České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra počítačů

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **Jiří Tužil**

Studijní program: Elektrotechnika a informatika (bakalářský), strukturovaný
Obor: Výpočetní technika

Název tématu: **Návrh a implementace úloh pro předmět JPO**

Pokyny pro vypracování:

Prostudujte stávající laboratorní úlohy používané v předmětu jednotky počítačů vyučovaném na katedře počítačů. Najděte a porovnejte nové přípravky osazené FPGA obvodem vhodné pro realizaci stávajících úloh. Vytvořte nový vzorový návrh sloužící jako šablona pro realizaci těchto laboratorních úloh s pomocí nástroje ISE firmy XILINX. Důraz bude kladen na možnost řešit úlohy buď ve schématu nebo s pomocí jazyka VHDL. Součástí řešení bude i knihovna základních prvku (multiplexor, klopný obvod, sčítačka). V rámci této práce vytvořte webovou stránku obsahující zadání 10 úloh a jejích řešení dokazující funkčnost vytvořených laboratorních úloh. Pro jedno zadání vytvořte kompletní popis jeho řešení.

Seznam odborné literatury:

Dodá vedoucí práce

Vedoucí: Ing. Pavel Kubalík, Ph.D.

Platnost zadání: do konce zimního semestru 2011/2012

doc. Ing. Miroslav Šnorek, CSc.
vedoucí katedry

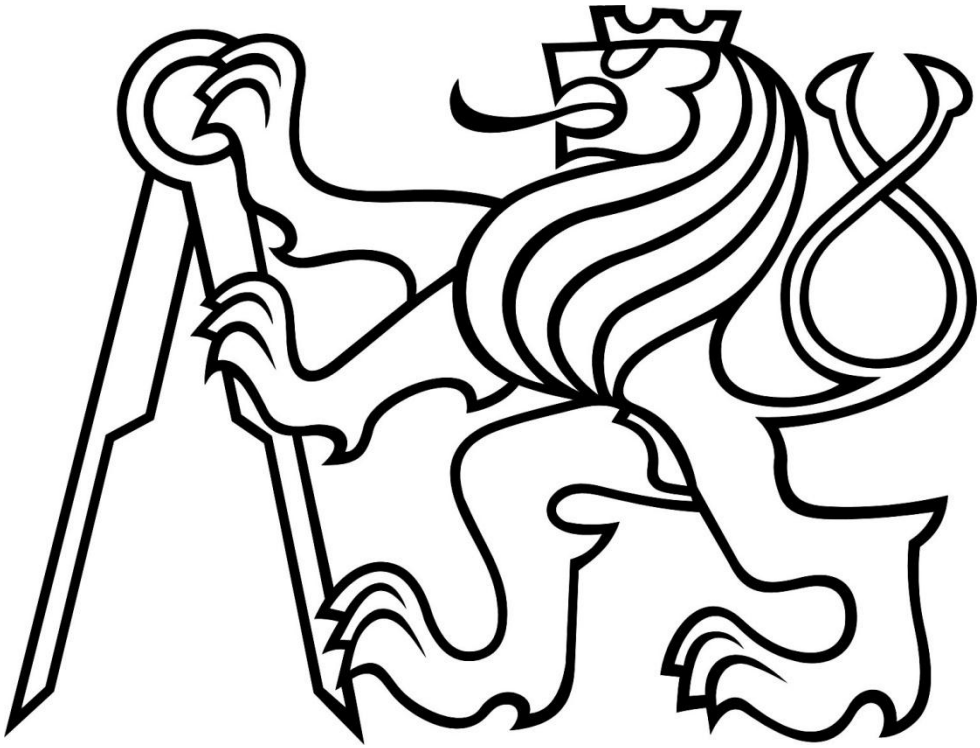
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta elektrotechnická
L.S.

prof. Ing. Boris Šimák, CSc.
děkan

V Praze dne 15. 3. 2011

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta Elektrotechnická



Bakalářská práce

Návrh a implementace úloh pro předmět JPO

Jiří Tužil

Vedoucí: Ing. Pavel Kubalík, Ph.D.

Studijní program: Elektrotechnika a informatika, strukturovaný, bakalářský
Obor: Výpočetní technika

6. leden 2012

Poděkování

Na tomto místě bych chtěl poděkovat

- mému vedoucímu práce ing. Pavlu Kubalíkovi za přínosné podněty z hlediska celého projektu
- ing. Josefu Hrázskému za praktické konzultace k řešení jednotlivých úloh

Prohlášení

Prohlašuji, že jsem svou bakalářskou práci vypracoval samostatně a použil jsem pouze podklady uvedené v přiloženém seznamu.

Nemám závažný důvod proti užití tohoto školního díla ve smyslu § 60 Zákona č.121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne .....

Abstract

Topic of this bachelor thesis is a design of a project that uses boards with FPGA chip according to requirements of school subject JPO. The board is used to practice data and control paths in the lessons. This bachelor thesis presents and solves typical tasks in the JPO subject with detailed description of the model task and manual, how to work with the project.

Abstrakt

Bakalářská práce se zabývá návrhem projektu za použití desek osazených FPGA čipem podle požadavků výuky předmětu jednotky počítačů (JPO) k procvičování datové a řídící cesty. Práce řeší typové úlohy probírané v tomto předmětu s podrobným popisem vzorové úlohy a vytvořením návodu, jak pracovat s projektem.

Obsah

b

Seznam obrázků

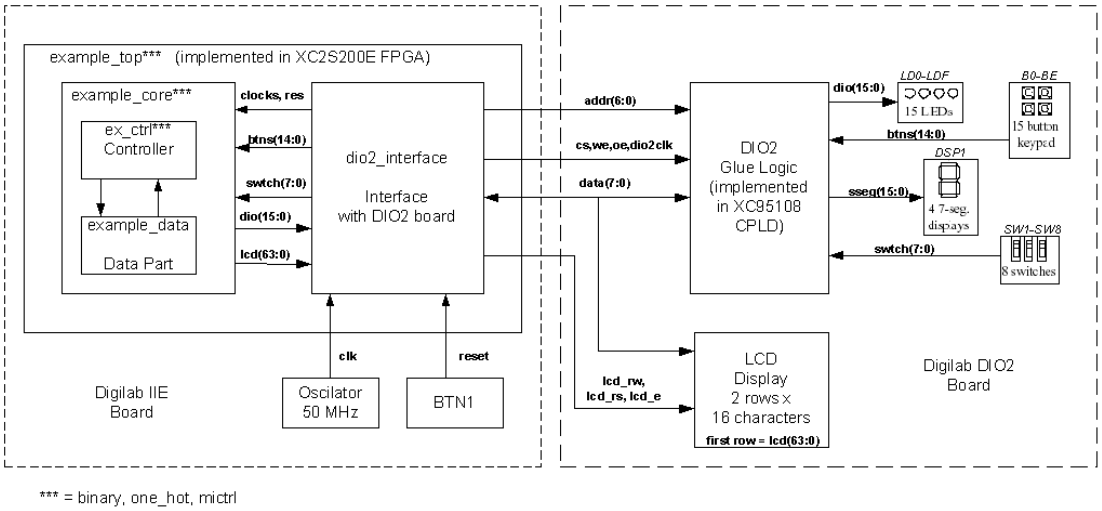
Seznam tabulek

1. Úvod

Cílem této práce je nastudovat stávající řešení laboratorních úloh v předmětu Jednotky počítačů (dále JPO), vybrat novou vhodnou vývojovou desku osazenou FPGA čipem, vytvořit vzorový projekt ve vývojovém prostředí ISE od firmy Xilinx, který bude sloužit pro řešení jednotlivých úloh na procvičování datové a řídící cesty v předmětu JPO. Součástí bude i knihovna obsahující základní stavební bloky, jako je registr, multiplexor a sčítačka. Součástí práce je i webová prezentace obsahující zadání deseti úloh a podrobný popis jedné úlohy, na které bude ukázáno, jak pracovat s projektem. Zadání úloh jsem převzal z laboratorních úloh z předchozích let.

2. Rešerše

Základem stávajícího řešení je vývojová deska Digilab D2E-DIO2. Samotná vývojová deska D2E nemá žádná ovládací tlačítka ani indikační diody LED, které by byly použitelné pro komunikaci s uživatelem. Z tohoto důvodu je k ní připojena rozšiřující deska DIO2. Vzhledem k tomu, že není možné přistupovat přímo k ovládacím tlačítkům, indikačním diodám LED, přepínačům a displeji LCD, přistupuje se k nim pomocí sběrnice s adresovou, datovou a řídící částí. Samotná realizace procvičované úlohy je ve dvou entitách, ex_ctrl*** a example_data (Obrázek 1). Oba soubory jsou zakomponovány do entity example_core***, kde jsou propojeny jednotlivé signály mezi datovou a řídící částí. Nyní je potřeba propojit example_core*** s rozhraním, které zajišťuje komunikaci s deskou DIO2.



Obrázek 1: Celkové blokové schéma obvodu a jeho realizace

Toto řešení přineslo možnost přistupovat k ovládacím tlačítkům, LED, indikačním diodám LED, přepínačům a displeji LCD tak, jako by byly připojeny přímo k čipu FPGA. Nevýhodou je nutnost zapojení všech signálů, a to i těch, které jsme nutně nemuseli využít.

