- pocet znaku v segmentu
constant pocet_mux : integer := 1;
```

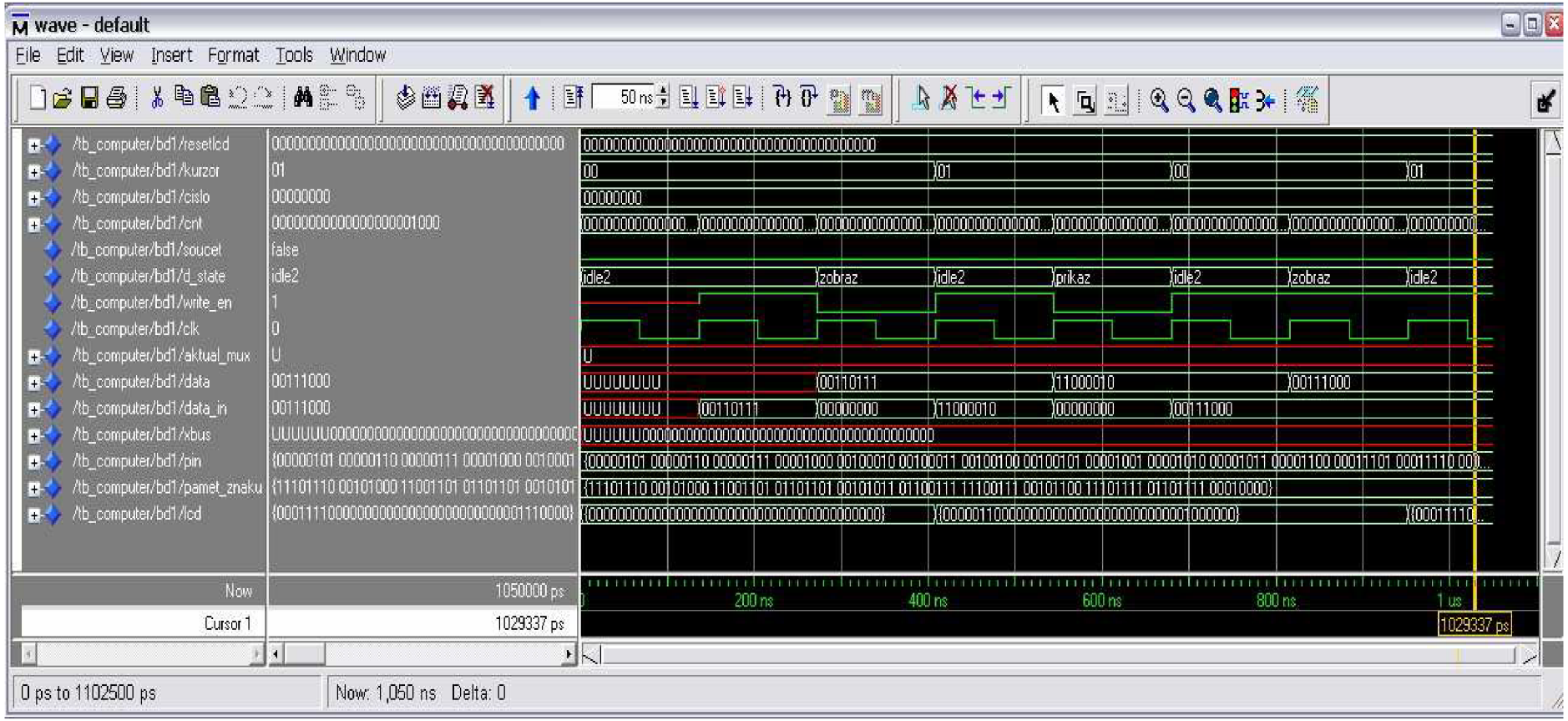
```
constant bitu_mux : integer := 1; -- pocet bitu reprezentujici cislo
                mux

