

Mérési segédlet a Mikroelektronika című tárgy 7. laboratóriumi gyakorlatához

Tartalomjegyzék

1	Standard logikák adatlapjai	4
2	A digitális tervezés menete Verilog nyelven, FPGA áramkörben implementálva	5
3	Logikai értékek hozzárendelése kimeneti lábakhoz	7
3.1	Feladat leírása	7
3.2	Megvalósítása Verilog nyelven	7
4	Négy kapcsoló állapotának hozzárendelése kimeneti lábakhoz	7
4.1	Feladat leírása	7
4.2	Megvalósítása Verilog nyelven	7
5	SN74HC151 – 8:1 Multiplexer	8
5.1	Feladat leírása	8
5.2	Megvalósítása Verilog nyelven	8
5.3	RTL szintű kapcsolási rajz	9
5.4	FPGA kihasználtsága	9
5.5	Szimulációs tesztkörnyezet	8
5.6	A teszt eredménye	9
6	SN74HC139 – Dual 2:4 Demultiplexer	11
6.1	Feladat leírása	11
6.2	Megvalósítása Verilog nyelven, logikai kapukkal	11
6.3	RTL szintű kapcsolási rajz	12
6.4	Megvalósítása Verilog nyelven, switch-case szerkezettel	13
6.5	FPGA kihasználtsága	14
6.6	RTL szintű kapcsolási rajz	14
6.7	Szimulációs tesztkörnyezet	11
6.8	A teszt eredménye	12
7	SN74LS49 – BCD-to-Seven Segment Decoder	15
7.1	Feladat leírása	18
7.2	Megvalósítása Verilog nyelven	18
7.3	RTL szintű kapcsolási rajz	19
7.4	FPGA kihasználtsága	21
7.5	Szimulációs tesztkörnyezet	19
7.6	A teszt eredménye	19
8	SN74HC163 – Szinkron, 4 bites számláló	15
8.1	Feladat leírása	15

8.2	Megvalósítása Verilog nyelven.....	15
8.3	RTL szintű kapcsolási rajz.....	16
8.4	FPGA kihasználtsága.....	17
8.5	Szimulációs tesztkörnyezet	16
8.6	A teszt eredménye.....	16
9	SN74HC166A – 8 bites shift regiszter	22
9.1	Feladat leírása	22
9.2	Megvalósítása Verilog nyelven.....	22
9.3	RTL szintű kapcsolási rajz.....	24
9.4	FPGA kihasználtsága.....	24
9.5	Szimulációs tesztkörnyezet	23
9.6	A teszt eredménye.....	23

1 Standard logikák adatlapjai

Minden elektronikai eszköz rendelkezik egy olyan adatlappal, amit a gyártó készített és bemutatja az eszköz működését, működési feltételeit és sok hasznos paramétert, ami szükséges ahhoz, hogy a saját áramkörünkbe be tudjuk illeszteni az eszközt és rendeltetésszerűen használni. Tipikusan a következő adatok szerepelnek benne:

- I. Az eszköz rövid leírása
- II. Abszolút maximum értékek
- III. Ajánlott működési értékek
- IV. Elektromos karakterisztikák
- V. „Live-insertion” specifikáció
- VI. Időzítési követelmények
- VII. Kapcsolási karakterisztikák
- VIII. Zaj karakterisztikák
- IX. Működési karakterisztikák
- X. Paraméterek mérésének leírása

Számunkra a legfontosabbak az első oldalon található információk, mivel ilyen, 74-es sorozatú alapáramkörök (SN74XYYYY) tervezésével fogunk foglalkozni. Ezeket az áramköröket a Digitális Technika című tárgyban részben már megismerték, ezért feltételezzük, hogy a logikai működéssel tisztában vannak. A laborgyakorlatokon az áramköröket hardverleíró nyelv (Hardware Description Language, HDL) segítségével valósítjuk meg (digitális IC tervezés) és egy speciális, programozható ASIC áramkört használva (FPGA) fogjuk kipróbálni a gyakorlatban. Minden tárgyalt eszköz kereskedelmi forgalomban kapható, a mai napig használt eszköz. A legfontosabb információk az adatlap első oldalán:

1. Adatlap címe, azonosító száma és a különböző dátumok (kiadás, revízió, stb.)
2. Főbb tulajdonságok bemutatása felsorolás jelleggel
3. Tokozás és lábkiosztással kapcsolatos információk
4. BGA tokozás felülnézeti rajza, lábkiosztás
5. Rendelési információk
6. Működés igazságtáblája
7. Logikai diagram
8. Az eszköz gyártási, illetve fejlesztési állapota

Nem csak ezen alaplogikák működésének ismerete nélkülözhetetlen, hanem az a képesség, amivel az áramköröket le tudjuk úgy írni, hogy később gyárthatók és szimulálhatók legyenek. Illetve ezeket az egyszerűbb áramköröket felhasználva bonyolultabb elektronikus eszközöket (mikrokontroller, PSoC, SoPC, stb.) tervezzünk, készítsünk és használjunk. Itt kell megjegyezni, hogy az áramkörök tervezése többféle absztrakciós szinten történik. (Logika, funkcionális, termikus, analóg, stb.) Az előző laborgyakorlatokon az analóg tervezésbe bepillantást nyertek, most pedig a logikai tervezés alapjait mutatjuk be.

A következő oldalon az SN7400-ás négy darab két-bemenetű NAND kapu IC adatlapjának első oldalát láthatjuk.

SN74ALVC00
QUADRUPLÉ 2-INPUT POSITIVE-NAND GATE

SCES115D – JULY 1997 – REVISED MARCH 2002

2

- Latch-Up Performance Exceeds 250 mA Per JESD 17
- ESD Protection Exceeds JESD 22
 - 2000-V Human-Body Model (A114-A)
 - 200-V Machine Model (A115-A)

4

description

This quadruple 2-input positive-NAND gate is designed for 1.65-V to 3.6-V V_{CC} operation.

The SN74ALVC00 performs the Boolean function $Y = \overline{A \cdot B}$ or $Y = \overline{A} + \overline{B}$ in positive logic.

3

D, DGV, NS, OR PW PACKAGE (TOP VIEW)

ORDERING INFORMATION

T_A	PACKAGE†		ORDERABLE PART NUMBER	TOP-SIDE MARKING
–40°C to 85°C	SOIC – D	Tube	SN74ALVC00D	ALVC00
		Tape and reel	SN74ALVC00DR	
	SOP – NS	Tape and reel	SN74ALVC00NSR	ALVC00
		TSSOP – PW	Tape and reel	SN74ALVC00PWR
	TVSOP – DGV	Tape and reel	SN74ALVC00DGV	VA00

6

† Package drawings, standard packing quantities, thermal data, symbolization, and PCB design guidelines are available at www.ti.com/sc/package.

FUNCTION TABLE (each gate)

INPUTS		OUTPUT
A	B	Y
H	H	L
L	X	H
X	L	H

7

logic diagram, each gate (positive logic)

8

9

PRODUCTION DATA information is current as of publication date. Products conform to specifications per the terms of Texas Instruments standard warranty. Production processing does not necessarily include testing of all parameters.