Dále jsem prostudoval stávající zadání laboratorních úloh. Tyto úlohy obsahují všechny základní matematické operace (sčítání, odčítání, násobení, dělení)

a logickou operaci posuv čísel. V úlohách se pracuje převážně s 8bitovými operátory, výjimkou jsou úlohy na posuv čísel.

3. Analýza

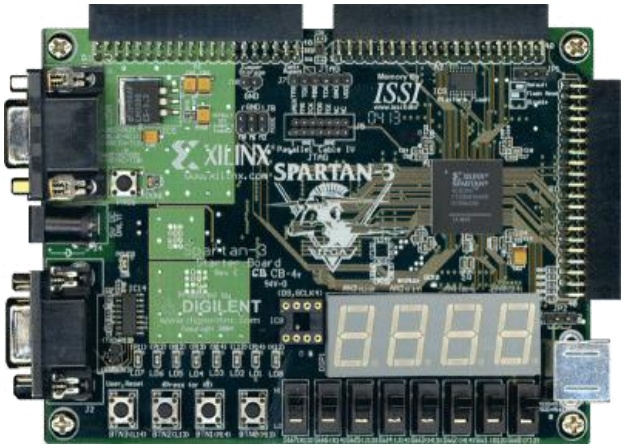
V této kapitole se zabývám výběrem vhodné vývojové desky, na které se budou řešit jednotlivé úlohy. Hlavním kritériem výběru je dostatečný počet ovládacích tlačítek, přepínačů, indikačních diod LED a sedmisegmentových displejů, případně LCD displejů. Pro porovnání vhodnosti vývojových desek zde uvádím zúžený výběr, a to jen dvou z nich. Existuje samozřejmě řada dalších možností, pro tuto aplikaci vzhledem k ceně a vybavení desek jsem zvolil toto řešení. Protože se v dílčích úlohách pracuje na vstupu maximálně s osmibitovými čísly, postačuje k jejich řešení pouze 8 přepínačů.

Pro tuto práci potřebujeme desku s dostatečným počtem tlačítek a přepínačů. V úvahu připadají desky Spartan-3 a Spartan-3E, přičemž verze 3E neobsahuje dostatečný počet přepínačů. Navíc je deska Spartan-3 podstatně jednodušší a levnější.

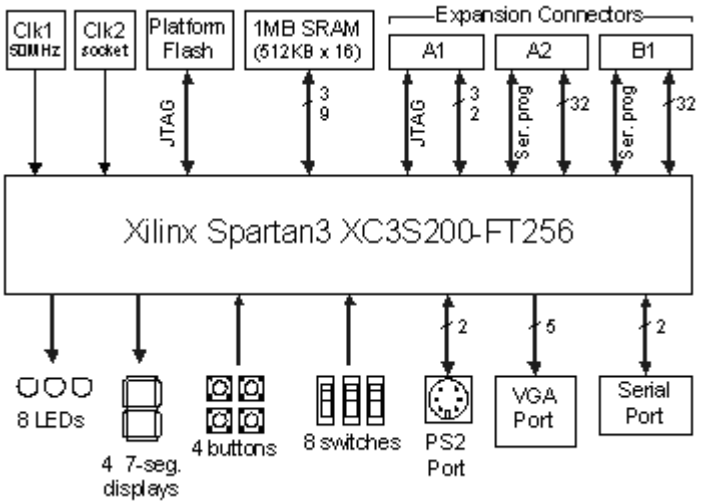
3.1. Digilent Spartan-3 Starter Kit

Parametry této desky jsou:
1 x RS232 port
1 x PS2 port
1 x VGA port
4 x tlačítko
8 x přepínač
8 x LED
1 x sedmisegmentový displej (4x1)
1 x SRAM (1 MB, 10 ns)
3x 40-pin rozšiřující slot

Na obrázku 2 je vyobrazena vývojová deska Diligent Spartan-3 Starter Kit a na obrázku 3 pak její blokové schéma.



Obrázek 2: Vývojová deska Digilent Spartan-3 Starter Kit



Obrázek 3: Blokové schéma vývojové desky Spartan 3

3.2. Digilent Spartan-3E Starter Kit:

2 x RS232

1 x PS2

1 x VGA

1 x Ethernet

1 x LCD (2 x16)

4 x přepínač

5 x tlačítko

1 x otočné tlačítko

8 x LED

Deska Spartan-3E je vyobrazena na obrázku 4.

Obrázek 4: Xilinx Spartan-3E Starter Kit

4. Návrh řešení

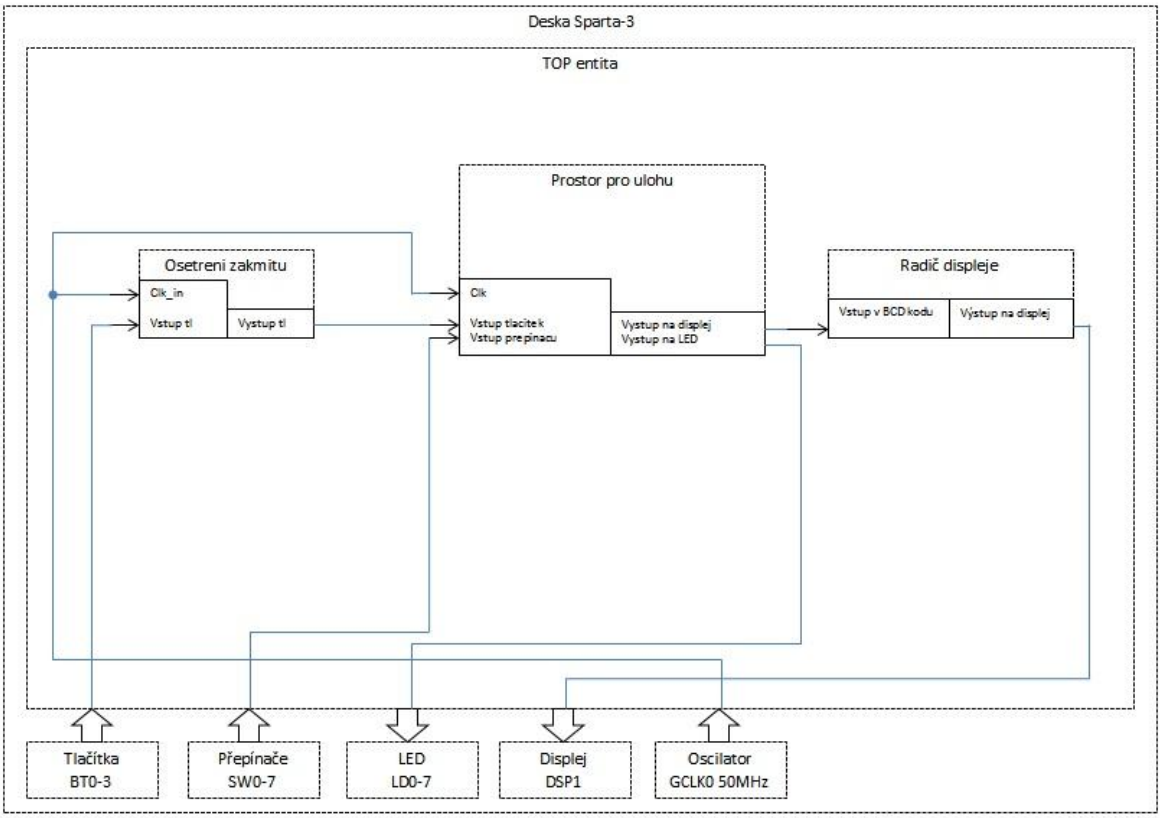
Návrh lze rozdělit do dvou částí - návrh projektu a zadání úloh, které budou sloužit k procvičování datové a řídící cesty.

4.1. Návrh řešení projektu

V projektu je potřeba vyřešit tyto dílčí části:

- odstranění zákmitů tlačítek
- vytvoření dynamického řadiče displeje
- vytvoření souboru, ve kterém bude řešena úloha
- propojení jednotlivých bloků.

Na obrázku 5 je nakresleno blokové schéma projektu.



Obrázek 5: Návrh blokového schéma

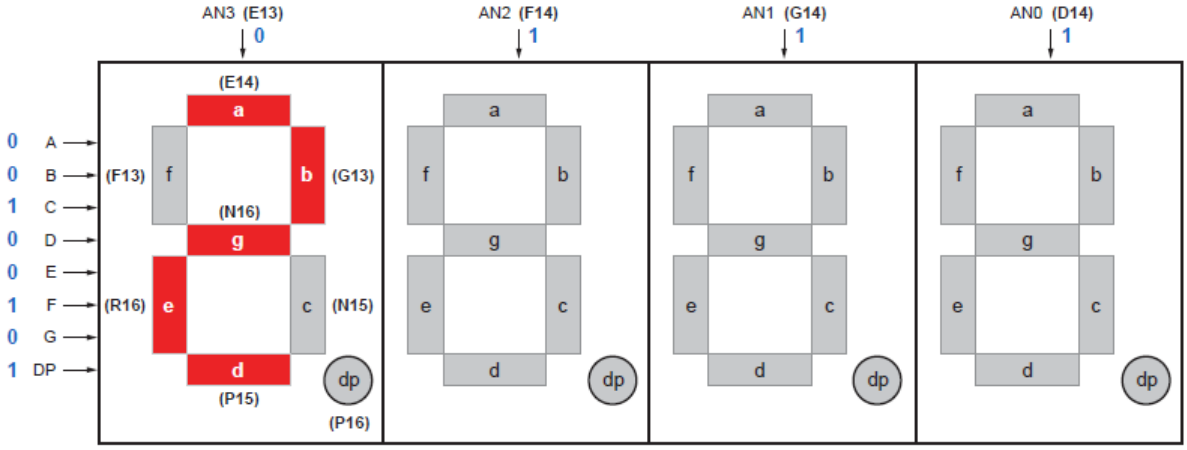
Úlohy bude možné řešit pomocí obvodového schématu nebo VHDL kódu. Proto vzniknou dva projekty, které se budou lišit pouze v bloku „Prostor pro úlohu“. V prvním případě se bude jednat o soubor se schématem a v druhém o soubor s VHDL kódem. Ačkoliv na předmětu X36PNO bylo doporučeno aby datová část a řídící část byly řešeny samostatně, rozhodl jsem se umístit obě části do jednoho schématu. Toto nestandardní řešení velmi zjednoduší práci v ISE a umožňuje pracovat na dílčích úlohách i studentům s minimálními znalostmi vývojového prostředí.

