APLICAŢIA 6
IMPLEMENTAREA CIRCUITELOR DE DEPLASARE BARREL SHIFTER

1. Rezumat

Acest laborator își propune implementarea unui circuit care realizează deplasarea cu un număr variabil de poziții – barrel shifter pe 4 biți. Acesta poate deplasa de la 0 la 3 poziții la dreapta. Se cere implementarea unui barrel shifter pe 4 biți cu deplasare la dreapta. Circuitul descris este un circuit combinațional care are 4 biți de intrare pentru operand, 2 intrări de selecție (indexul de shiftare) și 4 ieșiri corespunzătoare valorii deplasate la dreapta cu o valoare egală cu cea de la intrările de selecție.

Obiectivele lucrării

Obiectivul acestui laborator este acela de implementare a unui circuit combinațional clasic folosind construcția Verilog HDL *for ... generate*. De asemenea se cere descrierea Verilog a modulului și verificarea funcționării corecte folosind placa Nexys-2.

Obiective tehnice

1. Familiarizare cu construcţia *for .. generate.*
2. Implementare circuit folosind circuite simple de tip multiplexor 2-1.
3. Sinteză și implementare design pe placa FPGA Nexys-2.

Timp necesar

2-3 ore

Pregătirea pentru laborator

- Citiți documentul înainte de a începe realizarea practică.
- Salvați output-urile pentru fiecare cerință sau anunțați cadrul didactic în vederea prezentării rezultatelor.

Echipamente și Materiale

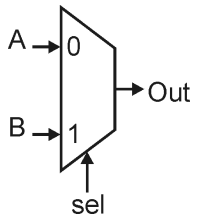
Acces la software-ul Xilinx

Necesar	Cantitate
Software ISE® WebPACK™ 14.4 de pe pagina de WEB Xilinx, www.xilinx.com	1
Plugin Digilent (www.digilent.com)	1
Placă Digilent Nexys 2	1
Cablu PMOD	1
Placă de expansiune - PMODSw	1

2. Unitate de deplasare de tip barrel shifter

2.1 Multiplexorul 2-1

Multiplexorul este un circuit logic combinaţional ce conectează ieşirea aceestuia la una din cele *n* intrări. Selecţia unuia din cele *n* intrări se face cu ajutorul a celor $\lceil \log_2 n \rceil$ intrări de selecţie. Multiplexorul poate fi privit ca un comutator digital. Cel mai simplu tip de multiplexor este cel cu 2 intrări de date, o intrare de selecţie şi o ieşire.



sel	Out
0	A
1	B

Fig. 6.1 – Schema bloc şi tabel de funcţionare a multiplexorului cu 2 intrari (sursa 1.[11])

Acest tip de multiplexor se poate modela în mod simplu folosind atât descrieri comportamentale, precum şi descrieri flux de date, exemplificate prin cele două fragmente de cod Verilog de mai jos.

```
module mux2_1_flux_date
        (input a,b,
        input sel,
        output o);

        assign o = sel ? b : a;

endmodule

module mux2_1_comp
        (input a,b,
        input sel,
        output reg o);

        always
          @(a,b,sel)
        begin
           o = a;
           if(sel)
                o = b;
           else
                o = a;
        end

endmodule
```

2.2 Unitate de deplasare de tip barrel shifter

Circuitele de tip barrel shifter sunt folosite pentru operaţiile de deplasare a unei valori binare cu un număr variabil de poziţii. Practic, având două numere *A* (pe *n* biți), respectiv *B* (pe *m* biţi), un barrel shifter e folosit pentru operaţii de tip *A>>B, A<<B*, respectiv operaţii de rotire a lui *A* cu *B*. În continuare vom descrie un astfel de circuit ce operează shiftări logice la dreapta, de tip *A>>B*. Un astfel de circuit este compus din mai multe niveluri de multiplexoare 2-la-1. Structura unui barrel shifter pe 4 biți cu 2 linii de intrare pentru selecție este prezentată în continuare:

