APLICAŢIA 3
IMPLEMENTAREA CIRCUITELOR COMBINAŢIONALE ÎN LIMBAJUL VERILOG HDL SUMATORUL PE 2 BIŢI

1. Rezumat

Acest laborator își propune prezentarea succintă a conceptelor de Verilog HDL necesare descrierii circuitelor combinaționale.
Vor fi trecute în revistă: tipurile de modele de descriere (flux de date, comportamental, structural, mixt) specifice unui circuit cu exemplificare pentru celula de însumare pe 1 bit, modulul Verilog HDL.
Circuitul descris este o celulă de însumare pe 1 bit cu 3 intrări. Se va folosi Xilinx ISE® pentru sinteză, simulare şi implementare.

Obiectivele lucrării

Obiectivul acestui laborator este acela de cunoștere a noțiunilor fundamentale de Verilog HDL necesare descrierii unui circuit combinațional. De asemenea folosirea acestore în vederea realizării unui design simplu: celula de însumare pe un bit.

Obiective tehnice

1. Familiarizare cu placa Nexys-2
2. Realizarea unui design simplu: sumator pe 2 biți folosind ISE® WebPACK™ și simularea lui.
3. Sinteza și implementare design.
4. Configurare 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. Scurtă introducere în limbajul VerilogHDL

Verilog-ul este un limbaj de descriere hardware (HDL) care permite modelarea unui sistem digital la diferite niveluri de abstractizare: algoritmic, comportamental, poartă logică, sau chiar la nivelul comutator. Acesta se bucură de o mare popularitate în rândurile designerilor de hardware şi a producătorilor de tool-uri CAD, datorită simplităţii acestuia, ce permite proiectarea şi verificarea rapidă a circuitelor.
Dintre principalele caracteristici ale acestui limbaj amintim:

- Definirea de primitive logice elementare pentru porţile logice
- Permite modelarea circuitelor sub formă de flux de date, comportamental, structural şi mixt;
- Descrierea circuitului se poate face la diferite niveluri de abstractizare: nivelul comutator, nivelul poartă logică, nivelul RTL (register transfer level), nivel algoritmic şi transactional.
- Modelarea execuţiei concurente şi a timpului

Modelarea unui sistem digital in limbajul Verilog se poate realiza în mai multe feluri. Elementul central in Verilog este constituit de noţiunea de modul. Acesta poate fi folosit pentru a modela atât elemente simple, cât şi sisteme complexe. În esenţă un design Verilog complex poate fi asimilat

cu o multitudine de module interconectate, reprezentând subansamblele sale. În cadrul unui modul se pot întalni următoarele stiluri de modelare:

- Modelarea sub formă de flux de date;
- Modelarea comportamentală;
- Modelarea structurală;
- Modelarea mixtă;

În continuare, vor fi prezentate în mod succinct modelarea de tip flux de date, modelarea comportamentală şi modelarea structurală pentru o celulă sumator pe 1 singur bit.

2.1 Modelarea prin flux de date

Aceasta se bazează pe instrucţiunea *assign*, ce reprezintă instrucţiunea de atribuire continuă. Ea realizează evaluarea continuă (reevaluează expresia de fiecare dată când unul dintre operanzi îşi modifică valoarea) a expresiei din dreapta egalului, şi atribuie noua valoare unui semnal sau vector de semnale de tipul *net*. Sintaxa este următoare:

assign #intarziere semnal = expresie;

Instrucţiunile de tip *assign* sunt executate concurrent, adică nu este importantă ordinea în care acestea au fost scrise, execuţia acestora fiind în paralel (simultan).

Un exemplu de modelare bazată pe flux de date este prezentat mai jos:

```
`timescale 1 ns/1 ps

module full_adder_cell
        (input a, b, c_in,
        output s, c_out)

        assign #2 c_out = (a & b) | (a & c_in) | (b & c_in);
        assign #2 s = a^b^c_in;

endmodule
```

După cum se poate observa în exemplul de mai sus, modelarea de tip flux de date se potriveşte atunci când se doreşte descrierea circuitelor prin ecuaţiile logice aferente acestuia.