4.1.1. Odstranění zákmitů tlačítek

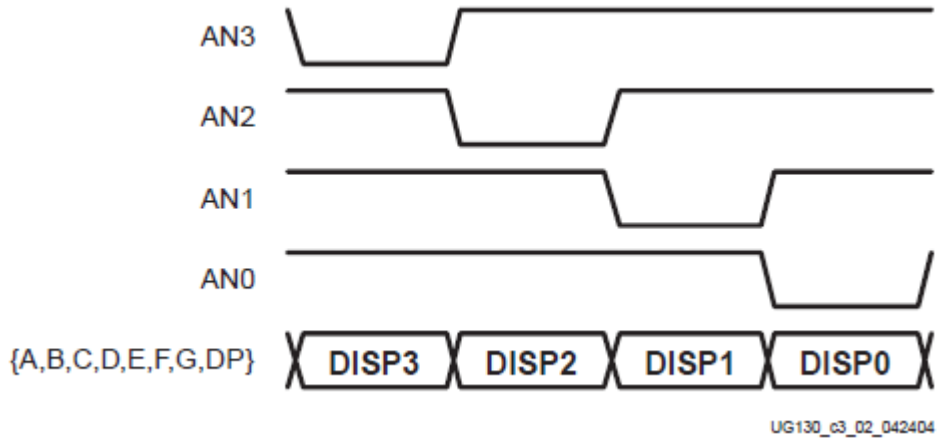
Blok „ošetření zákmitů“ je použit k odstranění zákmitů na tlačítkách při jejich stisku. Kdyby nebyla tlačítka ošetřena, mohlo by dojít k vytvoření několika nedefinovaných impulzů za sebou, než by se stav tlačítka ustálil.

4.1.2. Řadič displeje

Sedmisegmentový displej se skládá ze 4 sedmisegmentovek, které obsahují 8 společných datových vodičů, kde každá sedmisegmentovka má 1 samostatný anodový vstup (Obrázek 6). Musí být vytvořen dynamický řadič displeje, který bude nastavovat hodnoty na společných datových vstupech a přepínat anodové vstupy tak, aby v jediném časovém okamžiku byla aktivní vždy jen jedna sedmisegmentovka (Obrázek 7).



Obrázek 6: Sedmisegmentový displej



Obrázek 7: Průběh signálů na sedmisegmentovém displeji

4.1.3. Blok prostor pro úlohu

V tomto bloku (entitě) je prostor pro řešení dílčí úlohy. Jinak řečeno, bude zde navrženo schéma datové a řídící cesty úlohy. Budou zde připraveny potřebné vstupy a výstupy. Přepínače budou přímo napojeny na hardwarové přepínače, tlačítka budou napojena přes blok „ošetření zákmitů“. Výstup na diody bude napojen přímo na LED diody přípravku. Výstup na sedmisegmentový displej bude připojen přes blok „řadič displeje“, který převede BCD kód na 8 společných datových signálů a 4 anodové vstupy.

4.2. Návrh zadání úloh

Zadání úloh jsem se rozhodl převzít z předchozích let výuky JPO a přizpůsobil formulaci zadání tak, aby je bylo možné použít na zvolené vývojové desce Spartan-3. Úlohy pokrývají všechny základní matematické a logické operace, jako je sčítání, odčítání, násobení, dělení a posuvy čísel.

4.2.1. Úloha 1

Navrhněte sério-paralelní dvojkovou násobičku dvou 8bitových celých čísel bez znaménka.

Popis chování: V počátečním stavu čeká obvod na zadání hodnoty násobence A na přepínači swtch(7:0), hodnota je zobrazena na sedmisegmentovém displeji. Platnost hodnoty A je potvrzena stiskem tlačítka B0. Následně je obdobně zadána hodnota násobitele B, kdy hodnota je zobrazena na sedmisegmentovém displeji a opět potvrzena stiskem tlačítka B0. Po zadání násobitele je provedeno

násobení a výsledek (16 bitů) je zobrazen na sedmisegmentovém displeji v hexadecimálním kódu. Po opětovném stisknutí tlačítka B0 se obvod dostane do počátečního stavu.

Podmínky řešení: Při realizaci násobičky použijte pouze jednu 8bitovou sčítačku.

4.2.2. Úloha 2

Navrhněte sério-paralelní dvojkovou násobičku dvou 8bitových celých čísel v doplňkovém kódu Boothovou metodou.

Popis chování: V počátečním stavu čeká obvod na zadání hodnoty násobence A na přepínači swtch(7:0), hodnota je zobrazena na sedmisegmentovém displeji. Platnost hodnoty A je potvrzena stiskem tlačítka B0. Následně je obdobně zadána hodnota násobitele B, kdy hodnota je zobrazena na sedmisegmentovém displeji a opět potvrzena stiskem tlačítka B0. Po zadání násobitele je provedeno násobení a výsledek (16 bitů) je zobrazen na sedmisegmentovém displeji. Po opětovném stisku tlačítka B0 se obvod dostane do počátečního stavu.

Podmínky řešení: Při realizaci násobičky použijte pouze jednu 9bitovou sčítačku/odčítačku.

4.2.3. Úloha 3

Navrhněte sério-paralelní dvojkovou děličku dvou 8bitových celých čísel bez znaménka.

Popis chování: V počátečním stavu čeká obvod na zadání hodnoty dělence A na přepínači swtch(7:0), hodnota je zobrazena na sedmisegmentovém displeji. Platnost hodnoty A je potvrzena stiskem tlačítka B0. Následně je obdobně zadána hodnota dělitele B, kdy hodnota je zobrazena na sedmisegmentovém displeji a opět potvrzena stiskem tlačítka B0. Po zadání dělitele je provedeno dělení a podíl je zobrazen na sedmisegmentovém displeji. Stiskem tlačítka B0 se zobrazí zbytek na sedmisegmentovém displeji. Dělička detekuje dělení nulou a indikuje ho rozsvícením diody LD0. Po opětovném stisknutí tlačítka B0 se obvod dostane do počátečního stavu.

4.2.4. Úloha 4

Navrhněte sério-paralelní dvojkovou děličku dvou 8bitových celých čísel v přímém kódu

Popis chování: V počátečním stavu čeká obvod na zadání hodnoty dělence A na přepínači swtch(7:0), hodnota je zobrazena na sedmisegmentovém displeji. Platnost hodnoty A je potvrzena stiskem tlačítka B0. Následně je obdobně zadána hodnota dělitele B, hodnota je zobrazena na sedmisegmentovém displeji a opět potvrzena stiskem tlačítka B0. Po zadání dělitele je provedeno dělení a podíl je zobrazen na sedmisegmentovém displeji. Stiskem tlačítka B0 se zobrazí zbytek na sedmisegmentovém displeji. Dělička detekuje dělení nulou a indikuje ho rozsvícením diody LD0. Po opětovném stisknutí tlačítka B0 se obvod dostane do počátečního stavu.

Podmínky řešení: Znaménko výsledku určete XORem znamének jednotlivých operandů.

4.2.5. Úloha 5

Navrhněte dvojkovou sčítačku/odčítačku dvou 8bitových celých čísel v přímém kódu

Popis chování: V počátečním stavu čeká obvod na zadání hodnoty sčítance A na přepínači swtch(7:0), hodnota je zobrazena na sedmisegmentovém displeji. Platnost hodnoty A je potvrzena stiskem tlačítka B0. Následně je obdobně zadána hodnota sčítance B, kdy hodnota je zobrazena na sedmisegmentovém displeji a opět potvrzena stiskem tlačítka. Po zadání B je provedeno sčítání (bylo-li stisknuto B0) nebo odčítání (bylo-li stisknuto B1). Výsledek je zobrazen na sedmisegmentovém displeji. Typ operace je indikován pomocí stavu LED, LD0 - sčítání, LD1 - odčítání. Nyní obvod bude čekat na stisk tlačítka B0 a po jeho stisku se vrátí do počátečního stavu.

Podmínky řešení: K řešení použijte pouze jednu sčítačku/odčítačku čísel bez znaménka. Sčítání a odčítání musí být provedeno bez mezipřevodu do jiného kódu.

