

Digitális Technika

Dr. Oniga István
Debreceni Egyetem, Informatikai Kar

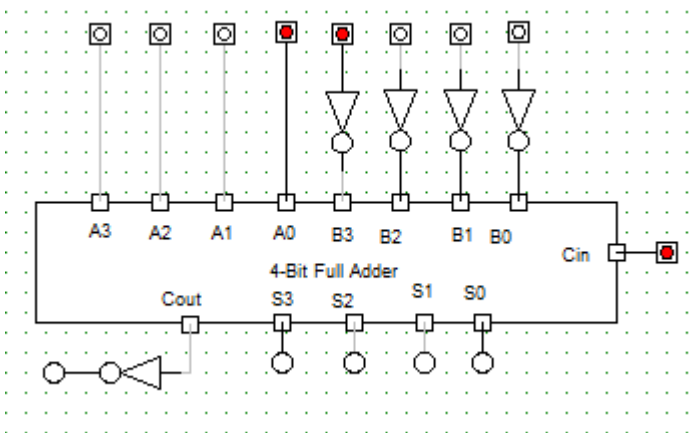
7. Laboratóriumi gyakorlat

Aritmetikai-logikai egységek

- 4 bites összeadó/kivonó
- 1 bites ALU
- 4-bites ALU – tervezése Verilog-ban
- 4-bites ALU – eredmény megjelenítése 7 szegmenses kijelzőn

Lab7_1a: 4 bites kivonó

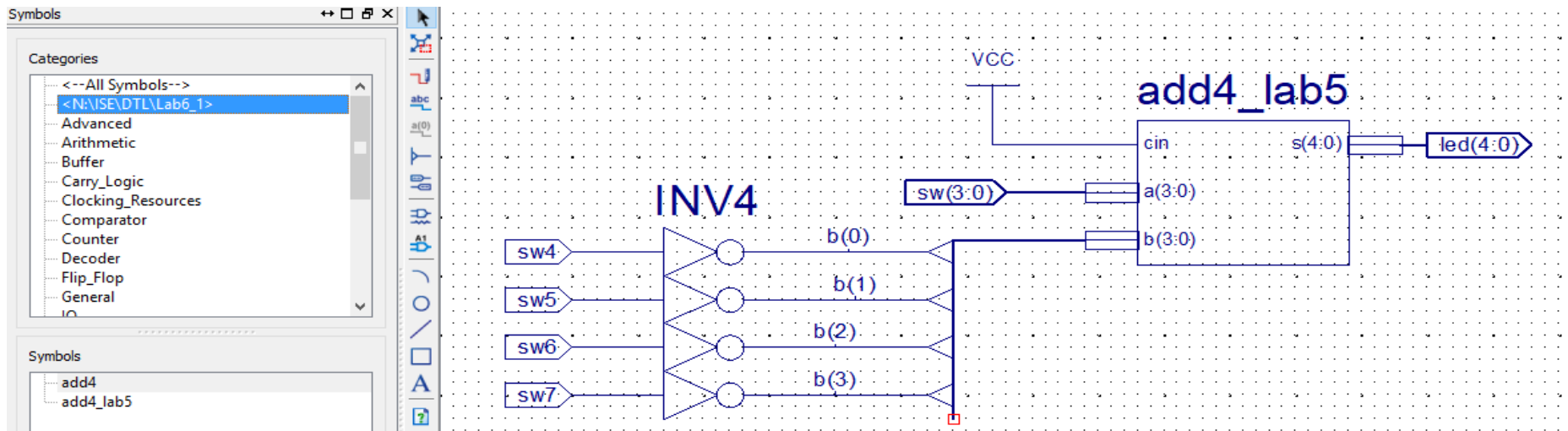
- Hozunk létre egy új projektet (Lab7_1)
- Adjunk hozzá (Add Copy of Source) a Lab6_5b feladathoz elkészített add4 .v forrásfájlt és adjunk hozzá egy cin nevű bemenetet.
- A Processes ablakban válasszuk ki a Create Schematic Symbol (dupla klikk)
- Adjunk hozzá a projekthez egy új Schematic típusú forrásfájlt (Sub4)
- A rajz szerkesztőben a Categories ablakban válasszuk ki a aktuális munka mappát
- A Symbols ablakban meg találjuk az imént létrehozott add4 szimbólumát. Ezt adjuk hozzá a rajzlaphoz és fejezzük be a rajzot mind az a következő lapon látható



