DIGITAL SYSTEM DESIGN APPLICATIONS

Experiment 3

VARIOUS IMPLEMENTATIONS OF BOOLE FUNCTIONS

Berfin Duman
040190108

Contents

1 Realization with SSI Library

You can find the testbench given to us in the report and all the switch, button, delay and loc placements we used for the constrain file in the appendixes.
First of all, I created a new folder called "week3" for experiment3. I opened a new rtl project named project_1 in the Experiment3 folder. I created three_different_methods.v as a source, and added the SSI_Lib.v and MSI_Lib.v files that I wrote in the previous weeks, by making a copy as written. I added the constrain file I downloaded from Ninova. Then I added the true FPGA board as requested in the report. And I made it ready to write the codes expected from us in the assignment.

1.1 Truth Table, Karnaugh Map and Circuit and Circuit Drawing

Before writing the code, I created boolean expressions by extracting the karnaugh map of the given truth table as requested. I demonstrated these expressions with two-port circuits. I made the drawing with my logis, because it was harder to erase and notice the wrong places when I drew it by hand.

I_3	I_2	I_1	I_0	f_3	f_2	f_1	f_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	1
0	1	1	0	0	0	1	0
0	1	1	1	0	0	1	1
1	0	0	0	0	0	0	0
1	0	0	1	0	0	1	0
1	0	1	0	0	1	0	0
1	0	1	1	0	1	1	0
1	1	0	0	0	0	0	0
1	1	0	1	0	0	1	1
1	1	1	0	0	1	1	0
1	1	1	1	1	0	0	1

Table 1: Truth Table of Experiment-3

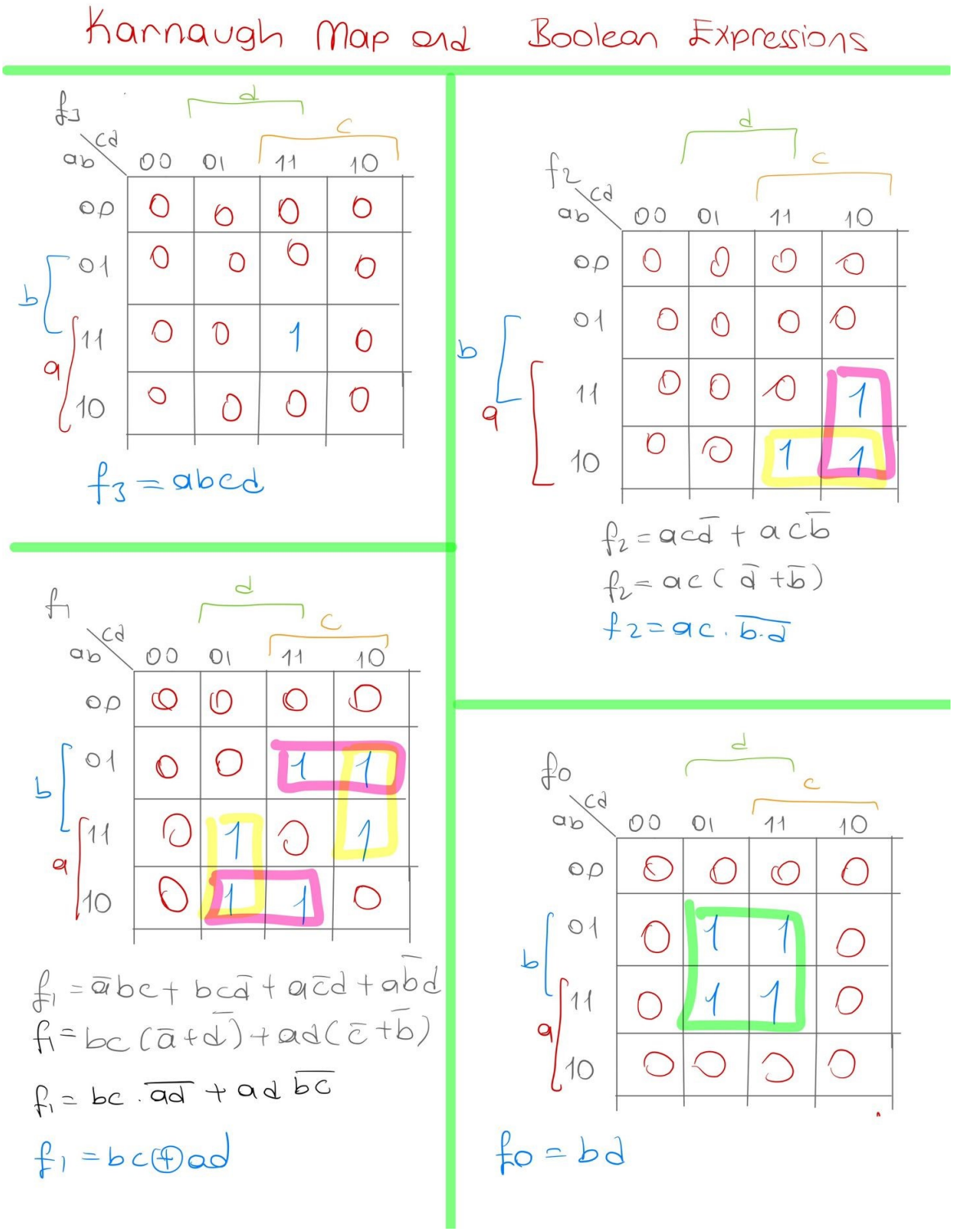


Figure 1: Karnaugh Map of Exp-3

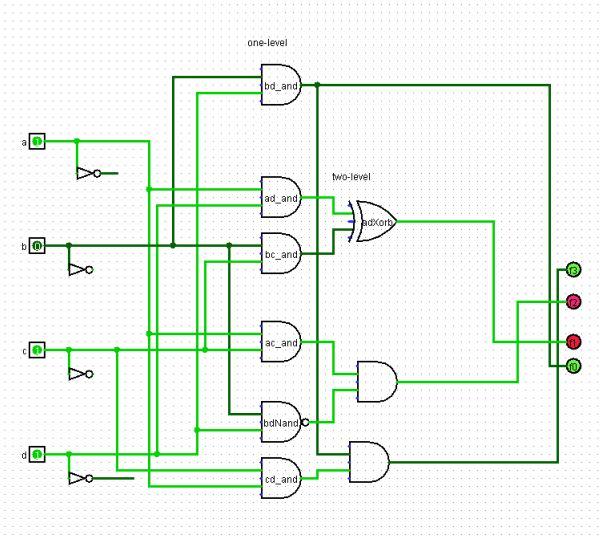


Figure 2: Logic Circuit Diagram of Exp-3

1.2 Verilog Code, Testbanch Code, Behavioral Simulation and Schematics

First, I created the with_SSI module under the three_different_methods module file. I gave the module a, b, c, d inputs, each of 1 bit, as input, and 1 bit outputs, f3, f2, f1, f0, as output. And I implemented 4 functions that obtained boolean expressions (nand, exor, and) using SSI_library. I have named the wires and modules for readability.

Listing 1: with_SSI_Lib.v

```
module with_SSI(
input a,b,c,d,
output f0,f1,f2,f3
    );
    AND f0_and(.O(f0),.I1(b),.I2(d));
    wire bcand_wire;
    wire adand_wire;
    AND f1_and1(.O(bcand_wire),.I1(b),.I2(c));
    AND f1_and2(.O(adand_wire),.I1(a),.I2(d));
    EXOR f1_exor(.O(f1),.I1(adand_wire),.I2(bcand_wire));
    wire f2_wire1;
    wire ac_andwire;
    NAND f2_and1(.O(f2_wire1),.I1(b),.I2(d));
    AND f2_and2(.O(ac_andwire),.I1(a),.I2(c));
    AND f2_and3(.O(f2),.I1(ac_andwire),.I2(f2_wire1));
    AND f3_and2(.O(f3),.I1(bcand_wire),.I2(adand_wire));
endmodule
```

I loaded the experiment3_tb.v file given in the homework files as simulation source. I added the simulation result and TCL output generated according to testbench to the report (listing 2, Fig14)

Then, as requested, I connected the inputs to the switches and the outputs to the LEDs in the order requested in the report: connected a, b, c, d inputs to switches SW3, SW2, SW1, SW0 respectively. Also, connected outputs f3, f2, f1, f0 to LED outputs LED3, LED2, LED1, LED0 respectively.