4.2.6. Úloha 6

Navrhněte obvod, který bude počítat klouzavý aritmetický průměr 4 po sobě zadaných 8bitových čísel v doplňkovém kódu.

Popis chování: Obvod obsahuje registry A0, A1, A2 a A3. Na začátku jsou všechny registry Ai vynulovány. Při stisku tlačítka B0 je provedeno nulování všech registrů Ai (vymazání historie). Při stisku tlačítka B1 je provedeno A3⇐A2, A2⇐A1, A1⇐A0, A0⇐swtch(7:0) a následně je spočítáno C⇐(A3+A2 +A1+A0)/4. Hodnota přepínačů je zobrazena na diodách (7:0), hodnota registru C je zobrazena na sedmisegmentovém displeji.

Podmínky řešení:

K řešení použijte pouze jednu sčítačku/odčítačku s minimálním nutným počtem bitů.

4.2.7. Úloha 7

Navrhněte obvod, který bude počítat výraz (A-B) + (C-D), kde A, B, C, D jsou 8bitová čísla v doplňkovém kódu

Popis chování: Obvod obsahuje registry A, B, C, D. Na začátku jsou všechny registry vynulovány. Stisk tlačítka B0 v jakémkoli okamžiku způsobí vynulování A, B, C, D a návrat do počátečního stavu (zadávání A). Hodnota přepínačů je po dobu zadávaní hodnot do registrů zobrazena na sedmisegmentovém displeji. Při prvním stisku tlačítka B1 je provedeno A⇐swtch(7:0), při dalším stisku tlačítka B1 je provedeno B⇐swtch(7:0) dále jsou obdobně načteny C⇐swtch(7:0) a D⇐swtch(7:0). Po zadání D je proveden výpočet výrazu a výsledek je zobrazen na sedmisegmentovém displeji. Případné přeplnění v průběhu výpočtu je indikováno stavem diody LED LD0.

Podmínky řešení: K řešení použijte pouze jednu sčítačku/odčítačku.

4.2.8. Úloha 8

Navrhněte obvod, který bude implementovat cyklický posuv 8bitového čísla vpravo o (-8 až +7 bitů)

Popis chování: V klidovém stavu obvod čeká na zadání 8-bitového operandu A pomocí přepínačů swtch(7:0), hodnota je zobrazena na sedmisegmentovém displeji. Operand je načten při stisku tlačítka B0. Následně obvod očekává délku posuvu L (4-bitové číslo v doplňkovém kódu) zadanou na swtch(3:0), kdy hodnota je zobrazena na sedmisegmentovém displeji. Délka posuvu je načtena při stisku tlačítka B0. Následně je proveden příslušný posuv a výsledek C je zobrazen na sedmisegmentovém displeji. Po opětovném stisku tlačítka B0 se obvod vrátí do počátečního stavu.

Podmínky řešení: Řešení může být sekvenční (formou posuvného registru) nebo kombinační (válcový posouvač).

4.2.9. Úloha 9

Navrhněte obvod, který bude implementovat cyklický posuv 8-bitového čísla vlevo nebo vpravo o 0-7 bitů

Popis chování: V klidovém stavu obvod čeká na zadání 8-bitového operandu A pomocí přepínačů swtch(7:0), hodnota je zobrazena na sedmisegmentovém displeji. Operand je načten při stisku tlačítka B0. Následně obvod očekává délku posuvu L (3-bitové číslo bez znaménka) zadanou na swtch(2:0), kdy hodnota je zobrazena na sedmisegmentovém displeji. Při stisku B0 je hodnota swtch(2:0) chápána jako délka posuvu vpravo, při stisku B1 je chápána jako délka posuvu vlevo. Poté je proveden příslušný posuv a výsledek C je zobrazen na sedmisegmentovém displeji. Směr posuvu je indikován pomocí stavu diod LED následovně:

svítí LD0 - posuv vpravo,

svítí LD1 - posuv vlevo.

Nyní obvod bude čekat na stisk tlačítka B0 a po jeho stisku se vrátí do počátečního stavu.

Podmínky řešení: Řešení může být sekvenční (formou posuvného registru) nebo kombinační (válcový posouvač).

4.2.10. Úloha 10

Navrhněte obvod, který bude implementovat cyklický, aritmetický a logický posuv 8-bitového čísla vpravo o 0-7 bitů

Popis chování: V klidovém stavu obvod čeká na zadání 8-bitového operandu A pomocí přepínačů swtch(7:0), hodnota je zobrazena na sedmisegmentovém displeji. Operand je načten při stisku tlačítka B0. Následně obvod očekává délku posuvu L (3-bitové číslo bez znaménka) zadanou na swtch(2:0), kdy hodnota je zobrazena na sedmisegmentovém displeji. Při stisku tlačítka B0 je hodnota swtch(2:0) chápána jako délka aritmetického posuvu v doplňkovém kódu vpravo, při stisku tlačítka B1 je chápána jako délka logického posuvu vpravo a při stisku tlačítka B2 je chápána jako délka cyklického posuvu vpravo. Poté je proveden příslušný posuv a výsledek C je zobrazen na sedmisegmentovém displeji. Typ posuvu je indikován pomocí stavu diod LED následovně:

svítí LD0 - aritmetický posuv,

svítí LD1 - logický posuv,

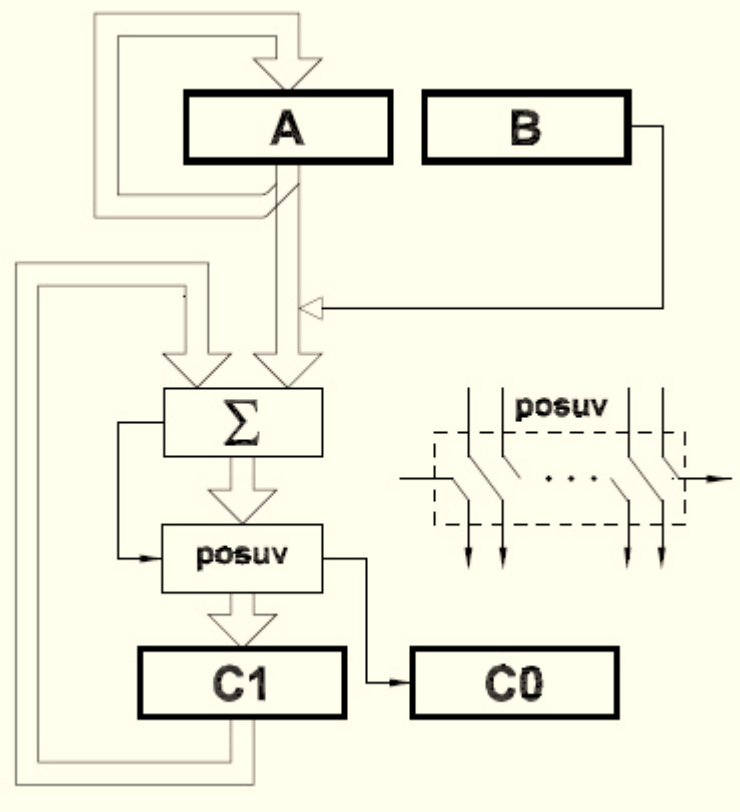
svítíLD2 - cyklický posuv.

Nyní obvod bude čekat na stisk tlačítka B0 a po jeho stisku se vrátí do počátečního stavu.

Podmínky řešení: Řešení může být sekvenční (formou posuvného registru) nebo kombinační (válcový posouvač).

4.3. Návrh řešení vzorové úlohy

Jako vzorovou úlohu jsem zvolil úlohu č.1 kapitola 4.2.1. Blokové schéma datové části je na obrázku 8. Toto blokové schéma je použito z přednášek předmětu JPO.

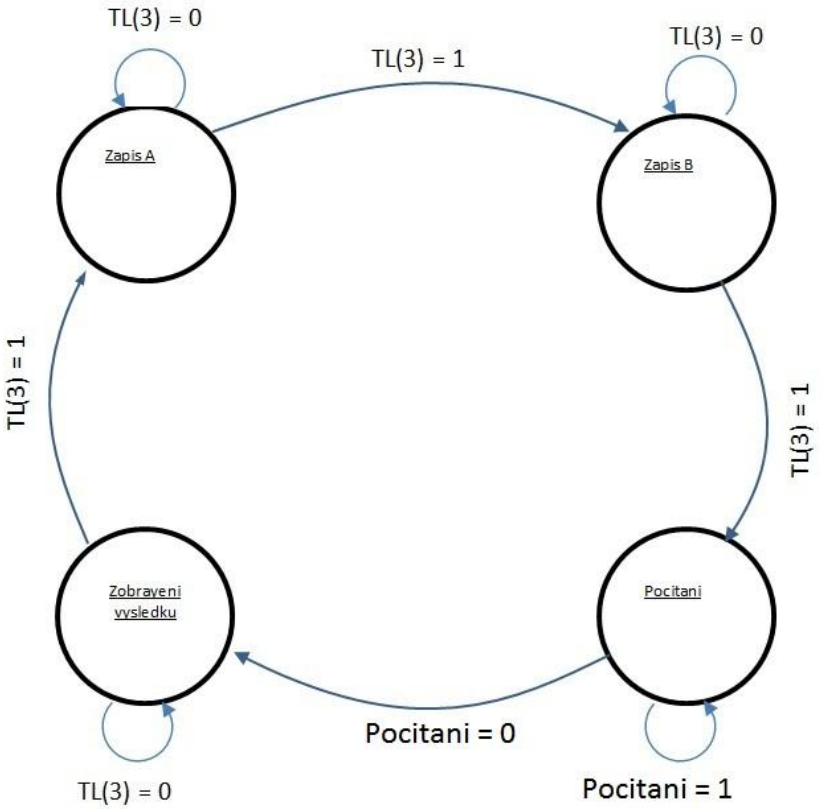


Obrázek 8: Blokové schéma vzorové úlohy

Řídící část se dá navrhnout jako graf přechodů. V tomto případě by mohl postačit automat se čtyřmi stavy (Obrázek 9).