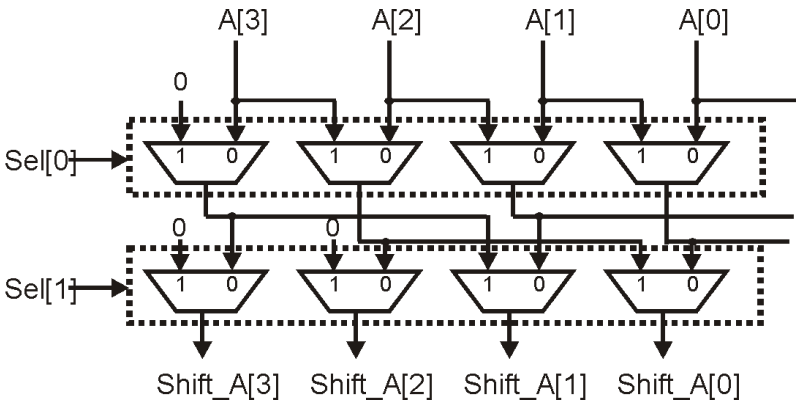


Fig. 6.2 – Schema bloc barrel shifter cu index pe 2 biți și operand pe 4 biți (sursa 1.[11])

Un barrel shifter funcţionează în felul următor:

- Prima linie de multiplexoare face selecţie între A şi A shiftat la dreapta cu o poziţie. Selecţia este realizată de rangul cel mai puţin semnificativ a lui B. Practic se va shiftarea la dreapta cu $2^0 * B[0]$ poziţii.
- A doua linie de multiplexoare face o selecţie între rezultatul primei linii şi acesta shiftat la dreapta cu două poziţii, selecţie realizată prin intermediul lui B[1]. Practic se realizează o shiftare cu $2^1 * B[0]$ poziţii. În mod analog, se comportă fiecare linie de multiplexoare.

2.3 Construcţia *for..generate*

Foarte utile pentru design-urile care necesită replicarea unor subcomponente sunt instrucţiunile de tip *generate*. Acestea permit selecţia sau replicarea unor secvenţe de cod în faza premergătoare simulării (faza de elaborare / elaboration time) în care toate modulele din design sunt conectate şi referinţele ierarhice sunt soluţionate. Un bloc *generate* este marcat de cuvintele cheie *generate … endgenerate*.
Trei tipuri de construcţii sunt posibile:

- Generate-loop
- Generate-case
- Generate-condiţionat

Generate-loop este folosit pentru replicare de cod în faza de elaborare a ierarhiei de module în vederea simulării. Este necesară declararea unei

variabile de tip *genvar* care să fie folosită drept contor al buclei for din generate-loop.

În continuare un exemplu pentru o poartă *xor* word-gate.

```
module xor_w
        (  input wire [7:0] in1,in2,
        output wire [7:0] x
        );

        //declararea variabilei pentru generate-loop
        genvar i;

        //generate-loop-ul
        generate
        for (i=0; i<8; i=i+1)
                begin: multiplicare_xor
                        assign x[i] = in1[i] ^ in2[i];
                end
        endgenerate

endmodule
```

Există şi posibilitatea realizării unei selecţii în faza de elaborare, prin testarea unei condiţii *statice* (depinde exclusiv de evaluarea unor constante şi/sau parametri, nu semnale sau variabile). Similar se poate folosi şi o construcţie de tip *case* pentru a realiza o selecţie condiţionată multiplă.
Dacă modificăm exemplul anterior, astfel încât primii 4 biţi sunt rezultatul unui *xor*, iar următorii ai unui *and* logic, codul Verilog HDL se modifică astfel :

```
… … …
genvar i;
generate
  for (i = 0; i < 8; i = i + 1)
   begin: multiplicare_xor
        if ( i <4 )
          assign x[i] = in1[i] ^ in2[i];
        else
          assign x[i] = in1[i] & in2[i];
```

```
   end
endgenerate
… … …
```

Pentru circuitul de tip barrel shifter descris anterior avem nevoie de generarea de 2 niveluri de circuite de tip multiplexor:

```
//declararea de semnale care sa capteze valorile de la fiecare nivel de
mux-uri
wire  [3:0] mux_li, mux_l0_1, mux_li_0; //2 niveluri de semnale de 4 biți
fiecare

assign mux_l0_1 = {1'b0, a[3], a[2], a[1]};
assign mux_l1_1 = {1'b0, 1'b0, mux_li [3], mux_li [2]};


genvar i;
generate
  for (i=0; i<4; i=i+1)
   begin: mux_level_i
        //primul nivel de mux-uri
        mux_2_1 mux_instance0 (.a(mux_l0_1[i]), .b(a[i]), .sel(sel[0]),
.out(mux_li[i]));
        //al doilea nivel de mux-uri
        mux_2_1 mux_instance1 (.a(mux_l1_1 [i]), .b(mux_li [i]),
.sel(sel[1]), .out(shift_out[i]));

   end
endgenerate
```

3. Implementarea unui circuit barrel shifter cu operand pe 4 biţi, selecţia pe 2 biţi

Se cere implementarea unui circuit barrel shifter folosind multiplexoare 2-1. Se va folosi oricare dintre descrierile aferente multiplexorului. Se va completa codul Verilog HDL de mai sus in vederea implementării circuitului barrel shifter.