type pin_konfig is array(0 to (pocet_segznak*pocet_seg)) of
                std_logic_vector(7 downto 0);
signal pin : pin_konfig := (
X"05",X"06",X"07",X"08",X"22",X"23",X"24",X"25", -- piny ovladajici
                prvni znak, v poradi e d c dp b a f g
X"09",X"0A",X"0B",X"0C",X"1D",X"1E",X"1F",X"20", -- piny ovladajici
                druhy znak znak,
X"0D",X"0E",X"0F",X"10",X"18",X"19",X"1A",X"1B", -- piny ovladajici
                treti znak,
X"11",X"12",X"13",X"1C",X"14",X"15",X"16",X"17", -- piny ovladajici
                ctvrty znak, v poradi e d c COL b a f g
others => "00000000");
type CHAR_RAM is array(0 to (pocet_znakseg*pocet_mux)-1) of
                std_logic_vector(0 to pocet_segznak-1);
signal pamet_znaku : CHAR_RAM := (
"11101110", -- znak 0
"00101000", -- znak 1
"11001101", -- 2
"01101101", -- 3
"00101011", -- 4
"01100111", -- 5
"11100111", -- 6
"00101100", -- 7
"11101111", -- 8
"01101111", -- 9
"00010000", -- DP, v pripade 4 znaku COL
others => "00000000");
```

Simulace:



obr. 25 Simulace řadiče s přímím řízením, zobrazení znaku.



obr. 26 Simulace řadiče s přímím řízením, zobrazeni znaku, posun doleva, a zobrazení jiného znaku

Frekvence nastavená v testbech.vhd odpovídá frekvenci SMCLK FITkitu, což odpovídá 7.3728 MHz.

První simulace (obr. 25) ukazuje zobrazení čísel 78 na displej. Po skončení stavu zobraz, se na výstupních pinech nastaví požadované úrovně. Znak čísla 7 je na výstupní piny zapsán v čase 410ns, dále je ve stejném čase změněna hodnota signálu kurzor, který již ukazuje na druhou pozici na displeji. Za další hodinový cyklus jsme opět ve stavu zobraz, který bude pomocí procedury výstup mapovat číslo 8. Na LCD se toto číslo zobrazí v čase 680 ns

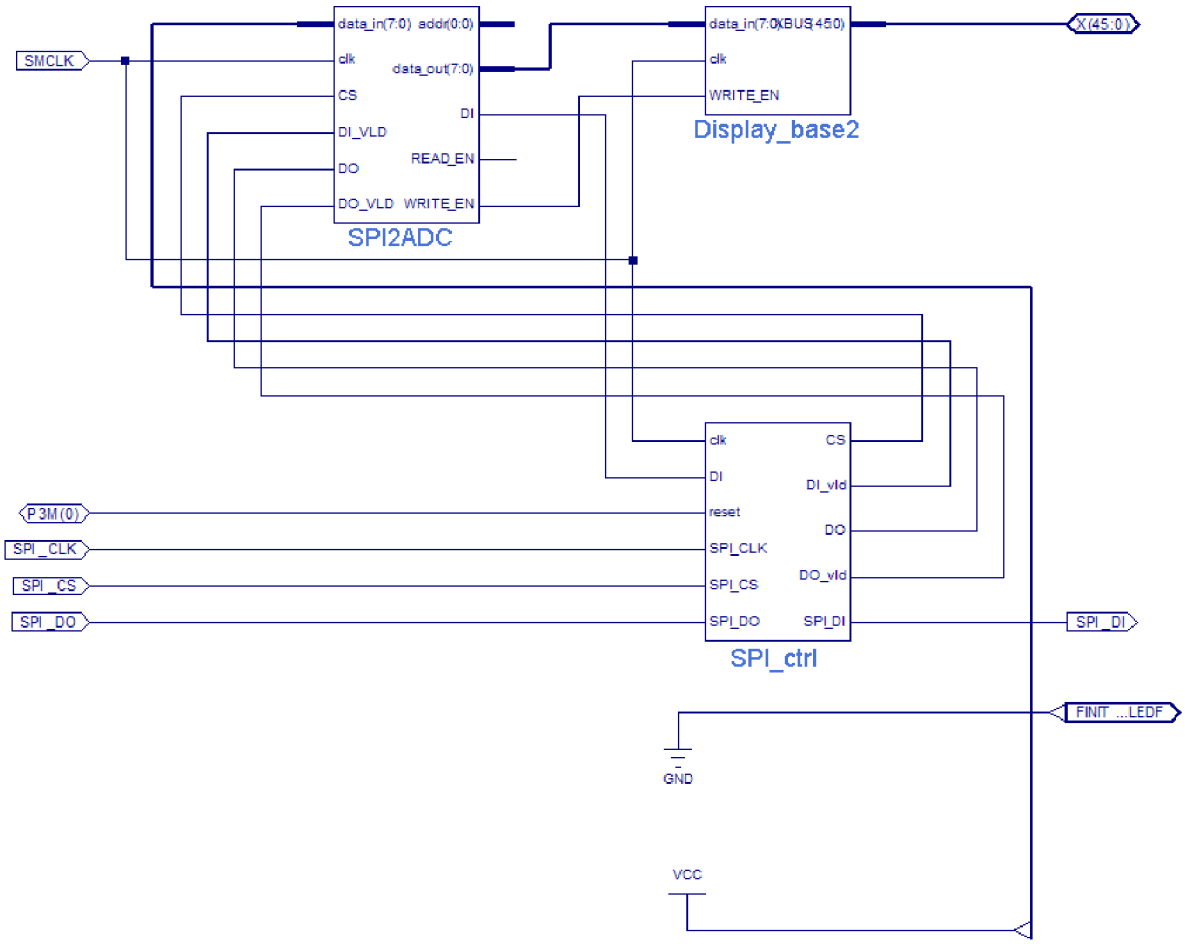
Druhá simulace (obr. 26) ukazuje zápis čísla 7, posun kurzoru o jednu pozici zpět vlevo, a zápis nového čísla 8. Číslo 7 je zobrazeno v čase 410ns, stav příkaz, ve kterém budeme posunovat kurzor do leva je aktivní v čase 540 – 680. Po tomto čase je kurzor nastaven zpět na hodnotu 00, která odpovídá první pozici na LCD. Dále je zobrazen znak 8 tzn., že dojde k přepsání předešlého čísla 7 na displeji. Celá tato simulace trvala 950 ns

Výsledky syntézy, která byla provedena v Xilinx ISE jsou následující:

Souhrné informace o využití FPGA			
Využití logických bloků	Použito	Dostupných	Využití
Počet slice klopných obvodů	129	1536	8%
Počet 4 vstupových LUT	231	1536	15%
Přidělené logické bloky			
Počet obsazených slice	173	768	22%
Počet slice obsahující pouze spojenou logiku	173	173	100%
Počet slice obsahující nespojenou logiku	0	173	0%
Celkový počet 4 vstupových LUT	263	1536	17%
Počet použitých jako logika	231		
Počet použitých jako cesta skrz	16		
Počet použitých jako dvouportová RAM	16		
Počet spojených vstupně výstupních bloku (IOB)	54	124	43%
IOB klopné obvody	3		
Počet GCLKs	1	8	12%
Celkový ekvivalent počtu hradel návrhu	3652		
Další počet JTAG hradel pro IOB	2592		

Pozn.: slice je označení pro menší logické elementy

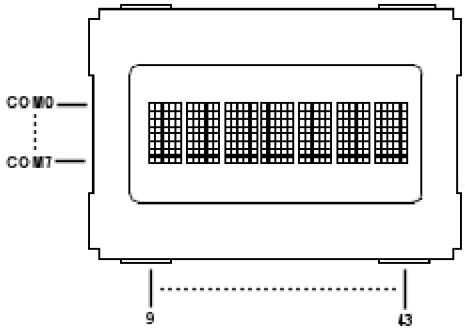
Na obr. 27 je ukázka blokového schématu celého LCD řadiče, tak jak je použito při hardwerové realizaci.



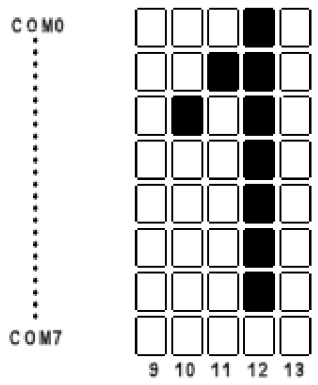
obr. 27 Blokové schéma LCD řadiče

6.2 LCD displej 7x1

Z důvodu nedostatku pinů na dostupném FPGA obvodu, jsem zvolil hypoteticky znakový displej 7x1 (obr. 28). Tento displej je ovládán signály COM0 až COM7 (multiplexovaný signál určující mikrořádek) a signály 9 až 43 (tyto signály určují sloupec). Na obr. 29 je ukázka obsazení pinů pro znak 1 na prvním místě tohoto dipleje.



obr. 28 Náčrt vývodů pro displej 7x1



obr. 29 Znak 1 ve výstupním poli

Nastavení konstant:

počet bitů pro dělení základní frekvence SMCLK, pro zajištění správné frekvence hodin clk

```
constant frekvence : integer := 16;
```

počet pinů displeje

```
constant pocet_pinu : integer := 44;
```

počet potřebných bitů pro zápis, který pin je aktivní, pro tabulku pinů

```
constant pocet_bitu : integer := 6;
```

počet segmentů určující znak, číslo je vypočítáno jako součet poučtu sloupců na znak + 1 ovládací signál (COM)

```
constant pocet_segznak : integer := 6;
```

počet znaků (segmentů) na displeji

```
constant pocet_seg : integer := 7;
```

počet bitů určující segment

```
constant bitu_seg : integer := 3;
```

počet znaků v segmentu

```
constant pocet_znakseg : integer := 128;
```

počet řídících signálů

```
constant pocet_mux : integer := 8;
```

počet bitů reprezentující číslo mux

```
constant bitu_mux : integer := 3;
```

Podle obr. 29 je zde ukázka nastavení pinů pro první pole znaku. A to tak, že na prvním místě je hodnota pinu, kterému patří daný multiplex, a dále je uvedeno 5 pinu sloupců, tyto sloupce patří právě prvnímu poli znaků.

```
type pin_konfig is array(0 to (pocet_segznak*pocet_seg*pocet_mux))
of std_logic_vector(7 downto 0);
signal pin : pin_konfig := (
X"01",X"09",X"0A",X"0B",X"0C",X"0D",
X"02",X"09",X"0A",X"0B",X"0C",X"0D",
X"03",X"09",X"0A",X"0B",X"0C",X"0D",
X"04",X"09",X"0A",X"0B",X"0C",X"0D",
X"05",X"09",X"0A",X"0B",X"0C",X"0D",
X"06",X"09",X"0A",X"0B",X"0C",X"0D",
X"07",X"09",X"0A",X"0B",X"0C",X"0D",
X"08",X"09",X"0A",X"0B",X"0C",X"0D",
others => "00000000"
);
```

Zde je ukázka nastavení paměti znaku, pro znak 1, kdy nejvyšší bit je vždy jedna, a to z důvodu použití multiplexu.