Listing 2: Simulation Results (with_SSI)

```
# run 1000ns
{a, b, c, d} = 0000 => {f3, f2, f1, f0} = 0000 -- TRUE
{a, b, c, d} = 0001 => {f3, f2, f1, f0} = 0000 -- TRUE
{a, b, c, d} = 0010 => {f3, f2, f1, f0} = 0000 -- TRUE
{a, b, c, d} = 0011 => {f3, f2, f1, f0} = 0000 -- TRUE
{a, b, c, d} = 0100 => {f3, f2, f1, f0} = 0000 -- TRUE
{a, b, c, d} = 0101 => {f3, f2, f1, f0} = 0001 -- TRUE
{a, b, c, d} = 0110 => {f3, f2, f1, f0} = 0010 -- TRUE
{a, b, c, d} = 0111 => {f3, f2, f1, f0} = 0011 -- TRUE
{a, b, c, d} = 1000 => {f3, f2, f1, f0} = 0000 -- TRUE
{a, b, c, d} = 1001 => {f3, f2, f1, f0} = 0010 -- TRUE
{a, b, c, d} = 1010 => {f3, f2, f1, f0} = 0100 -- TRUE
{a, b, c, d} = 1011 => {f3, f2, f1, f0} = 0110 -- TRUE
{a, b, c, d} = 1100 => {f3, f2, f1, f0} = 0000 -- TRUE
{a, b, c, d} = 1101 => {f3, f2, f1, f0} = 0011 -- TRUE
{a, b, c, d} = 1110 => {f3, f2, f1, f0} = 0110 -- TRUE
{a, b, c, d} = 1111 => {f3, f2, f1, f0} = 1001 -- TRUE
```

```
$finish called at time : 800 ns : File "C:/Xilinx/SST/week3/experiment3_tb.v"
Line 52
INFO: [USF-XSim-96] XSim completed. Design snapshot 'experiment3_tb_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
launch_simulation: Time (s): cpu = 00:00:04 ; elapsed = 00:00:06 . Memory (MB):
peak = 3824.926 ; gain = 0.000
```

As a result of the given testbench, I observed that the outputs in the TCL output were 100% compatible with the expected outputs. The outputs matched the truth table exactly as they should.

Listing 3: edit constrain file

```
set_property -dict { PACKAGE_PIN J15   IOSTANDARD LVCMOS33 } [get_ports { d }];
set_property -dict { PACKAGE_PIN L16   IOSTANDARD LVCMOS33 } [get_ports { c }];
set_property -dict { PACKAGE_PIN M13   IOSTANDARD LVCMOS33 } [get_ports { b }];
set_property -dict { PACKAGE_PIN R15   IOSTANDARD LVCMOS33 } [get_ports { a }];
set_property -dict { PACKAGE_PIN H17   IOSTANDARD LVCMOS33 } [get_ports { f0 }];
set_property -dict { PACKAGE_PIN K15   IOSTANDARD LVCMOS33 } [get_ports { f1 }];
set_property -dict { PACKAGE_PIN J13   IOSTANDARD LVCMOS33 } [get_ports { f2 }];
set_property -dict { PACKAGE_PIN N14   IOSTANDARD LVCMOS33 } [get_ports { f3 }];
```

Then, I obtained Rtl and technology schematics by performing Rtl analysis and synthesis, respectively.

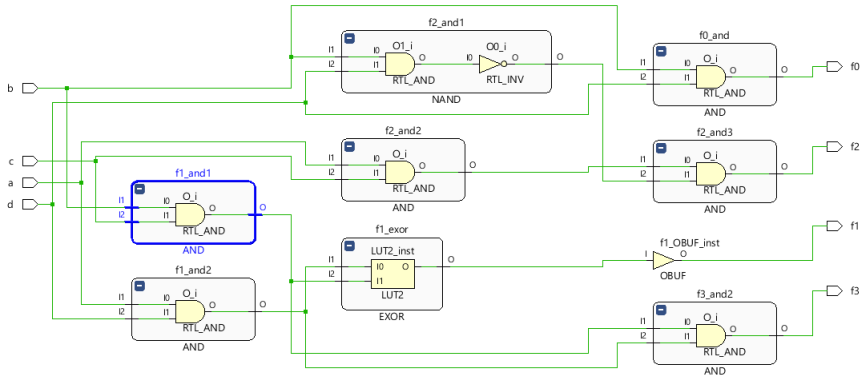


Figure 3: with_SSI RTL Schematic

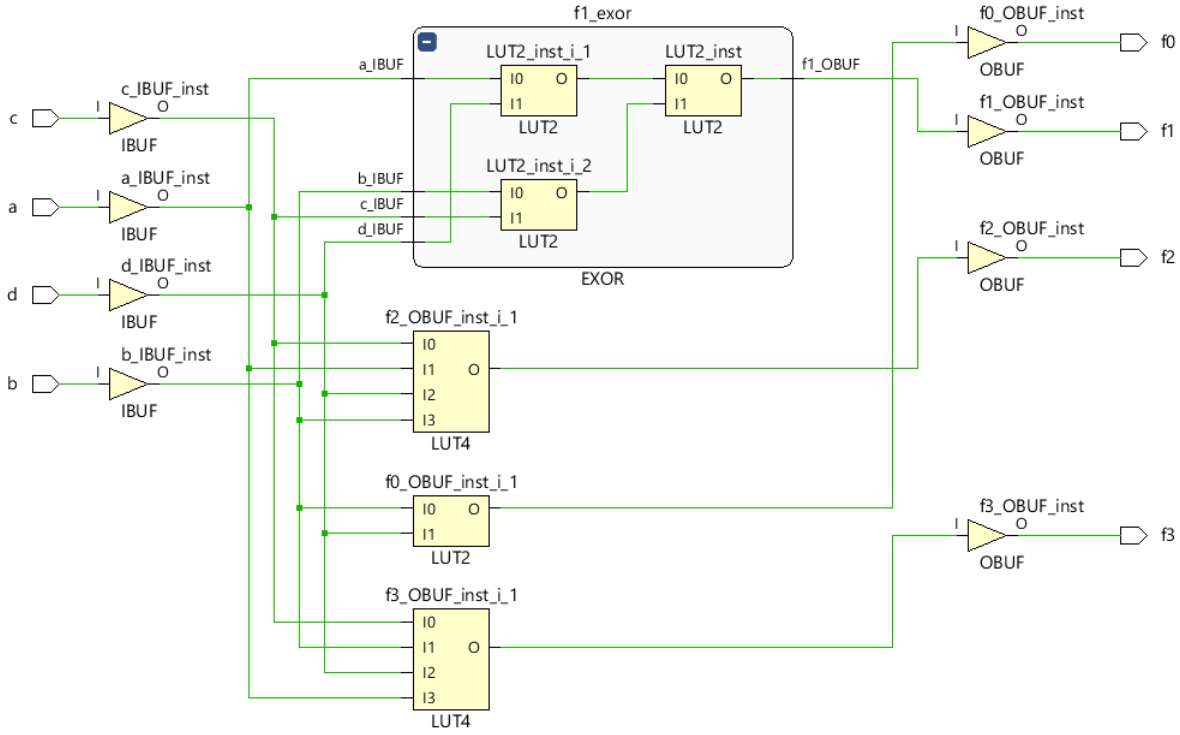


Figure 4: with_SSI Technology Schematic

In the RTL table, as we thought at the beginning, the functions are created with logic gates, and in the technology table, the functions are implemented with LUTs. It is consistent with the outputs as we expected. f0 is schematized with Lut2 because it only connects b and d, and one LUT4 since f2 and f3 are determined with only one canonical minterm, and f1 is schematized with 3 lut2 because it consists of three separate gates, 2 and and xor.

1.3 Implementation Parts and Reports

According to the post-implementation utilization report summary of the module we created, 5 and 8 I/O ports are as we see in the technology schematic.

