Digitális Technika

Dr. Oniga István
Debreceni Egyetem, Informatikai Kar

6. Laboratóriumi gyakorlat

Kombinációs logikai hálózatok 2.

- Komparátorok
- Paritásvizsgáló áramkörök
- Összeadok

Lab6_1: Két bites komparátor

- Hozunk létre egy új projektet
- Adjunk hozzá egy új „Verilog” forrásfájlt.
- Lab6_1.v forrásfájl specifikálása
- Funkcionális kód ellenőrzése szimulációval
- A Nexysx.ucf fájl hozzáadása és adaptálása
- Konfigurációs fájl generálása, letöltése és a működés tesztelése a kártyán

```
module compare_2 (output reg A_lt_B, A_gt_B, A_eq_B, input [1:0] A,B);

always @ (A or B)
        begin

                A_lt_B = 0;
                A_gt_B = 0;
                A_eq_B = 0;

                if (A==B) A_eq_B = 1;
                else if (A>B) A_gt_B = 1;
                else A_lt_B = 1;

        end

endmodule
```

Lab6_2: Paritásellenőrző áramkör

- Hozunk létre egy új projektet
- Adjunk hozzá egy új „Verilog” forrásfájlt
- Lab6_2.v forrásfájl specifikálása
- Funkcionális kód ellenőrzése szimulációval

```
module oddparity_for (output reg parity, input [7:0] data);
integer k;
always@(data)
begin
        parity = 1;
        for (k = 0; k <= 7; k = k+1)
        begin
                if (data[k] == 1)
                    parity = ~parity;
                end
        end
endmodule
```

```
25  module oddparity_test;
26
27      // Inputs
28      reg [7:0] data;
29
30      // Outputs
31      wire parity;
32
33      // Instantiate the Unit Under Test (UUT)
34      oddparity_for uut (
35          .parity(parity),
36          .data(data)
37      );
38      integer i;
39      initial begin
40          // Initialize Inputs
41          data = 0;
42
43          // Wait 100 ns for global reset to finish
44          #100;
45
46          for (i = 0; i <= 255; i = i+1)
47          begin
48              #50 data =i;
49
50          end
51
52      end
53
54  endmodule
```

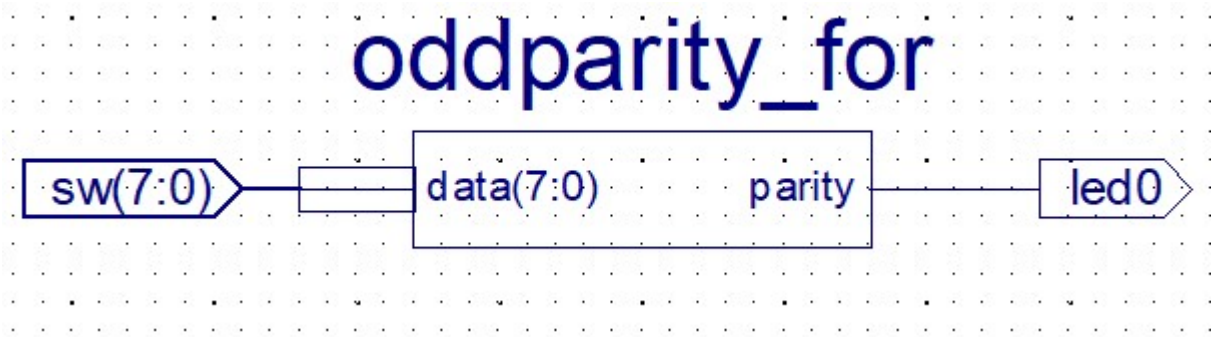
Lab6_2 Paritásellenőrző áramkör

szimuláció eredménye

Lab6_2 feladat

Paritásellenőrző áramkör implementáció

- A Nexysx.ucf fájl hozzáadása és adaptálása
- Konfigurációs fájl generálása, letöltése és a működés tesztelése a kártyán



Lab6_3 feladat

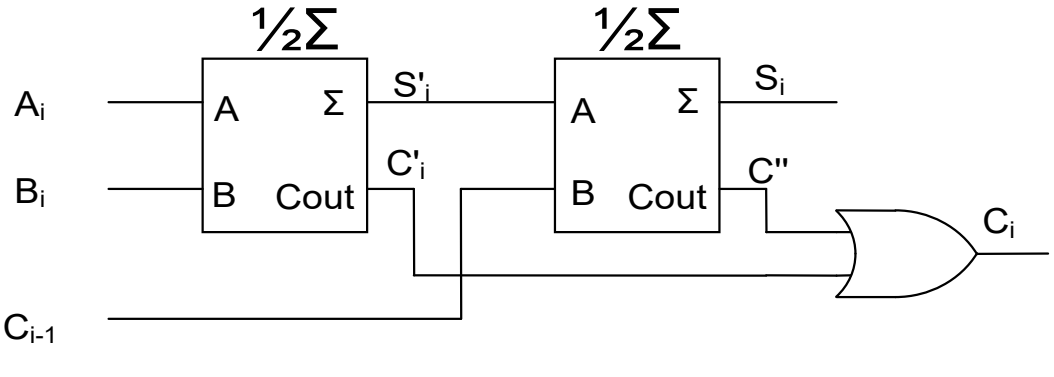
1-bites összeadó Verilog strukturális modellje