```
type CHAR_RAM is array(0 to (pocet_znakseg*pocet_mux)-1) of
std_logic_vector(0 to pocet_segznak-1);
signal pamet_znaku : CHAR_RAM := (
"100010",
"100110",
"101010",
"100010",
"100010",
"100010",
"100010",
"100000",
others => "000000");
```

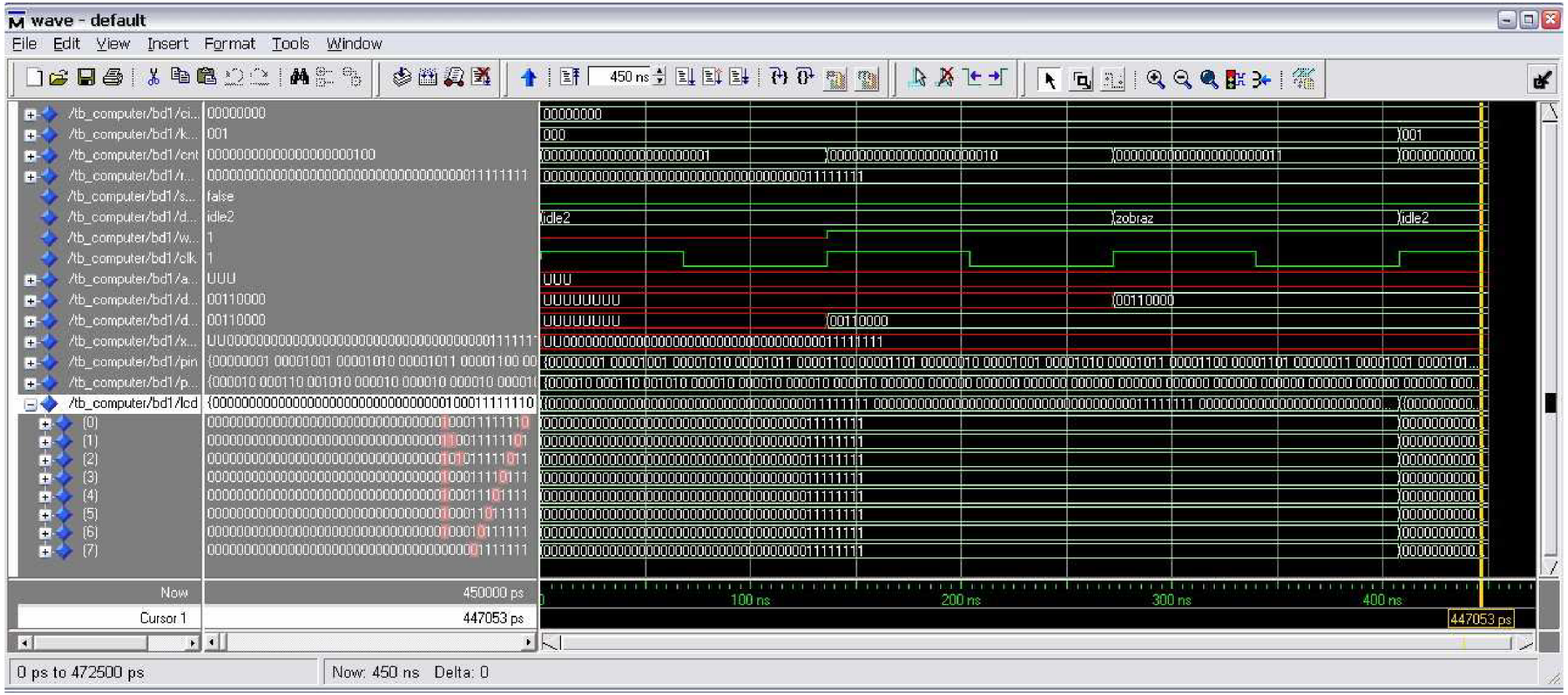
Jak je vidět na obr. 30 (na následující straně), je tento znak namapován do výstupního pole. Pro názornost je na obrázku tento znak zvýrazněn. Znak je zrcadlově obrácen z důvodu, že výstupní pole je ve formátu, kdy nejlevější je nejvyšší bit. Ale paměť znaků je ve formátu kdy nejvyšší bit je vpravo. Nuly, které tvoří diagonálu, souvisejí s COM piny. Jde o simulaci jednoduchého zápisu znaku na displej v multiplexovaném režimu. Na výstup se správné nastavení pinů dostane v čase 410 ns, od spuštění simulace (běhu řadiče).

Výsledky syntézy jsou následující:

Souhrné informace o využití FPGA			
Využití logických bloků	Použito	Dostupných	Využití
Celkový počet slice registrů	201	1536	13%
Počet použitých jako klopný obvod	198		
Počet použitých jako střadač	3		
Počet 4 vstupovych LUTs	456	1536	29%
Přidělené logické bloky			
Počet obsazených Slice	280	768	36%
Počet slice obsahující pouze spojenou logiku	280	280	100%
Počet slice obsahující nespojenou logiku	0	280	0%
Celkový počet 4 vstupových LUT	472	1536	30%
Počet použitých jako logika	456		
Počet použitých jako cesta skrz	16		
Počet spojených vstupně výstupních bloku (IOB)	54	124	43%
IOB klopné obvody	3		
Počet GCLKs	1	8	12%
Celkový ekvivalent počtu hradel návrhu	4617		
Další počet JTAG hradel pro IOB	2592		

Pozn.: slice je označení pro menší logické elementy

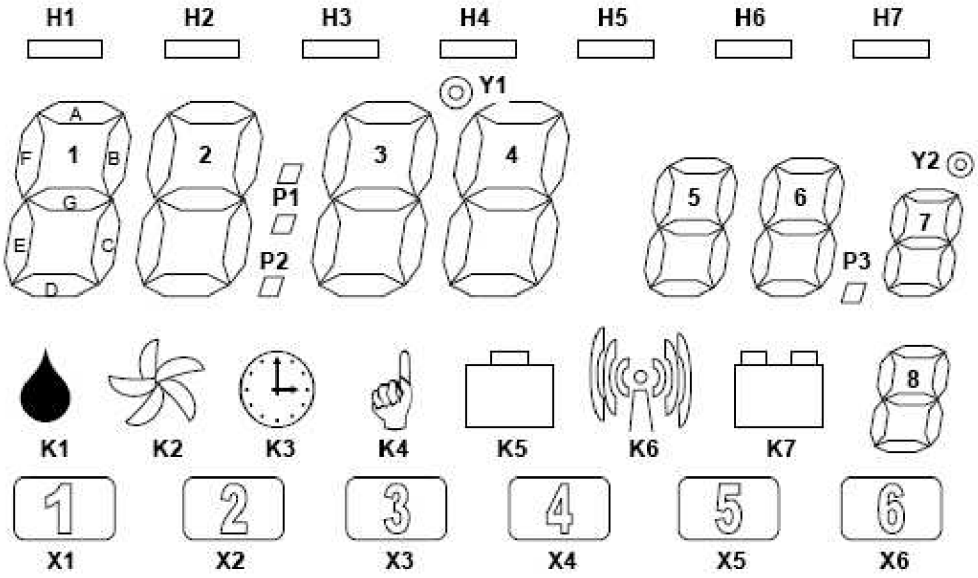
Simulace:



obr. 30 Simulace řadiče pro displej 7x1

6.3 LCD displej NTC/22K

Jako další testovací LCD byl vybrán segmentový displej z přípravku Starter Kit LJ12EVB (obr. 31), který je používán v předmětu IMP [13]. Skládá se z 81 segmentů a používá 4 multiplexy, díky tomu si tento displej vystačí z 26 piny.



obr. 31 Segmenty displeje NTC/22K

Nastavení segmentů:

BP1	BP0	FP22	FP21	FP19	FP18	FP17	FP16	FP15	FP14	FP13	FP12	FP11
1	2	3	4	5	6	7	8	9	10	11	12	13
	BP0	1F	1A	1B	2F	2A	2B	3F	3A	3B	4F	4A
BP1		1E	1G	1C	2E	2G	2C	3E	3G	3C	4E	4G
		K1	1D	K2	K3	2D	P1	K4	3D	K5	K6	4D
		X1	H1	X2	X3	H2	P2	X4	H3	X5	Y1	H4

FP10	FP9	FP8	FP7	FP6	FP5	FP4	FP3	FP2	FP1		BP3	BP2
14	15	16	17	18	19	20	21	22	23	24	25	26
4B	5F	5A	6F	6A	7F	7A	Y2	8F	8A			
4C	5G	5B	6G	6B	7G	7B		8G	8B			
K7	5E	5C	6E	6C	7E	7C		8E	8C			BP2
X6	5D	H5	6D	P3	7D	H6		8D	H7		BP3	

Jelikož má tento displej velké množství jednotlivých segmentů, je nutno tyto segmenty spojovat při zadávání do pole znaků. U klasické 7 segmentovky nám u tohoto displeje bude stačit do

pole pinu zadávat celkem 20 hodnot. Samostatné segmenty by bylo proto vhodné při konfiguraci vhodně spojit. A posléze při zobrazování takového segmentu do výstupního pole použít logickou funkci OR, která zajistí, že aktivní segmenty nebudou přepsaný (tato možnost je v VHDL kódu implementována a ovládá se signálem soucet).

Nastavení konstant:

počet bitů pro dělení základní frekvence SMCLK, pro zajištění správné frekvence hodin clk

```
constant frekvence : integer := 16;
```

počet pinů displeje

```
constant pocet_pinu : integer := 26;
```

počet potřebných bitů pro zápis, který pin je aktivní, pro tabulku pinů

```
constant pocet_bitu : integer := 5;
```

počet segmentů určující znak, číslo je vypočítáno jako součet poučtu sloupců na znak + 1 ovládací signál (COM)

```
constant pocet_segznak : integer := 5;
```

počet znaků (segmentů) na displeji