The screenshot shows the Quartus II software interface. On the left, the Hierarchy window displays the project structure: Lab6_1, xc7a100t-3csg324, Sub4 (Sub4.sch), and add4 - add4 (add4.v). The Processes window shows 'No Processes Running'. The Design Utilities window is open, showing options like 'Create Schematic Symbol', 'View HDL Instantiation Template', and 'Check Syntax'. On the right, the VHDL code for the add4 module is displayed:

```
1 timescale 1ns / 1ps
2 ///////////////////////////////////////////////////
3 // Company:
4 // Engineer:
5 //
6 // Create Date:    16:52:07 03/21/2018
7 // Design Name:
8 // Module Name:    add4
9 // Project Name:
10 // Target Devices:
11 // Tool versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 ///////////////////////////////////////////////////
21 module add4 (input [3:0] a, b, input cin, output [4:0] s);
22     assign s = a + b + cin;
23 endmodule
24
```

Lab7_1a: 4 bites kivonó



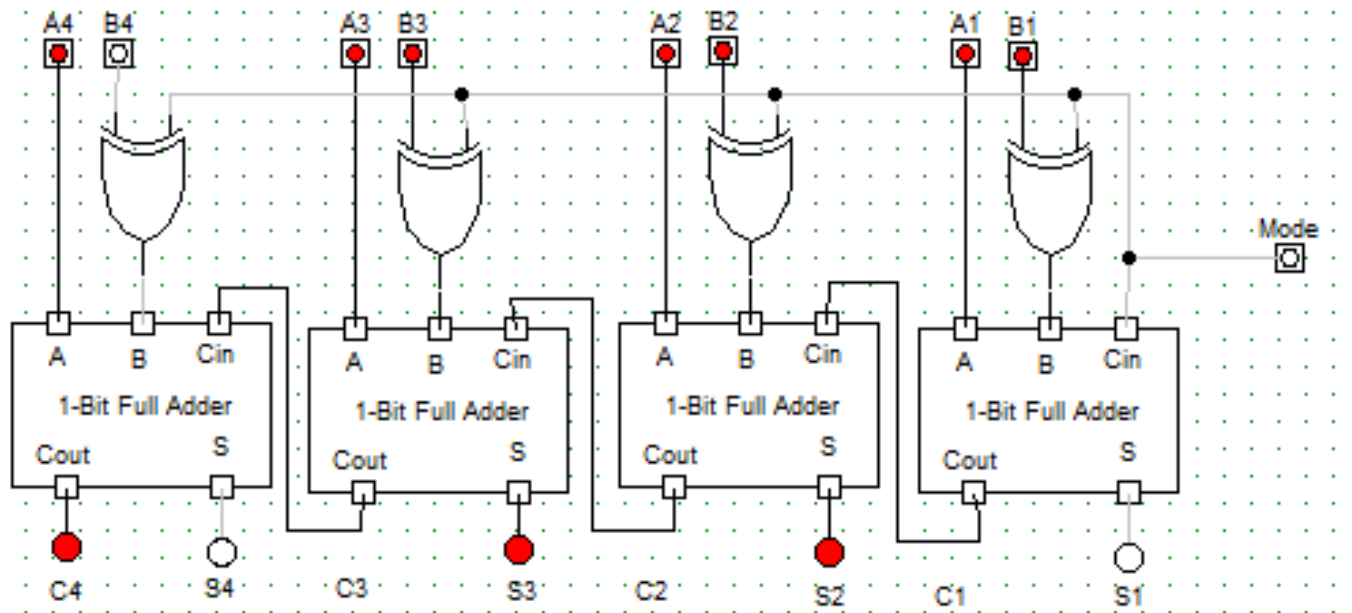
- A Nexys.ucf fájl hozzáadása és adaptálása. Figyeljük meg a sw(3:0) busz illetve sw4, sw5, sw6, sw7 bitek (vezetékek) specifikálási módját az ucf fájlban
- Konfigurációs fájl generálása, letöltése és a működés tesztelése a kártyán