- 1. stav – zápis do registru A
- 2. stav – zápis do registru B
- 3. stav – výpočet v osmi taktech
- 4. stav – zobrazení výsledku na sedmisegmentovém displeji

Následuje stisk tlačítka a přechod do prvního stavu.



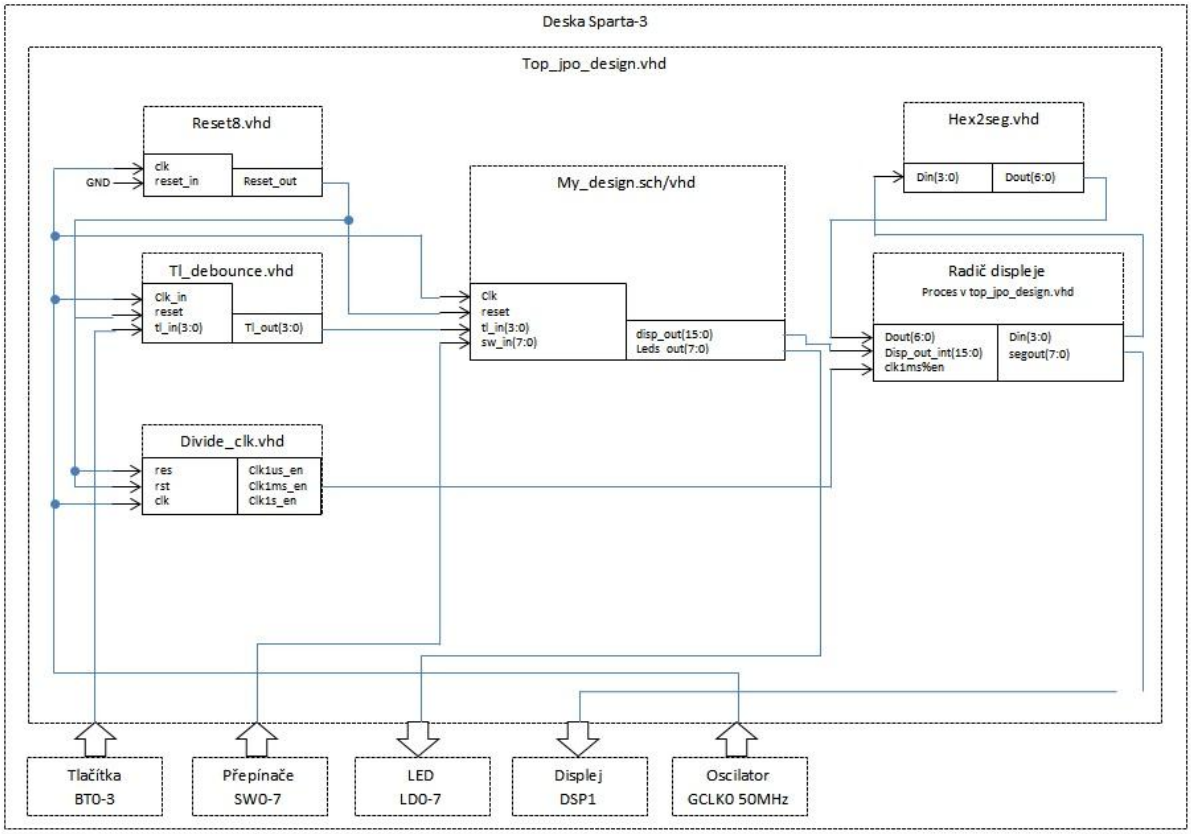
Obrázek 9: Návrh grafu přechodů vzorové úlohy

4.4. Návrh bloků knihovny

Pro řešení dílčích úloh jsou nezbytné tyto stavební bloky: registr, sčítačka, multiplexor, demultiplexor, posuv, klopný obvod.

5. Řešení

V této kapitole popisuji jednotlivé části projektu. Řeším vzorovou úlohu a popisuji jednotlivé bloky obsažené v knihovně, které jsem vytvořil a použil pro řešení dílčích úloh. Na obrázku 10 je nakresleno blokové schéma projektu.



Obrázek 10: Blokové schéma projektu

5.1. Řešení projektu

Celý projekt je řešen ve vývojovém prostředí ISE 13.3 od firmy Xilinx Inc a je psán ve VHDL kódu. Z požadavku na možnost řešení dílčích úloh pomocí schématu a VHDL kódu jsem vytvořil dva identické projekty s rozdílem jednoho souboru. Pro řešení pomocí schématu je projekt, který obsahuje soubor „my_design.sch“. Pro řešení ve VHDL kódu je projekt obsahující soubor my_design.vhd. Prázdný projekt se skládá z entity „Top_jpo_design“, pěti komponent a sedmi procesů v entitě „Top_jpo_design“. Výrazem „prázdný projekt“ je myšlen projekt, který neobsahuje řešení dílčí úlohy v souboru

„my_desgin.sch/vhd“. V entitě „Top_jpo_design.vhd“ jsou pomocí signálů pospojovány komponenty „reset8.vhd“, „tl_debounce.vhd“, „divide_clk.vhd“, „hex2seq.vhd“ a „my_design.sch/vhd“ (Obrázek 10).

5.1.1. Entita JPO_top_design

Tato entita obsahuje vstupní a výstupní porty. Na vstupu jsou napojena tlačítka tl_in(3:0), přepínače sw_in(7:0) a hodinový signál CLK. Na výstupu jsou napojeny LED diody leds(7:0) a výstup na displej segout(7:0) a selX(3:0). V souboru top_jpo_design.ucf je popsáno přiřazení pinů FPGA čipu na vstupní a výstupní signály v entitě.

Řadič displeje je řešen pomocí čtyř procesů v této entitě a komponenty hex2seq. Na obrázku 11 je zobrazen kód těchto čtyř procesů. Výstupem prvního procesu je 4 bitový signál sel_move, který udává, na jakou sedmisegmentovku budeme zobrazovat data. Druhý proces přiřazuje interní signál sel_move na výstupní signál selX a provádí jeho negaci, protože příslušná sedmisegmentovka je aktivní v logické nule (Obrázek 7). Ve třetím procesu se na základě hodnoty na signálu sel_move berou vždy 4 bity ze signálu disp_out_int, který je napojen na komponentu, v níž jsou řešeny dílčí úlohy. Tento signál o 4 bitech je připojen na signál vedoucí do komponenty hex2seq, kde se převádí BCD kód pro jednotlivé segmenty. Poslední proces přiřazuje výstup z komponenty hex2seq na výstupní signál připojený na společné vodiče sedmisegmentovky.

```
-- ================ move sel =======================
p_sel_move:process(clk, reset)
begin
   if (reset = '1') then
      sel_move <= (others => '0');
   elsif rising_edge(clk) then
      if (sel_move = x"0") then
         sel_move <= x"1";
      else
         if (clk1ms_en = '1') then
            sel_move <= sel_move (2 downto 0) & sel_move (3);
         end if;
      end if;
   end if;
end process;

--================ INSTANCE hex2seg ================
i_hex2seg : hex2seg

port map(
      din  => din,
      dout => dout
);
selX <= not sel_move;

p_sel:process(sel_move, disp_out_int)
begin
   case (sel_move) is
      when x"1" => din <= disp_out_int(3 downto 0) or x"0";
      when x"2" => din <= disp_out_int(7 downto 4) or x"0";
      when x"4" => din <= disp_out_int(11 downto 8) or x"0";
      when x"8" => din <= disp_out_int(15 downto 12) or x"0";
      when others => din <= "0000";
   end case;
end process;
segout <= all_in & dout when (sel_move=x"1") else '1' & dout;
```

Obrázek 11: Kód řadiče displeje

5.1.2. Komponenta hex2seq

Tato komponenta převádí BCD kód na jednotlivé segmenty (Tabulka 1). Vstup je 4 bitový signál din a výstupem je 7 bitový signál dout, odpovídající jednotlivým segmentům označených a, b, c, d, e, f, g (Obrázek 6).

Tabulka 1: Převod BCD kódu na segmenty

din	dout	znak
0000	1000000	0
0001	1111001	1
0010	0100100	2
0011	0110000	3
0100	0011001	4

0101	0010010	5
0110	0000010	6
0111	1111000	7
1000	0000000	8
1001	0010000	9
1010	0001000	A
1011	0000011	B
1100	1000110	C
1101	0100001	D
1110	0000110	E
1111	0001110	F

5.1.3. Komponenta reset8t

Tato komponenta nastaví signál reset do logické jedničky na dobu 8 hodinových taktů. Tento signál slouží k resetování všech bloků, jako jsou klopné obvody, registr atd. Je generován vždy pouze po spuštění projektu. Reset je důležitý pro nastavení všech bloků do definovaných počátečních hodnot.