TEXAS INSTRUMENTS

POST OFFICE BOX 655303 • DALLAS, TEXAS 75265

Copyright © 2002, Texas Instruments Incorporated

1

2 A digitális tervezés menete Verilog nyelven, FPGA áramkörben implementálva

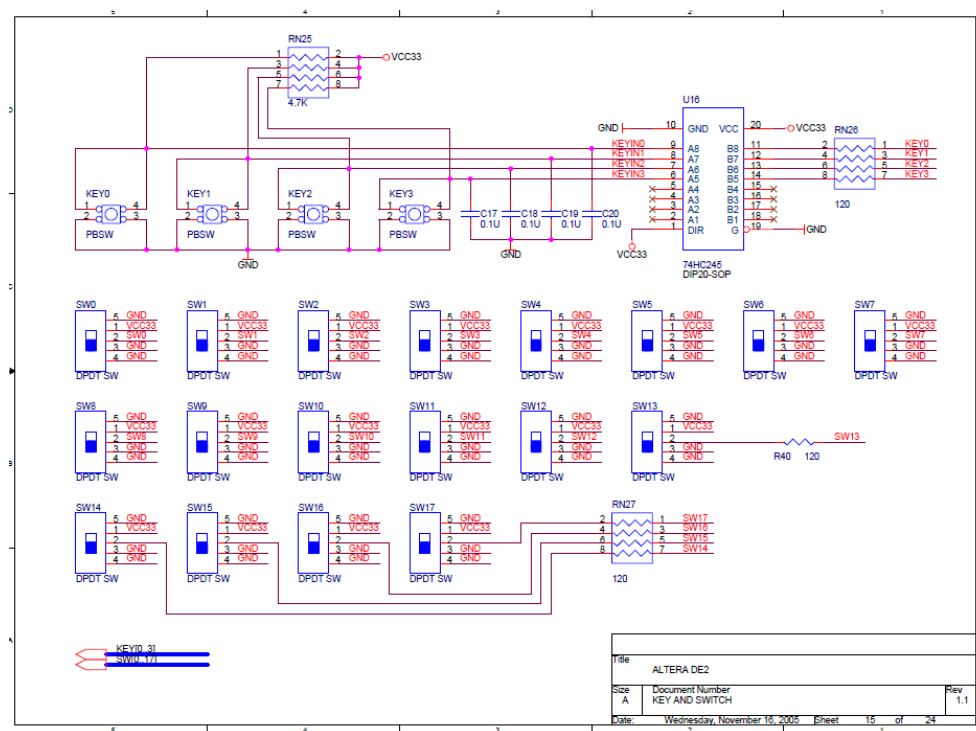
A laboratóriumi gyakorlatok során az Altera cég Quartus II nevű integrált fejlesztői környezetét fogjuk használni arra, hogy Verilog nyelven néhány alap logikai áramkört megtervezzünk, majd szintetizálás után egy Cyclone FPGA-ba letöltsük és megvizsgáljuk a tervezett áramkör működését. Az FPGA-s

környezet egy DE0 fejlesztői panelt jelent, amin az FPGA, a programozó interfész és sok-sok periféria kapott helyet. Mi a kapcsolókat, LED-eket és a 7-segmenses kijelzőt fogjuk használni.

A fejlesztés menete:

1. Hozzunk létre új projektet (lásd „quartus_verilog_intro.pdf”).
2. Készítsük el a modulunkat Verilog nyelven.
3. Rendeljük a modulunk ki és bemeneteihez az FPGA megfelelő lábait (A panel kapcsolási rajza megtalálható a „DE0_UserManual.pdf”-ben).
4. Szintetizáljuk a hardvert.
5. Vizsgáljuk meg, hogy mennyi erőforrást használtunk az FPGA-ból.
6. Vizsgáljuk meg, hogy a logikánk az FPGA melyik részében lett implementálva.
7. Vizsgáljuk meg, hogy a szintézer szerint milyen RTL szintű kapcsolási rajzot írtunk le.
8. Töltsük rá a programozó fájlt a fejlesztői panelre és vizsgáljuk meg a működést.

A lábhozzárendelést minden esetben ellenőrizzük a kapcsolási rajzon (DE0_UserManual.pdf)! Például a kapcsolók, nyomógombok bekötése:



A következő fejezetekben egy egyszerű logikai érték hozzárendelésével kezdünk és a gyakorlat végére alapvető logikai feladatokat megvalósító áramkörök megtervezéséig jutunk el.

3 Logikai értékek hozzárendelése kimeneti lábakhoz

3.1 Feladat leírása

A legelső feladat, hogy megismerjük azokat a lépéseket, amik ahhoz szükségesek, hogy a Verilog kódunkat szintetizálni tudjuk, majd letölteni a célhardverre. Első lépésként rendeljük 4 fizikai lábhoz fix logikai értéket. Rendeljük össze a lábakat a modulunk ki és bemeneteivel (Pin Assignment)! Ha ezzel megvagyunk, szintetizáljuk, nézzük át a fordítási jelentést (Compilation Report), majd töltsük le a programozó fájlt.

3.2 Megvalósítása Verilog nyelven

```
module LogikaiErtek (LEDs);  
    output [3:0] LEDs;  
  
    assign LEDs = 4'b1010;  
  
endmodule
```

Látható, hogy az FPGA szinte semmilyen erőforrását sem terheltük le, viszont a lábhozzárendelés nagy odafigyelést igényel. Első lépésként a Verilog modulunk négy kivezetését rendeljük a DE0-ás fejlesztőpanel négy LED-jéhez. Ezek azonosítói megtalálhatóak a „DE0_User_manual.pdf” 4.2. fejezetében (PIN_J1, PIN_J2, PIN_J3, PIN_H1). A lábhozzárendelés már a paneltervezést követően el tudjuk készíteni és mivel a legtöbb alkatrész fix lábakhoz van rendelve, elég egyszer alaposan eljárni és helyes hozzárendelési táblázatot készíteni. A kapcsolási rajz alapján döntünk el, hogy a LED-ek melyik esetben világítanak: ha logikai 1-est, vagy logikai 0-ást rendelünk a lábhoz (3.3V vagy 0V)?

4 Négy kapcsoló állapotának hozzárendelése kimeneti lábakhoz

4.1 Feladat leírása

Miután az első feladatban fix értéket rendeltünk a lábakhoz, most haladjunk tovább és próbáljunk változtatható értéket hozzárendelni. Vagyis olvassunk be 4 kapcsoló állapotát és azt rendeljük a LED-eket vezérlő lábakhoz.