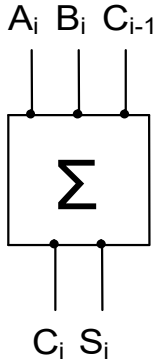
A_i	B_i	C_{i-1}	S_i	C_i
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

$$S_i = A_i \oplus B_i \oplus C_{i-1}$$

$$C_i = A_i B_i + A_i C_{i-1} + B_i C_{i-1}$$

Teljes összeadó két félösszeadóból:

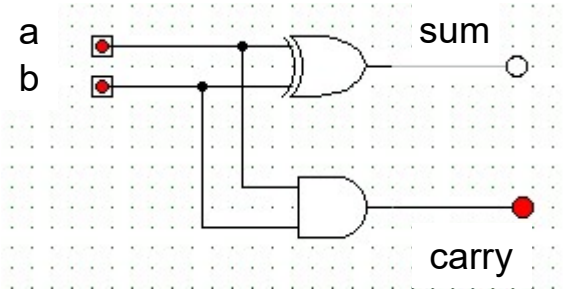




Lab6_3a feladat

1-bites fél összeadó Verilog strukturális modellje

- Hozunk létre egy új projektet (Lab6_3)
- Adjunk hozzá egy új „Verilog” forrásfájlt (half_add.v)
- half_add.v forrásfájl specifikálása




```
module half_add (output sum, carry, input a, b);
        xor (sum, a, b);     // exclusive OR
        and (carry, a ,b);   // AND
endmodule
```

Lab6_3a feladat

1-bites fél összeadó szimuláció

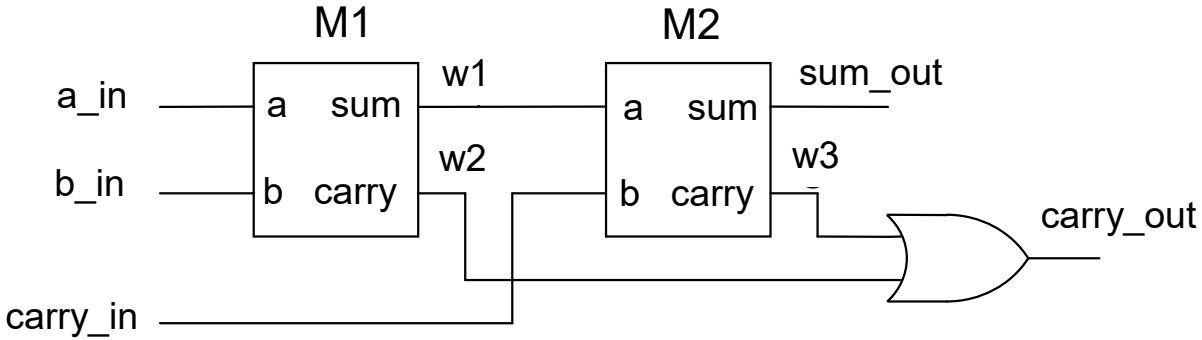
- Funkcionális kód ellenőrzése szimulációval (half_add_test.v)
- half_add_test.v forrásfájl specifikálása (Gerjesztő jelek specifikálása)
- Funkcionális kód ellenőrzése szimulációval

```
// Add stimulus here
#100
        a=1;
#100
        a=0;
        b=1;
#100
        a=1;
        b=1;
```

Lab6_3b feladat

1-bites teljes összeadó Verilog strukturális modellje

- Adjunk hozzá a projekthez egy új „Verilog” forrásfájlt (full_add.v)
- A forrásfájl specifikálása z alábbi ábra szerint



Névszerinti összekötés

```
module full_add (output sum_out, carry_out, input a_in, b_in, carry_in );

wire w1, w2, w3;

        half_add M1 (.a(a_in), .sum(w1), .b(b_in), .carry(w2));
        half_add M2 (.sum(sum_out), .b(w1), .carry(w3), .a(carry_in));

        or (carry_out, w2, w3);
endmodule
```

- A Nexysx.ucf fájl hozzáadása és adaptálása
- Konfigurációs fájl generálása, letöltése és a működés tesztelése a kártyán