- NET "sw<0>" LOC=J15 | IOSTANDARD=LVCMOS33; #IO_L24N_T3_RS0_15
- NET "sw<1>" LOC=L16 | IOSTANDARD=LVCMOS33; #IO_L3N_T0_DQS_EMCCCLK_14
- NET "sw<2>" LOC=M13 | IOSTANDARD=LVCMOS33; #IO_L6N_T0_D08_VREF_14
- NET "sw<3>" LOC=R15 | IOSTANDARD=LVCMOS33; #IO_L13N_T2_MRCC_14
- NET "sw4" LOC=R17 | IOSTANDARD=LVCMOS33; #IO_L12N_T1_MRCC_14
- NET "sw5" LOC=T18 | IOSTANDARD=LVCMOS33; #IO_L7N_T1_D10_14
- NET "sw6" LOC=U18 | IOSTANDARD=LVCMOS33; #IO_L17N_T2_A13_D29_14
- NET "sw7" LOC=R13 | IOSTANDARD=LVCMOS33; #IO_L5N_T0_D07_14
- NET "led<0>" LOC=H17 | IOSTANDARD=LVCMOS33; #IO_L18P_T2_A24_15
- NET "led<1>" LOC=K15 | IOSTANDARD=LVCMOS33; #IO_L24P_T3_RS1_15
- NET "led<2>" LOC=J13 | IOSTANDARD=LVCMOS33; #IO_L17N_T2_A25_15
- NET "led<3>" LOC=N14 | IOSTANDARD=LVCMOS33; #IO_L8P_T1_D11_14
- NET "led<4>" LOC=R18 | IOSTANDARD=LVCMOS33; #IO_L7P_T1_D09_14

Lab7_1b: 4 bites összeadó/kivonó

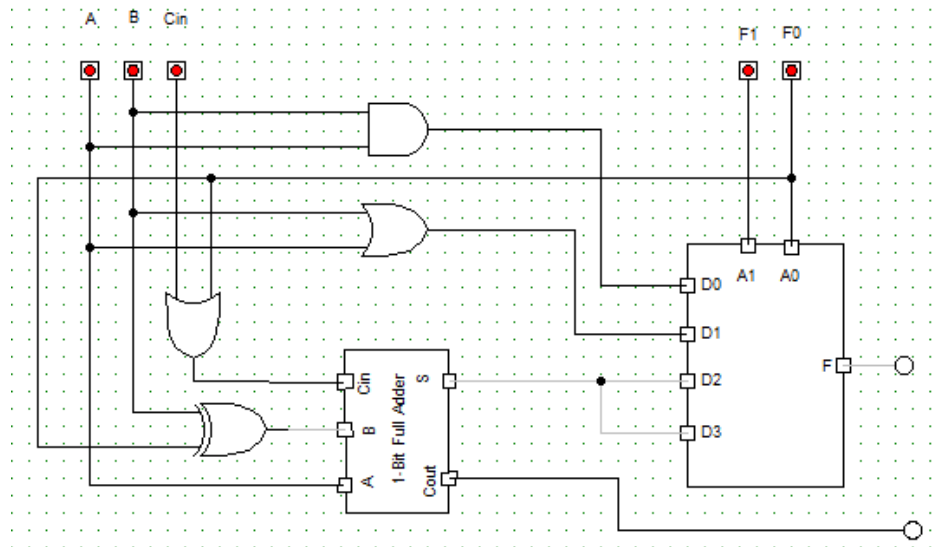
(szorgalmi feladat)

- Hozunk létre egy új projektet (Lab7_1b)
- Adjunk hozzá (Add Copy of Source) a Lab6_4 feladathoz elkészített add1_full.v forrásfájlt.
- Ebül készítsünk egy új rajz szimbólumot.
- Adjunk hozzá a projekthez egy új Schematic típusú forrásfájlt (Lab7_1b.v)
- A rajz szerkesztőben készítsük el a következő rajzot
- A Nexysx.ucf fájl hozzáadása és adaptálása
- Konfigurációs fájl generálása, letöltése és a működés tesztelése a kártyán