4.2 Megvalósítása Verilog nyelven

```
module Vezetek (SWs, LEDs);  
    output [3:0] LEDs;  
    input [3:0] SWs;  
  
    assign LEDs = SWs;  
  
endmodule
```

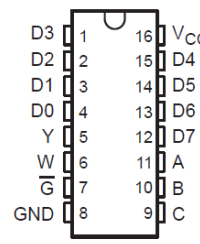
Szintén érdemes megnézni a kapcsolási rajzot, hogy a kapcsoló hogyan van bekötve az FPGA lábához. Innen sejthetjük, hogy melyik állapotban (nyitott, zárt) milyen feszültség szintet (3.3V, 0V) várhatunk és az milyen logikai értéknek (1, 0) felel meg. A lábhozzárendelést szintén a „DE0_User_manual.pdf”

4.2. fejezete alapján tudjuk megtenni. Az eddigi négy LED mellett még 4 kapcsoló PIN-jére is szükségünk van (SW0-SW3: **PIN_G4**, **PIN_H6**, **PIN_H5**, **PIN_J6**).

5 SN74HC151 – 8:1 Multiplexer

5.1 Feladat leírása

A következő feladat egy 8:1-es multiplexer megvalósítása. Mivel ez már egy kész szabványos logika megvalósítása, ezért törekedjünk annak minél tökéletesebb leírására. Nézzük meg a ki- és bemeneti portjait, a működést leíró táblázatot.



INPUTS				OUTPUTS	
SELECT			STROBE G	Y	W
C	B	A			
X	X	X	H	L	H
L	L	L	L	D0	$\overline{D0}$
L	L	H	L	D1	$\overline{D1}$
L	H	L	L	D2	$\overline{D2}$
L	H	H	L	D3	$\overline{D3}$
H	L	L	L	D4	$\overline{D4}$
H	L	H	L	D5	$\overline{D5}$
H	H	L	L	D6	$\overline{D6}$
H	H	H	L	D7	$\overline{D7}$

D0, D1 . . . D7 = the level of the respective D input

5.2 Megvalósítása Verilog nyelven

```
module SN74HC151 (A, B, C, D, Gn, Y, W);
    input A, B, C, Gn;
    input [7:0] D;
    output W;
    output reg Y;

    assign W = !Y;

    always @ (A or B or C or D or Gn) begin
        if(Gn == 0) begin
            Y <= D[{C, B, A}];
        end else begin
            Y <= 1'b0;
        end
    end
endmodule
```

Az „always” blokk érzékenységi listájában használhattuk volna a *-ot! A mintamegoldásban az egyes jelek vezetékként lettek deklarálva, nem pedig buszként. Ez befolyásolta az egyik értékadást, mint észrevehető.

5.3 Szimulációs tesztkörnyezet

```
`timescale 1ns / 10ps

module SN74HC151_TestBench;
    reg [7:0] D;
    reg [2:0] CNTR;
    reg Gn;
    wire Y, W;

    // Instantiate the Dual 2:4 Decoder/Demultiplexer
```



```
// (named UUT {unit under test})
SN74HC151 UUT(CNTR[0], CNTR[1], CNTR[2], D, Gn, Y, W);

initial begin
    CNTR = 3'b000;           // Counter
    Gn   = 1'b1;             // Disable MUX
    D    = 8'b10101010;     // Test Data

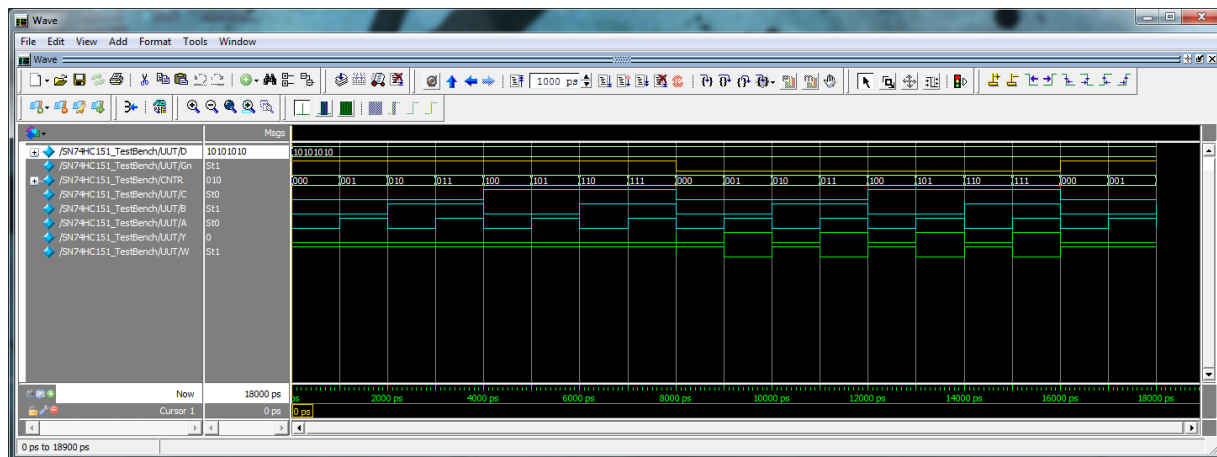
    #8
    Gn   = 1'b0;             // Enable MUX

    #8
    Gn   = 1'b1;             // Disable MUX
end

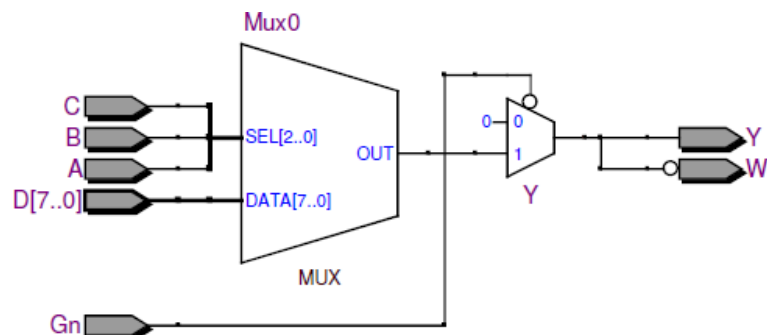
// Increment counter
always #1 CNTR <= CNTR + 1;

endmodule
```

5.4 A teszt eredménye



5.5 RTL szintű kapcsolási rajz



5.6 FPGA kihasználtsága

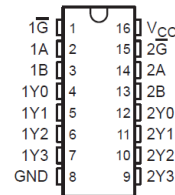
Flow Status	Successful - Sun Sep 04 11:19:12 2011
Quartus II Version	11.0 Build 157 04/27/2011 SJ Web Edition

Revision Name	R1
Top-level Entity Name	SN74HC151
Family	Cyclone II
Device	EP2C35F672C6
Timing Models	Final
Total logic elements	5 / 33,216 (< 1 %)
Total combinational functions	5 / 33,216 (< 1 %)
Dedicated logic registers	0 / 33,216 (0 %)
Total registers	0
Total pins	14 / 475 (3 %)
Total virtual pins	0
Total memory bits	0 / 483,840 (0 %)
Embedded Multiplier 9-bit elements	0 / 70 (0 %)
Total PLLs	0 / 4 (0 %)

6 SN74HC139 – Dual 2:4 Demultiplexer

6.1 Feladat leírása

Most egy dupla 2:4 Demultiplexert tartalmazó IC leírása következik, méghozzá két eltérő módon megvalósítva. Először a „szokásos” logikai kapukkal történő leírását adjuk meg, majd utána a Verilog lehetőségeit kihasználva egy sokkal olvashatóbb, átláthatóbb leírását. Célszerű a gyakorlatban nem mindent logikai kapukból felépíteni, mivel a HDL nyelvek ennél kifinomultabb tervezést tesznek lehetővé.