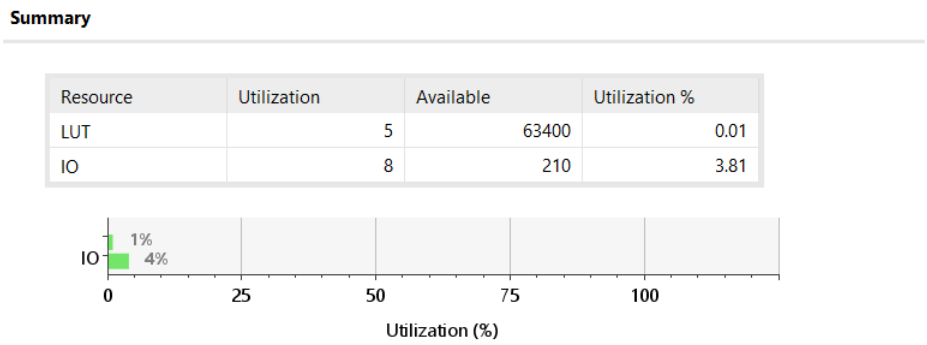


Figure 5: with_SSI Utilization Reports

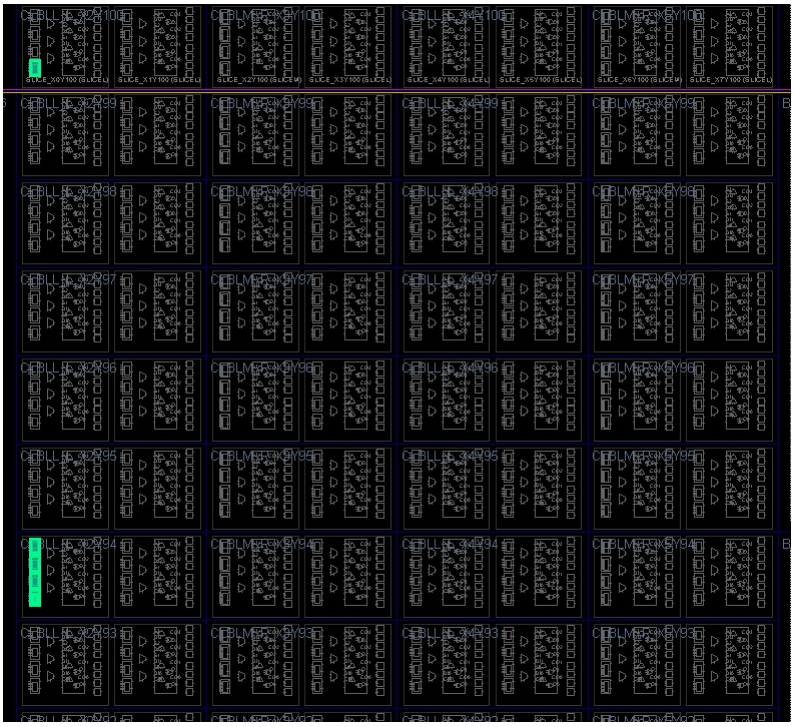
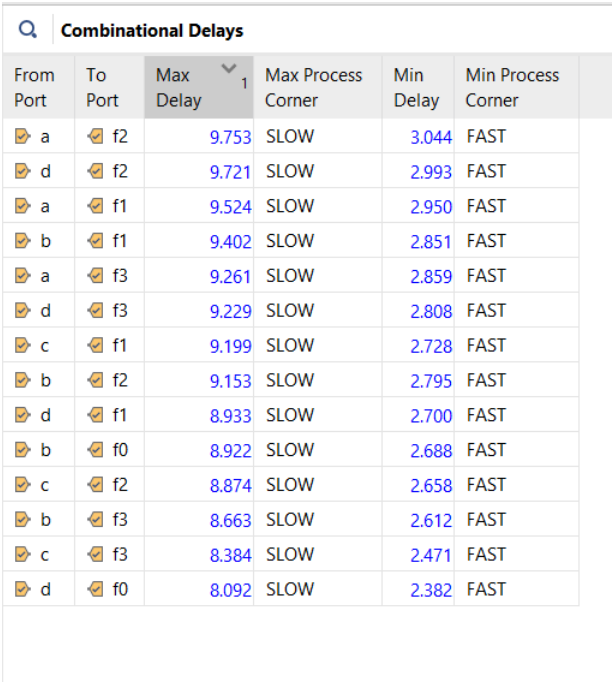


Figure 6: with_SSI Place of LUTS

I implemented it after the synthesis and when I pulled out the timing report as written in the instructions, it was seen that the highest delay was 9.753 at the f2 output from the "a" input. We can also state that the delay of 8 of our connections is above 9.



Combinational Delays

From Port	To Port	Max Delay	Max Process Corner	Min Delay	Min Process Corner
a	f2	9.753	SLOW	3.044	FAST
d	f2	9.721	SLOW	2.993	FAST
a	f1	9.524	SLOW	2.950	FAST
b	f1	9.402	SLOW	2.851	FAST
a	f3	9.261	SLOW	2.859	FAST
d	f3	9.229	SLOW	2.808	FAST
c	f1	9.199	SLOW	2.728	FAST
b	f2	9.153	SLOW	2.795	FAST
d	f1	8.933	SLOW	2.700	FAST
b	f0	8.922	SLOW	2.688	FAST
c	f2	8.874	SLOW	2.658	FAST
b	f3	8.663	SLOW	2.612	FAST
c	f3	8.384	SLOW	2.471	FAST
d	f0	8.092	SLOW	2.382	FAST

Figure 7: with_SSI Timing Delay

Then, to reduce the delay, I add the line that will cause the constraint to my constrain file (this line will force the structure to ensure that the delay does not exceed 9 and will try to optimize accordingly):

Listing 4: edit constrain file

```
set_max_delay 9 -from [all_inputs] -to [all_outputs]
```

I added this code and ran it. It completed the implementation but gave an error, the error code above right is "Implementation completed, Failed Timing!"
According to this error, the timing could not go down to 9 ns or below as we wanted.

Combinational Delays

From Port	To Port	Max Delay	Max Process Corner
a	f1	9.008	SLOW
b	f0	8.825	SLOW
d	f0	8.777	SLOW
a	f2	8.696	SLOW
b	f1	8.659	SLOW
c	f2	8.635	SLOW
b	f2	8.611	SLOW
d	f2	8.506	SLOW
c	f1	8.478	SLOW
d	f1	8.452	SLOW
a	f3	8.177	SLOW
d	f3	8.168	SLOW
c	f3	8.099	SLOW
b	f3	7.975	SLOW

Figure 8: with_SSI Timing Delay After Adding Limit

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	-0.008 ns	Worst Hold Slack (WHS):	inf	Worst Pulse Width Slack (WPWS):	NA
Total Negative Slack (TNS):	-0.008 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	NA
Number of Failing Endpoints:	1	Number of Failing Endpoints:	0	Number of Failing Endpoints:	NA
Total Number of Endpoints:	4	Total Number of Endpoints:	4	Total Number of Endpoints:	NA

Timing constraints are not met.

Figure 9: with_SSI Timing Report- Negatif Slack

As we saw in the implementation result, when we look at the delays table above, we see that the connection from a to f1 cannot go below 9 and shows a stretch of 0.008. This is a negative slack, and this is not an unexpected situation, and even the fact that it reduced it by approximately 0.75 ns may be pleasing, considering that this is directly automatically optimized. Negative slack is digital design and FPGA/ASIC applications refers to the difference between the arrival time and the required time for a signal, indicating when a path fails to meet specified timing constraints. In automatic optimization, optimization can be done on the FPGA side (placement, etc.) by taking into account the constraint we impose on the constraine, but this may consume power.

1.4 LOC

Listing 5: Set LOC- Constrain File

```
set_property LOC SLICE_X12Y67 [get_cells f0_OBUF_inst_i_1]
set_property LOC SLICE_X12Y66 [get_cells f2_OBUF_inst_i_1]
```

```
set_property LOC SLICE_X13Y67 [get_cells f1_exor/LUT2_inst]
set_property LOC SLICE_X14Y64 [get_cells f1_exor/LUT2_inst_i_1]
```

Then, during the loc phase, I made the necessary additions to the constrain file and implemented it again. As a result of the implementation, I checked that the new locations of the cells were in the desired locations.

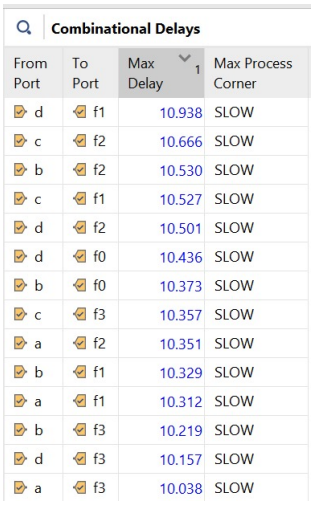
Figure 10: position of the lut on FPGA

Combinational Delays

From Port	To Port	M a	Max Process Corner	Min Delay	Min Process Corner
d	f1	12.048	SLOW	4.053	FAST
d	f2	11.757	SLOW	3.980	FAST
d	f3	11.478	SLOW	3.887	FAST
c	f1	11.380	SLOW	3.767	FAST
b	f1	11.186	SLOW	3.715	FAST
c	f2	11.096	SLOW	3.691	FAST
a	f1	11.072	SLOW	3.605	FAST
d	f0	10.977	SLOW	3.641	FAST
c	f3	10.812	SLOW	3.598	FAST
b	f0	10.692	SLOW	3.526	FAST
b	f2	10.686	SLOW	3.554	FAST
b	f3	10.406	SLOW	3.463	FAST
a	f2	10.297	SLOW	3.346	FAST
a	f3	10.015	SLOW	3.253	FAST

Figure 11: LOC Constrain Pad-to Pad Delay

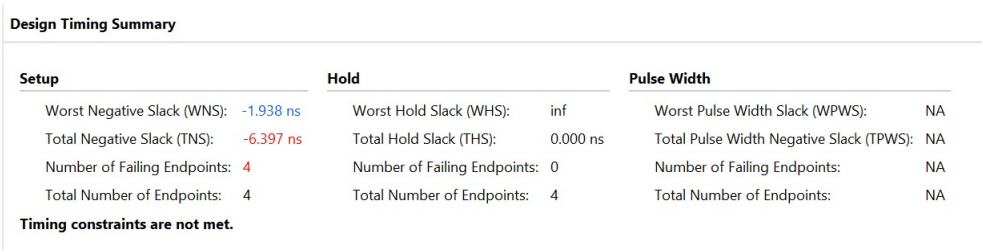
By removing the set_max_delay code from the comment, I observed both time constrain and location constrain conditions at the same time:



Combinational Delays

From Port	To Port	Max Delay	Max Process Corner
d	f1	10.938	SLOW
c	f2	10.666	SLOW
b	f2	10.530	SLOW
c	f1	10.527	SLOW
d	f2	10.501	SLOW
d	f0	10.436	SLOW
b	f0	10.373	SLOW
c	f3	10.357	SLOW
a	f2	10.351	SLOW
b	f1	10.329	SLOW
a	f1	10.312	SLOW
b	f3	10.219	SLOW
d	f3	10.157	SLOW
a	f3	10.038	SLOW

Figure 12: LOC and Set Delay Constrain Pad-to Pad Delay



Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	-1.938 ns	Worst Hold Slack (WHS):	inf	Worst Pulse Width Slack (WPWS):	NA
Total Negative Slack (TNS):	-6.397 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	NA
Number of Failing Endpoints:	4	Number of Failing Endpoints:	0	Number of Failing Endpoints:	NA
Total Number of Endpoints:	4	Total Number of Endpoints:	4	Total Number of Endpoints:	NA

Timing constraints are not met.