Pas 1 – Crearea unui proiect Xilinx ISE şi descrierea unei circuit barrel shifter

Succint vor fi punctate etapele realizării unui proiect nou:

- Pentru pornire ISE: deschideți un terminal și tastați *ise*
- Creați un proiect nou în directorul workspace: *barrel_shifter_hex*
- În continuare realizați utilizând limbajul de descriere hardware Verilog componenta din figura de mai jos. La *Hierarchy* în tab-ul de *Design* selectați *Project→New source* deschide fereastra *New Source Wizard*. Pentru implementarea folosind descrierea Verilog HDL alegeți la *Select Source Type – Verilog Module.*

  Proiectul va avea patru fișiere sursă*:*
  - *mux2_1* – este modulul ce modelează multiplexorul
  - *barrel_shifter* – este modulul aferent barrel shifter şi va instanţia modulul *mux2_1*
  - *hex_7display* – este modulul care va implementa decodificatorul pentru afişajul cu 7 segmente;
  - *barrel_shifter_hex* – instanţiază o unitate *barrel_shifter* şi un *hex_7displaz*

- Adăugați la proiect un fișier de tip testbench . *Project→New source* deschide fereastra *New Source Wizard*, alegeți la *Select Source Type – Verilog Test Fixture*

Fișierul testbench este următorul.

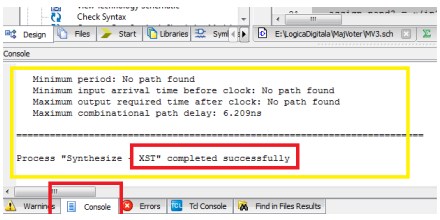
```
module barrel_shifter_tb;

    // Inputs
    reg [3:0] data; //intrări date
    reg  [1:0] sel;   //intrări de selecție

    //Outputs
    wire [3:0] data_out;

    // Instantiate the Unit Under Test (UUT)
    // realizati instanta pentru circuitul testat
    barrel_shifter uut (.a(data), .sel(sel), .shift_out(data_out));
```

```
    initial begin
        // Initialize Inputs
        data=0;
        sel = 0;

        // Wait 100 ns for global reset to finish
        #100;
        data=4'b0101; //numărul de deplasat
        sel=2'b10; //deplasare cu 2 pozitii
        #20;
        data=4'b0101; //numărul de deplasat
        sel=2'b01; //deplasare cu 1 pozitie
        #20;
        //toate combinațiile posibile

    end

endmodule
```

Simulaţi circuitul folosind simulatorul ISIM.

Pas 3 – Sinteza circuitului

La *Hierarchy* în tab-ul de *View* selectați *Implementation*.Se poate observa că fișierul testbench a dispărut.
În continuare selectați modulul care doriți să-l setați ca și top-level (cel al cărui design va fi programat pe placa FPGA) – *barrel_shifter_hex*.
În tabul de *Design* dați click pe *Synthesize->Run*. Alternativa este să dați dublu click pe *Synthesize*.
Remarcați la output-ul din tab-ul *Console*, finalizarea cu succes a operației de sinteză.



Pas 3 – Implementarea circuitului

Înainte de a trece la configurarea design-ului pe placă mai aven nevoie de crearea fişierului .UCF. Placa folosită este Nexys-2 cu FPGA-ul Spartan3-E 500 FG320. Toate aceste informații se găsesc specificate în manualul plăcii (Nexys-2 Board Reference Manual).
Circuitul pe care dorim sa-l verificăm foloseşte 6 comutatoare pentru intrări şi 1 afişaj cu 7 segmente.
Va fi folosită componenta PmodSWT care este conectata la interfața PMOD2, atunci trebuie consultat manualul aferent acestuia şi trebuie identificați pinii pentru conectorul PMOD2 ai plăcii Digilent Nexys2.

