APLICAŢIA 4
MINIMIZAREA FUNCŢIILOR LOGICE
METODA KARNAUGH

1. Rezumat

Acest laborator își propune prezentarea succintă a tehnicii de minimizare bazate pe diagrame Karnaugh, precum și sinteza cu porți logice a expresiei funcției astfel obținute.
Se cere minimizarea funcțiilor logice care calculează dacă două numere pe 2 biți sunt: mai mic sau egal, respectiv mai mare sau egal.
Circuitul descris este un circuit combinațional care are 2 biți de intrare pentru operanzi (fiecare dintre operanzi) și 2 ieșiri corespunzătoare funcțiilor logice.

Obiectivele lucrării

Obiectivul acestui laborator este acela de cunoștere a tehnicii de minimizare bazate pe diagrame Karnaugh. De asemenea se cere descrierea Verilog a modulului și verificare funcționării corecte folosind placa Nexys-2.

Obiective tehnice

1. Minimizarea funcțiilor logice folosind diagrame Karnaugh.
2. Realizarea unui design simplu pentru funcțiile minimizate.
3. Sinteza și implementare design pe placa FPGA Nexys-2.

Timp necesar

2-3 ore

Pregătirea pentru laborator

- Citiţi 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. Minimizarea funcţiilor logice bazată pe diagrame Karnaugh

2.1 Etapele proiectării unui circuit logic combinaţional

Circuitele logice combinaţionale sunt circuite ale căror valori de ieşire depind numai şi numai de valoarea de intrare (nu şi de stările precedente ale acestora).
Proiectarea acestor tipuri de circuite presupune următorii paşi:

1. Se pleacă de la descrierea în cuvinte a enunţării problemei
2. Se construieşte tabelul de adevăr corespunzător.
3. Se scriu ecuaţiile în forma canonică.
4. Se procedează la simplificarea ecuaţiilor.
5. Se implementează schema aferentă.

În continuare, vom exemplifica prin implementarea unui circuit de votare folosit in circuite TMR (*Triple Modular Redundancy*). Acest circuit analizează trei intrări şi are ca ieşire valoarea majoritară (2-din-3). Cele trei intrări vor fi denumite *a,b,c* iar ieşirea *f*.

Tabelul de adevăr aferent acestui circuit este prezentat mai jos.

a	b	c	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Ecuaţia logică în formă canonică pe baza tabelului este:

$f = \bar{a}bc + a\bar{b}c + ab\bar{c} + abc$

Ecuaţia minimizată obţinută în urma unei minimizări este:

$f = ab + ac + bc$

2.2 Termeni folosiţi la minimizare

Prin minimizare se înţelege simplificarea ecuaţiilor logice booleene. Este un procedeu de importanţă mare în proiectarea circuitelor digitale deoarece reduce numărul de porţi pe de o parte, iar pe de altă parte este redus numărul de intrări al porţilor.

Pentru o mai bună înţelegere a tehnicilor de minimizare se vor explicita termenii folosiţi în tabelul 4.1.

Tabelul 4.1 – Definiţii

Termen	Definiţie
Literal	Variabilă booleană sau complementul ei
Termen Produs (Product Term)	Literal sau produs logic (ŞI) între mai mulţi literali
Termen Sumă (Sum Term)	Literal sau sumă logică (OR) între mai mulţi literali
Sum of Products (SOP)	Sumă logică (OR) între mai mulţi termeni produs
Products of Sums (POS)	Produs logic (ŞI) între mai mulţi termeni sumă
Minterm	Caz particular de termen produs, care conţine toate variabilele de intrare o singură dată
Maxterm	Caz particular de termen sumă, care conţine toate variabilele de intrare o singură dată

Sumă de produse canonică	Sumă logică (OR) de acei mintermi aferenţi rândurilor din tabelul de adevăr al funcţiei de ieşire unde aceasta are valoarea 1 logic
Produs de sume canonic	Produs logic (ŞI) de acei mintermi aferenţi rândurilor din tabelul de adevăr al funcţiei de ieşire unde aceasta are valoarea 0 logic

Astfel, pentru funcţia logică *f* descrisă în exemplul de mai sus, avem următoarele:

Termen	Exemplu
Literal	$a, b, c, \bar{a}, \bar{b}, \bar{c}$
Termen Produs (Product Term)	$ab, abc, a\bar{c}, b, \bar{b}c, \bar{a}\bar{b}\bar{c}$
Termen Sumă (Sum Term)	$a+b, a+b+c, a+\bar{c}, b, \bar{b}+c, \bar{a}+b+\bar{c}$
Sum of Products (SOP)	$ab + abc + a\bar{c} + b + \bar{b}c + \bar{a}\bar{b}\bar{c}$
Products of Sums (POS)	$(\bar{a}+\bar{b})(\bar{a}+\bar{b}+\bar{c})(\bar{a}+c)\bar{b}(b+\bar{c})(a+\bar{b}+c)$
Minterm	$\bar{a}\bar{b}\bar{c}, \bar{a}\bar{b}c, \bar{a}b\bar{c}, \bar{a}bc, a\bar{b}\bar{c}, a\bar{b}c, ab\bar{c}, abc$
Maxterm	$a+b+c, a+b+\bar{c}, a+\bar{b}+c, a+\bar{b}+\bar{c},$ $\bar{a}+b+c, \bar{a}+b+\bar{c}, \bar{a}+\bar{b}+c, \bar{a}+\bar{b}+\bar{c}.$
Sumă canonică de produse	$f = \bar{a}bc + a\bar{b}c + ab\bar{c} + abc$ $f = m_3 + m_5 + m_6 + m_7$
Produs canonic de sume	$f = (a+b+c)(\bar{a}+\bar{b}+c)(\bar{a}+b+\bar{c})(a+\bar{b}+\bar{c})$ $f = M_0 + M_1 + M_2 + M_4$

2.3 Diagrama Karnaugh

Metoda de minimizare bazată pe diagrame Karnaugh are la bază proprietatea de distibutivitatea (x(y+z)=xy+xz), respectiv proprietatea complementului ($x+\bar{x}=1$).
Diagramele Karnaugh se aplică atât pentru ecuaţiile logice descrise sub formă canonică de sumă de produse (SOP), cât şi pentru ecuaţiile logice descrise sub formă canonică de produs de sume (POS). Datorită faptului

că la ora actuală sunt folosite aproape in exclusivitate ecuaţiile logice sub formă de SOP, vom trata doar minimizarea pentru SOP.
Diagramele Karnaugh constituie o matrice de pătrate cu proprietatea ca două celule vecine corespund unor mintermi adiacenţi. Doi vectori sunt adiacenţi dacă diferă valoric printr-un singur bit.
În continuare este prezentată construcţia diagramelor Karnaugh pentru funcţii cu 3 variabile de intrare, respectiv 4 variabile de intrare:

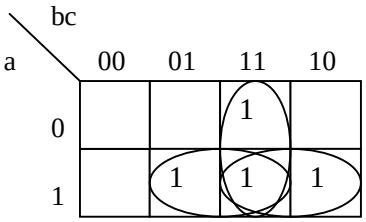
- Diagrama Karnaugh aferentă funcţiilor cu 3 variabile de intrare

a	00	01	11	10
0	000	001	011	010
1	100	101	111	110

- Diagrama Karnaugh aferentă funcţiilor cu 4 variabile de intrare

ab \ cd	00	01	11	10
00	0000	0001	0011	0010
01	0100	0101	0111	0110
11	1100	1101	1111	1110
10	1000	1001	1011	1010

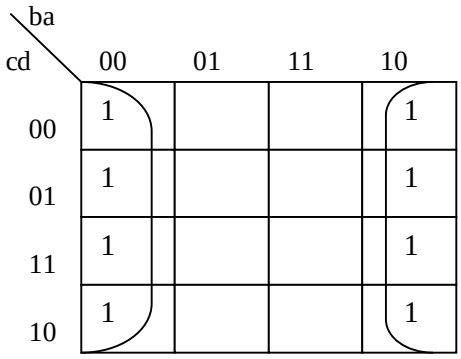
În urma introducerii mintermilor în diagramă conform tabelului de adevăr, se încearcă formarea unor grupe de mintermi bazate pe reguli de adiacenţă. Aceste grupări vor avea în diagrame forma unor dreptunghiuri/pătrate ce vor conţine mintermi. Astfel din totalul de *m* variabile booleene a funcţiei, termenul asociat grupării formate va conţine *n* variabile. Cele *m-n* variabile rămase sunt cele care apar în toţi mintermi ai grupării doar în formă nenegată sau doar în formă negată. Pentu funcţia *f* considerată anterior avem următoarea diagramă Karnaugh.



$f = ab + ac + bc$

Dacă la o astfel de grupare nu mai pot fi adăugaţi mintermi înseamnă că s-a obţinut un *implicant prim*. Dacă un anumit implicant prim conţine cel puţin un minterm care nu poate apare în alt implicanţi primi atunci acesta este un *implicant prim esenţial*. Ecuaţia minimizată va conţine toţi implicanţii primi esenţiali, si uneori si implicanţi primi neesenţiali, astfel încât toate celule marcate cu 1 logic să fie acoperite.

