Univerzitet u Kragujevcu
Fakultet inženjerskih nauka

Seminarski rad iz predmeta
Osnovi računarske tehnike 2

Tema:
Hello World

Student:
Aleksandar Kondić, 575/2015

Predmetni profesor:
Aleksandar Peulić

Kragujevac 2016.

Sadržaj:

1. UVOD

Živimo u vremenu u kome život čoveka postaje sve više povezan sa tehnologijom. Posmatrajući porast korišćenja uređaja kao što su mobilni telefoni poslednjih godina, prirodno je zapitati se kojim sve sferama života će tehnologija biti koristan dodatak. Da li će sve biti automatizovano? Da li će ljudi imati koristi od znanja iz programiranja I elektronike i van njihove karijere? Da li će postojati uređaji koji imaju jednostavne komande za brzo programiranje njihovog rada, da čak I neko ko ne zna parvo programiranje može da utiče na njihovo ponašanje?

Ovaj projekat neće se baviti, niti biti srodan, bilo kojim od prethodno postavljenih pitanja. Tema nije ni automatizacija, ni pravljenje lako programabilnog uređaja, niti bilo šta drugo što bi možda bilo korisno. Ali je svakako zanimljivo razmišlati o takvim pitanjima. U svakom slučaju, ovaj projekat je vezan za razumevanje i korišćenje nekih od mogućnosti koje nam pruža ciljana platforma za koju se razvija projekat. Prikazaćemo tekst “Hello World.” korišćenjem četiri sedmosegmentnih displeja i dodati nekoliko komandi za upravljanje tim tekstom.

2. ARHITEKTURA

Ciljana platforma za naš projekat je Digilent Nexys2, razvojna ploča koja sadrži Xilinx Spartan 3E FPGA (*eng. **F**ield-**P**rogrammable **G**ate **A**rray*, više reči o ovome kasnije) sa 500 000 logičkih kapija. Navedena ploča sadrži razne komponente koje pomažu u razvoju programa i logičkih kola.

Komponente koje će nama biti korisne su:

- Četiri sedmosegmentnih displeja;
- Osam prekidača i LED dioda;
- Četiri dugmića; i
- Oscilator od 50MHz.

Ostalie komponente koje sadrži ova ploča, o kojima treba obratiti pažnju:

- 16MB SDRAM-a;
- 16MB Flash ROM-a;
- Razni I/O portovi kao što su:
    - USB2 port, koji služi za napajanje i programiranje ploče;
    - PS/2 port;
    - Serijski portovi;
    - Četiri 12-pinska Pmod konektora; i
    - Hirose FX2 port

Glavni deo Nexys2 ploče, odnosno onaj deo ploče koji se zapravo programira jeste Xilinx Spartan 3E FPGA. Za razliku od mikrokontrolera, koji imaju definisanu arhitekturu za koju se pišu programi, FPGA je programabilno integrisano kolo, koje se sastoji od niza međusobno povezanih, osnovnih logičkih jedinica koji se zovu logički blokovi. Logički blokovi su programabilni i prilikom njihovog programiranja mogu da zamene bilo koju logičku kapiju kao što su AND, OR, NOR, XOR, itd.

FPGA se programira preko vrste programskih jezika koji se zovu HDL (*eng. **H**ardware **D**escription **L**anguage*). Mada HDL-ovi liče na standardne programske jezike, njihova funkcija je drugačija. Programski jezici služe za generisanje mašinskog koda koji može da izvršava procesor određene arhitekture, dok HDL konfiguriše logičke blokove unutar određenog FPGA. U suštini, preko HDL-a se prave logička kola koja su realizovana unutar FPGA.

Pošto se preko HDL-ova mogu praviti digitalna kola, moguće je realizovati procesor unutar FPGA, pa za njega generisati mašinski kod pomoću nekog od standardnih programskih jezika.

Posvetimo na trenutak pažnju jednom detalju arhitekture naše ploče. Ulazi za svaki od sedmosegmentih displejeva su međusobno povezani, tako da je nemoguće istovremeno prikazati različite sadržaje na različitim displejima. Ovaj problem se rešava upotrebom 4 dodatnih ulaza koji su označeni redom **AN0**, **AN1**, **AN2**, i **AN3**. Svaki od ovih ulaza je povezan za jedan sedmosegmentni displej,

i pozitivan signal ka nekim od ovih ulaza isključuje odgovarajući displej. Korišćenjem ove činjenice i oscilatora od 50MHz moguće je napraviti kolo koje u svakom trenutku vremena prikazuje sadržaj na samo jednom sedmosegmentnom displeju, dok su ostali isključeni. Ako se ova operacija dovoljno brzo izvrši za svaki pojedinačni displej, stvara se iluzija u ljudskom oku da se istovremeno prikazuju različiti sadržaji na različitim displejima.