Figure 13: LOC and Set Delay Constraint Negative Slack

When we added the LOC command, the maximum delay was 12.048, while it was observed that it decreased to 10.938 ns when we set both loc and time limits.
Without any restrictions: 9.753
When max_delay constraint is introduced: 9.008
By loc edit:12.048
Loc + max_delay together: 10.938
When all 4 cases were examined, minimum max_delay was observed only in the case where we added the set delay command. One reason why changing the LOC has a negative effect may be that the LUT states it places are more optimized, and our command disrupts the optimization.

1.5 Compare of Simulations

Figure 14: SSİ_lib.v Behavioral Simulation Results

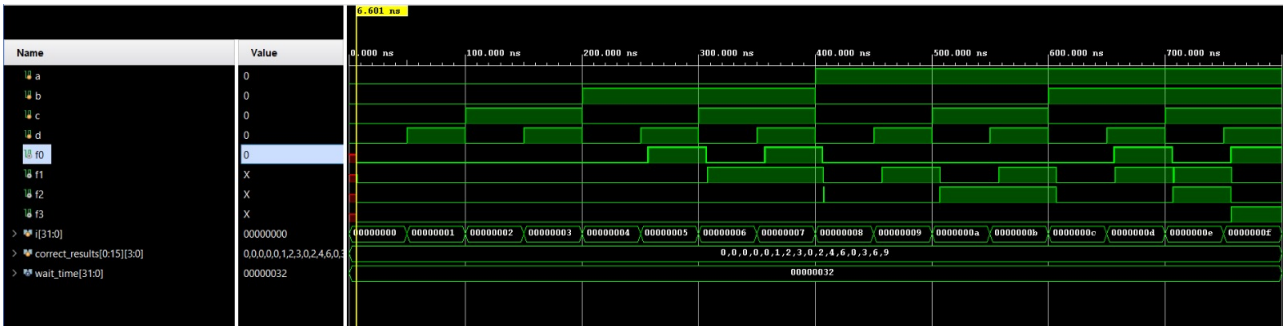


Figure 15: SSİ_lib.v Post Synthesis Timing Simulation Results

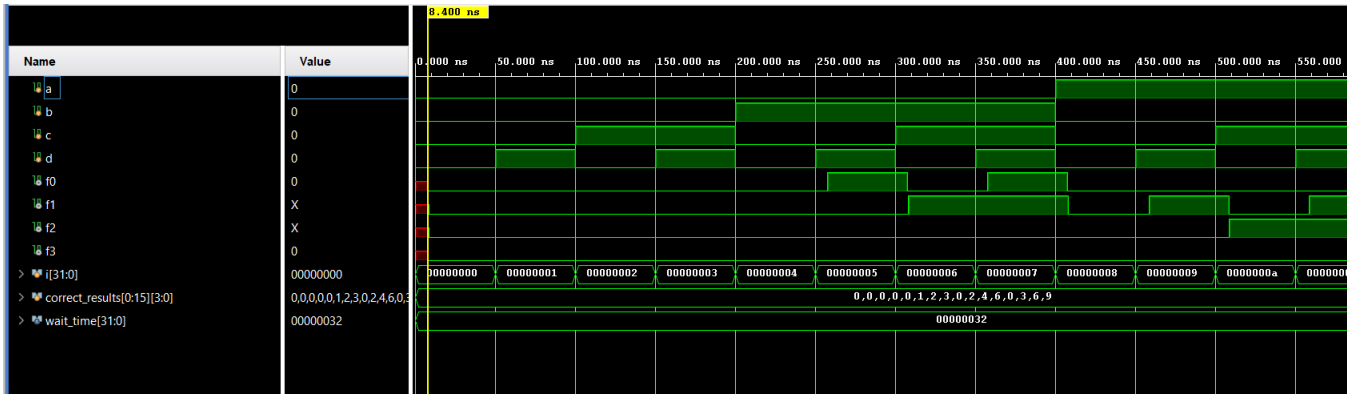


Figure 16: SSİ_lib.v Post Implementation Timing Simulation Results

In behavioral simulation it is only tested logically. Therefore, there is no delay in the simulation result. However, in the simulation after the synthesis and implementation, the hardware level has now been reached and therefore delays occur. In the synthesis, the necessary netlists are determined and in the implementation, as is now recognized in our FPGA card, results are produced by taking into account the situations that may occur during hardware. A delay of 6,601 ns is observed during synthesis and a delay of 8,840 ns after implementation.

2 Realization with Decoder

2.1 Hand-drawing Decoder-based Circuit Schematic

When I came to the decoder part, I first drew the circuit diagram (Fig 17) to perform the functions in the decoder.

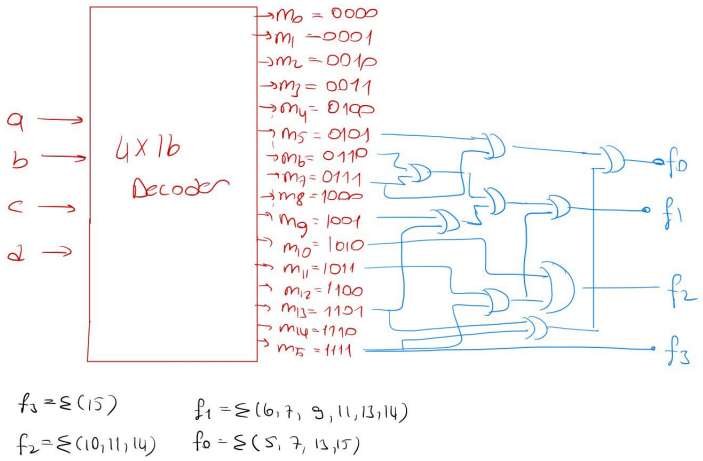


Figure 17: with_SSI Hand-drawing Decoder Circuit

2.2 Verilog Code, Testbanch Code and Behavioral Simulation

According to the circuit diagram I drew, I implemented the with_decoder module in the three_different_module.v file. I kept the input and output inputs the same as the first module.

Listing 6: with_decooder module verilog code

```
DECODER decoder1(.IN({a,b,c,d}),.OUT(OUT1));
wire f1_or2;
wire f1_or3;
wire f1_or4;
wire f12_or1;
wire f0_or1;
wire f0_or2;
```

```
OR f12or1(.O(f12_or1),.I1(OUT1[14]),.I2(OUT1[11]));
OR f2or2(.O(f2),.I1(f12_or1),.I2(OUT1[10]));
OR f1or2(.O(f1_or2),.I1(OUT1[9]),.I2(OUT1[7]));
OR f1or3(.O(f1_or3),.I1(OUT1[13]),.I2(OUT1[6]));
OR f1or4(.O(f1_or4),.I1(f12_or1),.I2(f1_or2));
OR f1or5(.O(f1),.I1(f1_or4),.I2(f1_or3));
OR f0o1(.O(f0_or1),.I1(OUT1[5]),.I2(OUT1[7]));
OR f0o2(.O(f0_or2),.I1(OUT1[13]),.I2(OUT1[15]));
OR f0o3(.O(f0),.I1(f0_or1),.I2(f0_or2));
assign f3 = OUT1[15];
endmodule
```

I gave the newly written module to the uut section of the testbench, which was prepared for testing with the test code:

Listing 7: Set Testbench for Testing Decoder Schematic

```
//with_SSI
    with_decoder
    //with_MUX
    UUT (.a(a),
         .b(b),
         .c(c),
         .d(d),
         .f0(f0),
         .f1(f1),
         .f2(f2),
         .f3(f3));
```

I set the with_decoder module to top and got the behavioral simulation.