FUNCTION TABLE

G	INPUTS		OUTPUTS			
	B	A	Y0	Y1	Y2	Y3
H	X	X	H	H	H	H
L	L	L	L	H	H	H
L	L	H	H	L	H	H
L	H	L	H	H	L	H
L	H	H	H	H	H	L

6.2 Megvalósítása Verilog nyelven, logikai kapukkal

```

module SN74HC139_Assign(Gn1, A1, B1, Y1, Gn2, A2, B2, Y2);
    input Gn1, A1, B1, Gn2, A2, B2;
    output [3:0] Y1, Y2;

    assign Y1[0] = (!(Gn1 & !B1 & !A1));
    assign Y1[1] = (!(Gn1 & !B1 & A1));
    assign Y1[2] = (!(Gn1 & B1 & !A1));
    assign Y1[3] = (!(Gn1 & B1 & A1));

    assign Y2[0] = (!(Gn2 & !B2 & !A2));
    assign Y2[1] = (!(Gn2 & !B2 & A2));
    assign Y2[2] = (!(Gn2 & B2 & !A2));
    assign Y2[3] = (!(Gn2 & B2 & A2));

endmodule

```

6.3 Szimulációs tesztkörnyezet

```

`timescale 1ns / 10ps

module SN74HC139_TestBench;
    wire [3:0] Y1, Y2;
    reg Gn1, A1, B1, Gn2, A2, B2;

    // Instantiate the Dual 2:4 Decoder/Demultiplexer
    // (named UUT {unit under test})

    SN74HC139_Assign UUT(Gn1, A1, B1, Y1, Gn2, A2, B2, Y2);
    //SN74HC139_Switch UUT(Gn1, A1, B1, Y1, Gn2, A2, B2, Y2);

    initial begin

        A1  = 1'b0;
        B1  = 1'b0;
        Gn1 = 1'b1;
    end

```

```

A2  = 1'b0;
B2  = 1'b0;
Gn2 = 1'b1;

#9;
Gn1 = 1'b0;
Gn2 = 1'b0;

#10;
A1 = 1'b1;
B1 = 1'b0;
A2 = 1'b1;
B2 = 1'b0;

#10;
A1 = 1'b0;
B1 = 1'b1;
A2 = 1'b0;
B2 = 1'b1;

#10;
A1 = 1'b1;
B1 = 1'b1;
A2 = 1'b1;
B2 = 1'b1;

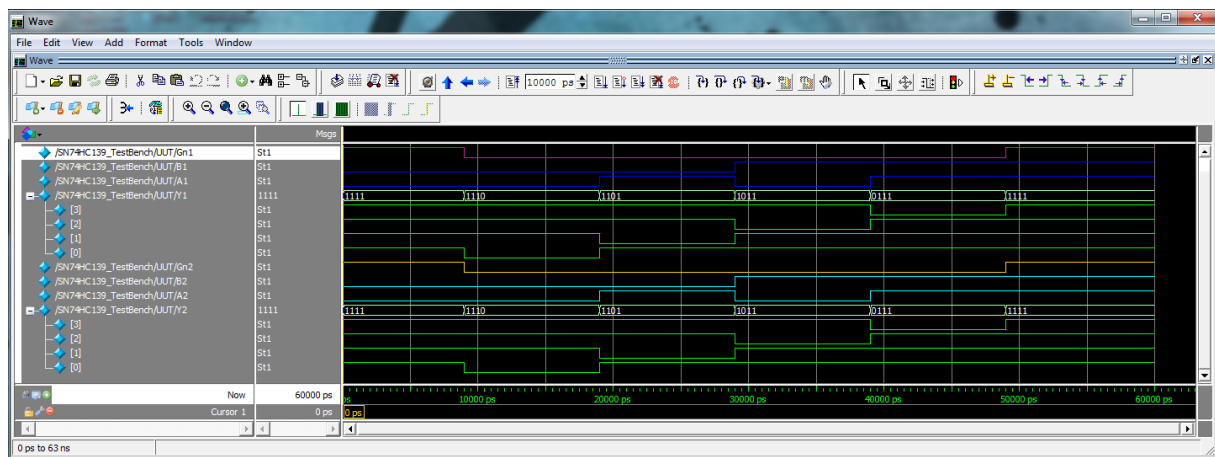
#10;
Gn1 = 1'b1;
Gn2 = 1'b1;

end

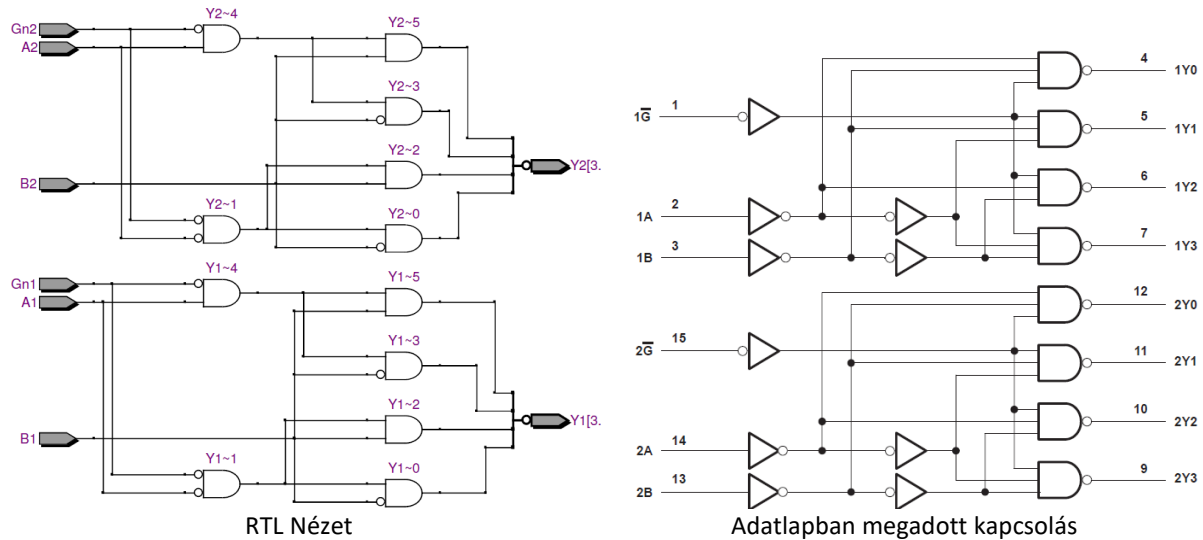
endmodule

```

6.4 A teszt eredménye



6.5 RTL szintű kapcsolási rajz



Érdemes összehasonlítani az eltérő megvalósítási formákat. Mi az, amit a Verilog kódunkba írtunk, mi az ami az FPGA-ban rendelkezésre áll és mi az, amit az IC gyártók a „legszívesebben” használnak? Miért?