3. PROJEKTNI ZADATAK

Naš projektni zadatak je da korišćenjem 2 diode i 4 sedmosegmentna displeja kao izlaze i 2 prekidača i 4 dugmad kao ulaze, prikažemo poruku “Hello World.” Pošto poruka ove dužine ne može stati na samo 4 displeja, u svakom trenutku vremena će se prikazivati određeni deo poruke, tako da se stvori efekat *scroll-ovanja*.

Da bismo učinili naš zadatak zanimljivijim, koristićemo ulaze koje nam obezbeđuje Nexys2 ploča da manipulišemo “tekstom” koji je prikazan na sedmosegmentnim displejevima.

Dalje sledi detaljan opis svih ulaznih jedinica i njihovih komandi digitalnom kolu:

- **BTN0**: povećavanje brzine skrolovanja;
- **BTN1:** smanjivanje brzine skrolovanja;
- **BTN2:** povećavanje ukupne dužine prikazanog teksta, dodavajući prazne karaktere na kraju “Hello World.” poruke po potrebi. Najveća moguca dužina teksta je 63 karaktera;
- **BTN3:** smanjivanje ukupne dužine prikazanog teksta, pri tom da se ne prikazuju poslednji karakteri poruke “Hello World.” ako je trenutna dužina teksta manja od dužine same poruke. Najmanja moguća dužina teksta je 4 karaktera;
- **SW1:** promena smera skrolovanja – dok je ovaj signal pozitivan, istovremeno se izvršava i blinkanje diode **LD1**, koja se nalazi direktno iznad prekidača;
- **SW0:** pauzira skrolovanje teksta i istovremeno uključuje diodu **LD0**, koja se nalazi direktno iznad prekidača. Ovaj prekidač takođe ima efekta na neke od ostalih ulaza i izlaza, kao što su:
    - Promena funkcije dugmeta **BTN0** u funkciju skrolovanja prikazanog teksta za jedno mesto udesno, omogućujući ručno skrolovanje kada je automatsko skrolovanje isključeno
    - Promena funkcije dugmeta **BTN1** u funkciju skrolovanja prikazanog teksta za jedno mesto ulevo
    - Ako se šalje signal iz prekidača **SW1**, umesto blinkanja, dioda LD1 konstantno svetli

4. REALIZACIJA PROJEKTNOG ZADATAKA

Za programiranje FPGA na našoj ploči koristićemo alat *Xilinx ISE Design Suite*. On nam omogućuje programiranje Xilinx Spartan 3E FPGA, koje ćemo izvršiti koristeći HDL jezik *Verilog*.

Vodeći se preporučljivim principima iz proceduralnog i objektno-orijentisanog programiranja – apstrakcijom i enkapsulacijom podataka, olakšaćemo naš zadatak pravljenjem hijerarhije modula.

4.1. Modul "karakter"

Na dnu hijerarhije se nalazi modul *karakter*, koji će kao ulaz dobiti osmobitni celobrojni podatak koji označava poziciju u tekstu "Hello World." Kao izlaz će vratiti jedan osmobitni podatak koji predstavlja slovo na datom mestu u stringu, u formatu koji omogućuje pravilan prikaz datog karaktera na sedmosegmentom displeju. Ako je ulazna pozicija mesto van stringa, podrazumeva se prazan karakter.

Da bi se vratio pravilan karakter, potrebno je implementirati neku vrstu grananja "programa" u odnosu na ulazni parametar koji određuje poziciju u stringu. Uobičajen način rešavanja ovakvog problema u Verilog-u je korišćenje instrukcije *if* ili *case*, međutim rad sa tim instrukcijama uglavnom podrazumeva korišćenje registara ili pravljenje sekvencijalnih kola. Moguće je realizovati kombinaciono logičko kolo koje rešava naš problem korišćenjem istrukcija *assign* i *(boolean) ? true_value : false_value*. (pogledati prilog 7.1.)

Pošto sedmosegmentni displej koji koristimo je predviđen za prikaz dekadnih i heksadekadnih cifara, postoji problem pri ispisivanju slova "W". Ovaj problem smo delimično rešili tako što smo za slovo "W" rezervisali 2 mesta na skupu displeja, tako da ono više izgleda kao spoj slova "I" i "U".

4.2. Modul "slova_display"

Ovaj modul kao ulaz prima osmobitni celobrojni podatak koji označava poziciju u stringu i još jedan osmobitni celobrojni podatak koji označava željenu dužinu teksta. Na izlazu daje 4 osmobitna podataka u vidu jednog 32-bitnog podatka, koji kazuju koja 4 karaktera treba ispisati na sedmosegmentnim displejima. Ovaj modul koristi modul *karakter* kao podmodul.