Listing 8: Simulation Results (with_decoder

```
# run 1000ns
{a,b,c,d}=0000 => {f3,f2,f1,f0} = 0000 -- TRUE
{a,b,c,d}=0001 => {f3,f2,f1,f0} = 0000 -- TRUE
{a,b,c,d}=0010 => {f3,f2,f1,f0} = 0000 -- TRUE
{a,b,c,d}=0011 => {f3,f2,f1,f0} = 0000 -- TRUE
{a,b,c,d}=0100 => {f3,f2,f1,f0} = 0000 -- TRUE
{a,b,c,d}=0101 => {f3,f2,f1,f0} = 0001 -- TRUE
{a,b,c,d}=0110 => {f3,f2,f1,f0} = 0010 -- TRUE
{a,b,c,d}=0111 => {f3,f2,f1,f0} = 0011 -- TRUE
{a,b,c,d}=1000 => {f3,f2,f1,f0} = 0000 -- TRUE
{a,b,c,d}=1001 => {f3,f2,f1,f0} = 0010 -- TRUE
{a,b,c,d}=1010 => {f3,f2,f1,f0} = 0100 -- TRUE
{a,b,c,d}=1011 => {f3,f2,f1,f0} = 0110 -- TRUE
{a,b,c,d}=1100 => {f3,f2,f1,f0} = 0000 -- TRUE
```

```
{a,b,c,d}=1101 => {f3,f2,f1,f0} = 0011 -- TRUE
{a,b,c,d}=1110 => {f3,f2,f1,f0} = 0110 -- TRUE
{a,b,c,d}=1111 => {f3,f2,f1,f0} = 1001 -- TRUE
$finish called at time : 800 ns : File "C:/Xilinx/SST/week3/experiment3_tb.v" Line 5

INFO: [USF-XSim-96] XSim completed. Design snapshot 'experiment3_tb_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
launch_simulation: Time (s): cpu = 00:00:09 ; elapsed = 00:00:07 . Memory (MB): peak
```

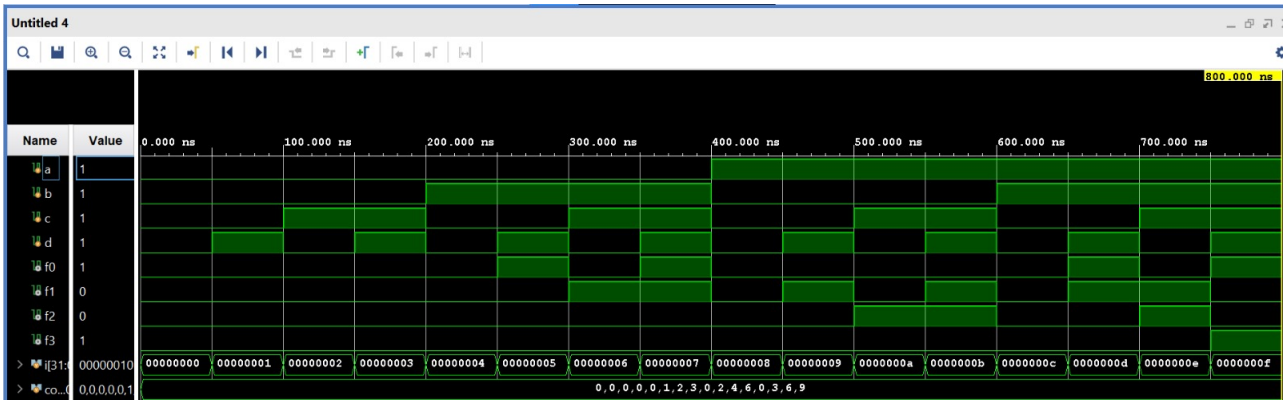


Figure 18: Behavioral Simulation with_decoder module

2.3 Rtl and Technology Schematics, Pad to Pad Delay

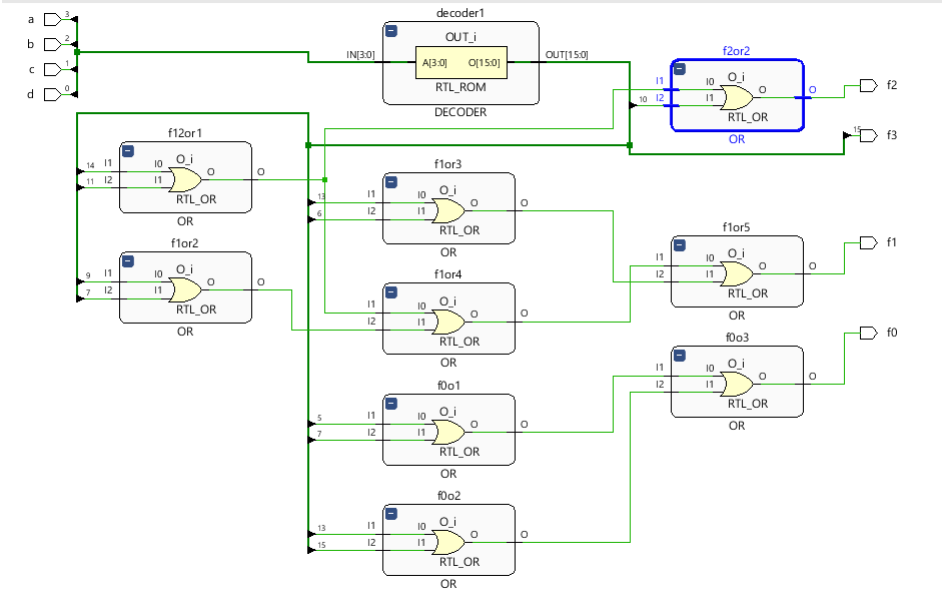


Figure 19: Rtl Schematic with_decoder module

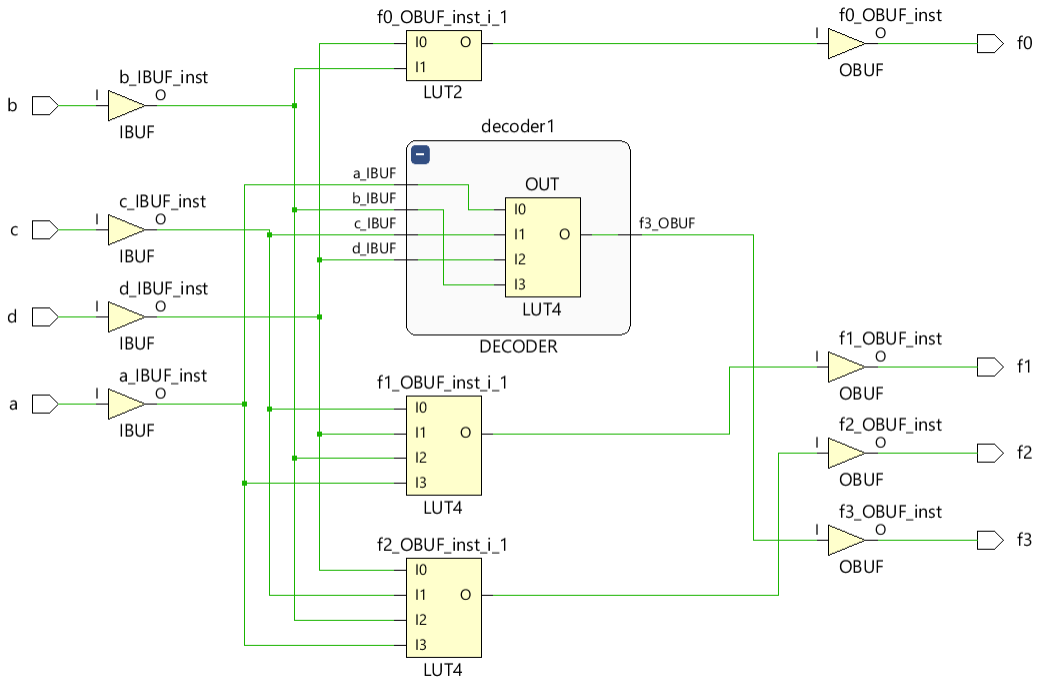


Figure 20: Technology Schematic with_decoder module

Combinational Delays

From Port	To Port	M a 1	Max Process Corner	Min Delay	Min Process Corner
a	f2	9.497	SLOW	2.979	FAST
a	f1	9.422	SLOW	2.895	FAST
a	f3	9.261	SLOW	2.866	FAST
b	f0	8.922	SLOW	2.688	FAST
d	f2	8.895	SLOW	2.722	FAST
d	f1	8.818	SLOW	2.635	FAST
b	f2	8.668	SLOW	2.659	FAST
b	f3	8.652	SLOW	2.603	FAST
c	f2	8.619	SLOW	2.591	FAST
b	f1	8.559	SLOW	2.579	FAST
c	f1	8.543	SLOW	2.508	FAST
d	f3	8.431	SLOW	2.533	FAST
c	f3	8.285	SLOW	2.445	FAST
d	f0	8.091	SLOW	2.381	FAST

Figure 21: Pad-to-Pad Delay with_decoder module

If we do not impose any restrictions on the constrain file, the max_delay from a input to f1 output is 9.497. It is observed that all delays have values of 8 and above.

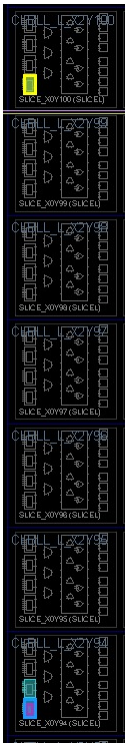


Figure 22: placement of luts on FPGA board

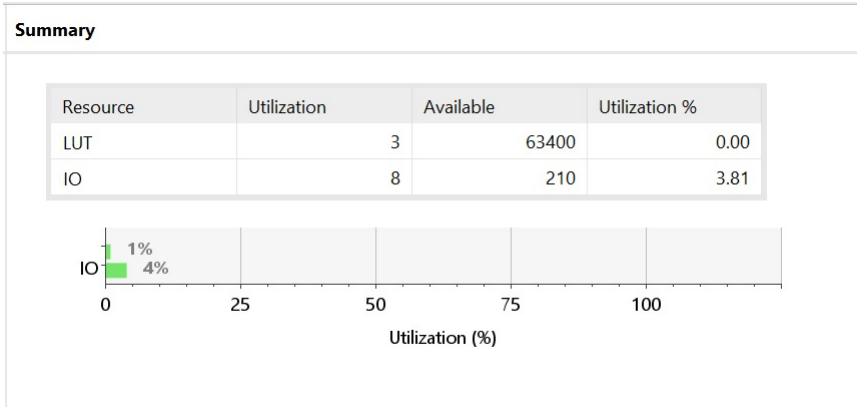


Figure 23: Utilization Report after implementation

When we compare it with our first module, while the number of luts was 5 in SSI, it decreased to 3 in the decoder. More efficient placement has been made.