6.6 Megvalósítása Verilog nyelven, switch-case szerkezettel

```

module SN74HC139_Switch(Gn1, A1, B1, Y1, Gn2, A2, B2, Y2);
    input Gn1, A1, B1, Gn2, A2, B2;
    output reg [3:0] Y1, Y2;

    always @ (Gn1 or A1 or B1) begin
        if(Gn1 == 0) begin
            case ({B1, A1})
                0: Y1 <= 4'b1110;
                1: Y1 <= 4'b1101;
                2: Y1 <= 4'b1011;
                3: Y1 <= 4'b0111;
            endcase
        end else begin
            Y1 <= 4'b1111;
        end
    end

    always @ (Gn2 or A2 or B2) begin
        if(Gn2 == 0) begin
            case ({B2, A2})
                0: Y2 <= 4'b1110;
                1: Y2 <= 4'b1101;
                2: Y2 <= 4'b1011;
                3: Y2 <= 4'b0111;
            endcase
        end else begin
            Y2 <= 4'b1111;
        end
    end

end

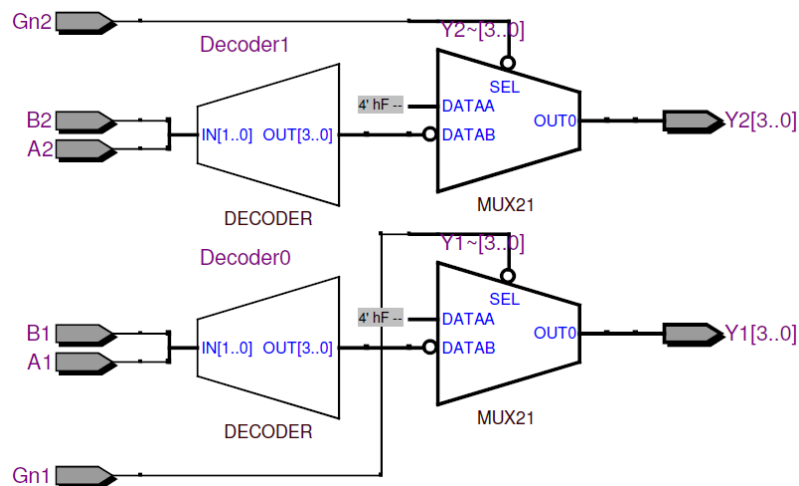
endmodule

```

6.7 FPGA kihasználtsága

Flow Status	Successful - Sun Sep 04 10:21:15 2011
Quartus II Version	11.0 Build 157 04/27/2011 SJ Web Edition
Revision Name	R1
Top-level Entity Name	SN74HC139
Family	Cyclone II
Device	EP2C35F672C6
Timing Models	Final
Total logic elements	8 / 33,216 (< 1 %)
Total combinational functions	8 / 33,216 (< 1 %)
Dedicated logic registers	0 / 33,216 (0 %)
Total registers	0
Total pins	14 / 475 (3 %)
Total virtual pins	0
Total memory bits	0 / 483,840 (0 %)
Embedded Multiplier 9-bit elements	0 / 70 (0 %)
Total PLLs	0 / 4 (0 %)

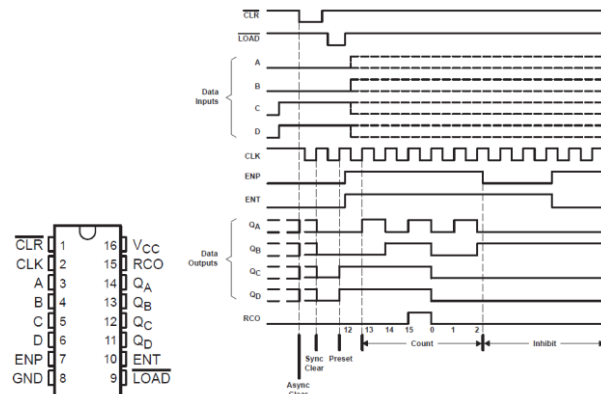
6.8 RTL szintű kapcsolási rajz



7 SN74HC163 – Szinkron, 4 bites számláló

7.1 Feladat leírása

A számlálók talán a legelterjedtebb blokknak számítanak, ezért most mi is egy 4 bites szinkron engedélyezhető, tölthető, törölhető számlálót készítünk. Az adatlap leírása egyértelmű, és a kapcsolási rajz pedig segít megfejtetni a két engedélyező jel funkcióját. Bár megtehetnénk, hogy flip-flop szinten írjuk le a számlálót, de mivel a Verilog nem erre való, ezért a funkcionalitás felől közelítjük meg a feladatot.



7.2 Megvalósítása Verilog nyelven

```
module SN74HC163 (CLRn, LOADn, ENT, ENP, CLK, ABCD, RCO, Q);
    input CLRn, LOADn, ENT, ENP, CLK;
    input [3:0] ABCD;
    output RCO;
    output reg [3:0] Q;

    // ENP -> Counter Enable/Disable
    // RCO -> ENT Enable Next Stage
    // Ripple-Carry Output
    assign RCO = &Q & ENT;

    // Only for simulation! Initial value.
    initial begin
        Q <= 4'b0000;
    end

    // Synchronous operation
    always @ (posedge CLK) begin
        if(ENT & ENP) begin
            // If Enabled
            if(CLRn == 1'b0) begin
                // Clear
                Q <= 4'b0000;
            end else begin
                if(LOADn == 1'b0) begin
                    // Load
                    Q <= ABCD;
                end else begin
                    // Count
                    Q <= Q + 1'b1;
                end
            end
        end else begin
            // Disabled
            Q <= Q;
        end
    end
endmodule
```