```
constant pocet_seg : integer := 11;
```

počet bitů určující segment

```
constant bitu_seg : integer := 4;
```

počet znaků v segmentu

```
constant pocet_znakseg : integer := 128;
```

počet řídících signálů

```
constant pocet_mux : integer := 4;
```

počet bitů reprezentující číslo mux

```
constant bitu_mux : integer := 3;
```

Pro první 7. segmentovku na displeji je nastavení pole pin následující:

```
type pin_konfig is array(0 to (pocet_segznak*pocet_seg*pocet_mux))
of std_logic_vector(7 downto 0);
```

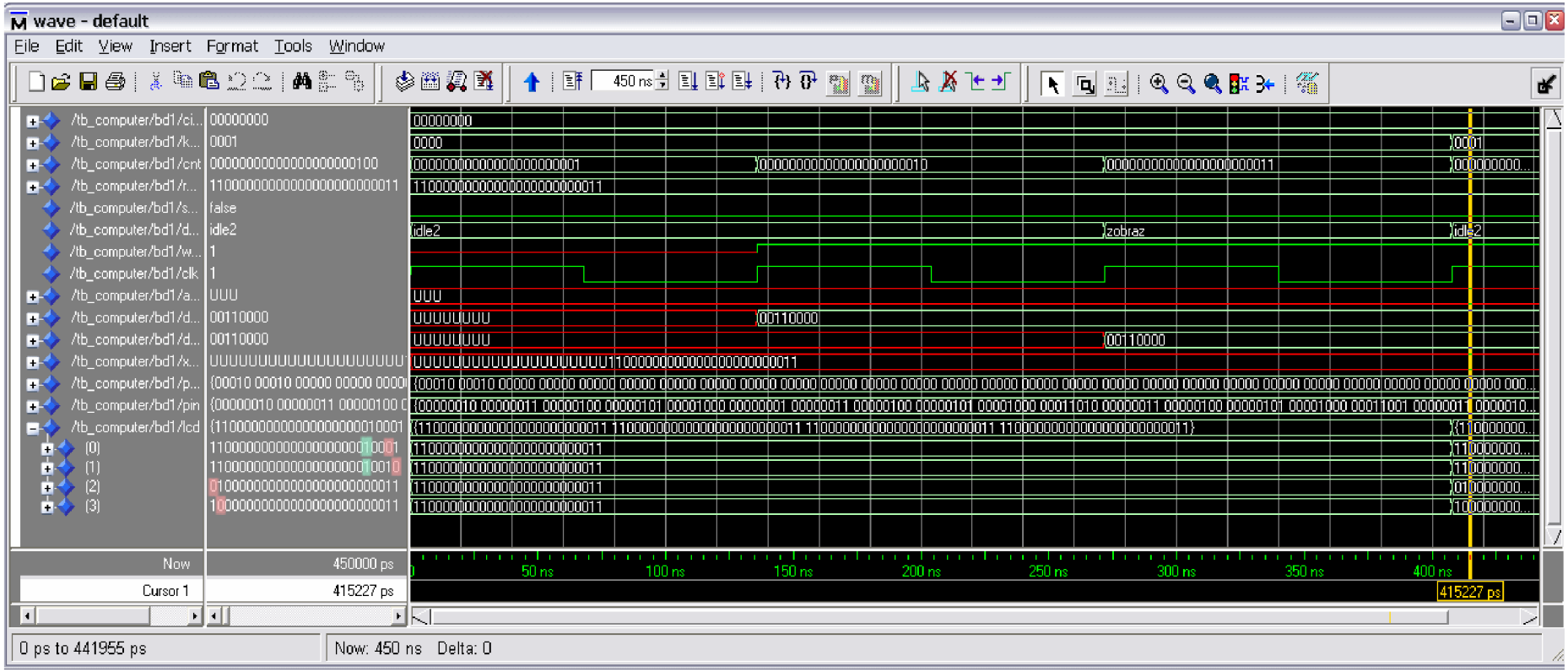
```
signal pin : pin_konfig := (
X"02",X"03",X"04",X"05",X"08",
X"01",X"03",X"04",X"05",X"08",
X"1A",X"03",X"04",X"05",X"08",
X"19",X"03",X"04",X"05",X"08",
others => "00000000"
);
```

Zde je ukázka nastavení paměti znaku, pro znak 1, kdy nejvyšší bit je vždy jedna, a to z důvodu použití multiplexu.