2.3.1 Comparison of Simulation After Synthesis and Implementation Part

An initial delay was observed in both. 6.601 ns delay after synthesis and 8.157 ns delay after implementation part. As we expected, there was more initial delay after implementation. Because it takes into account the situations related to our FPGA in the implementation.

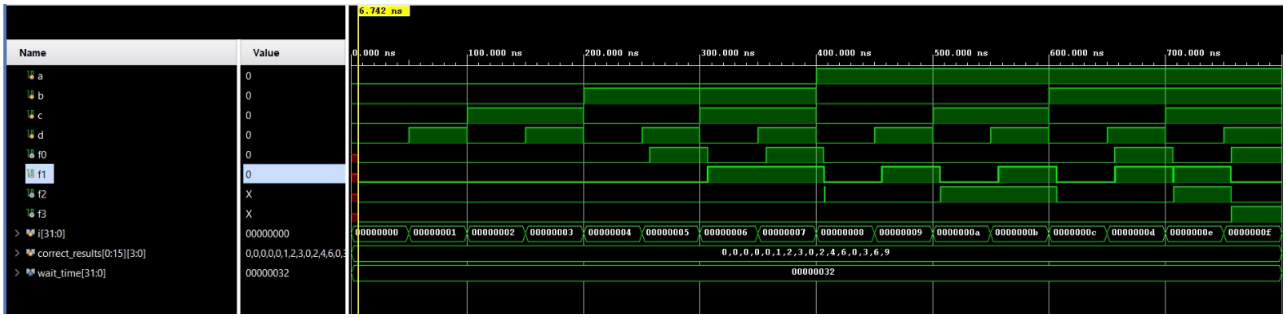


Figure 24: Post Synthesis Timing Simulation

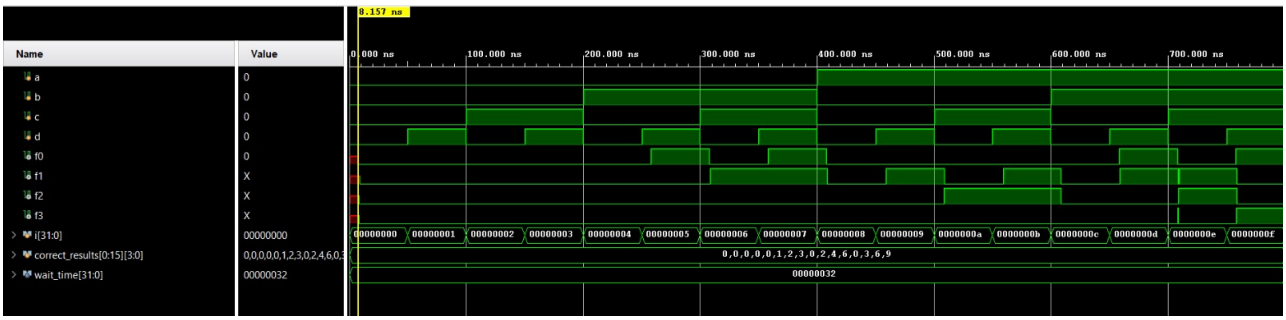


Figure 25: Post Implementation Timing Simulation

2.4 Implementation Parts and Comparison Reports After Edit Constrain

Combinational Delays

From Port	To Port	Max Delay	Max Process Corner
a	f2	8.703	SLOW
c	f2	8.635	SLOW
b	f2	8.622	SLOW
d	f2	8.488	SLOW
b	f0	8.305	SLOW
a	f1	8.296	SLOW
d	f1	8.281	SLOW
d	f0	8.180	SLOW
d	f3	8.150	SLOW
b	f1	8.139	SLOW
c	f1	8.132	SLOW
c	f3	8.099	SLOW
b	f3	7.986	SLOW
a	f3	7.980	SLOW

Figure 26: Pad-to-Pad Delay After Forced to 6ns Max Delay

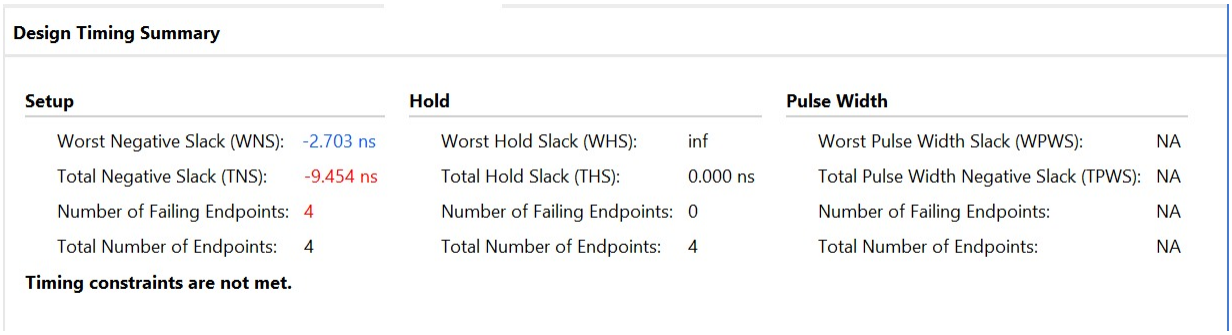


Figure 27: Negatif Slack After Forced to 6ns Max Delay

Although the 6 ns requirement was not met, max_delay was reduced whenever possible. It decreased from 9,467 to 8,703. For this reason, negative slack -2.703 was formed.

