Digitális Technika

Dr. Oniga István
Debreceni Egyetem, Informatikai Kar

5. Laboratóriumi gyakorlat

Kombinációs logikai hálózatok 1. (HDL kódok készítése, szimulációja, implementáció.)

- Kódolok strukturális és viselkedési leírása
- Dekódolok
- Multiplexerek

Lab5_1 feladat:

Kódolók

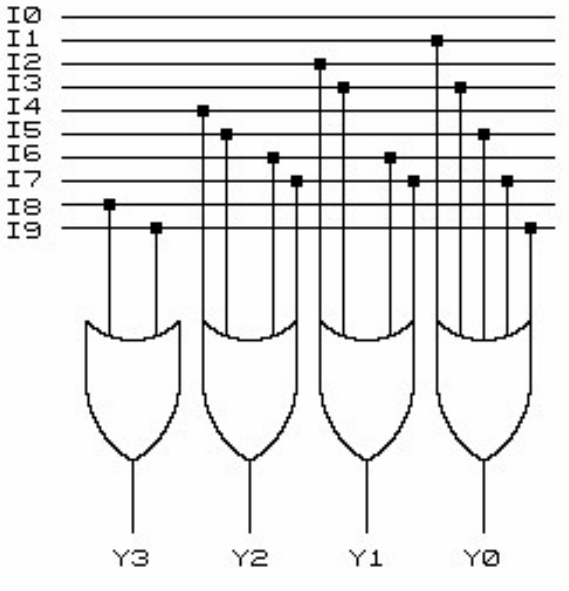
- Hozunk létre egy új projektet
- Adjunk hozzá egy új „Verilog” forrásfájlt. Lab5_1.v forrásfájl mintakeret specifikálása: sw[7:0], bt[3:0], ld[7:0]
- A lab5_1x feladatok specifikálása a funkcionális kódrészletekkel
- Funkcionális kód ellenőrzése szimulációval
- A Nexysx.ucf fájl hozzáadása és adaptálása
- Konfigurációs fájl generálása, letöltése és a működés tesztelése a kártyán

Lab5_1a feladat:

Decimális – BCD kódoló strukturális leírás

I	Y_3	Y_2	Y_1	Y_0
I_0	0	0	0	0
I_1	0	0	0	1
I_2	0	0	1	0
I_3	0	0	1	1
I_4	0	1	0	0
I_5	0	1	0	1
I_6	0	1	1	0
I_7	0	1	1	1
I_8	1	0	0	0
I_9	1	0	0	1