```
type CHAR_RAM is array(0 to (pocet_znakseg*pocet_mux)-1) of
std_logic_vector(0 to pocet_segznak-1);
signal pamet_znaku : CHAR_RAM := (
"00010",
"00010",
"00000",
"00000",
others => "00000");
```

Jak je vidět na obr. 32 (na následující straně), je tento znak namapován do výstupního pole. Pro názornost jsou potřebné segmenty zvýrazněny zeleně, aktivní multiplexy červeně. Jde o simulaci jednoduchého zápisu znaku 1 na první pozici na displeji v multiplexovaném režimu. Na výstup se správné nastavení pinů dostane v čase 410 ns, od spuštění simulace (běhu řadiče).

Simulace:



obr. 32 Symulce řadiče pro displej NTC/22K

Výsledky syntézy jsou následující:

Souhrné informace o využití FPGA			
Využití logických bloků	Použito	Dostupných	Využití
Celkový počet slice registrů	144	1536	9%
Počet použitých jako klopný obvod	141		
Počet použitých jako střadač	3		
Počet 4 vstupovych LUTs	237	1536	15%
Přidělené logické bloky			
Počet obsazených Slice	154	768	20%
Počet slice obsahující pouze spojenou logiku	154	154	100%
Počet slice obsahující nespojenou logiku	0	154	0%
Celkový počet 4 vstupových LUT	253	1536	16%
Počet použitých jako logika	237		
Počet použitých jako cesta skrz	16		
Počet spojených vstupně výstupních bloku (IOB)	54	124	43%
IOB klopné obvody	3		
Počet GCLKs	1	8	12%
Celkový ekvivalent počtu hradel návrhu	2706		
Další počet JTAG hradel pro IOB	2592		

Pozn.: slice je označení pro menší logické elementy

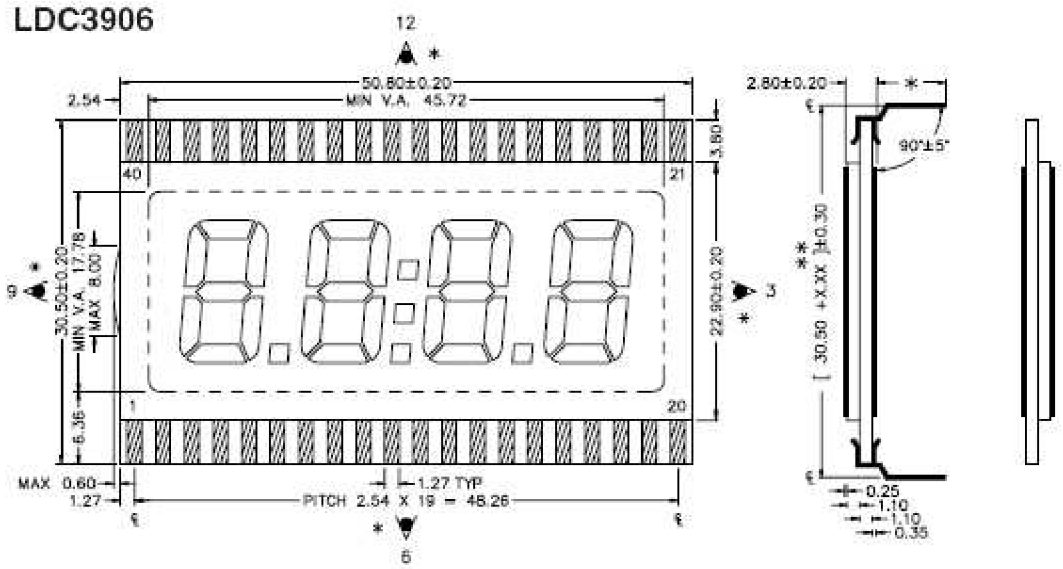
6.4 Shrnutí výsledků syntézy

Shrnutí výsledků syntézy z pohledu využití FPGA			
	7. segmentový displej	Znakový displej 7x1	NTC/22K
Počet 4 vstupovych LUTs	15%	29%	15%
Počet obsazených Slice	22%	36%	20%
Celkový počet 4 vstupových LUT	17%	30%	16%
Počet spojených vstupně výstupních bloku (IOB)	43%	43%	43%
Počet GCLKs	12%	12%	12%
Celkový ekvivalent počtu hradel návrhu	3652	4617	2706

Z výsledků syntézy je patrné, že nejnáročnější na zdroje FPGA je znakový displej 7x1, tento výsledek bych přičítal velkému počtu segmentů, které musí řadič obvodově řešit (celkem je to 280 segmentů).

7 Realizace pro vybraný LCD displej

Z důvodu velmi omezeného výběru dostupných LCD displejů neobsahující již integrovaný řadič, jsem byl nucen vybrat segmentový displej. Tento displej je podle katalogu dostupný v prodejně GME. Přesné katalogové označení tohoto displeje je LDC3906. Tento displej obsahuje čtyři 7 segmentovky, desetinné tečky a dvojtečku (obr. 33).



obr. 33 Schéma LCD displeje

Je to displej pro přímé řízení.

Konfigurace pinů [12]:

SEG.	4A	4F	4G	3B	3A	3F	3G	COL	2B	2A	2F	2G	NC	1B	1A	1F	1G	NC	NC	BP
PIN#	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
SEG.	BP	NC	NC	NC	1E	1D	1C	DP1	2E	2D	2C	DP2	3E	3D	3C	DP3	4E	4D	4C	4B
PIN#	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

Popis zkratek:

BP – back plane, spodní vodivá strana, musí být vždy uzemněna (tzn. logická hodnota 0).

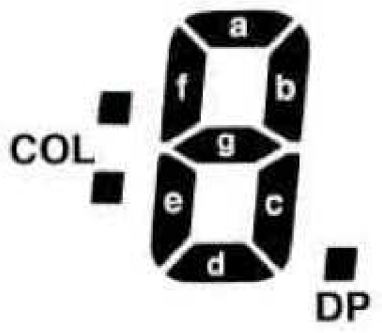
NC – not connect, pin nepřipojen

xE, xD, ... – jednotlivé segmenty znaku na pozici x, pozice je počítána z leva do prava.

DPx – tečka za znakem na pozici x

COL – dvojtečka

Konfigurace segmentů znaku viz. obr. 34.

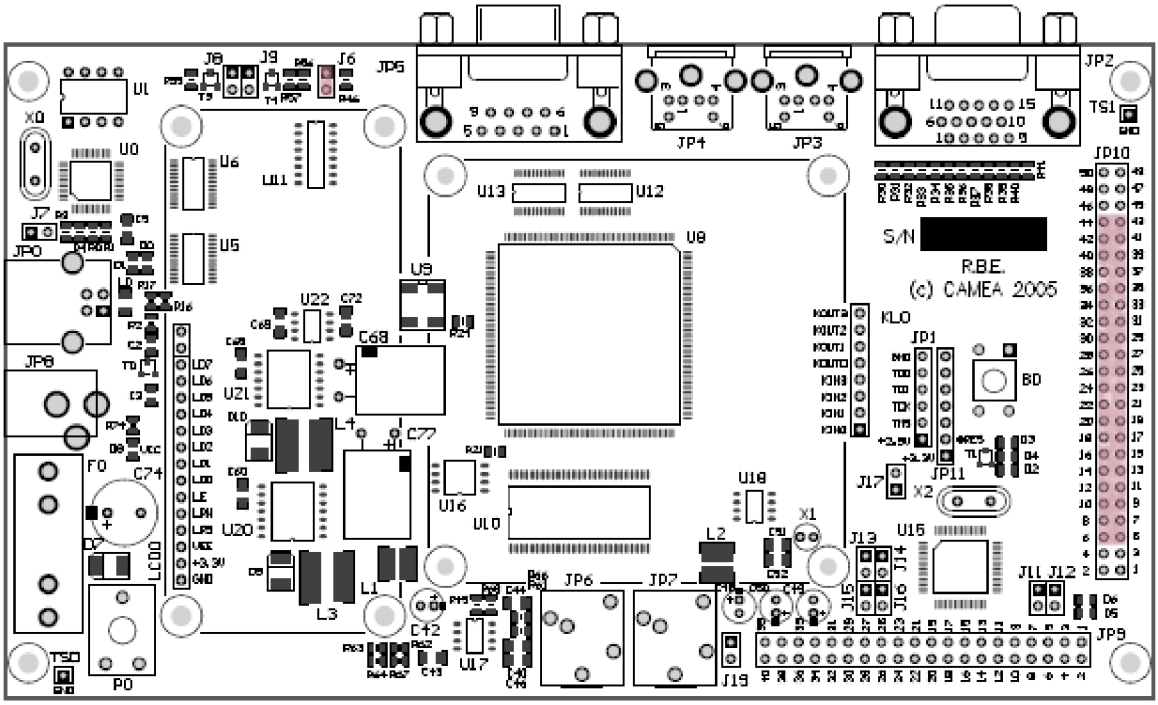


obr. 34 Označení segmentů

Kompletní datasheet je k dispozici na internetu [12].

7.1 Propojení s FITkitem

Displej je přímo napojen na vstupně výstupní pole pinů FITkitu (je označen jako JP10). A to konkrétně na piny 5 až 46. Pomocí JP6 je zrušeno propojení na konektory PS2, RS232, VGA. (na obr. 35 zvýrazněno)



obr. 35 FITkit

7.2 Programování FPGA

Pro naprogramování FPGA jsem v úvodu využil již existujícího projektu, a to konkrétně zobrazeni na LCD, které je součástí kitu. Byly modifikovány vstupní a výstupní signály entity Display_base. A to především signály zajišťující komunikaci s okolím. Mezi další komponenty patří SPI_ctrl a SPI2adc zajišťující komunikaci po SPI sběrnici, na těchto komponentách nebyly provedeny žádné změny. Dále došlo k úpravě programu pro MCU, aby obsahoval mnou potřebné funkce, kterými řídím implementovaný řadič.

7.3 Programování mikrokontroleru

U projektu, ze kterého jsem vycházel, jsem musel upravit a také přidat některé funkce.

Konkrétně jde o funkce:

void inline left_cursor(void) – pohyb kurzoru o jednu pozici do leva

void inline right_cursor(void) – pohyb kurzoru doprava

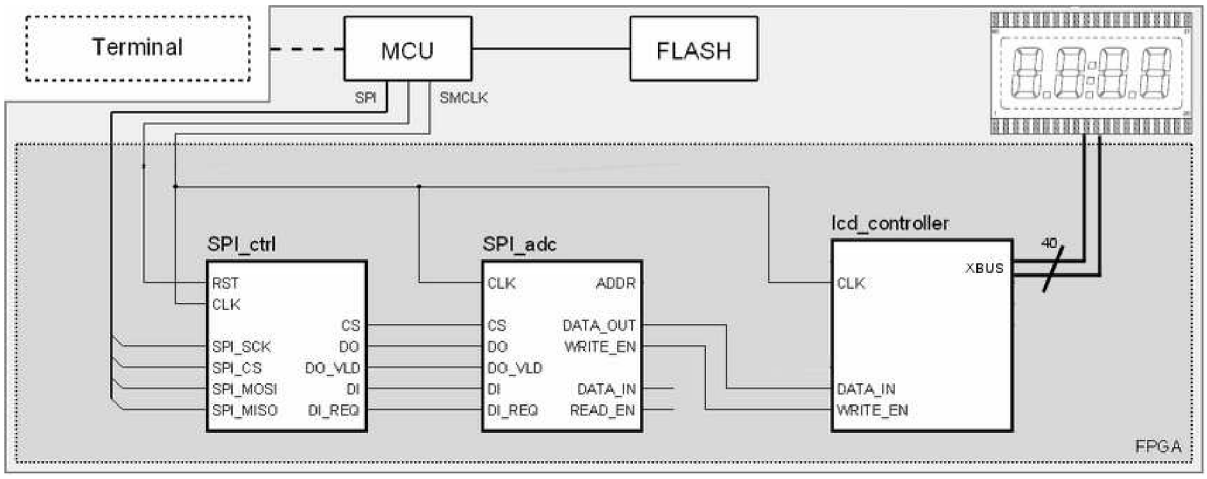
void inline DISPLAY_CLEAR(void) – upravená funkce pro smazání displeje

void zapis_znak(char *data) – funkce pro zápis nového znaku do řadiče LCD

U všech výše napsaných funkcí se využívá nejdůležitější funkce send_bit_diaplay, která zajistí poslání dat po sběrnici SPI z MCU do fpga.

7.4 Průběh komunikace PC -> LCD

Jak je ukázáno na blokovém schématu (obr. 36), skládá se VHDL kód z 3 bloků.



obr. 36 blokové schéma

Po připojení FITkitu k PC pomocí USB, musíme do MCU nahrát správný program. Program je psán v jazyce C, a využívám již hotových knihoven pro komunikaci s FITkitem, jako je například libfitkit.

Po naprogramování a připojení pomocí terminálu [14], musíme naprogramovat FPGA, a to bud příkazem prog fpga zapsaným do konzole (nebo příkazem flash w fpga a následným resetem FITkitu) a vybráním zkompilovaného binárního souboru (při použití připraveného make souboru je výstup pojmenován jako output.bin). Při posílání souborů v konzole je nutné nastavit přenos jako XMODEM -1KCRC.

Pokud máme všechny předešlé úkony hotovy, a také máme připojen LCD displej na správné piny JP10, a také rozpojen J6, můžeme do konzole psát příkazy a sledovat reakce displeje.

7.4.1 Popis komunikace od příkazu až k zobrazení na LCD

Po napsání příkazu do konzole se tento řetězec odešle do MCU, zde je zpracován. V případě, že příkaz neexistuje, je zpět do konzole zaslán řetězec, který říká, že daný příkaz neexistuje. V případě, že je příkaz správný, je zpracován. Příkazy pro práci s připojeným LCD jsou tyto:

dischar - zobrazení znaku a posun kurzoru doprava

dis - zobrazeni řetězce, od pozice 1

posl - posun kurzoru doleva

posr - posun kurzoru doprava

disclean - smazání displeje

zzn - uložení nového znaku do paměti znaků

Tyto příkazy volají funkce:

dischar = void send_char_diaplay(char ch)

dis = void send_string_diaplay(char *data)

posl = void inline left_cursor(void)

posr = void inline right_cursor(void)

disclean = void inline DISPLAY_CLEAR(void)

zzn = void zapis_znak(char *data)

Popis jednotlivých funkcí:

```
void send_char_diaplay(char ch){
```

daný znak předám funkci send_bit_diaplay, která zařídí zápis dat do SPI

```
        send_bit_diaplay(char2char_dis(ch), 1);
         return;}
```

```
void send_string_diaplay(char *data){
    int i;
    int len = strlen(data);
    if (len > 4) len = 4;
```

napřed smaži displej (tím také zajistíme posun kurzoru na první pozici)

```
    DISPLAY_CLEAR();
```

v cyklu posílám postupně jeden znak za druhým

```
    for (i=0; i < len; i++)
    {
        send_char_diaplay(data[i]);
    }
    return;}
```

```
void inline DISPLAY_CLEAR(void){
```

zašleme do FPGA hodnotu 192, to odpovídá "11000000"

```
    send_bit_diaplay(192,0);
    return;}
```

```
void inline left_cursor(void){
```

zašleme do FPGA hodnotu 194, to odpovídá "11000010"

```
    send_bit_diaplay(194,0);
    return;}
```

```
void inline right_cursor(void){
```

zašleme do FPGA hodnotu 193, to odpovídá "11000001"

```
    send_bit_diaplay(193,0);
    return;}
```

```
void zapis_znak(char *data){
    int i;
    int cislo1 = 0;
    int cislo2 = 0;
    int nasobek;
    int len = strlen(data);
```

převod řetězce na číselnou hodnotu

```
    for (i=0; i < len; i++)
    {
        if (data[i] == ' ') {cislo2 = cislo1; cislo1 =0;}
        else {cislo1 = cislo1*10;
             cislo1+=data[i]-48;
             }
    }
```

poslání příkazu do FPGA, hodnota 195 říká že jde o instrukci vložení nového znaku do paměti znaků

```
    send_bit_diaplay(195,0);
```

zaslání hodnoty pozice znaku v paměti znaků

```
    send_bit_diaplay(cislo2,0);
```

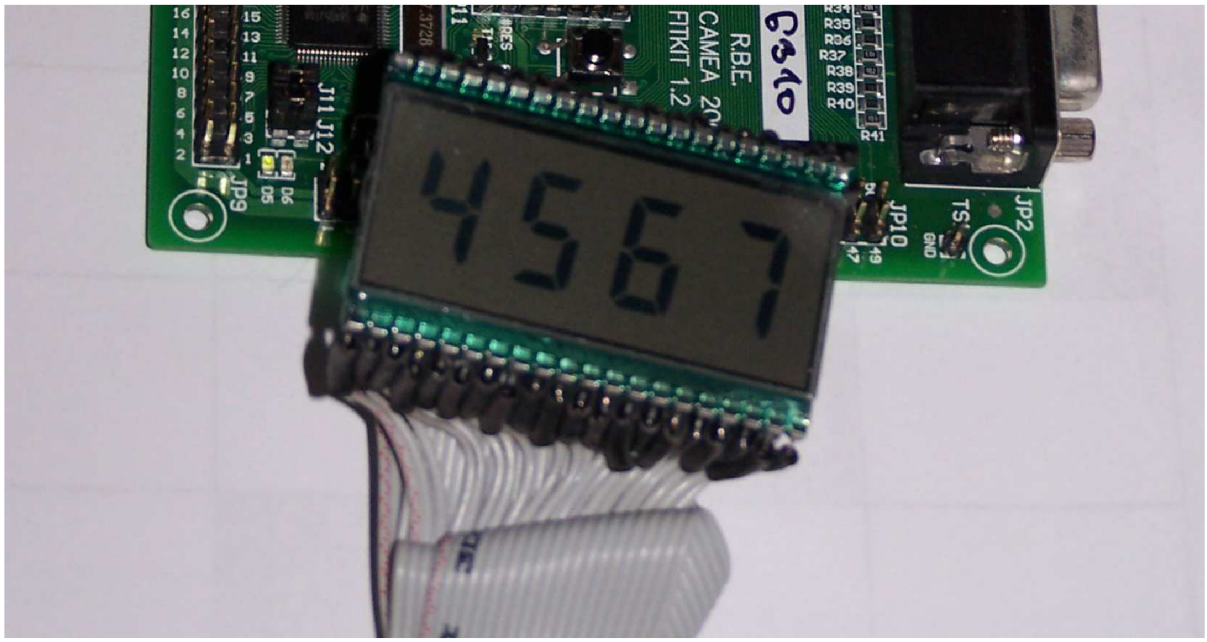
zaslání znaku