5.1.4. Komponenta divide_clk

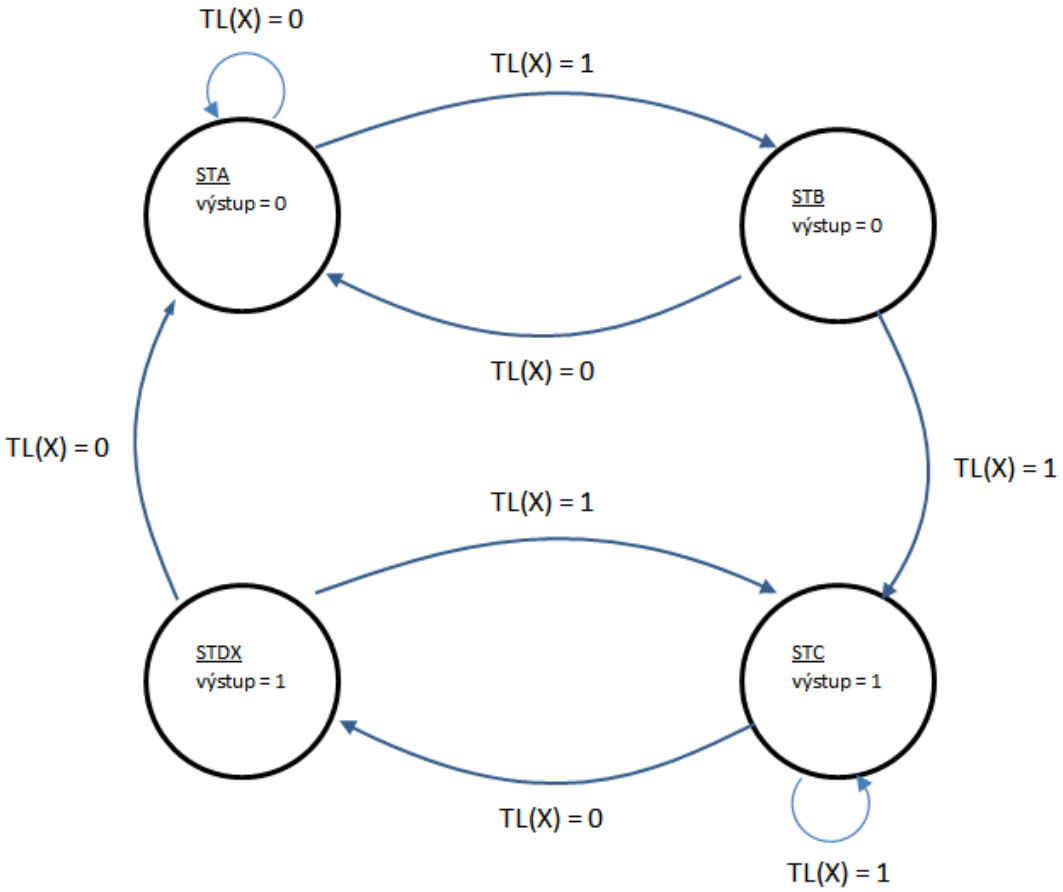
V této komponentě jsou odvozeny hodinové signály. Na přípravku je krystal běžící na frekvenci 50MHz. Od této frekvence jsou odvozeny 3 signály, a to „clk1us“, „clk1ms“ a „clk1s“. Délka jednoho hodinového cyklu je 20ns => 50*clk = 1us, z čehož je zřejmé, že délka hodinového signálu clk1us je 1 us, clk1ms je 1ms, clk1s je 1s.

5.1.5. Komponenta tl_debounce

Při řešení jsem zjistil, že je nezbytné, aby při stisku tlačítka, došlo k vytvoření pouze jediného impulsu. Tento impuls je dlouhý jako jeden hodinový takt tj. 20ns.

Základem je automat o 4 stavech, který odstraňuje zákmity na tlačítkách. Automat běží na frekvenci přibližně 47Hz. Při stisku tlačítka automat na náběžnou hranu hodin přejde do stavu STB. V případě, že je i na další náběžnou

hranu stisknuto stejné tlačítko, přejde automat do stavu STC a uvede výstup do logické „1“.



Obrázek 12: Automat odstranění zákmitů na tlačítkách

Na následujícím obrázku 13 je zobrazen kód, který modifikuje signál z výstupu automatu, a to tak, že ze signálu vytvoří jeden impuls o 20ns.

```
--================= edge detect ======================
edge_dt:process(clk_in, reset)
begin
   if (reset = '1') then
      tl_edge <= (others => '0');
   elsif rising_edge(clk_in) then
      tl_edge <= tlx_out;
   end if;
end process;

tl_outf : for I in 3 downto 0 generate
   tl_out(I) <= '1' when ((tl_edge(I) = '0') and (tlx_out(I) = '1')) else '0';
end generate;
```

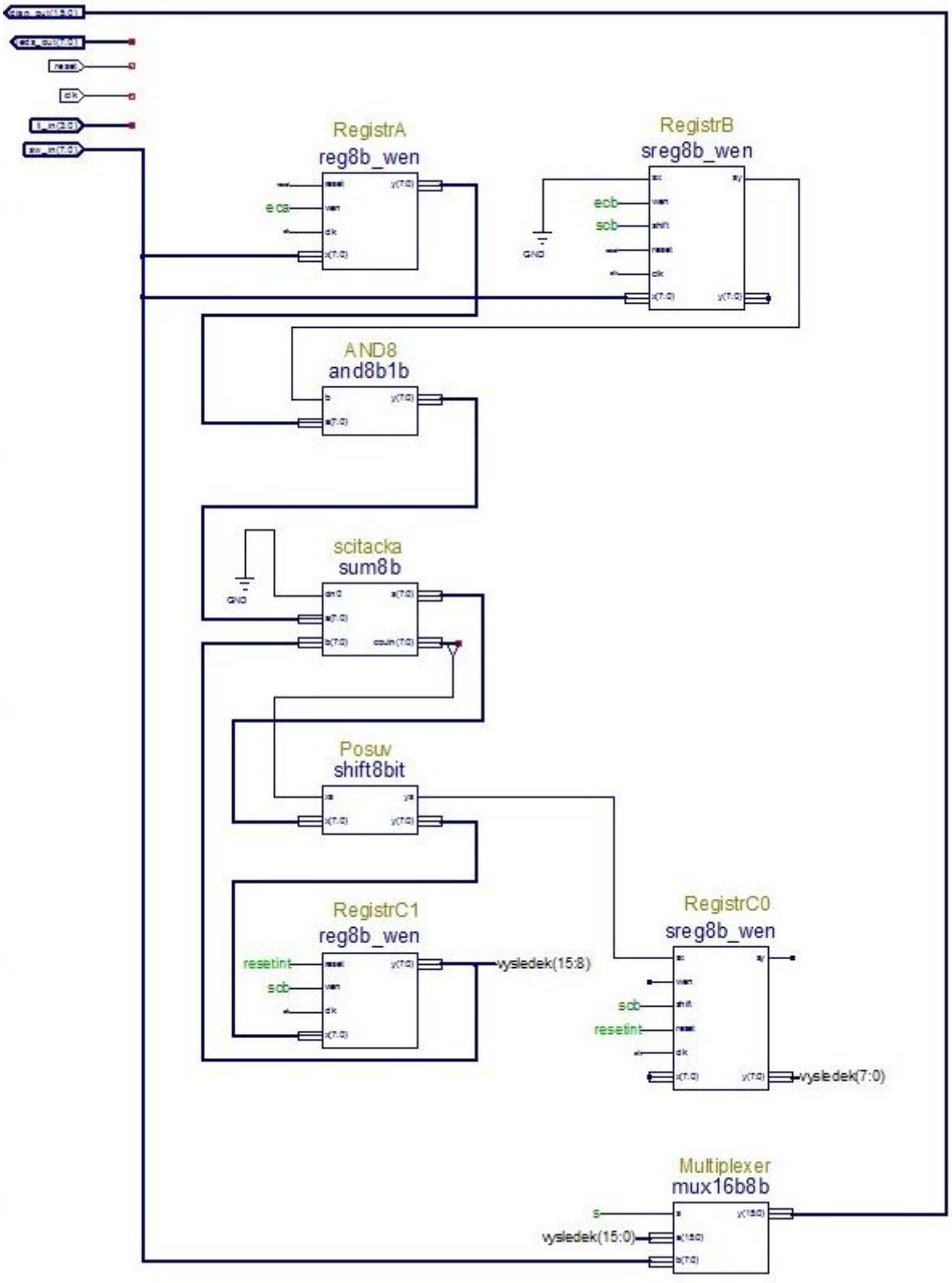
Obrázek 13: VHDL kód vytvářející jeden impuls

5.2. Řešení vzorové úlohy

Vzorovou úlohu jsem řešil ve schématu a snažil jsem se o co nejjednodušší zapojení. Zadání viz kapitola 4.2.1.