- $Y_0=I_1+I_3+I_5+I_7+I_9$
- $Y_1=I_2+I_3+I_6+I_7$
- $Y_2=I_4+I_5+I_6+I_7$
- $Y_3=I_8+I_9$

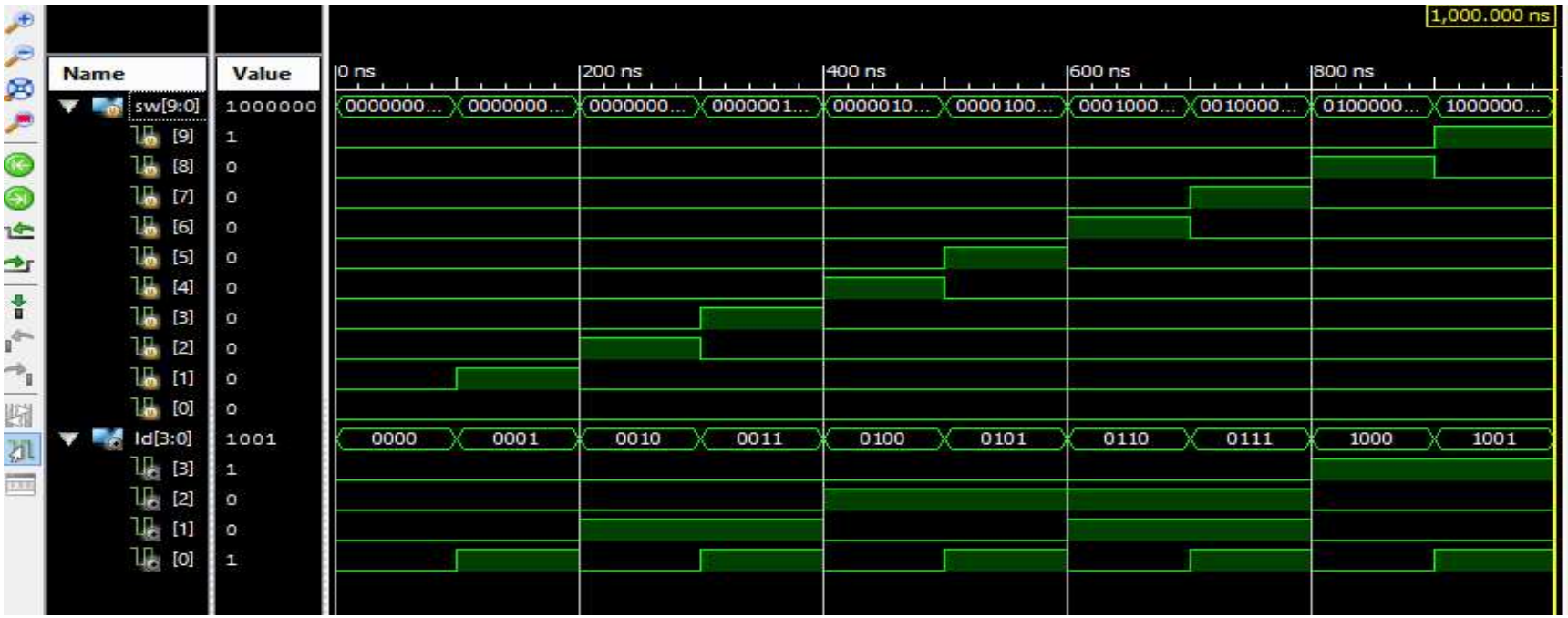



```
module BCDmod(
   input [9:0] sw,
output [3:0] ld );
            assign ld[0]=sw[1]|sw[3]|sw[5]|sw[7]|sw[9];
            assign ld[1]=sw[2]|sw[3]|sw[6]|sw[7];
            assign ld[2]=sw[4]|sw[5]|sw[6]|sw[7];
            assign ld[3]=sw[8]|sw[9];

endmodule
```

Lab5_1a Decimális – BCD kódoló szimuláció

I	Y_3	Y_2	Y_1	Y_0
I_0	0	0	0	0
I_1	0	0	0	1
I_2	0	0	1	0
I_3	0	0	1	1
I_4	0	1	0	0
I_5	0	1	0	1
I_6	0	1	1	0
I_7	0	1	1	1
I_8	1	0	0	0
I_9	1	0	0	1

```
// Add stimulus here

sw[1]=1;
#100; sw[1]=0; sw[2]=1;
#100; sw[2]=0; sw[3]=1;
#100; sw[3]=0; sw[4]=1;
#100; sw[4]=0; sw[5]=1;
#100; sw[5]=0; sw[6]=1;
#100; sw[6]=0; sw[7]=1;
#100; sw[7]=0; sw[8]=1;
#100; sw[8]=0; sw[9]=1;
```



Lab5_1b feladat:

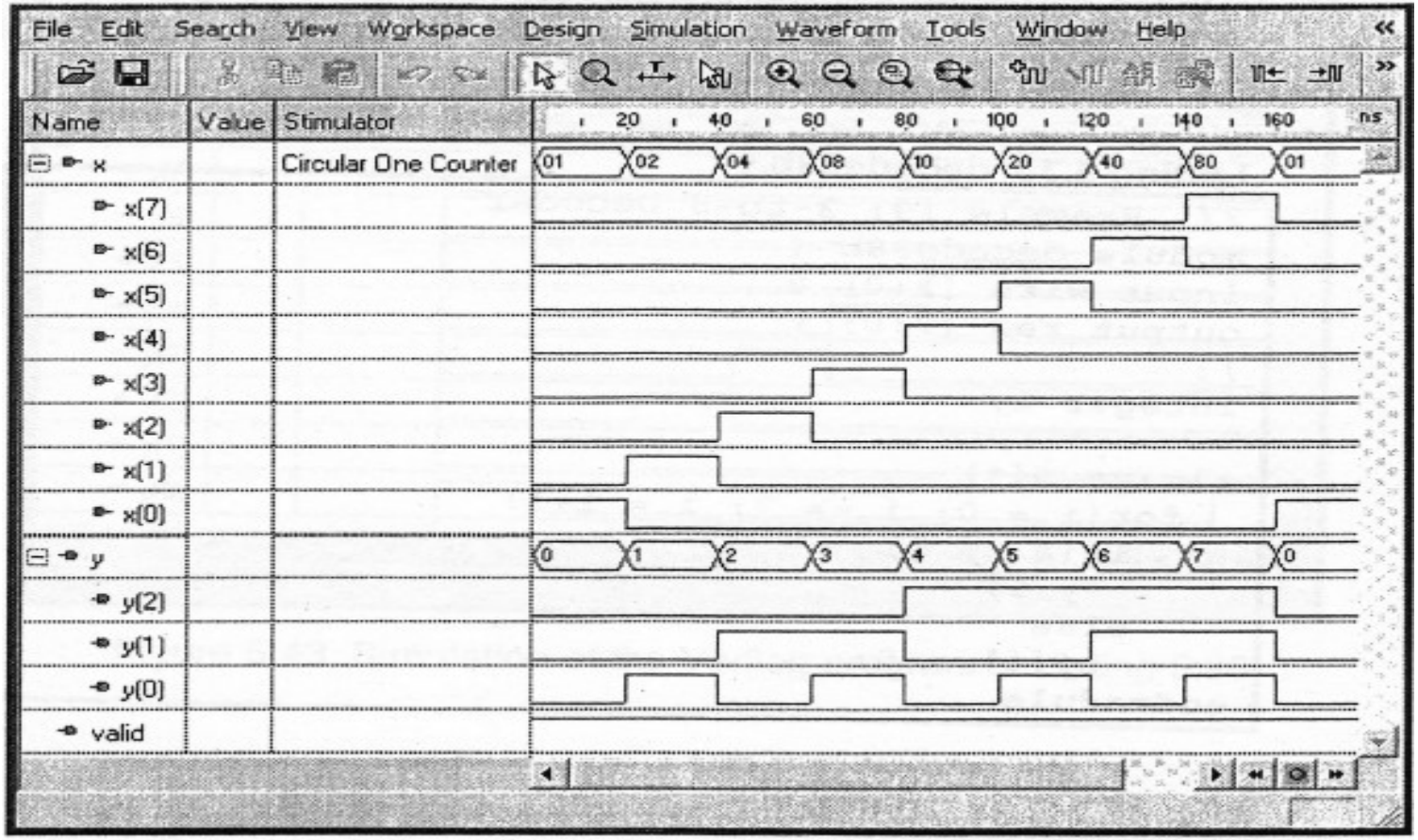
8-ból:3-ba kódoló strukturális leírás

```
module encode83a (
input wire [7:0] x ,
output wire [2:0] y ,
output wire valid
);

assign y[2] = x[7] | x[6] | x[5] | x[4];
assign y[1] = x[7] | x[6] | x[3] | x[2];
assign y[0] = x[7] | x[5] | x[3] | x[1];
assign valid = |x;

endmodule
```

- Adja hozzá a Verilog test fixture fájlt és végezze el a szimulációt.
- Rajzolja le a kódoló igazságtáblázatát.

8:3 kódoló - szimuláció

A test fixture fájlban hozza létre az alábbi ábra szerint a nyolc bites gerjesztő jelet és ellenőrizze az y kimenetet.



8-ból:3-ba kódoló implementálása

Adja hozzá a projekthez a következő Top modul fájlt

```
module encode83a_top (
input wire [7:0] sw ,
output wire [2:0] ld ,
output wire dp
);
wire valid;
assign dp = ~valid;

encode83a E1 (.x(sw),
      .y(ld),
      .valid(valid)
);

endmodule
```

Adja hozzá a megfelelő ucf fájlt.
Konfigurációs fájl generálása, letöltése és a működés tesztelése a kártyán

Lab5_1c feladat

Prioritásos kódoló viselkedési leírása - if-el

```
// 3-Bit 1-of-9 Priority Encoder
//Verilog 1995

module v_priority_encoder_1 (sw, ld);
   input  [7:0] sw;
   output [2:0] ld;
   reg    [2:0] ld;

   always @(sw)
   begin
      if      (sw[0]) ld = 3'b000;
      else if (sw[1]) ld = 3'b001;
      else if (sw[2]) ld = 3'b010;
      else if (sw[3]) ld = 3'b011;
      else if (sw[4]) ld = 3'b100;
      else if (sw[5]) ld = 3'b101;
      else if (sw[6]) ld = 3'b110;
      else if (sw[7]) ld = 3'b111;
      else            ld = 3'bxxx;
   end

endmodule
```

```
// 3-Bit 1-of-9 Priority Encoder
//Verilog 2001

module v_priority_encoder_1 (input  [7:0] sw,
                             output reg [2:0] ld);

   always @(sw)
   begin
      if      (sw[0]) ld = 3'b000;
      else if (sw[1]) ld = 3'b001;
      else if (sw[2]) ld = 3'b010;
      else if (sw[3]) ld = 3'b011;
      else if (sw[4]) ld = 3'b100;
      else if (sw[5]) ld = 3'b101;
      else if (sw[6]) ld = 3'b110;
      else if (sw[7]) ld = 3'b111;
      else            ld = 3'bxxx;
   end

endmodule
```

Adja hozzá a megfelelő ucf fájlt.