```
send_bit_diaplay(cislo1,0);
return;}
```

Dále tyto příkazy zpracuje konečný automat v FPGA a podle daných hodnot rozhodne o jakou akci se bude jednat (kód FPGA je již rozebrán výše, takže se o něm zde nerozepisuji).

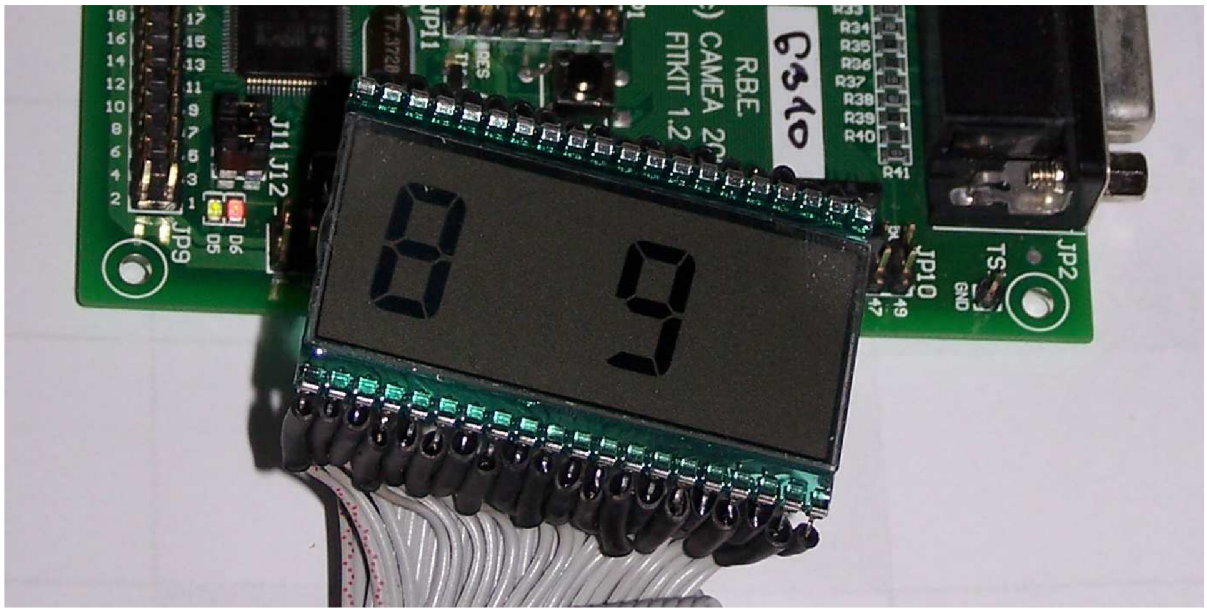
7.4.2 Příklady použití

Do konzole zapíši tento příkaz „dis 4567“, jsou odeslány jednotlivé znaky do FPGA a dále zobrazeny na LCD (obr. 37).



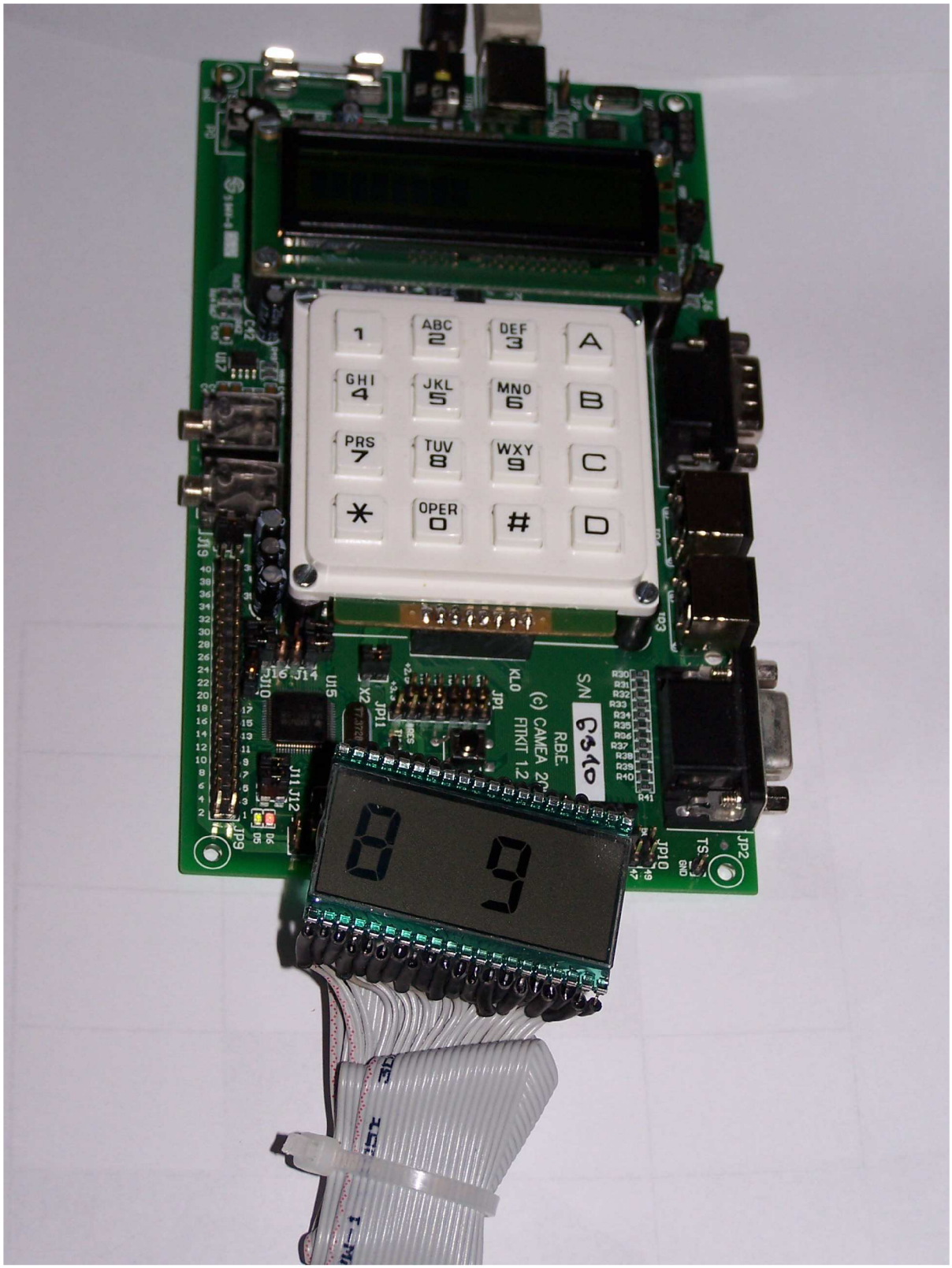
obr. 37 Stav displeje po příkazu „dis 4567“

Dále následovali příkazy „disclear“, „dischar 8“, „posr“, „dischar 9“, vysledek viz. obr. 38.



obr. 38 Stav displeje po sekvenci příkazů

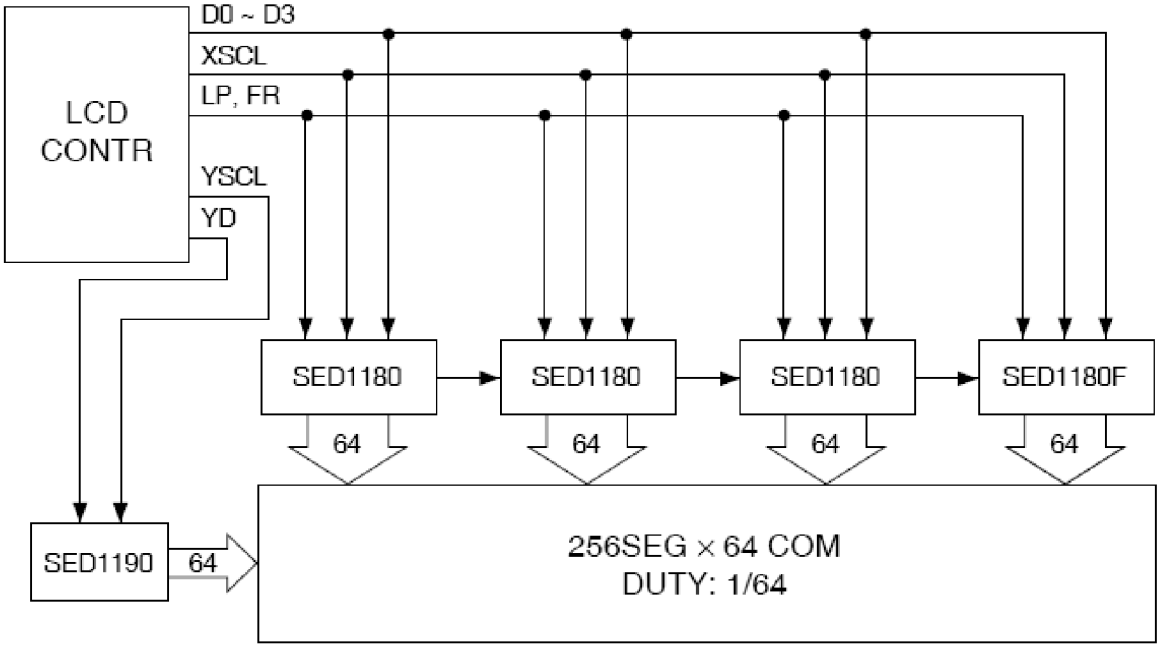
Na závěr fotografie zapojeného FITkitu s LCD (obr. 39).



obr. 39 Ukázka zapojeného LCD

8 Závěr

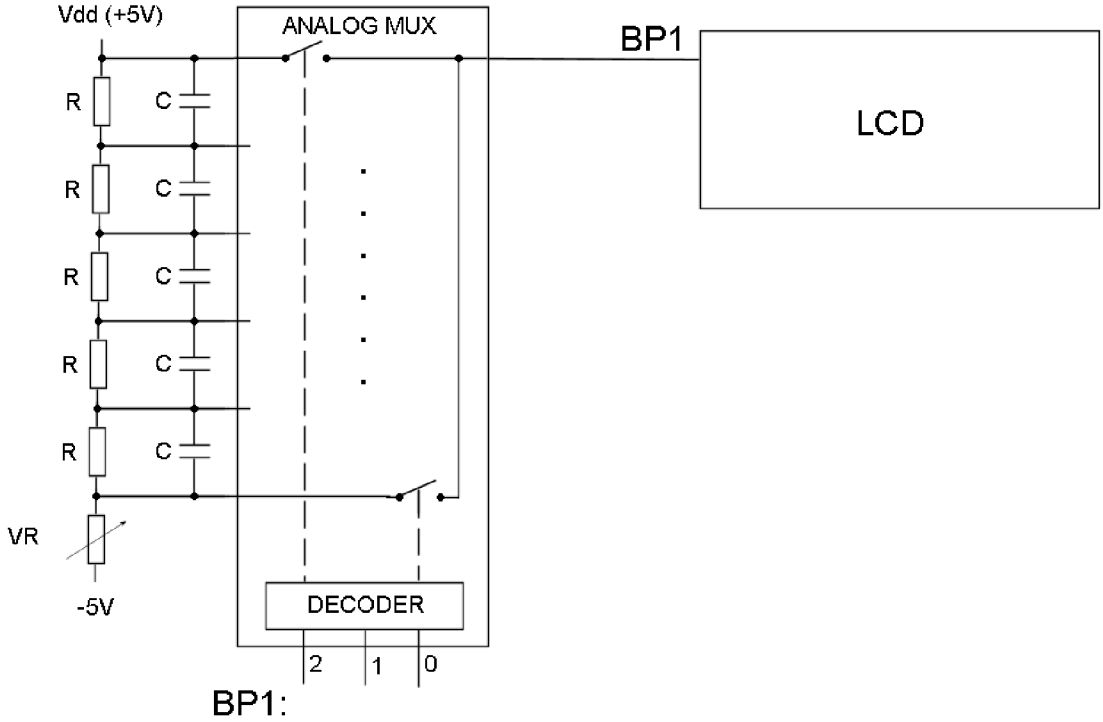
Jako nejdůležitější výsledek této práce bych označil funkční HW řešení s LCD displejem. Jelikož jde o univerzální řadič, je v něm obsažena složitá práce s pamětí reprezentující nastavení výstupních pinů. Vytvořený řadič je směrován spíše pro displeje segmentové, případně displeje textové. Dal by se samozřejmě využít i pro displeje grafické, ale zde by muselo dojít k větším úpravám především v bloku zajišťujícím přímé řízení LCD, protože u těchto displejů se využívá dalších součástek jako adresní dekodéry, na obr. 40 se jedná o bloky SED1180 (Segment driver) [15] a SED1190 (Common drivers) [16].



obr. 40 Blokové schéma řadiče a podpůrných obvodů pro grafický displej

Další vývoj by se mohl týkat zvýšení počtu dalších funkcí pro ovládání displeje, ale zde již můžeme narazit na nedostatek bloků v použitém FPGA.

Pro řízení multiplexovaných LCD je dále nutno vytvořit podpůrné obvody, jako je například v této práci zmíněný odporový žebřík k zajištění potřebných víceúrovňových budících signálů. A dále by bylo nutné definovat (případně generovat podle nějakého nastavení) tyto víceúrovňové průběhy, abychom jejich pomocí byli schopni ve správný čas nastavit odpovídající napěťovou úroveň. Pokud signály BP a SEG popíšeme více bity, a to tak že v daný časový okamžik přiřadíme tomuto signálu požadovanou hodnotu, pak pomocí analogového multiplexoru vygeneruje požadované napětí, viz schematicky obr. 41.



obr. 41 Přepínání napětí pro víceúrovňový signal BP1

Jako další rozšíření této diplomové práce by mohlo být zaměřeno na ovládání složitějších displejů, např. tak jak je v této naznačeno. Při tomto rozšiřování by bylo nutné upravit části týkající se generování víceúrovňových signálů, v případě grafických displejů řešit komunikaci z dalšími obvody. Pro grafické displeje by se také hodily jiné instrukce, protože v této práci jsem byl zaměřen na displeje znakové.

Literatura