7.3 Szimulációs tesztkörnyezet

```

`timescale 1ns / 10ps

module SN74HC163_TestBench;
    reg CLRn, LOADn, ENT, ENP, CLK;
    reg [3:0] ABCD;
    wire RCO;
    wire [3:0] Q;

    // Instantiate the 4-bit Synchronous Counter
    // (named UUT {unit under test})
    SN74HC163 UUT(CLRn, LOADn, ENT, ENP, CLK, ABCD, RCO, Q);

    initial begin
        CLK      = 1'b0;
        CLRn     = 1'b1;    // Don't Reset Counter
        LOADn    = 1'b1;    // Don't Load Value to the Counter
        ENT      = 1'b0;    // Disbale Counter and RCO
        ENP      = 1'b0;    // Disbale Counter
        ABCD     = 4'b1100; // Load Value

        #4
        ENT      = 1'b1;    // Enable Counter and RCO
        ENP      = 1'b1;    // Enable Counter
        LOADn    = 1'b0;    // Load Value to the Counter
        #2
        LOADn    = 1'b1;

        #16
        CLRn     = 1'b0;    // Reset Counter
        #2
        CLRn     = 1'b1;

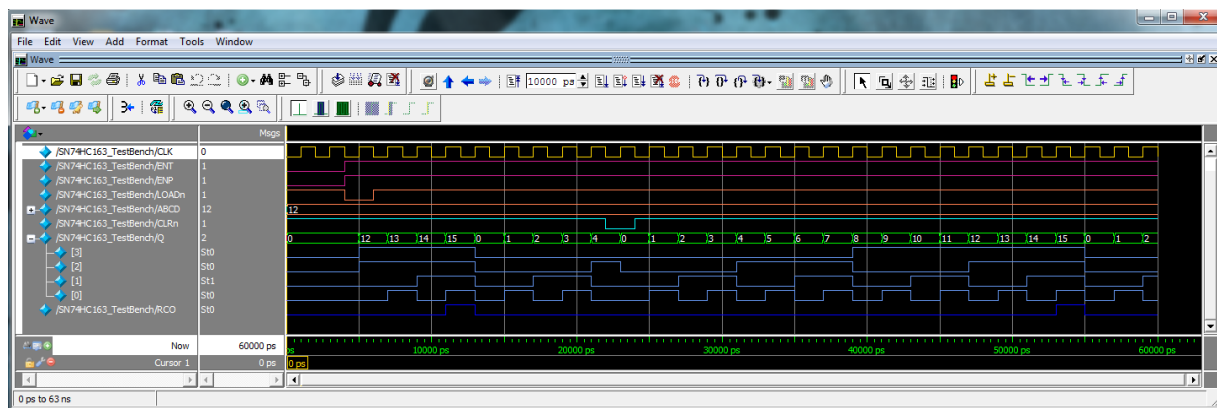
    end

    //Increment counter
    always #1 CLK <= !CLK;

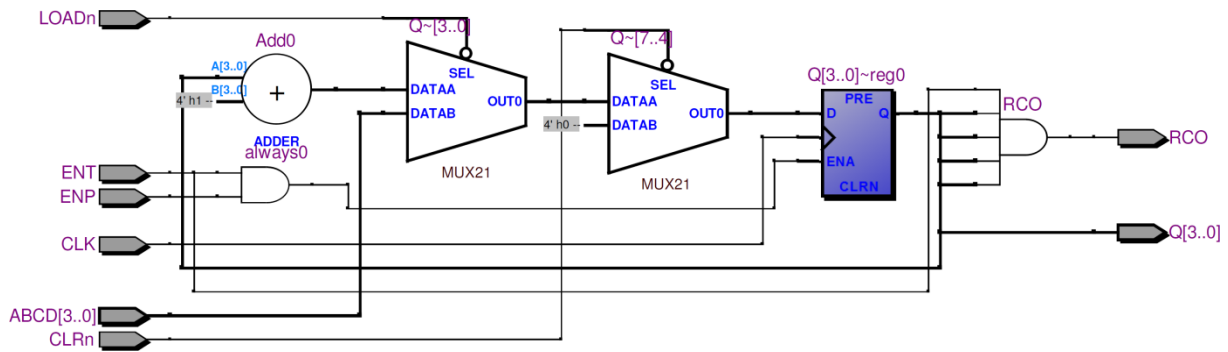
endmodule

```

7.4 A teszt eredménye



7.5 RTL szintű kapcsolási rajz



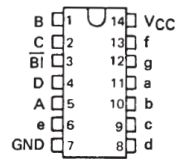
7.6 FPGA kihasználtsága

Flow Status	Successful - Sun Sep 04 14:16:56 2011
Quartus II Version	11.0 Build 157 04/27/2011 SJ Web Edition
Revision Name	R1
Top-level Entity Name	SN74HC163
Family	Cyclone II
Device	EP2C35F672C6
Timing Models	Final
Total logic elements	10 / 33,216 (< 1 %)
Total combinational functions	10 / 33,216 (< 1 %)
Dedicated logic registers	4 / 33,216 (< 1 %)
Total registers	4
Total pins	14 / 475 (3 %)
Total virtual pins	0
Total memory bits	0 / 483,840 (0 %)
Embedded Multiplier 9-bit elements	0 / 70 (0 %)
Total PLLs	0 / 4 (0 %)

8 SN74LS49 – BCD-to-Seven Segment Decoder

8.1 Feladat leírása

Ez a feladat már egy látványosabb funkció megvalósítását tűzi ki célul. Készítsünk egy olyan áramkört, aminek a bemenetén megjelenő bináris szám hatására a kimeneten olyan vezérlő jelek jelennek meg, amit egy 7-segmenses kijelzőre kötve a kijelzőn hexadecimális számok jelennek meg. A megvalósítás során annyiban célszerű eltérni az adatlaptól, hogy a DE0 panelen a szegmensek meghajtása eltérő logikával történik, mint amit az IC leír.



'LS49
FUNCTION TABLE (T3)

DECIMAL OR FUNCTION	INPUTS					OUTPUTS							NOTE
	D	C	B	A	BI	a	b	c	d	e	f	g	
0	L	L	L	L	H	H	H	H	H	H	L	L	1
1	L	L	L	H	H	L	H	H	L	L	L	L	
2	L	L	H	L	H	H	H	L	H	H	L	H	
3	L	L	H	H	H	H	H	H	H	L	L	H	
4	L	H	L	L	H	L	H	H	L	L	H	H	
5	L	H	L	H	H	H	L	H	H	L	H	H	
6	L	H	H	L	H	L	L	H	H	H	H	H	
7	L	H	H	H	H	H	H	H	L	L	L	L	
8	H	L	L	L	H	H	H	H	H	H	H	H	
9	H	L	L	H	H	H	H	H	L	L	H	H	
10	H	L	H	L	H	L	L	L	H	H	L	H	
11	H	L	H	H	H	L	L	H	H	L	L	H	
12	H	H	L	L	H	L	H	L	L	L	H	H	
13	H	H	L	H	H	H	L	L	L	H	L	H	
14	H	H	H	L	H	L	L	L	L	H	H	H	
15	H	H	H	H	H	L	L	L	L	L	L	L	
BI	X	X	X	X	L	L	L	L	L	L	L	L	2

8.2 Megvalósítása Verilog nyelven

```

module SN74LS49(IN, BIn, OUT);
    input BIn;
    input [3:0] IN;
    output reg [6:0] OUT;

    always @ (IN or BIn) begin
        if(BIn == 1) begin
            case (IN)
                0: OUT = 7'b1000000;
                1: OUT = 7'b1111001;
                2: OUT = 7'b0100100;
                3: OUT = 7'b0110000;
                4: OUT = 7'b0011001;
                5: OUT = 7'b0010010;
                6: OUT = 7'b0000010;
                7: OUT = 7'b1111000;
                8: OUT = 7'b0000000;
                9: OUT = 7'b0010000;
                10: OUT = 7'b0001000;
                11: OUT = 7'b0000011;
                12: OUT = 7'b1000110;
                13: OUT = 7'b0100001;
                14: OUT = 7'b0000110;
                15: OUT = 7'b0001110;
            endcase
        end else begin
            OUT = 7'b1111111;
        end
    end

endmodule