Lab7_2: 1 bites ALU

- Hozunk létre egy új projektet (Lab7_2)
- Adjunk hozzá (Add Copy of Source) a Lab6_4 feladathoz elkészített add1_full.v forrásfájlt.
- A Processes ablakban válasszuk ki a Create Schematic Symbol (dupla klikk)
- Adjunk hozzá a projekthez egy új Schematic típusú forrásfájlt (Sub4)
- A rajz szerkesztőben a Categories ablakban válaszunk ki a aktuális munka mappát
- A Symbols ablakban meg találjuk az imént létrehozott add4 szimbólumát. Ezt adjuk hozzá a rajzlaphoz és készítsük el a következő rajzot.
- Hasonlóképpen készítsünk egy multiplexer szimbólumot a Lab5_3c feladatnál használt Verilog kódból. (Alternatíva: a rajz szerkesztő programba létezik egy M4_1E szimbólum. Az E engedélyező lábát Vcc-re kell kötni))



Lab7_2: 1 bites ALU

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

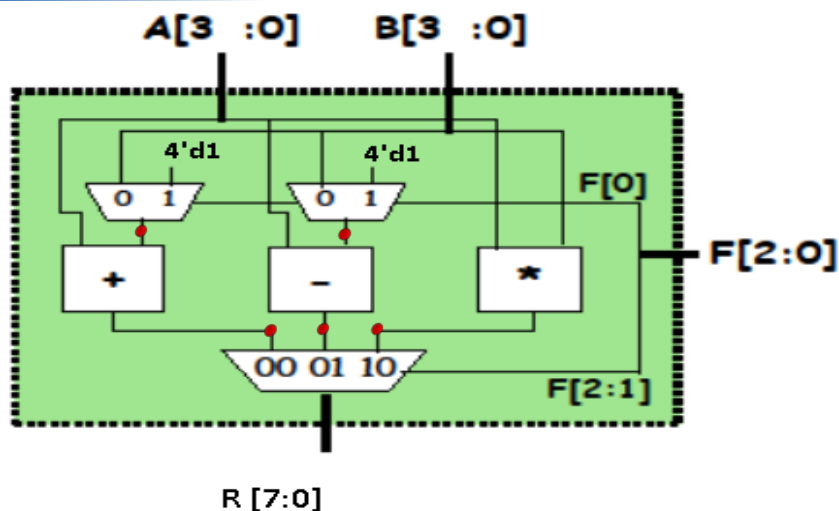
Logikai műveletek

F1	F0	A	B	Cin	F
0	0	0	0	x	
0	0	0	1	x	
0	0	1	0	x	
0	0	1	1	x	
0	1	0	0	x	
0	1	0	1	x	
0	1	1	0	x	
0	1	1	1	x	
1	0				

Aritmetikai műveletek

F1	F0	A	B	Cin	F	Cout
1	0	0	0	0		
1	0	0	1	0		
1	0	1	0	0		
1	0	1	1	0		
1	0	0	0	1		
1	0	0	1	1		
1	0	1	0	1		
1	0	1	1	1		
1	1	0	0	0		
1	1	0	1	0		
1	1	1	0	0		
1	1	1	1	0		
1	1	0	0	1		
1	1	0	1	1		
1	1	1	0	1		
1	1	1	1	1		

Lab7_3: 4 bites ALU



F2	F1	F0	Function
0	0	0	A + B
0	0	1	A + 1
0	1	0	A - B
0	1	1	A - 1
1	0	X	A * B

- 4 és 8 bites belső vezetékek

- Hozunk létre egy új projektet Lab7_3
- Adjunk hozzá egy új „Verilog” forrásfájlt (alu_top). Ez a modul köti össze az ALU moduljait

```
module alu_top(input [3:0] a, b, input [2:0] f, output [7:0] r ); wire
[3:0] addmux_out, submux_out;
wire [7:0] add_out, sub_out, mul_out;
    mux2_4 adder_mux(b, 4'd1, f[0], addmux_out);
    mux2_4 sub_mux(b, 4'd1, f[0], submux_out);
    add4 our_adder(a, addmux_out, add_out);
    sub4 our_subtracrer(a, submux_out, sub_out);
    mul4 our_multiplier(a,b,mul_out);
    mux3_8 output_mux(add_out, sub_out, mul_out, f[2:1], r);
endmodule
```