Konfigurációs fájl generálása, letöltése és a működés tesztelése a kártyán

Lab5_1d feladat (szorgalmi)

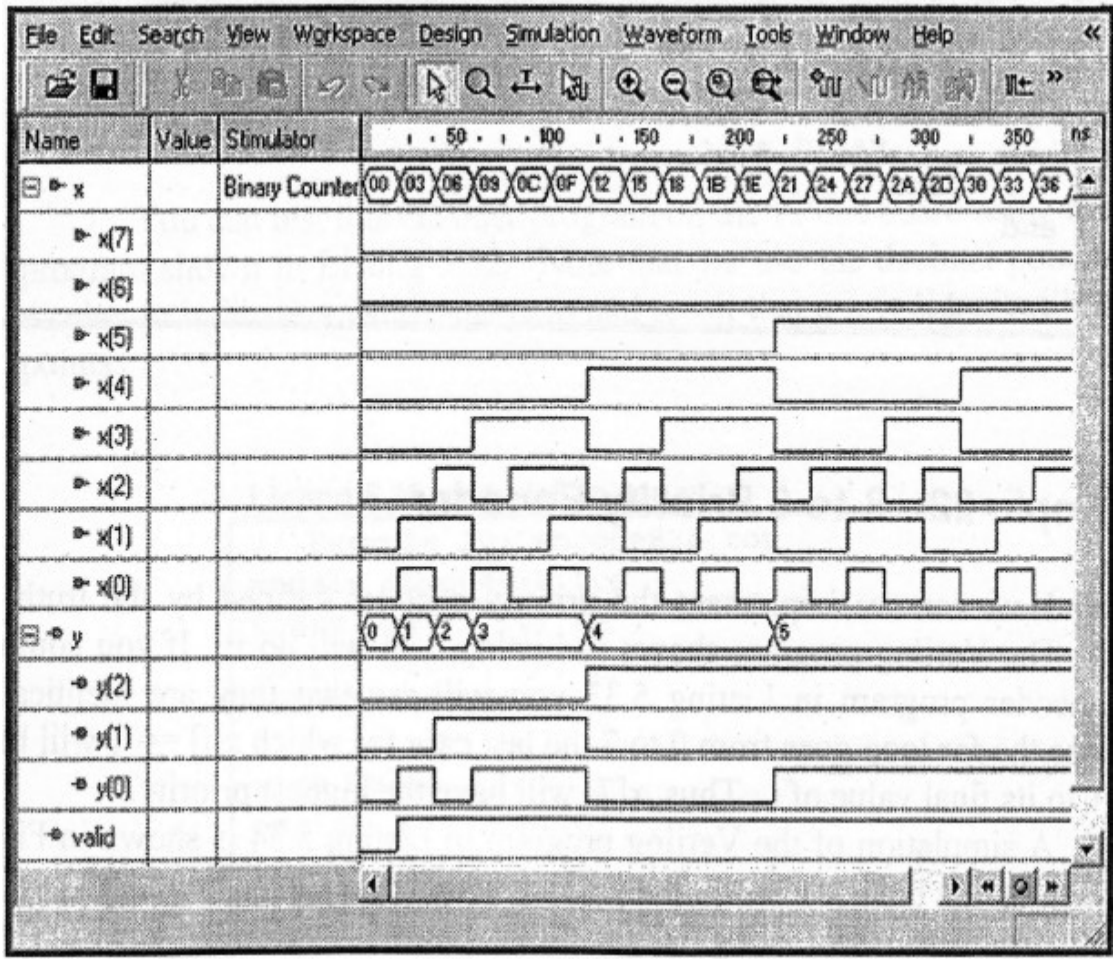
Prioritásos kódoló – for Loop-al

x0	x1	x2	x3	x4	x5	x6	x7	y2	y1	y0
1	0	0	0	0	0	0	0	0	0	0
X	1	0	0	0	0	0	0	0	0	1
X	X	1	0	0	0	0	0	0	1	0
X	X	X	1	0	0	0	0	0	1	1
X	X	X	X	1	0	0	0	1	0	0
X	X	X	X	X	1	0	0	1	0	1
X	X	X	X	X	X	1	0	1	1	0
X	X	X	X	X	X	X	1	1	1	1