Să se minimizeze funcţia $p = f(a,b,c,d) = \sum(0,2,4,6,8,10,12,14)$



$p = \overline{a}$

În acest caz avem un singur implicat prim esenţial format din toţi mintermi din definiţia funcţiei.

Nu toate funcţiile logice sunt definite complet. Pentru unele valori ale variabile de intrare funcţia nu este specificată. În acest caz, pentru acele valori se spune că funcţia are „valoarea” *don’t care*, acest lucru este specificat de regulă în tabelul de adevăr printr-un *d*. În acest caz, în diagrama Karnaugh se va lua în considerare valoarea care ne convine pentru d (0 sau 1) şi care permite realizarea unei grupări mai mari de mintermi. Atenţie: aceste celule nu TREBUIE acoperite! Ele sunt considerate în grupări, în măsura in care sunt utile.

3. Implementarea comparatoarelor pe 2 biţi

Se cere implementarea comparatoarelor pe 2 biţi. Acestea au ca şi intrare 2 vectori fiecare pe 2 biţi(*a* şi *b*), şi 2 ieşiri, una care indică că vectorul *a* este mai mic sau egal decât *b*, iar cea de-a doua ieşire indică că vectorul *a* este mai mare sau egal decât *b*.

Pas 1 – Tabelul de adevăr şi minimizarea funcţiilor logice

Se va completa tabelul de adevăr aferent celor două funcţii logice. Tabelul de adevăr va avea 4 variabile de intrare.
Se vor minimiza cele 2 funcţii logice şi se va scrie forma lor minimizată.

Pas 2 – Crearea unui proiect Xilinx ISE şi descrierea unei circuit comparator 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: *comp_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 o singură surse*:*

- *comp_2bits* – este modulul care va implementa circuitul dorit; descrierea acestuia va fi una de tip flux de date, bazat pe funcţiile logice minimizate; interfaţa modulului este dată mai jos

```
`timescale 1 ns/1 ps

module comp_2bits
        (input [1:0] a, b;
        output sm_eq,
        output gr_eq);

//DE COMPLETAT LISTA DE SEMNALE INTERNE
```

```
//DE COMPLETAT DESCRIEREA MODULULUI

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*

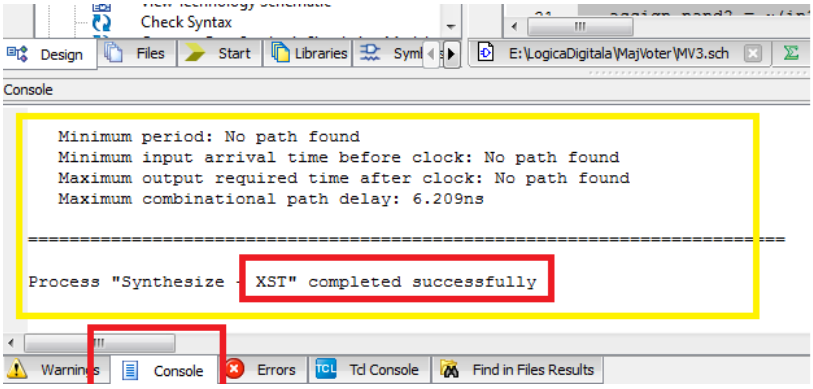
Fișierul testbench este următorul.

```
module comp_2bits_tb;

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

    //Outputs
    wire gr_eq;
    wire sm_eq;

    // Instantiate the Unit Under Test (UUT)
    // completați cu instanta pentru circuitul testat

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

   always //toggle inputs for two bit comparator
   begin
            //adaugați combinatiile de numere
  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 FPGA) – *comp_2bits*.
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 4 comutatoare pentru intrări şi 2 led-uri pentru ieşiri.
Va fi folosită componenta PmodSWT care este conectata la interfața PMOD2, atunci trebuie consultat manualul aferent acestuia și trebuie identificați pinii pentru conectorul PMOD2 ai plăcii Digilent Nexys-2.

Pentru placa Nexys-2, din manual studiați specificația pentru PMOD2 și 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

Creați fişierul *comp_2bit.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 comp_2bit.bit

4. Exerciţii

Realizati paşii indicați. Completați liniile de cod lipsă. Realizați design-ul şi construiți tabelul de adevăr pentru un unitatea comparator 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.