2.2 Modelarea comportamentală

Are la bază construcţii de tip *always*. Acestea permit o descriere de nivel înalt a comportamentului unui dispozitiv hardware prin folosirea unor instrucţiuni care se execută secvenţial similare cu cele întâlnite în limbajele de programare, precum cele de decizie (*if, case*), repetitive (*for, while*), etc.
O clauză *always* trebuie să fie prevăzută cu un fel de control pentru timp. Acesta poate fi reprezentat fie de întârzieri (*wait* – aşteaptă un interval de timp), fie de producerea unui eveniment (aşteaptă modificarea unui semnal din lista de senzitivităţi). În lipsa acestora, codul este executat la infinit.
Semnalele ce iau valori în cadrul blocurilor *always* sunt de tipul *reg*. În Verilog HDL, cuvântul cheie *reg* denotă un semnal sau variabilă care poate memora o valoare la un moment dat. E important să nu fie confundat cu noțiunea de registru!

În exemplu de mai jos, apare descrierea comportamentală a celulei sumator complet pe 1 bit.

```
`timescale 1 ns/1 ps

module full_adder_cell
        (input a, b, c_in,
        output reg s, c_out)

        always
        @(a,b,c_in)
        begin
                s = a^b^c_in;
                c_out = (a & b) | (a & c_in) | (b & c_in);
        end

endmodule
```

2.3 Modelarea structurală

Modelarea structurală se bazează pe instanţierea unor module în interiorul modului descris, precum şi conectarea modulelor instanţiate prin semnale de tip *wire*. Instanţierea unor module se poate face fie folosind module anterior descrise sau modelate, fie folosind primitive.

În exemplul de mai jos (Fig. 3.1), este prezentată modelarea celulei sumator complet pe 1 bit folosind primitive pentru porţile logice elementare.

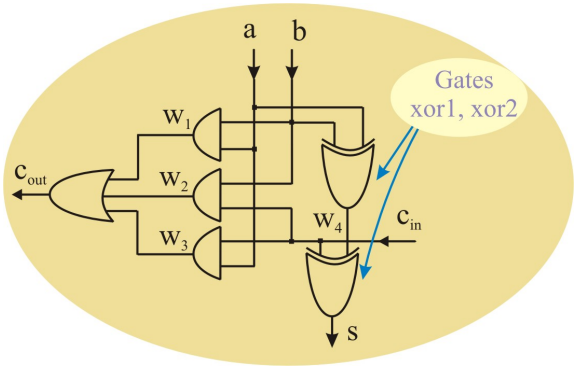


Fig. 3.1 – Celula sumator descriere structură 1.[10])

```
`timescale 1 ns/1 ps

module full_adder_cell
        (input a, b, c_in,
        output reg s, c_out)

        wire w1, w2, w3;
        wire w4;

        and a1(w1, a, b);
        and a2(w2, a, c_in);
        and a3(w3, b, c_in);

        or o1(c_out, w1,w2,w3);

        xor x1(w4, a,b);
```

```
    xor x2(s, s1, c_in);
endmodule
```

3. Implementarea unui circuit sumator pe 2 biţi

Reamintim paşii necesari implementării circuitului pentru dispozitive FPGA Xilinx:

Pas 1: Descrierea design-ului

- Circuitul va fi descris in Verilog HDL. Se va folosi descrierea comportamentală sau flux de date pentru celula sumator complet pe 1 bit, respective descriere strcuturală pentru sumatorul pe 2 biți alcătuit din 2 celule sumator complet.

Pas 2: Sinteza design-ului

- Translatarea codului Verilog HDL și/sau fișierelor cu scheme într-un format standard – fișier EDIF.

Pas 3: Implementarea design-ului

- Translatare, mapare pe componentele FPGA-ului, alocarea componentelor specifice dispozitivului FPGA, și rutare în vederea stabilirii interconexiunilor dintre componente. Ieșirea acestui process este un fișier (.bit) folosit pentru programarea FPGA-ului.

Pas 4: Configurare echipament Xilinx

- Descărcarea fișierului pe FPGA

Pas 1 – Crearea unui proiect Xilinx ISE şi descrierea unei circuit sumator pe 2 biţi

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: *Adder_2bits*
- 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 două surse*:*

- *adder_1bit_cell* – aceasta reprezintă descrierea celulei sumator complet pe 1 bit; ea va avea în interfaţă 3 semnale de intrare *(a,b,c_in)* şi 2 semnale de ieşire *(s, c_out)*; descriera acestui module este una de tip flux de date sau comportamental
- *adder_2bits* – este modulul care va implementa circuitul dorit; descrierea acestuia va fi structurală. Astfel, vom folosi 2 instanţe de celule sumator complet pe 1 bit; mai jos este descrisă interfaţa unui astfel de sumator. În descrierea de mai jos, trebuiesc completate semnalele interne modulului, precum și cele 2 instanțe de celule însumare.

```
`timescale 1 ns/1 ps

module adder_2bits
        (input c_in,
        input [1:0] a, b;
        output [1:0] s,
        output c_out);

//DE COMPLETAT LISTA DE SEMNALE INTERNE

//DE COMPLETAT DESCRIEREA STRUCTURALA

endmodule
```