```

8.3 Szimulációs tesztkörnyezet

```

`timescale 1ns / 10ps

module tb_top;
    reg BIn;
    reg [3:0] CNTR;
    wire [6:0] OUT;

    // Instantiate the BCD-To-Seven-Segment Decoder
    // (named UUT {unit under test})
    SN74LS49 UUT(CNTR, BIn, OUT);

    initial begin

        // time = 0
        CNTR = 1'b0;           // Counter
        BIn = 1'b1;           // Blanking Input

        #17
        BIn = 1'b0;           // Blanking Input

        #16
        BIn = 1'b1;           // Blanking Input

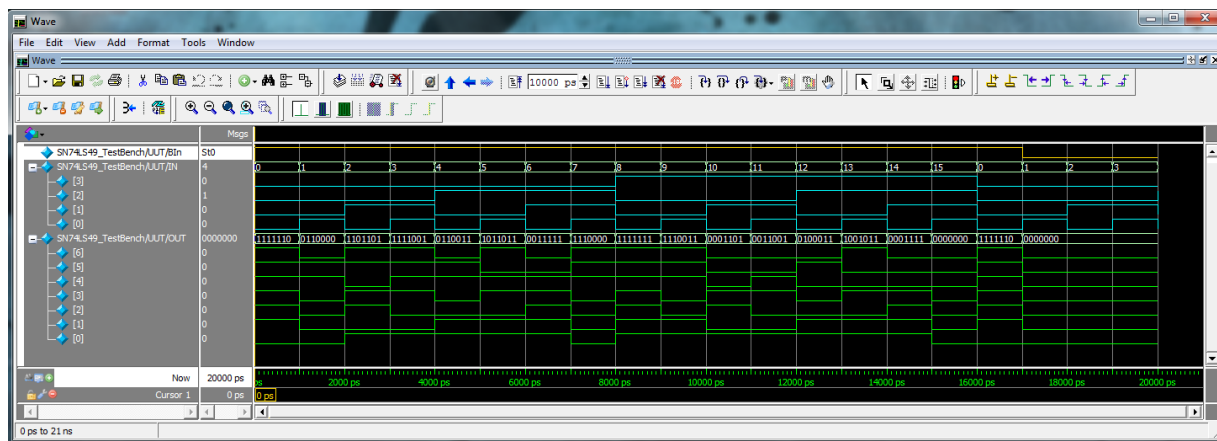
    end

    //Increment counter
    always #1 CNTR <= CNTR + 1;

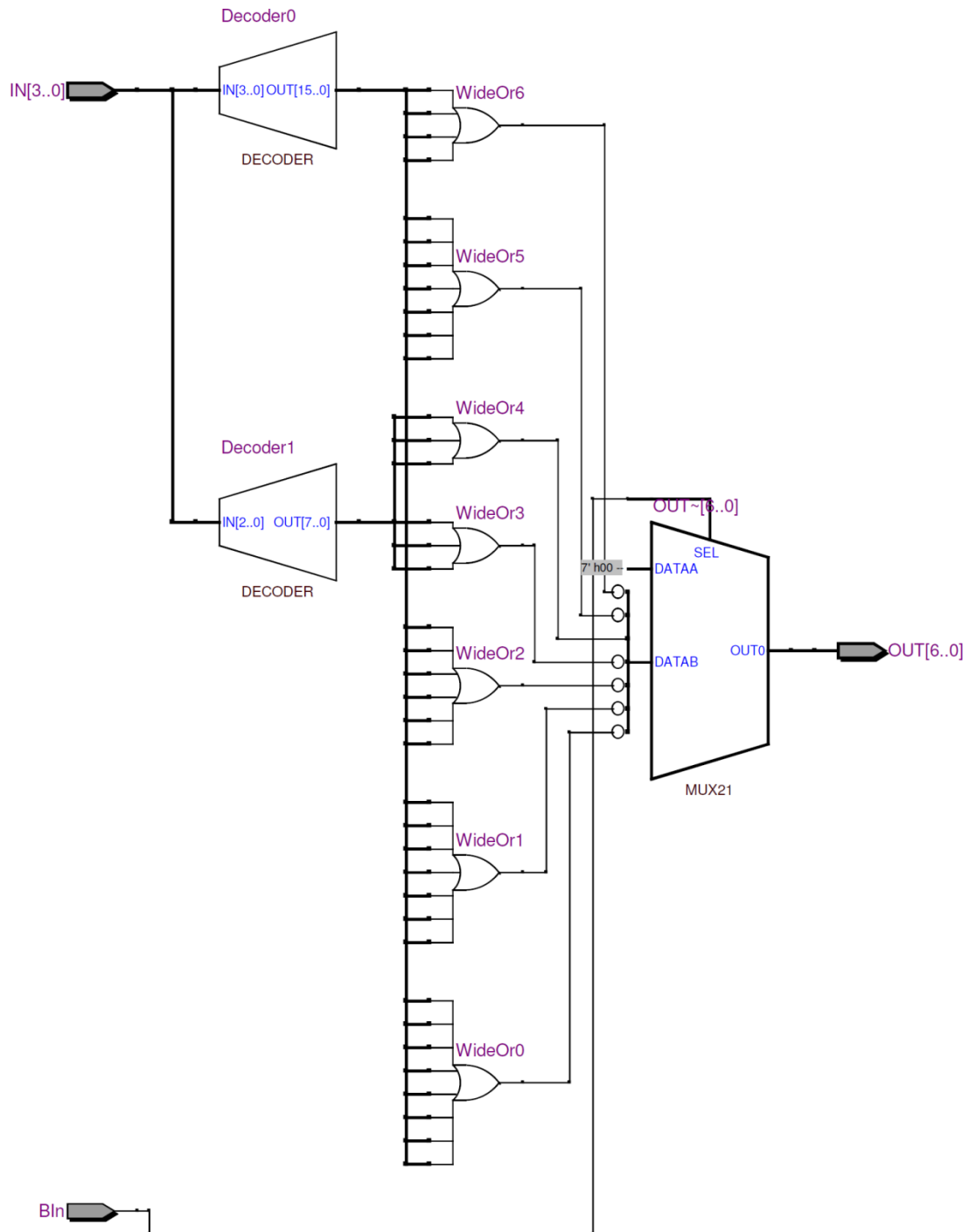
endmodule

```

8.4 A teszt eredménye



8.5 RTL szintű kapcsolási rajz



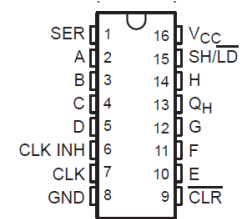
8.6 FPGA kihasználtsága

Flow Status	Successful - Sun Sep 04 13:08:32 2011
Quartus II Version	11.0 Build 157 04/27/2011 SJ Web Edition
Revision Name	R1
Top-level Entity Name	SN74LS49
Family	Cyclone II
Device	EP2C35F672C6
Timing Models	Final
Total logic elements	12 / 33,216 (< 1 %)
Total combinational functions	12 / 33,216 (< 1 %)
Dedicated logic registers	0 / 33,216 (0 %)
Total registers	0
Total pins	12 / 475 (3 %)
Total virtual pins	0
Total memory bits	0 / 483,840 (0 %)
Embedded Multiplier 9-bit elements	0 / 70 (0 %)
Total PLLs	0 / 4 (0 %)

9 SN74HC166A – 8 bites shift regiszter

9.1 Feladat leírása

A Shift-Regiszter áramkörök is közkedveltek, szinte minden beágyazott soros kommunikáció alapja a párhuzamos-soros átalakítás (UART, SPI, I²C, CAN, One-Wire, stb.). A mintamegvalósítás egy plusz kivezetést tartalmaz, hogy látszódjanak a belső regiszterek értékei.