5.2.1. Datová cesta vzorové úlohy

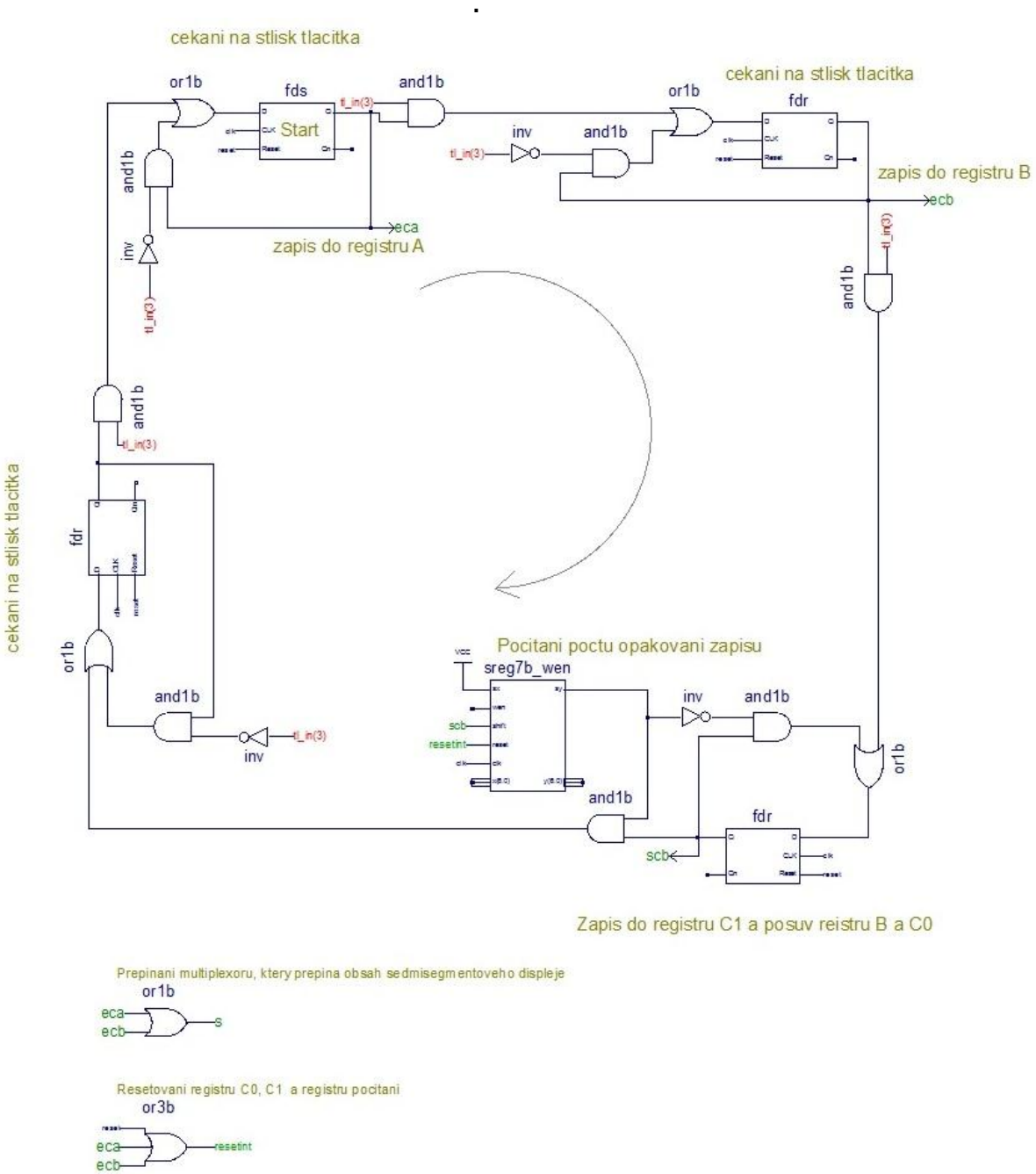
V řešení jsem použil celkem čtyři 8bitové registry, z toho jsou dva posuvné registry. Dále jsem použil jednu 8bitovou sčítačku, jeden 8bitový AND, jeden 8bitový posuv a multiplexor. Zapojení datové cesty je na obrázku 14. Pro větší přehlednost zapojení nejsou řídící signály kresleny, ale jsou pouze pojmenovány vstupy. Malým modrým písmem jsem označil signály reset a clk, které jsou připojeny na vstupní porty. Zeleně jsou označeny signály ECA, ECB, SCB, S a resetint, které generuje řídící část (automat).



Obrázek 14: Schéma datové cesty vzorové úlohy

5.2.2. Řídící část

Graf přechodů jsem realizoval jako řadič s řídícími řetězci. V zapojení jsem použil dva typy klopných obvodů. První má po přivedení signálu reset na výstupu logickou „0“, druhý má logickou „1“. Rozmístění klopných obvodů na obrázku 15 odpovídá rozmístění jednotlivých stavů v návrhu automatu (Obrázek 9).



Obrázek 15: Schéma zapojení automatu vzorové úlohy

Výstupy řadiče jsou řídící signály ECA, ECB, SCB, S a RESETINT. Počáteční stav je označen nápisem „Start“. V tomto stavu automat čeká na stisk tlačítka a po celou dobu drží signály ECA, S a RESETINT v logické „1“. Zde je nutno použít klopný obvod FDS, který po přivedení signálu reset na výstupu Q má logickou „1“. Ve všech ostatních stavech je použit klopný obvod FDR, který má po přivedení signálu „reset“ na výstupu „Q“ logickou „0“. Touto vhodnou volbou klopných obvodů je definován výchozí stav řadiče. Po stisku tlačítka přejde řadič do druhého stavu, kde opět čeká na stisknutí tlačítka a po celou dobu drží signály ECB, S a RESETINT v logické „1“. Ve třetím stavu zůstane po dobu osmi hodinových taktů. Po tuto dobu bude signál SCB v logické „1“. Během těchto osmi hodinových taktů proběhne operace násobení dvou 8bitových celých čísel bez znaménka. Po osmém hodinovém taktu řadič přejde do posledního stavu, kde je zobrazen výsledek na sedmisegmentovém displeji a čeká na stisk tlačítka. Po stisku tlačítka přejde řadič do prvního stavu. Tuto sekvenci stále opakuje.

6. Základní prvky knihovny

Knihovna základních prvků je dodána na přiloženém DVD disku. Jednotlivé prvky v knihovně jsou popsány pomocí VHDL kódu. V knihovně jsou obsaženy následující prvky: registr, klopný obvod, sčítačka, posuv, logická hradla, konstanty a speciální prvky pro usnadnění řešení úloh. Nyní podrobněji popíši jednotlivé prvky.

6.1. Registr

V knihovně jsou obsaženy dva typy registrů. Prvek s názvem reg*b_wen.vhd je registr pouze s paralelním vstupem a výstup, druhým prvkem je posuvný registr s názvem reg*b_wen.vhd. Znak „*“ v názvu souboru zastupuje číslo udávající počet bitů. Popis funkce je vysvětlen v tabulce 2 a 3.

Tabulka 2: Nbitový registr

Vstup				Výstup
x	wen	CLK	RESET	Y
	N	N	1	0
	0	↑	0	Bez změny
	1	↑	0	X

Tabulka 3: Posuvný registr

Vstup						Výstup	
x	sx	wen	shift	CLK	RESET	Y	sy
		N	N	N	1	0	0
		0	0	↑	0	Bez změny	Bez změny
		1	N	↑	0	X	x(0)
		0	1	↑	0	Sx & x(*:1)	X(1)

6.2. Klopný obvod

Pro řešení úloh jsem použil dva typy klopných obvodů D. Název těchto souborů je „FDS.vhd“ a „FDR.vhd“. Funkce těchto obvodů je zřejmá z tabulek 4 a 5.

Tabulka 4: Klopný obvod FDR

Vstup			Výstup	
D	CLK	RESET	Q	Qn
	N	1	0	1
	↑	0	D	D´

Tabulka 5: Klopný obvod FDS

Vstup			Výstup	
D	CLK	RESET	Q	Qn
	N	1	1	0
	↑	0	D	D´

6.3. Sčítačka

Sčítačka je navržena jako úplná sčítačka. Signály „a“ a „b“ jsou nbitové sčítance. Signál „Cin0“ je přenos z nižšího řádu. Signál „s“ je výsledek součtu. Signál „Coutn“ je nbitový signál, na kterém jsou přenosy do vyššího řádu ze všech bitů.

Tabulka 6: Nbitová úplná sčítačka

Vstup			Výstup		
a	b	Cin0	s	Coutn(0)	Coutn(n)
			a ⊕ b ⊕ cin	a(0)*b(0)+((a(0)⊕b(0))*cin)	a(n)*b(n)+((a(n)⊕b(n))*cin)

Sčítačka se skládá ze tří souborů sum1b.vhd, sumxb.vhd a sum*b.vhd. Znak „*“ v názvu souboru zastupuje číslo udávající počet bitů.

6.4. Posuv

Pro zjednodušení řešení částí úloh jsem vytvořil tyto posuvy:

- shift*bit.vhd

Znak „*“ v názvu souboru zastupuje číslo udávající počet bitů. Signál „x“ je paralelní nbitový vstup, „xs“ je sériový vstup. Signál „y“ je paralelní nbitový výstup a signál „ys“ je sériový výstup.