3 Realization with MUX

3.1 Hand-drawing Mux-based Circuit Schematic

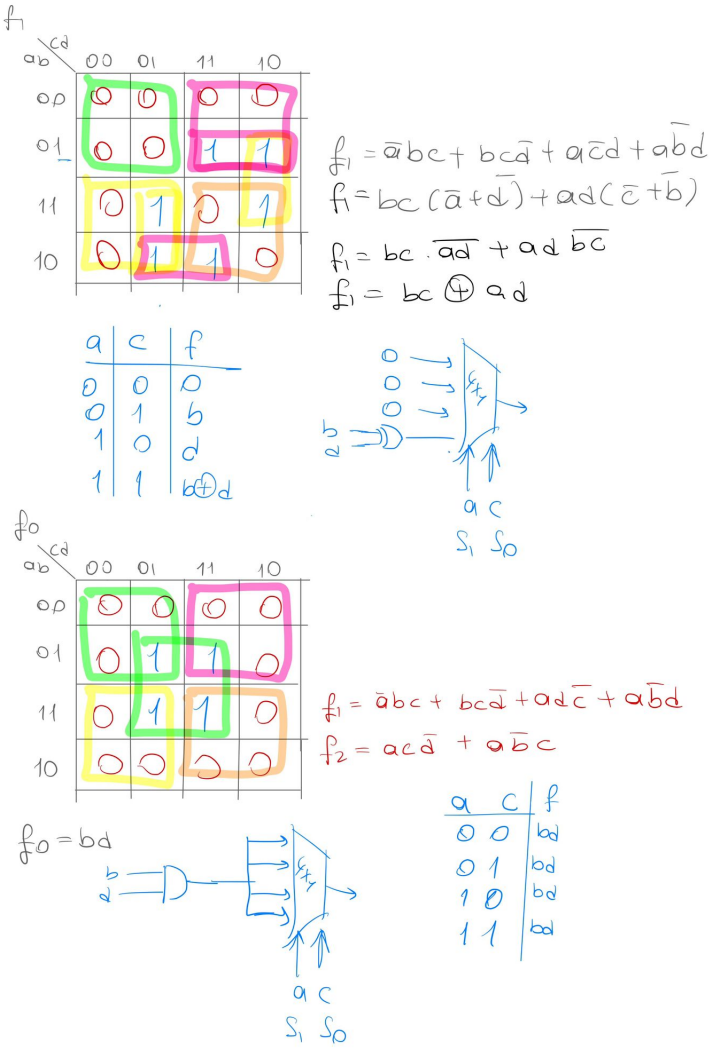


Figure 28: Hand-drawing MUX Circuit-1

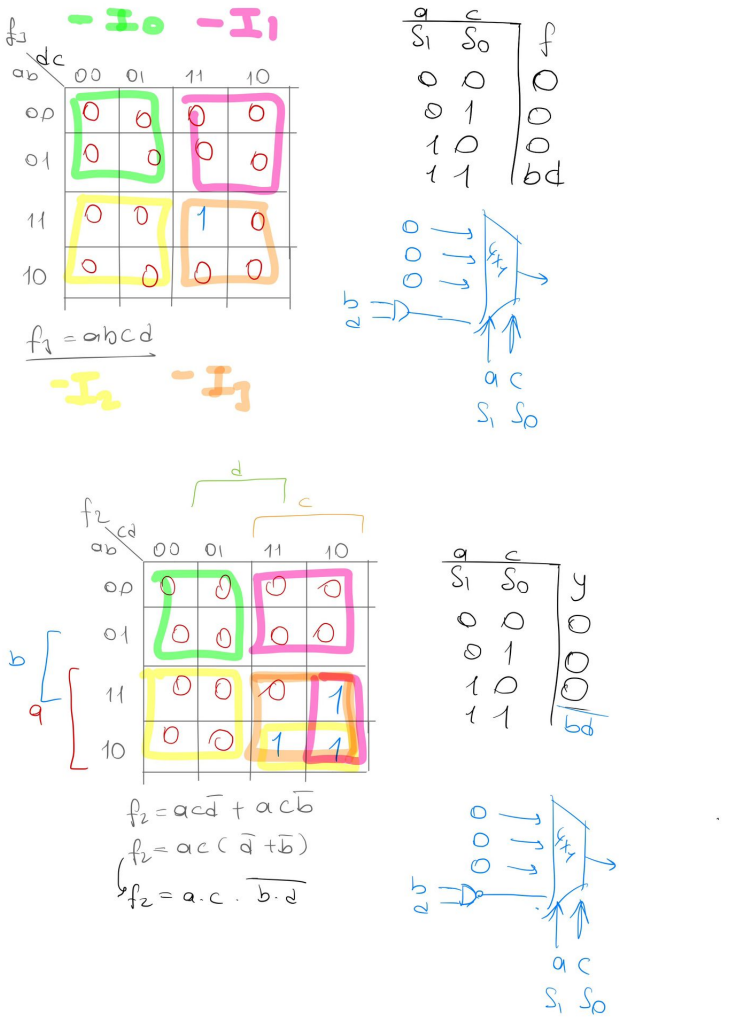


Figure 29: Hand-drawing MUX Circuit-2

3.2 Verilog Code, Testbanch Code and Behavioral Simulation

Listing 9: with_MUX Verilog Code

```
module with_MUX(
input a,b,c,d,
output f0,f1,f2,f3
    );
wire nand_bd;
wire and_bd;
wire xor_bd;
AND bd_and(.O(and_bd),.I1(b),.I2(d));
NAND bd_nand(.O(nand_bd),.I1(b),.I2(d));
EXOR bd_exor(.O(xor_bd),.I1(b),.I2(d));
```

```
11 MULTIPLEXER mux_f3(.D({and_bd, 1'b0, 1'b0, 1'b0}),.S({a,c})
       ,.O(f3));
12 MULTIPLEXER mux_f2(.D({nand_bd, 1'b0, 1'b0, 1'b0}),.S({a,c})
       ,.O(f2));
13 MULTIPLEXER mux_f1(.D({xor_bd, d, b, 1'b0}),.S({a,c}) ,.O(f1));
14 MULTIPLEXER mux_f0(.D({and_bd,and_bd,and_bd,and_bd}),.S({a,c})
       ,.O(f0));
15 endmodule
```

After writing the module, when we organize the test bench according to the mux module and run it, according to our TCL output, it is observed that our module works in accordance with the truth table as we expected.

Listing 10: Simulation Results (with_MUX)

```
# run 1000ns
{a, b, c, d} = 0000 => {f3, f2, f1, f0} = 0000 -- TRUE
{a, b, c, d} = 0001 => {f3, f2, f1, f0} = 0000 -- TRUE
{a, b, c, d} = 0010 => {f3, f2, f1, f0} = 0000 -- TRUE
{a, b, c, d} = 0011 => {f3, f2, f1, f0} = 0000 -- TRUE
{a, b, c, d} = 0100 => {f3, f2, f1, f0} = 0000 -- TRUE
{a, b, c, d} = 0101 => {f3, f2, f1, f0} = 0001 -- TRUE
{a, b, c, d} = 0110 => {f3, f2, f1, f0} = 0010 -- TRUE
{a, b, c, d} = 0111 => {f3, f2, f1, f0} = 0011 -- TRUE
{a, b, c, d} = 1000 => {f3, f2, f1, f0} = 0000 -- TRUE
{a, b, c, d} = 1001 => {f3, f2, f1, f0} = 0010 -- TRUE
{a, b, c, d} = 1010 => {f3, f2, f1, f0} = 0100 -- TRUE
{a, b, c, d} = 1011 => {f3, f2, f1, f0} = 0110 -- TRUE
{a, b, c, d} = 1100 => {f3, f2, f1, f0} = 0000 -- TRUE
{a, b, c, d} = 1101 => {f3, f2, f1, f0} = 0011 -- TRUE
{a, b, c, d} = 1110 => {f3, f2, f1, f0} = 0110 -- TRUE
{a, b, c, d} = 1111 => {f3, f2, f1, f0} = 1001 -- TRUE
$finish
```

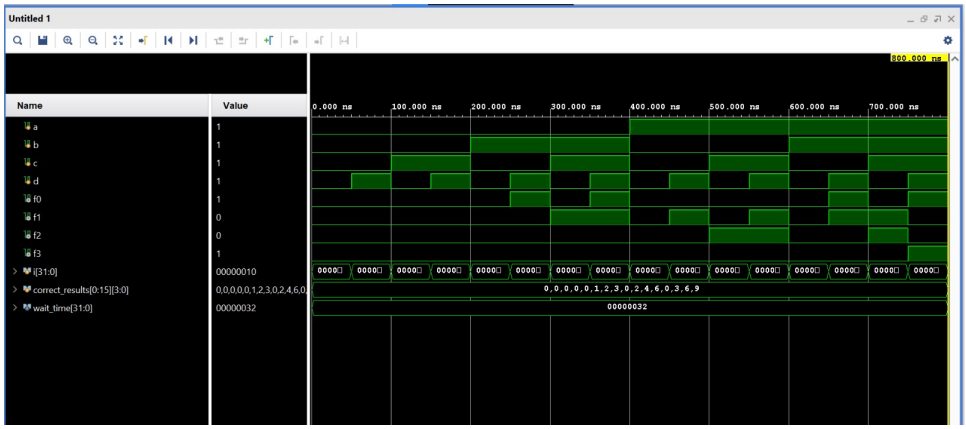


Figure 30: Mux Behavioral Simulation

3.3 Rtl and Technology Schematic

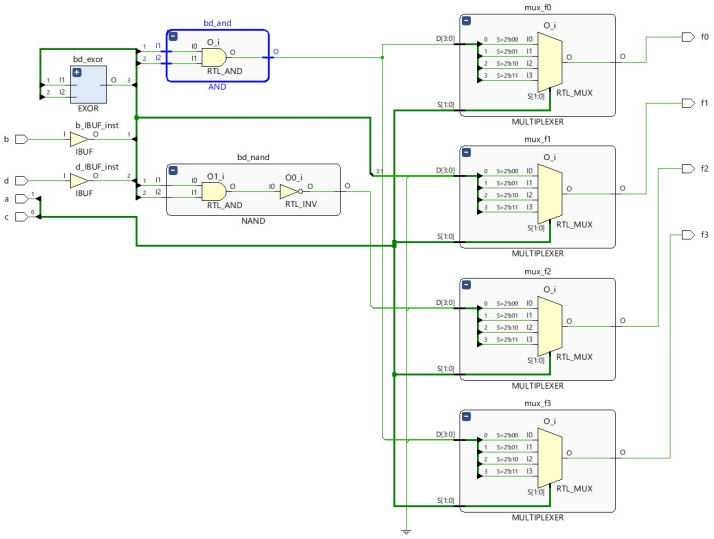


Figure 31: Mux Rtl Schematic

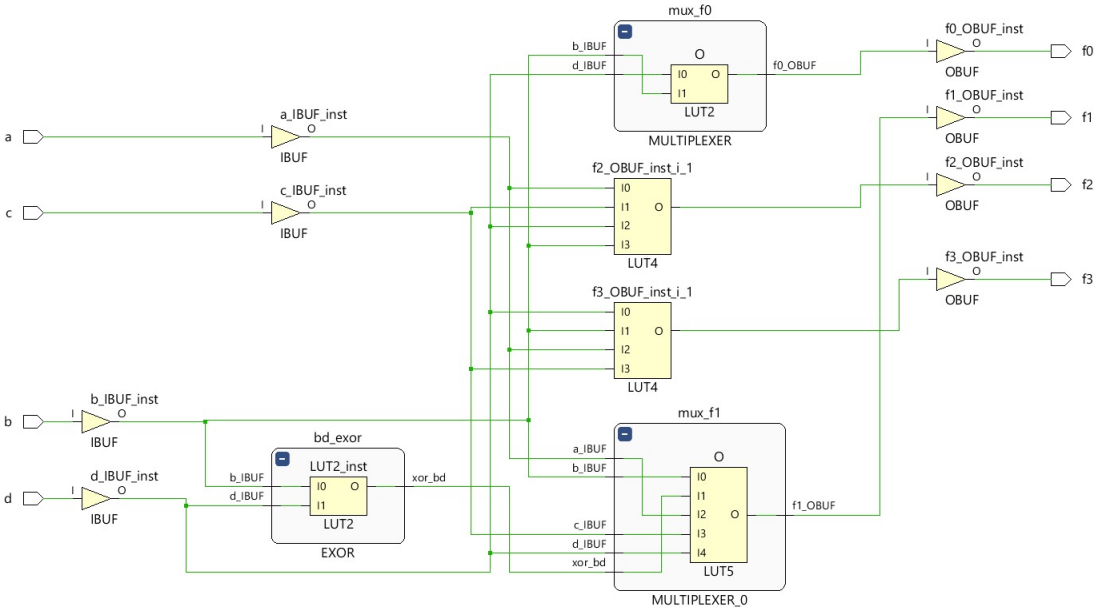


Figure 32: Mux Technology Schematic

Combinational Delays

From Port	To Port	M a 1	Max Process Corner	Min Delay	Min Process Corner
a	f2	9.754	SLOW	3.044	FAST
d	f2	9.500	SLOW	2.922	FAST
a	f3	9.262	SLOW	2.859	FAST
b	f2	9.162	SLOW	2.804	FAST
a	f1	9.047	SLOW	2.783	FAST
d	f3	9.010	SLOW	2.735	FAST
d	f1	8.933	SLOW	2.646	FAST
b	f0	8.922	SLOW	2.688	FAST
b	f1	8.901	SLOW	2.530	FAST
c	f2	8.677	SLOW	2.627	FAST
b	f3	8.672	SLOW	2.621	FAST
c	f3	8.219	SLOW	2.438	FAST
d	f0	8.092	SLOW	2.382	FAST
c	f1	8.006	SLOW	2.359	FAST