9.2 Megvalósítása Verilog nyelven

```
module SN74LV166A (ABCDEFGH, CLK, CLKINH, CLRN, LOADn, SER, QH, DEMO);
    input CLK, CLKINH, CLRN, LOADn, SER;
    input [7:0] ABCDEFGH;
    output QH;
    output [7:0] DEMO;

    // Internal Register
    reg [7:0] SHIFTREG;
    wire CLR;

    // Serial Output
    assign QH = SHIFTREG[7];
    assign CLR = !CLRN;
    assign DEMO = SHIFTREG;

    initial begin
        SHIFTREG <= 8'h00;
    end

    // Synchronous operation with async. reset
    always @ (posedge CLK or posedge CLR) begin
        if (CLR == 1'b1) begin
            // Clear
            SHIFTREG <= 8'b00000000;
        end else begin
            if (CLKINH == 1'b0) begin
                // Clock Enabled
                if (LOADn == 1'b0) begin
                    // Load
                    SHIFTREG <= ABCDEFGH;
                end else begin
                    // Shift
                    SHIFTREG[0] <= SER;
                    SHIFTREG[7:1] <= SHIFTREG[6:0];
                end
            end else begin
                // Clock Inhibit
                SHIFTREG <= SHIFTREG;
            end
        end
    end

endmodule
```

9.3 Szimulációs tesztkörnyezet

```

`timescale 1ns / 10ps

module SN74LV166A_TestBench;
    reg CLK, CLKINH, CLRn, LOADn, SER;
    reg [7:0] ABCDEFGH;
    wire QH;

    // Instantiate the 8-bit Shift Register
    // (named UUT {unit under test})
    SN74LV166A UUT(ABCDEFGH, CLK, CLKINH, CLRn, LOADn, SER, QH);

    initial begin
        CLK      = 1'b0;    // Clock
        CLKINH   = 1'b1;    // Clock Inhibit
        CLRn     = 1'b1;    // Don't Reset Shift Register
        LOADn    = 1'b1;    // Don't Load Value to the Shift Register
        SER      = 1'b0;    // Serial Input
        ABCDEFGH = 8'b11001100; // Load Value

        #4
        CLKINH = 1'b0;    // Enable Clocking
        LOADn  = 1'b0;    // Load Value to the Shift Register
        #2
        LOADn  = 1'b1;

        #12
        SER = 1'b1;
        #2
        SER = 1'b0;
        #2
        SER = 1'b1;
        #2
        SER = 1'b0;
        #2
        SER = 1'b1;
        #2
        SER = 1'b0;
        #2
        SER = 1'b1;

        #32
        CLRn = 1'b0;    // Reset Shift Register
        #2
        CLRn = 1'b1;

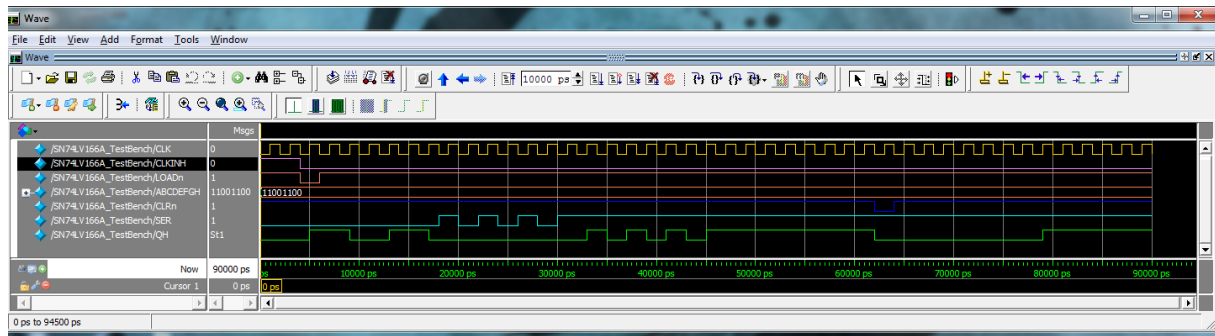
    end

    //Increment counter
    always #1 CLK <= !CLK;

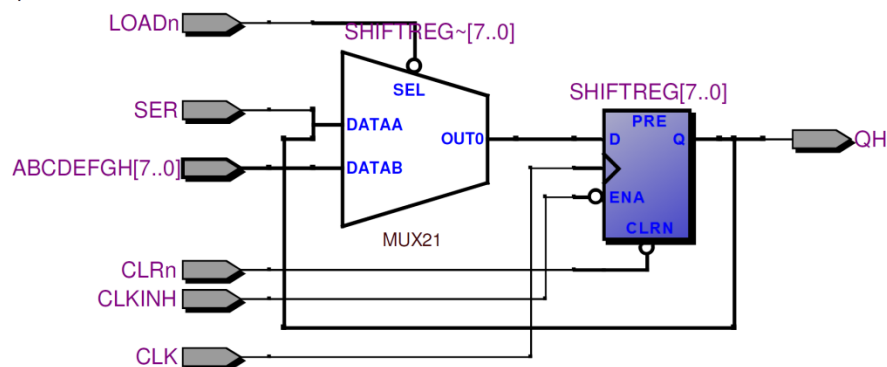
endmodule

```

9.4 A teszt eredménye



9.5 RTL szintű kapcsolási rajz



9.6 FPGA kihasználtsága

Flow Status	Successful - Sun Sep 04 15:17:06 2011
Quartus II Version	11.0 Build 157 04/27/2011 SJ Web Edition
Revision Name	R1
Top-level Entity Name	SN74LV166A
Family	Cyclone II
Device	EP2C35F672C6
Timing Models	Final
Total logic elements	8 / 33,216 (< 1 %)
Total combinational functions	8 / 33,216 (< 1 %)
Dedicated logic registers	8 / 33,216 (< 1 %)
Total registers	8
Total pins	14 / 475 (3 %)
Total virtual pins	0
Total memory bits	0 / 483,840 (0 %)
Embedded Multiplier 9-bit elements	0 / 70 (0 %)
Total PLLs	0 / 4 (0 %)