- Adjunk hozzá egy új „Verilog” forrásfájlt (alu4_modulok)
- A forrásfájlok specifikálása (Másoljuk bele a következő oldalon lévő modul leírásokat)

Modulok leírása

```
module mux2_4(input [3:0] i0, i1, input sel, output [7:0] out);
    assign out = sel ? i1 : i0;
endmodule

module mux3_8(input [7:0] i0, i1, i2, input [1:0] sel, output reg [7:0] out);
    always @(i0 or i1 or i2 or sel)
        begin
            case (sel)
                2'b00: out = i0;
                2'b01: out = i1;
                2'b10: out = i2;
                default: out = 8'bx;
            endcase
        end
endmodule

module add4(input [3:0] i0, i1, output [7:0] sum);
    assign sum=i0+i1;
endmodule

module sub4(input [3:0] i0, i1, output [7:0] diff);
    assign diff=i0-i1;
endmodule

module mul4(input [3:0] i0, i1, output [7:0] prod);
    assign prod=i0*i1;
endmodule
```

ALU implementálása és tesztelése

- A Nexysx.ucf fájl hozzáadása és adaptálása (F[2:0] <=> sw [15:13]; a [3:0] <=> sw [3:0], b [3:0] <=> sw [7:4], r [7:0] <=> led [7:0])
- Konfigurációs fájl generálása, letöltése és a működés tesztelése a kártyán
- A sw [7:0] kapcsolókkal alítsa be a következő operandusokat:
 - a = 3, b = 2 és sw [15:13] kapcsolók segítségével alítsa be egymás után az 5 lehetséges műveletet. Minden estben írja be a táblázatba a művelet eredményét és véleményezze a helyességét.

F2 sw[15]	F1 sw[14]	F0 sw[13]	r[7]	r[6]	r[5]	r[4]	r[3]	r[2]	r[1]	r[0]
0	0	0								
0	0	1								
0	1	0								
0	1	1								
1	0	X								

- Végezze el a fenti méréseket újabb operandusokkal: a = 10, b = 12 és véleményezze az eredmények helyességét. Bizonyítsa be algebrai módszerekkel hogy a bináris kivonás jól működik vagy nem.
- Mennyi lesz a szorzás legnagyobb eredménye?

```

NET "a<0>" LOC=J15 | IOSTANDARD=LVCMOS33;
NET "a<1>" LOC=L16 | IOSTANDARD=LVCMOS33;
NET "a<2>" LOC=M13 | IOSTANDARD=LVCMOS33;
NET "a<3>" LOC=R15 | IOSTANDARD=LVCMOS33;

NET "b<0>" LOC=R17 | IOSTANDARD=LVCMOS33;
NET "b<1>" LOC=T18 | IOSTANDARD=LVCMOS33;
NET "b<2>" LOC=U18 | IOSTANDARD=LVCMOS33;
NET "b<3>" LOC=R13 | IOSTANDARD=LVCMOS33;

NET "f<0>" LOC=U12 | IOSTANDARD=LVCMOS33;
NET "f<1>" LOC=U11 | IOSTANDARD=LVCMOS33;
NET "f<2>" LOC=V10 | IOSTANDARD=LVCMOS33;
## LEDs
NET "r<0>" LOC=H17 | IOSTANDARD=LVCMOS33;
NET "r<1>" LOC=K15 | IOSTANDARD=LVCMOS33;
NET "r<2>" LOC=J13 | IOSTANDARD=LVCMOS33;
NET "r<3>" LOC=N14 | IOSTANDARD=LVCMOS33;
NET "r<4>" LOC=R18 | IOSTANDARD=LVCMOS33;
NET "r<5>" LOC=V17 | IOSTANDARD=LVCMOS33;
NET "r<6>" LOC=U17 | IOSTANDARD=LVCMOS33;
NET "r<7>" LOC=U16 | IOSTANDARD=LVCMOS33;
    
```