- Ha egyidejűleg egynél több bemenet aktív, az eredmény nem meghatározható.
- A megoldás: a bemenetekhez prioritást rendelünk
- Ha egy vagy több bemenet aktív, akkor a legmagasabb prioritásúnak megfelelő eredményt kapjuk

```
module pencode83 (
input wire [7:0] x ,
output reg [2:0] y ,
output reg valid
);
integer i;

always @(*)
begin
  y = 0;
  valid = 0;
  for(i = 0; i <= 7; i = i+1)
    if(x[i] == 1)
      begin
         y = i;
         valid = 1;
      end
end

endmodule
```

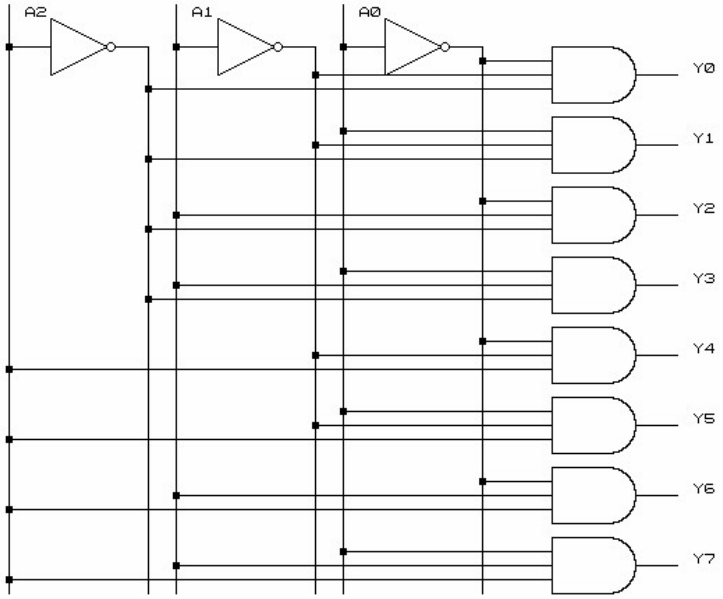
Prioritás kódoló – szimuláció



Lab5_2 feladat: Dekódolók

Lab5_2a feladat: Bináris dekódoló 3-ról 8-ra

A_2	A_1	A_0	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1




```
module decod(
    input A0,
    input A1,
    input A2,
    output [7:0] led
    );
            assign led[0]=~A0&~A1&~A2;
            assign led[1]=A0&~A1&~A2;
            assign led[2]=~A0&A1&~A2;
            assign led[3]=A0&A1&~A2;
            assign led[4]=~A0&~A1&A2;
            assign led[5]=A0&~A1&A2;
            assign led[6]=~A0&A1&A2;
            assign led[7]=A0&A1&A2;

endmodule
```

Lab5_2b feladat: Bináris dekódoló 3-ról 8-ra

- viselkedési leírás -

```
// 1-of-8 decoder (One-Hot)
module v_decoders_1 (input [2:0] sel, output reg [7:0] res);
   always @(sel or res)
   begin
      case (sel)
         3'b000 : res = 8'b00000001;
         3'b001 : res = 8'b00000010;
         3'b010 : res = 8'b00000100;
         3'b011 : res = 8'b00001000;
         3'b100 : res = 8'b00010000;
         3'b101 : res = 8'b00100000;
         3'b110 : res = 8'b01000000;
         default : res = 8'b10000000;
      endcase
   end
endmodule
```

```
// 1-of-8 decoder (One-Cold)
module v_decoders_1 (input [2:0] sel, output reg [7:0] res);
   always @(sel)
   begin
      case (sel)
         3'b000 : res = 8'b11111110;
         3'b001 : res = 8'b11111101;
         3'b010 : res = 8'b11111011;
         3'b011 : res = 8'b11110111;
         3'b100 : res = 8'b11101111;
         3'b101 : res = 8'b11011111;
         3'b110 : res = 8'b10111111;
         default : res = 8'b01111111;
      endcase
   end
endmodule
```

A dekódoló leírása a funkcionális kódrészlettel (válaszon egyet a fentiek közül)
Adja hozzá a megfelelő ucf fájlt. (sel [2:0] <=> sw [2:0]; res [7:0] <=> ld [7:0])
Konfigurációs fájl generálása, letöltése és a működés tesztelése a kártyán