Pentru placa Nexys-2, din manual studiați specificația pentru PMOD2 şi extrageți informațiile referitoare la pini. Vor fi folosiţi pinii indicaţi mai jos:

Table 3: Nexys2 Pmod Connector Pin Assignments

Pmod JA		Pmod JB		Pmod JC		Pmod JD	
JA1: L15	JA7: K13	JB1: M13	JB7: P17	JC1: G15	JC7: H15	JD1: J13	JD7: K14[1]
JA2: K12	JA8: L16	JB2: R18	JB8: R16	JC2: J16	JC8: F14	JD2: M18	JD8: K15[2]
JA3: L17	JA9: M14	JB3: R15	JB9: T18	JC3: G13	JC9: G16	JD3: N18	JD9: J15[3]
JA4: M15	JA10: M16	JB4: T17	JB10: U18	JC4: H16	JC10: J12	JD4:; P18	JD10: J14[4]

Notes: [1] shared with LD3 [2] shared with LD3 [3] shared with LD3 [4] shared with LD3

În ceea ce priveşte afişajul cu 7 segmente, se va folosi modulul de expansiune PMODSSD. Acesta va fi conectat la portul de expansiune JC.

Table 3: Nexys2 Pmod Connector Pin Assignments

Pmod JA		Pmod JB		Pmod JC		Pmod JD	
JA1: L15	JA7: K13	JB1: M13	JB7: P17	JC1: G15	JC7: H15	JD1: J13	JD7: K14[1]
JA2: K12	JA8: L16	JB2: R18	JB8: R16	JC2: J16	JC8: F14	JD2: M18	JD8: K15[2]
JA3: L17	JA9: M14	JB3: R15	JB9: T18	JC3: G13	JC9: G16	JD3: N18	JD9: J15[3]
JA4: M15	JA10: M16	JB4: T17	JB10: U18	JC4: H16	JC10: J12	JD4:; P18	JD10: J14[4]

Notes: [1] shared with LD3 [2] shared with LD3 [3] shared with LD3 [4] shared with LD3

Va fi creat fişierul *barrel_shifter_hex.ucf*.
Se va continua prin implementarea şi crearea fişierului de configurare .bit.

Pas 4 – Configurare placă FPGA

Ultimul pas constă în descărcarea design-ului pe placă.

Din Terminal tastați:

djtgcfg prog -d *Nexys2* -i 0 -f barrel_shifter_hex.bit

4. Exerciţii

Realizati pașii indicați. Completați liniile de cod lipsă. Realizați design-ul şi verificați funcționarea pentru barrel shiter-ul modelat.

Bibliografie:

[1] Xilinx - Xilinx UG695 ISE In Depth Tutorial - http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_1/ise_tutorial_ug695.pdf , 2012

[2] C. Kief, A. Vera, A. Haddad, Q. Cao. COSMIAC FPGA Tutorials http://cosmiac.org/thrust-areas/education-and-workforce-development/fpga/ate-developed-material/.

[3] J. F. Wakerly – Digital Design: Principles and Practices, 3rd Edition, Prentice Hall, 2000

[4] J. Bhasker - A Verilog HDL Primer, Third Edition - Star Galaxy Publishing, 2005

[5] P. Chu - RTL Hardware Design Using VHDL: Coding for Efficiency, Portability, and Scalability, Wiley – IEEE Press, 2006

[6] S. Brown, Z. Vrsaniec - Fundamentals of Digital Logic with Verilog Design - McGraw-Hill, 2007

[7] R. Haskell, D. Hanna - Introduction to Digital Design Using Digilent FPGA Boards – Block Diagram/Verilog Examples – LBE Books, 2009

[8] Digilent Nexys 2 Reference Manual - https://www.digilentinc.com/Data/Products/NEXYS2/Nexys2_rm.pdf

[9] Digilent PMODSSD Reference Manual - http://www.digilentinc.com/Products/Detail.cfm?Prod=PMOD-SSD

[10] Digilent PMODSWT Reference Manual - https://www.digilentinc.com/Data/Products/PMOD-SWITCH/Pmod%20SWT_rm.pdf

[11] O. Boncalo, A. Amăricăi. “Proiectarea circuitelor digitale folosind Verilog HDL – Analiza si Sinteza”. Editura Politehnica, 2011.