[1] STRUNA, V. *LCD* [online]. 2000 [cit. 2007-01-03]. Dostupný z WWW: <http://www.sous.cz/struna/cd/>.

[2] ELECTRONIC INVENTORY ONLINELCD. *Information and Technical Forum* [online]. 2005 [cit. 2008-01-22] Dostupný z WWW: <http://eio.com/lcdintro.htm>

[3] Pacific Display Devices. *Products and Services* [online]. 1996-2007 [cit. 2007-01-03]. Dostupný z WWW: <http://www.pacificdisplay.com/products.htm>.

[4] STRUNA, V. *Napájení a typy podsvícení LCD* [online]. 2000 [cit. 2007-01-03]. Dostupný z WWW: <http://www.sous.cz/struna/cd/aplikace.html#nap%E1jen%ED%20a%20typy%20podsv%EDcen%ED%20LCD>.

[5] Pacific Display Devices. *Static Drive LCD Technology* [online]. 1996-2007 [cit. 2007-01-03]. Dostupný z WWW: <http://www.pacificdisplay.com/lcd_static_drive.htm>.

[6] Pacific Display Devices. *Multiplex Drive and Bias of LCD Technology* [online]. 1996-2007 [cit. 2007-01-03]. Dostupný z WWW: <http://www.pacificdisplay.com/lcd_multiplex_drive.htm>.

[7] SCHWARZ, Josef, RŮŽIČKA, Richard, STRNADEL, Josef. *Mikroprocesorové a vestavěné systémy : Studijní opora.* [s.l.] : [s.n.], [2006]. 168 s. Dostupný z WWW: <https://www.fit.vutbr.cz/study/courses/IMP/private/VYUKA/PREDNASKY/STUDIJNI_OPORA/IMP_opora_pr.pdf>.

[8] FIT VUT. *Popis kitu* [online]. 2006 – 2008 [cit. 2008-01-22] Dostupný z WWW: <http://merlin.fit.vutbr.cz/FITkit/hardware.html>

[9] FIT VUT. *FITkit* [online]. 2006 – 2008 [cit. 2008-01-22] Dostupný z WWW: <http://merlin.fit.vutbr.cz/FITkit/>

[10] VAŠÍČEK, Zdeněk. *Propojovací systém FITkitu* [online]. 2006 – 2008 [cit. 2008-01-22] Dostupný z WWW: <http://merlin.fit.vutbr.cz/FITkit/docs/firmware/spifitkit.html>

[11] MARKOVIČ, Ján. *Knihovna libfitkit* [online]. 2006 – 2008 [cit. 2008-01-22] Dostupný z WWW: <http://merlin.fit.vutbr.cz/FITkit/docs/firmware/20060501lib.html>

[12] Display Elektronik GmbH. *DATA SHEET LCD STANDART PANEL DE 119 – SERIES* [online]. 23. 5. 2003 [cit. 2008-01-22] Dostupný z WWW: <http://www.sos.sk/a_info/resource/d/dem/DE119-RS-20_75.pdf>

[13] Beta Control. *Starter Kit LJ12EVB* [online]. [2002] [cit. 2008-01-22] Dostupný z WWW: < https://www.fit.vutbr.cz/study/courses/IMP/private/VYUKA/CVICENI_A_LABORATORE/hc908lj12_en.pdf>

[14] MARKOVIČ, Ján. *FITkit Tutoriál* [online]. 2006 – 2008 [cit. 2008-01-22] Dostupný z WWW“ <http://merlin.fit.vutbr.cz/FITkit/docs/navody/20060210a.html>

[15] *CMOS LCD 64-SEGMENT DRIVER* [online]. [2007] [cit. 2008-01-22]. Dostupný z WWW: <http://www.pacificdisplay.com/ics_app%20notes/epson/SED1180.pdf>.

[16] *CMOS LCD 64-COMMON DRIVERS* [online]. [2007] [cit. 2008-01-22]. Dostupný z WWW: <http://www.pacificdisplay.com/ics_app%20notes/epson/SED1190.pdf>.

Seznam příloh

Příloha 1. CD obsahující:

Zdrojové soubory,

Technickou zprávu v PDF.