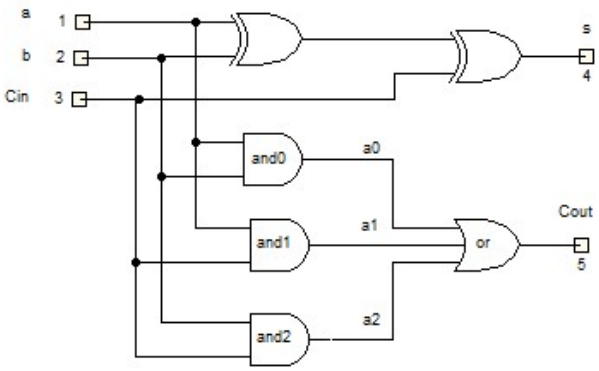
Lab6_3b eredmények

Működés tesztelése a kártyán

Cin	A	B	Sum	Cout
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

Lab6_4: 1-bites teljes összeadó leírása

```
// A verzió
module add1_full (input a, b, cin, output cout, s);
xor3_m xor(.i0(a), .i1(b), .i2(cin), .o(s));
wire a0, a1, a2;
and2_m and0(.i0(a), .i1(b), .o(a0));
and2_m and1(.i0(a), .i1(cin), .o(a1));
and2_m and2(.i0(b), .i1(cin), .o(a2));
or3_m or(.i0(a0), .i1(a1), .i2(a2) , .o(cout))
endmodule
```




```
// B verzió
module add1_full (input a, b, cin, output cout, s);
assign s = a ^ b ^ cin;
assign cout = (a & b) | (a & cin) | (b & cin);
endmodule
```

```
// C verzió
module add1_full (input a, b, cin, output cout, s);
assign {cout, s} = a + b + cin;
endmodule
```

- Hozunk létre egy új projektet (Lab6_4)
- Adjunk hozzá egy új „Verilog” forrásfájlt (add1_full.v)
- A teljes összeadó leírása a funkcionális kódrészlettel (válaszon egyet a fentiek közül)
- A Nexysx.ucf fájl hozzáadása és adaptálása
- Konfigurációs fájl generálása, letöltése és a működés tesztelése a kártyán

Lab6_4 eredmények

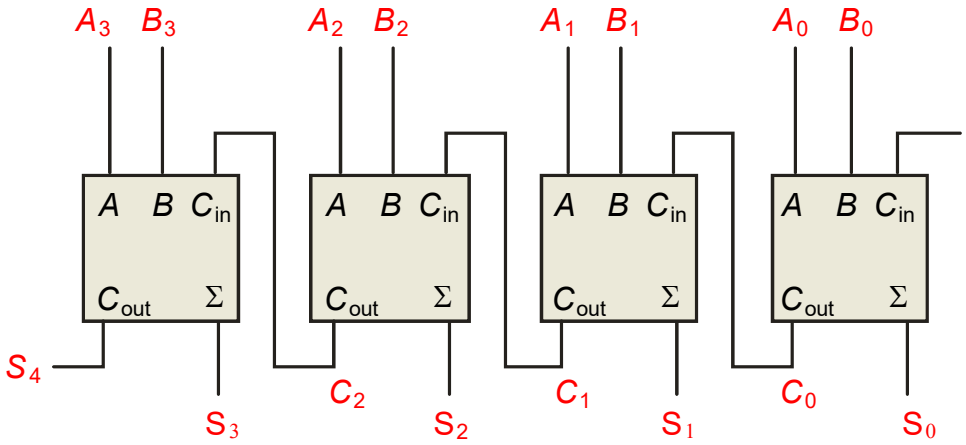
Működés tesztelése a kártyán

Cin	A	B	Sum	Cout
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

Lab6_5:Több bites összeadók

A több bites számokat teljes összeadókból építhetjük meg

Soros átvitelű 4 bites összeadó (Ripple carry adder): 7483



- Hozunk létre egy új projektet (Lab6_5a)
- Adjunk hozzá egy új „Verilog” forrásfájlt (add4 .v)
- A teljes összeadó leírása a funkcionális kódrészlettel (a következő oldalon)

Lab6_5a: 4 bites soros átvitelű összeadó

- Funkcionális kódrészlet

```
// 5_5a 4 bites soros átvitelű összeadó
module add4 (input [3:0] a, b, output [4:0] s);
wire [3:0] c;
add1_full add0(.a(a[0]), .b(b[0]), .cin(1'b0),    .cout(c[0]), .s(s[0]));
add1_full add1(.a(a[1]), .b(b[1]), .cin(c[0]), .cout(c[1]), .s(s[1]));
add1_full add2(.a(a[2]), .b(b[2]), .cin(c[1]), .cout(c[2]), .s(s[2]));
add1_full add3(.a(a[3]), .b(b[3]), .cin(c[2]), .cout(s[4]),    .s(s[3]));
endmodule
```

- Az előző feladatban létrehozott forrásfájl (add1_full .v) hozzáadása projekthez (Ad Copy of Source)
- A Nexysx.ucf fájl hozzáadása és adaptálása
- Konfigurációs fájl generálása, letöltése és a működés tesztelése a kártyán

Lab6_5b: 4 bites összeadó viselkedési leírás

- Hozunk létre egy új projektet (Lab6_5b)
- Adjunk hozzá egy új „Verilog” forrásfájlt (add4 .v)
- A teljes összeadó leírása a funkcionális kódrészlettel

```
// 5_5b
module add4 (input [3:0] a, b, output [4:0] s);
assign s = a + b;
endmodule
```

- A Nexysx.ucf fájl hozzáadása és adaptálása
- Konfigurációs fájl generálása, letöltése és a működés tesztelése a kártyán