Lab5_3a: Multiplexerek

- 2:1 multiplexer

```
module mux_21 (input in0, in1, sel, output r);
assign r = (sel==1'b1) ? in1 : in0;
endmodule
```

Assign

```
module mux_21 (input in0, in1, sel, output reg r);
always @ (*)
if (sel==1'b1) r <= in1;
else           r <= in0;
endmodule
```

If

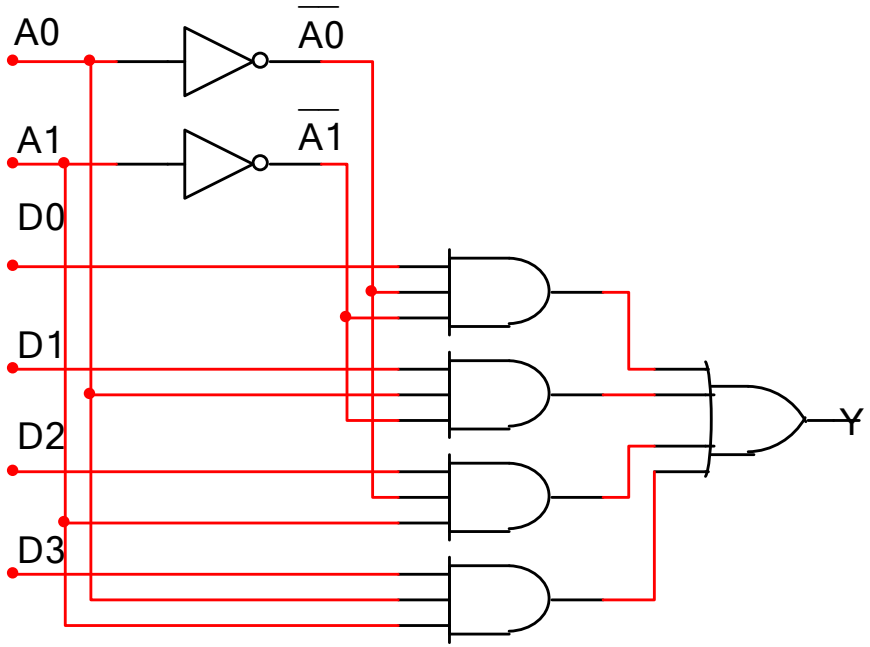
```
module mux_21 (input in0, in1, sel, output reg r);
always @ (*)
case(sel)
    1'b0:    r <= in0;
    1'b1:    r <= in1;
endmodule
```

Case

- A Multiplexer leírása a funkcionális kódrészlettel (válaszon egyet a fentiek közül)
- A Nexysx.ucf fájl hozzáadása és adaptálása (az inputok = sw kapcsolok, az r kimenet egy LED)
- Konfigurációs fájl generálása, letöltése és a működés tesztelése a kártyán

Lab5_3b feladat:

- 4:1 multiplexer



Y = ~A1 & ~A0 & D0 | ~A1 & A0 & D1 | A1 & ~A0 & D2 | A1 & A0 & D3

Lab5_3c feladat:

- 4:1 multiplexer

```
module mux_41 (input in0, in1, in2, in3, input [1:0] sel, output reg r);
always @ (*)
case(sel)
        2'b00: r <= in0;
        2'b01: r <= in1;
        2'b10: r <= in2;
        2'b11: r <= in3;
endcase
endmodule
```

- A Multiplexer leírása a fenti kódrészlettel
- A Nexysx.ucf fájl hozzáadása és adaptálása
- Konfigurációs fájl generálása, letöltése és a működés tesztelése a kártyán
- Rajzolja le a kódoló igazságtáblázatát.

Lab5_3d feladat:

Generikus Multiplexer

```
module mux2g
#(parameter N = 4)
(input wire [N-1:0] a,
 input wire [N-1:0] b,
 input wire s,
 output reg [N-1:0] y
);

always @(*)
      if(s == 0)
     y = a;
  else
     y = b;

endmodule
```

```
module mux28(
input wire [7:0] a,
input wire [7:0] b,
input wire s,
output wire [7:0] y
);

mux2g #(
  .N(8))
M8 (.a(a),
  .b(b),
  .s(s),
  .y(y)
);

 endmodule
```

Feladat:

Írjon Verilog kódban egy 4 bites bementekkel rendelkező 3:1 multiplexert