Lab7_3b: 4 bites ALU – szimuláció

(szorgalmi feladat)

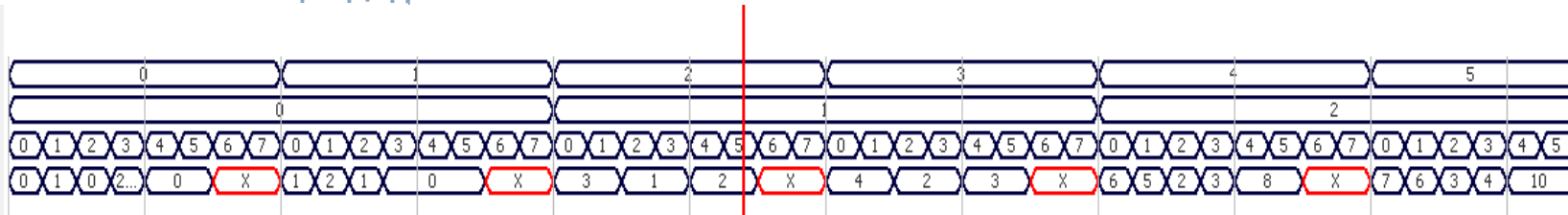
ALU teszt modul leírása

```

initial begin
    a = 0;
    b = 0;
    f = 0;
end

always # 400
    a = a+1;
always # 800
    b = b+1;
always # 50
    f = f+1;
    
```

Name	Value
a[3:0]	2
b[3:0]	1
f[2:0]	5
r[7:0]	2



Belső vezetékek állapotának megvizsgálása

The screenshot displays a digital logic simulator interface with the following components:

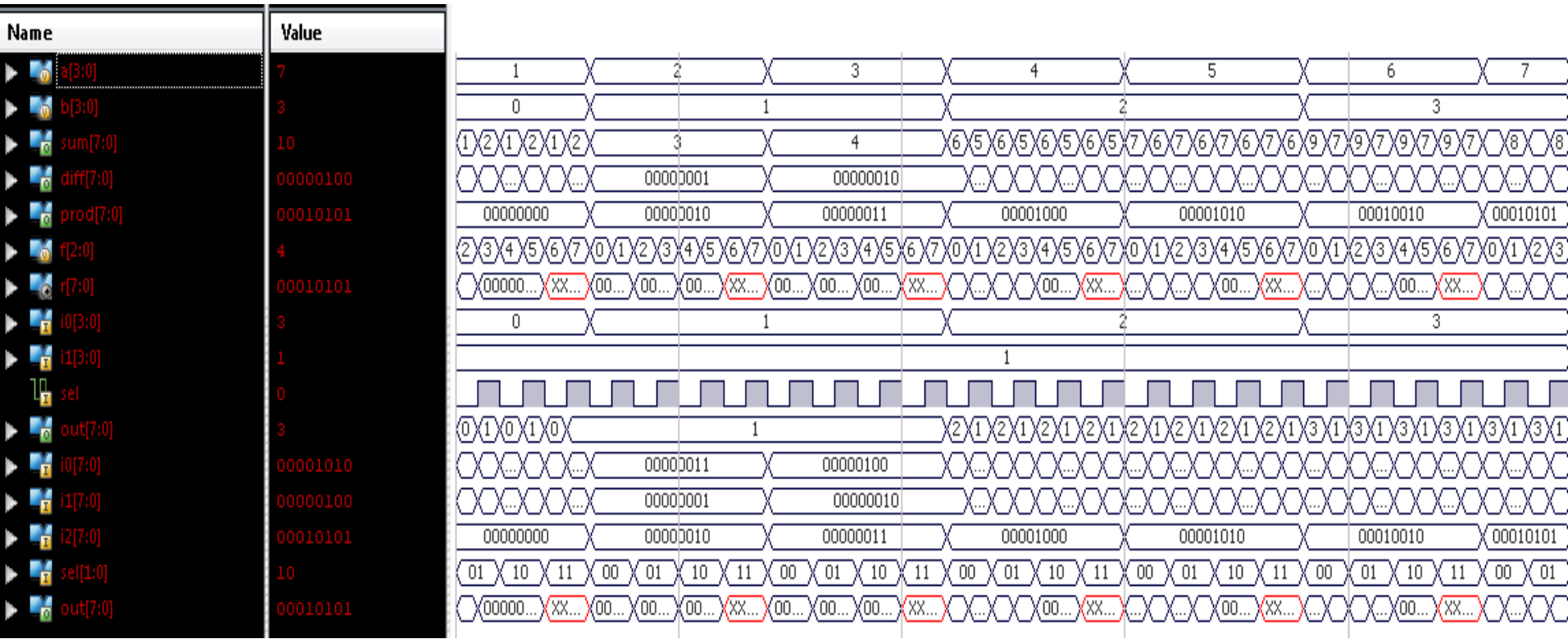
- Instances and Processes:** A tree view on the left showing the hierarchy of components, with 'adder_mux' selected under the 'uut' instance.
- Objects:** A table showing simulation objects for 'adder_mux' with their current values.
- Signal List:** A central table listing internal signals and their values.
- Timing Diagram:** A waveform viewer on the right showing the temporal behavior of various signals.