Tabulka 7:Nbitový posuv

Vstup		Výstup	
x	xs	y	Ys
		Xs & x(3:1)	X(0)

- alfashift8b.vhd

Signály „x“ a „y“ jsou 8bitové. Signál „alfa“ je 1bitový řídící vstup

Tabulka 8: 8bitový alfa posuv

Vstup		Výstup
x	alfa	y
	0	x & 0
	1	0 & x

- alfashiftq16b.vhd

Signál „x“ je 16bitový vstup a „q“ je 1bitový vstup. Signál „alfa“ je řídící 1bitový vstup. Signál „y“ je 16 bitový výstup.

Tabulka 9: 16bitový alfa posuv

Vstup			Výstup
x	q	alfa	y
		0	x
		1	x(14:0) & q

- betashift.vhd

Tento obvod rozšiřuje vstup o jeden bit. Signál „x“ je 8bitový vstup. Signál „y“ je 9bitový výstup.

Tabulka 10: Beta posuv

Vstup			Výstup
x	minus	beta	y
		0	minus & x
		1	x & ’0’

h

- BarShift8b.vhd

Je kombinační obvod, který posouvá číslo na signálu „a“ o hodnotu na signálu „b“. Signály „left“, „cy“, „ar“ a „lo“ jsou 1bitové řídící signály a udávají směr a typ posuvu.

Tabulka 11: Válcový posouvač

Vstup						Výstup
a	b	left	cy	ar	Lo	y
		0	0	0	1	logický posun a o hodnotu b vpravo
		0	0	1	0	aritmetický posun a o hodnotu b vpravo
		0	1	0	0	cyklický posun a o hodnotu b vpravo
		1	0	0	1	logický posun a o hodnotu b vlevo
		1	0	1	0	aritmetický posun a o hodnotu b vlevo
		1	1	0	0	cyklický posun a o hodnotu b vlevo

6.5. Konstanty

Pro potřeby připojit na signály stálou hodnotu jsem vytvořil tyto prvky obsahující uvedenou hodnotu v hexadecimální soustavě v názvu prvku. Rovněž v názvu prvku je uvedeno kolika bitový je výstup. Ukázka názvu prvku: „consr8b_80.vhd“. Výstupem je 8bitový signál s hodnotou „10000000“ .

6.6. Speciální prvky

Dále jsem vytvořil prvky zjednodušující zapojení datových cest. Tyto prvky zvyšují, případně snižují, počet vodičů signálu. V knihovně se názvy těchto prvků skládají z názvu operace, kolik bitů je na vstupu a kolik bitů je na výstupu. Ukázka názvů „reduce9bto8b.vhd“, „extend8bto10b.vhd“. Pravdivostní tabulky k těmto prvkům jsou součástí webové prezentace.

7. Testování

Funkčnost projektu jsem odzkoušel pomocí simulace v programu ISim od společnosti Xilinx inc, jenž je součástí vývojové prostředí ISE. Dále jsem odzkoušel funkčnost zapojení projektu na vývojové desce Spartan-3. Průběh simulace odpovídal zadání projektu.

7.1. Testování vzorové úlohy

Vzorovou úlohu jsem odzkoušel pomocí simulace v programu ISim. V tomto programu jsem testoval pouze soubor „my_design“ obsahující schéma zapojení vzorové úlohy. K tomuto účelu jsem vytvořil soubor „test_my_design.vhd“ (Obrázek 16), v němž přiřazuji logické hodnoty na signály tlačítek, přepínačů, hodin a reset. Nejdříve jsem nastavil výchozí hodnoty na signálech tlačítek a přepínačů, to znamená uvedení těchto signálu do logické „0“. Poté jsem nastavil signál reset na 8 hodinových taktů do logické jedničky. Signál musím nastavit, protože simuluji pouze soubor „my_design.sch“. Důvod proč tento signál musí být definován a použit, je vysvětlen v kapitole 5.1.3. Následně nastavím na signály přepínačů hodnotu 0xFF a signál tlačítka 3 nastavím na logickou „1“ na dobu jednoho hodinového taktu (kapitola 5.1.5). Pro lepší přehlednost průběhu simulace vyčkám cca 100nS a nastavím na signálech přepínačů novou hodnotu 0xAA. Opět uvedu signál tlačítka 3 do logické „1“. Nyní bude probíhat simulace výpočtu trvající 8 hodinových taktů. Po skončení simulace výpočtu uvedu signál tlačítka 3 opět do logické „1“ na dobu jednoho hodinového cyklu.

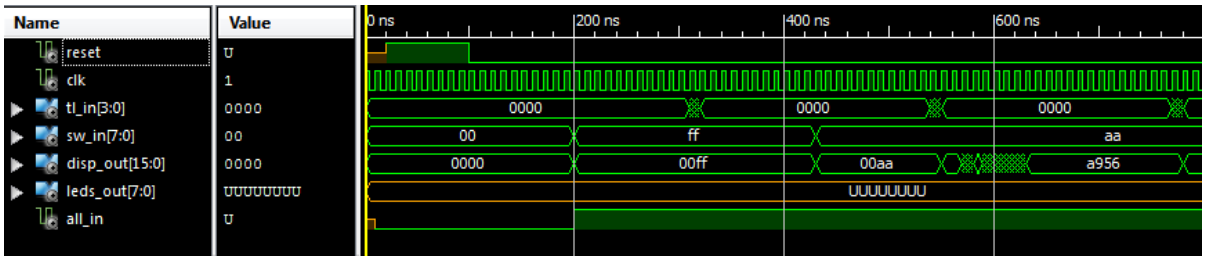
```
-- *** Test Bench - User Defined Section ***
   clkprocess : PROCESS
   BEGIN
      clk <='1';
      wait for 5 ns;
      clk <='0';
      wait for 5 ns;
   END PROCESS;


   tb : PROCESS
   BEGIN
      ----------------------------------------
      -- nastaveni vychozich hodnot
      tl_in <= "0000";
      sw_in <= "00000000";
      -- Signal reset
      wait for 20 ns;
      reset <= '1';
      wait for 80 ns;
      reset <= '0';
      ----------------------------------------
      -- nastaveni hodnoty na prepinacich2
      wait for 100 ns;
      sw_in <= "11111111";
      wait for 100 ns;
      -- stisk tlacitka
      wait for 10 ns; tl_in(3) <= '1'; wait for 10 ns; tl_in(3) <= '0'; wait for 10 ns;
      -- nastaveni nove hodnoty na prepinacich
      wait for 100 ns;
      sw_in <= "10101010";
      wait for 100 ns;
      -- stisk tlacitka
      wait for 10 ns; tl_in(3) <= '1'; wait for 10 ns; tl_in(3) <= '0'; wait for 10 ns;
      -- ted probiha vypocet operace nasobeni
      wait for 100 ns;
      wait for 100 ns;
      -- stisk tlacitka
      wait for 10 ns; tl_in(3) <= '1'; wait for 10 ns; tl_in(3) <= '0'; wait for 10 ns;


      WAIT; -- will wait forever
   END PROCESS;
-- *** End Test Bench - User Defined Section ***
```

Obrázek 16: Přirazení hodnot na signály v simulaci

Na následujícím obrázku 17 je ukázka průběhu simulace vzorové úlohy. V simulaci jsou vidět průběhy vstupních a výstupních signálů. Signál „leds_out“ nemá definovanou hodnotu. V této úloze nejsou použity indikační LED diody. Z průběhu signálu „disp_out“ jsou vidět hodnoty, které by byly vidět na displeji. Zadali jsme hodnoty 0xFF a 0xAA, což je po vynásobení 0xA956.



Obrázek 17: Ukázka simulace vzorové úlohy

8. Závěr

Hlavní přínos své práce vidím ve zpřehlednění a zjednodušení práce studentů v předmětu JPO, které povede k efektivnější výuce.

Při své práci jsem využil podle mého názoru nejefektivnější variantu přípravků osazených FPGA obvodem tak, aby odpovídala současným nárokům. Řešení úloh je možné jak ve schématu, tak s pomocí VHDL.

Detailní vzorový návrh řešení jedné z úloh, navržený na bázi ISE XILINX, může sloužit jako obecná šablona pro řešení dalších úloh, které jsem prezentoval. Závěry je však možno aplikovat obecně.

K této práci připojuji CD s prezentací obsahující návod k tomuto projektu. Pro ni jsem použil systém DokuWiki, který je běžně využíván pro studium na ČVUT. Navíc se na CD nachází prázdný projekt jako šablona pro starší verzi vývojového prostředí ISE 9.2 od firmy Xilinx Inc.

9. Literatura

[1] Doc. Ing. Alois Pluháček, CSc. Přednášky předmětu X36JPO, 2006

[2] Přednášky předmětu X36PNO

[3] Spartan-3 Starter Kit Board User Guide,
http://www.digilentinc.com/Data/Products/S3BOARD/S3BOARD_RM.pdf

[4] Online VHDL, http://www.vhdl-online.de/tutorial/englisch/inhalt.htm

10. Přílohy

1. Obsah CD disku
2. CD disk

Obsah CD disku

CD disk obsahuje zdrojové kódy prezentace pro systém DokuWiki, knihovnu základních prvků, vzorový projekt, prázdný projekt jako šablonu pro ISE 13.3 a 9.2, projekty obsahující řešení jednotlivých úloh.