Treba voditi računa da ako je ulazna pozicija (manja ali) veoma blizu željenje dužine teksta, jedan deo displeja treba prikazati krajnje karaktere stringa, dok drugi deo displeja treba prikazati početne karaktere stringa, da bi se stvorio efekat skrolovanja.

Ovaj modul je takođe realizovan kao kombinaciono logičko kolo korišćenjem istrukcija *assign* i *(boolean) ? true_value : false_value*. (pogledati prilog 7.2.)

4.3. Glavni (Top) modul "tekst_scroll"

Ovaj modul je realizovan kao sekvencijalno logičko kolo. Kao ulazne signale prima jednobitne podatke koji su opisani u poglavlju 3 i povezani sa odgovarajućim fizičkim komponentama na ploči, uz dodatak jednobitnog signala *clk* koji je direktno povezan za oscilator na ploči. Kao izlaz ima jedan osmobitni podatak koji označava trenutni sadržaj jednog od sedmosegmentnih displeja, jedan četvorobitni podatak koji odgovara portovima na ploči označeni sa **AN0**, **AN1**, **AN2** i **AN3**; i n a kraju dva jednobitna signala koji odgovaraju diodama na ploči, koje oznalavaju stanja pauze i obrnutog smera automatskog skrolovanja.

Ovaj modul omogućava da kolo reaguje na ulazne signale onako kako je opisano u poglavlju 3. U detalje implementacije nećemo mnogo zalaziti (izvorni Verilog kod ovog modula može se naći u prilogu 7.3.) Napomenućemo samo da ovaj modul koristi modul *slova_display* kao podmodul, i da se sve operacije izvršavaju unutar jednog *always* bloka, koji čeka da signal *clk* bude na pozitivnoj ivici. Unutar ovog bloka se vrši detekcija pozitivne ivice signala koji potiču iz ulaza **BTN0**, **BTN1**, **BTN2** i **BTN3**, tako što se pamte vrednosti ovih signala iz prethodnog ciklusa u *always* bloku; i samo se pri pozitivnoj ivici signala izvršavaju odgovarajuće komande ovih ulaza.

5. ZAKLJUČAK

Tehnologija i programabilni uređaji jesu, i biće, veoma korisni za čovečanstvo. Postoje mnogi projekti koji su olakšali život i podigli životni standard u civilizaciji. A onda postoje projekti kao što je projekat koji je tema ovog seminarskog rada.

Ako išta, ovaj projekat je bar pomogao jednom studentu u sticanju znanja i iskustva iz programiranja FPGA uređaja.

6. LITERATURA

Digilent Nexys2 Board Refference Manual; revision July 11, 2011

Wikipedia; https://en.wikipedia.org/wiki/Main_Page

7. PRILOG KODOVI

7.1. Implemenacija modula "karakter"

```
`define karakter_space 8'b00000000
`define karakter_d 8'b01111010
`define karakter_E 8'b10011110
`define karakter_H 8'b01101110
`define karakter_L 8'b00011100
`define karakter_O 8'b11111100
`define karakter_r 8'b00001010
`define karakter_U 8'b01111100
`define karakter_W1 8'b01100000
`define karakter_W2 `karakter_U

`define karakter_0 `karakter_H
`define karakter_1 `karakter_E
`define karakter_2 `karakter_L
`define karakter_3 `karakter_L
`define karakter_4 `karakter_O
`define karakter_5 `karakter_space
`define karakter_6 `karakter_W1
`define karakter_7 `karakter_W2
`define karakter_8 `karakter_O
`define karakter_9 `karakter_r
`define karakter_10 `karakter_L
`define karakter_11 `karakter_d | 8'b1 // Dodaj tacku

module karakter(out, index);
output wire [7:0] out;
input wire [4:0] index;

assign out = (index == 0) ? `karakter_0 :
             (index == 1) ? `karakter_1 :
             (index == 2) ? `karakter_2 :
             (index == 3) ? `karakter_3 :
             (index == 4) ? `karakter_4 :
             (index == 5) ? `karakter_5 :
             (index == 6) ? `karakter_6 :
             (index == 7) ? `karakter_7 :
             (index == 8) ? `karakter_8 :
             (index == 9) ? `karakter_9 :
             (index == 10) ? `karakter_10 :
             (index == 11) ? `karakter_11 : `karakter_space;
endmodule
```