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

Fişierul testbench este următorul.

```
    module adder_2bits_tb;

        // Inputs
        reg [1:0] a; //operand 1
        reg [1:0] b; //operand 2
```

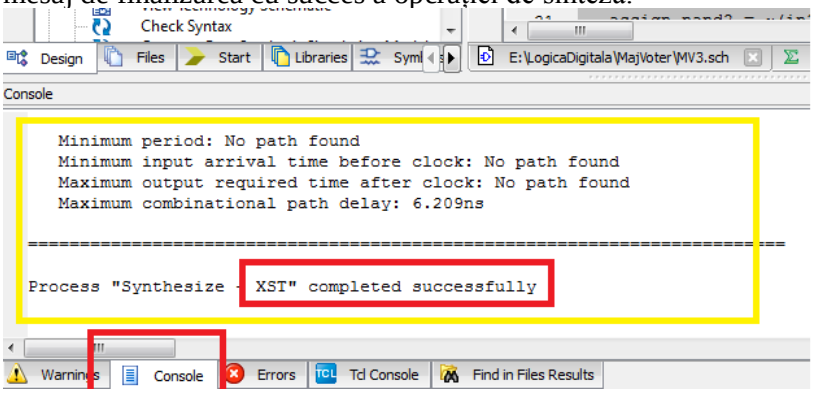
```
    //Outputs
    wire [1:0]sum;
    wire cout;
    // Instantiate the Unit Under Test (UUT)
    adder_2bits uut (
            .a(a),
            .b(b),
            .c_in(1'b0),
            .s(sum),
            .c_out(cout)
    );

    initial begin
            // Initialize Inputs
            a = 0;
            b = 0;

            // Wait 100 ns for global reset to finish
            #100;

    end
  always //toggle inputs for two bit adder
  begin
    #25 a = a+1'b1;
    #50 b = b+1'b1;
  end

endmodule
```

Simulaţi circuitul folosind simulatorul ISIM.

Pas 2 – Sinteza circuitului

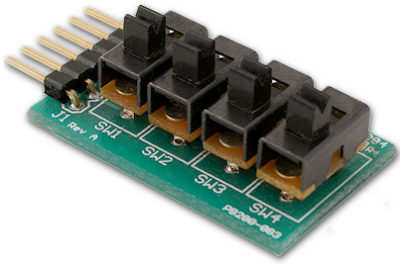
La *Hierarchy* în tab-ul de *View* selectați *Implementation*. Se poate observa că fișierul testbench a dispărut.
În continuare selectați modulul care doriți să-l setați ca și top-level (cel al cărui design va fi programat pe FPGA) – *adder_2bits*.
În tabul de *Design* dați click pe *Synthesize→Run*. Alternativa este să dați dublu click pe *Synthesize*.

Remarcați output-ul din tab-ul *Console*. Dacă nu sunt erori apare un mesaj de finalizarea cu succes a operației de sinteză.



Pas 3 – Implementarea circuitului

Înainte de a trece la configurarea design-ului pe placă mai avem nevoie de crearea fişierului .UCF. Acesta precizează modalitatea prin care pinii din design-ul nostru sunt conectati la pinii externi ai plăcii FPGA. 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 4 comutatoare pentru intrări şi 3 led-uri pentru ieşiri.
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 Nexys-2.
Conform manualului componenta PmodSWT se prezintă astfel:



(sursa **Error! Reference source not found.**)
Aceste nume, împreună cu pinii aferenți mai pot fi identificați:

- De pe placă;
- Din Master UCF (de la producătorul plăcii);

Pentru placa Nexys-2, din manual studiați specificația pentru PMOD2 și extrageți informațiile referitoare la pini. Vor fi folosiți pinii indicați 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

(sursa **Error! Reference source not found.**)

Va fi creat fişierul *adder_bit.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 adder_2bits.bit

4. Exerciţii

Realizati pașii indicați. Completați liniile de cod lipsă. Realizați design-ul și construiți tabelul de adevăr pentru un sumator cu operanzi pe 2 biți.

Verificați funcționarea corectă a design-ului pe placă!

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

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