Simulation Objects for adder_mux	
Object Name	Value
i0[3:0]	0011
i1[3:0]	0001
sel	0
out[7:0]	00000011

Name	Value
a[3:0]	7
b[3:0]	3
sum[7:0]	10
diff[7:0]	00000100
prod[7:0]	00010101
f[2:0]	4
r[7:0]	00010101
i0[3:0]	3
i1[3:0]	1
sel	0
out[7:0]	3
i0[7:0]	00001010
i1[7:0]	00000100

The timing diagram shows waveforms for signals like 'a', 'b', 'sum', 'diff', 'prod', 'f', 'r', 'i0', 'i1', 'sel', and 'out'. A scale bar indicates a duration of 3,000,000 ns.

Belső vezetékek állapotának megvizsgálása



Lab7_4:

4-bites ALU eredménye 7szegmenses kijelzőn

- Hozunk létre egy új projektet Lab7_4
- Adjunk hozzá (Add copy of source) a Lab7_3 feladat összes forrásfájlját (alu_top, alu4_modulok).
- Adjunk hozzá a Lab4_3 feladatnál elkészített hét szegmenses kijelző leírásának egy másolatát (hex7seg.v)
- Adjunk hozzá egy új „Verilog” forrásfájlt (alu4_top). Ez a modul köti össze az alu_top és a hex7seg modulokat.

```
module alu4_top( input [3:0] a, input [3:0] b, input [2:0] f,
                output [6:0] a_to_g, output [7:0] an, output dp);
    wire [7:0] r1;

    assign an = 8'b11111110; // 1 digit on, 7 digits off
    assign dp = 1; // dp off

    hex7seg D4 (.x(r1[3:0]),.a_to_g(a_to_g));
    alu_top D1 (.a(a), .b(b), .f(f), .r(r1));

endmodule
```

ALU implementálása és tesztelése

- A Nexysx.ucf fájl hozzáadása és adaptálása (F[2:0] \Leftarrow sw [15:13]; a [3:0] \Leftarrow sw [3:0], b [3:0] \Leftarrow sw [7:4], a_to_g [6:0] \Leftarrow a_to_g [6:0], an[7:0] \Leftarrow an[7:0], dp \Leftarrow dp)
- Konfigurációs fájl generálása, letöltése és a működés tesztelése a kártyán
- A sw [7:0] kapcsolókkal alítsa be a következő operandusokat:
 - a = 3, b = 2 és sw [15:13] kapcsolók segítségével alítsa be egymás után az 5 lehetséges műveletet. Minden esetben jegyezze fel a kijelzőn látható eredményét és véleményezze a helyességét.

F2	F1	F0	Function
0	0	0	A + B
0	0	1	A + 1
0	1	0	A - B
0	1	1	A - 1
1	0	X	A * B

F2 sw[15]	F1 sw[14]	F0 sw[13]	
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	X	

- Végezze el a fenti méréseket újabb operandusokkal: és véleményezze az eredmények helyességét.
- Milyen korlátai vannak az fenti áramkörnek?
- Mennyi lesz a szorzás legnagyobb eredménye?