7.2. Implementacija modula "slova_display"

```
module slova_display(out, pozicija, duzina_teksta);
output wire [31:0] out;
input wire [7:0] pozicija;
input wire [7:0] duzina_teksta;

karakter k0(out[7:0],     (pozicija >= duzina_teksta) ? pozicija - duzina_teksta : pozicija);
karakter k1(out[15:8],    ((pozicija + 1) >= duzina_teksta) ? pozicija + 1 - duzina_teksta : pozicija + 1);
karakter k2(out[23:16],  ((pozicija + 2) >= duzina_teksta) ? pozicija + 2 - duzina_teksta : pozicija + 2);
karakter k3(out[31:24],  ((pozicija + 3) >= duzina_teksta) ? pozicija + 3 - duzina_teksta : pozicija + 3);

endmodule
```

7.3. Implementacija modula "tekst_scroll"

```
`define scroll_wait_min 524287
module tekst_scroll(out_digit, out_ans, pause_out, reverse_out, pause, reverse, faster, slower, longer, shorter,
clk);
output reg [7:0] out_digit = 0;
output reg [3:0] out_ans = 4'b0001;
output wire pause_out;
output wire reverse_out;
input wire pause, reverse, faster, slower, longer, shorter;
input wire clk;

reg reverse_blink = 1'b1;

assign pause_out = pause;
assign reverse_out = reverse & (pause | reverse_blink);

reg [7:0] duzina_teksta = 8'd16;
reg [7:0] pozicija = 8'd0;

reg [25:0] scroll_cnt = 0, scroll_wait = 24000000;
reg [22:0] blink_cnt = 0, blink_wait = 32767;
reg [25:0] reverse_blink_cnt = 0, reverse_blink_wait = 20000000;
reg [1:0] ans_pos = 2'd0;

wire [31:0] brojevi;

reg prethodno_faster = 1'b0;
reg prethodno_slower = 1'b0;
reg prethodno_longer = 1'b0;
reg prethodno_shorter = 1'b0;

slova_display disp(brojevi, pozicija, duzina_teksta);
```

```
always @(posedge clk)
begin
	//posedge detekcija kod drugih signala
	if(faster == ~prethodno_faster)
	begin
		if(faster == 1'b1)
		begin
			if(pause == 1'b1) begin //rucno skrolovanje
				pozicija = (pozicija > 0) ? pozicija - 1 : duzina_teksta - 1;
			end else begin
				scroll_wait = ((scroll_wait  / 2) < `scroll_wait_min) ? `scroll_wait_min :
(scroll_wait / 2);
			end
		end
		prethodno_faster = faster;
	end

	if(slower == ~prethodno_slower)
	begin
		if(slower == 1'b1)
		begin
			if(pause == 1'b1) begin //rucno skrolovanje
				pozicija = (pozicija + 1 >= duzina_teksta) ? 0 : pozicija + 1;
			end else begin
				scroll_wait = ((scroll_wait * 2) < scroll_wait) ? scroll_wait : (scroll_wait * 2);
			end
		end
		prethodno_slower = slower;
	end

	if(longer == ~prethodno_longer)
	begin
		if(longer == 1'b1) begin
			duzina_teksta = ((duzina_teksta + 1) == 0) ? duzina_teksta : duzina_teksta + 1;
		end
		prethodno_longer = longer;
	end

	if(shorter == ~prethodno_shorter)
	begin
		if(shorter == 1'b1) begin
			duzina_teksta = (duzina_teksta > 4) ? duzina_teksta - 1 : duzina_teksta;
		end
		prethodno_shorter = shorter;
	end

	scroll_cnt = scroll_cnt + 1;
	blink_cnt = blink_cnt + 1;

	if(blink_cnt >= blink_wait)
	begin
		blink_cnt = 0;
		ans_pos = ans_pos + 1;
```

```
                out_ans = ~(1 << ans_pos);
        end

        case(ans_pos)
                2'd0 : out_digit = ~brojevi[7:0];
                2'd1 : out_digit = ~brojevi[15:8];
                2'd2 : out_digit = ~brojevi[23:16];
                2'd3 : out_digit = ~brojevi[31:24];
        endcase

        if(!pause && (scroll_cnt >= scroll_wait))
        begin
                scroll_cnt = 0;

                if(reverse == 1'b1) begin
                        pozicija = (pozicija > 0) ? pozicija - 1 : duzina_teksta - 1;
                end else begin
                        pozicija = (pozicija + 1 >= duzina_teksta) ? 0 : pozicija + 1;
                end
        end

        if(reverse)
        begin
                reverse_blink_cnt = reverse_blink_cnt + 1;

                if(reverse_blink_cnt >= reverse_blink_wait)
                begin
                        reverse_blink_cnt = 0;
                        reverse_blink = ~reverse_blink;
                end
        end
        else
        begin
                reverse_blink_cnt = 0;
                reverse_blink = 1'b1;
        end
end

endmodule
```