Figure 33: Pad-to-Pad delay after Implementation

Summary

Resource	Utilization	Available	Utilization %
LUT	4	63400	0.01
IO	8	210	3.81

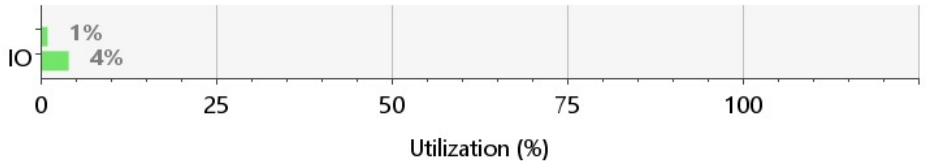


Figure 34: Utilization Report after Implementation

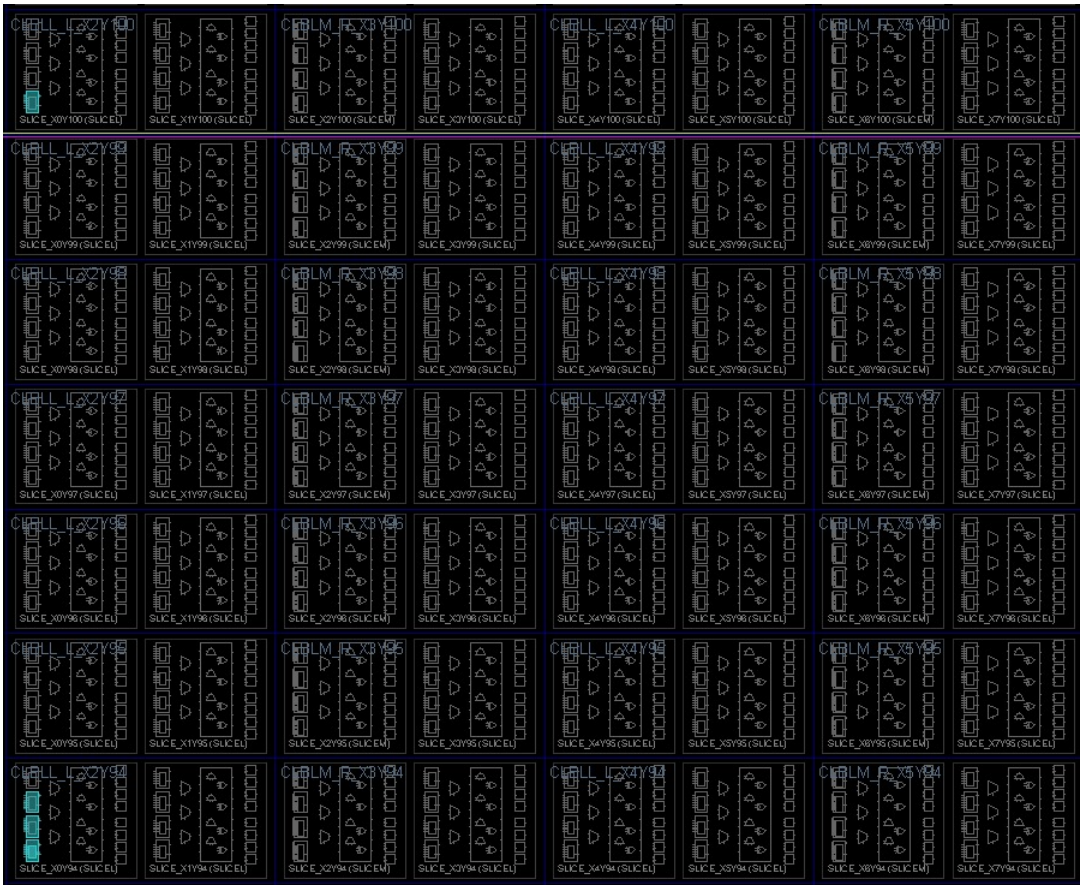


Figure 35: Place Of LUTS

3.3.1 Comparison of Simulation After Synthesis and Implementation Part

An initial delay was observed in both. 8.157 ns delay after synthesis and 8.400 ns delay after implementation part. As we expected, there was more initial delay after implementation. Because it takes into account the situations related to our FPGA in the implementation.

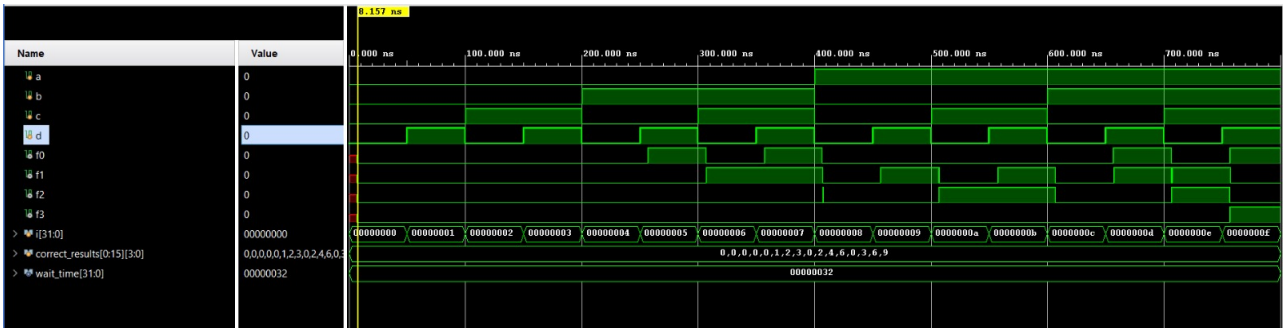


Figure 36: Post Synthesis Timing Simulation

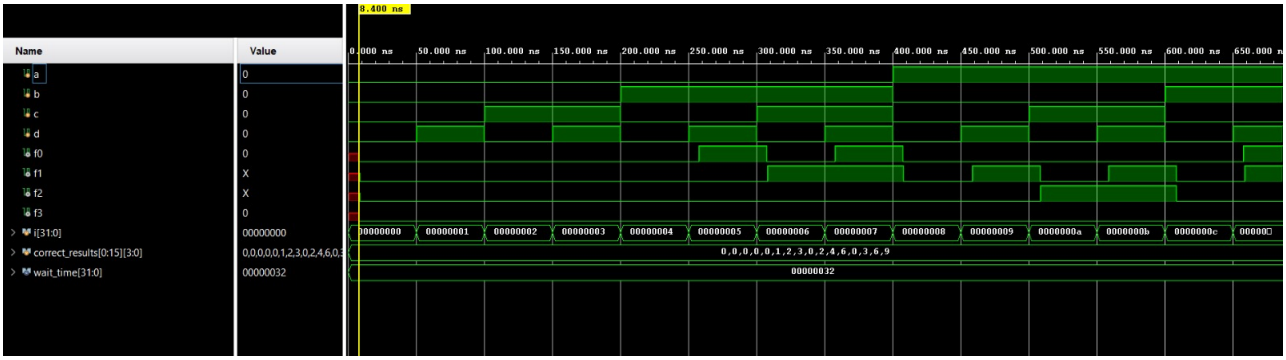


Figure 37: Post Implementation Timing Simulation

3.4 Compare of Luts Places on FPGA

We observe a decrease in lut due to the design difference. We observe 5 lutes in the primitive gates, 3 in the decoder and 4 in the mux, respectively.

3.5 Implementation Parts and Reports After Edit Constrain

Combinational Delays

From Port	To Port	Max Delay	Max Process Corner
a	f2	8.814	SLOW
d	f2	8.650	SLOW
c	f2	8.622	SLOW
b	f2	8.563	SLOW
b	f1	8.520	SLOW
a	f1	8.510	SLOW
d	f1	8.393	SLOW
b	f0	8.305	SLOW
d	f0	8.183	SLOW
d	f3	8.155	SLOW
c	f1	8.129	SLOW
c	f3	8.101	SLOW
b	f3	7.986	SLOW
a	f3	7.978	SLOW

Figure 38: Pad-to-pad Max Delay After Forcing 6ns Max Delay

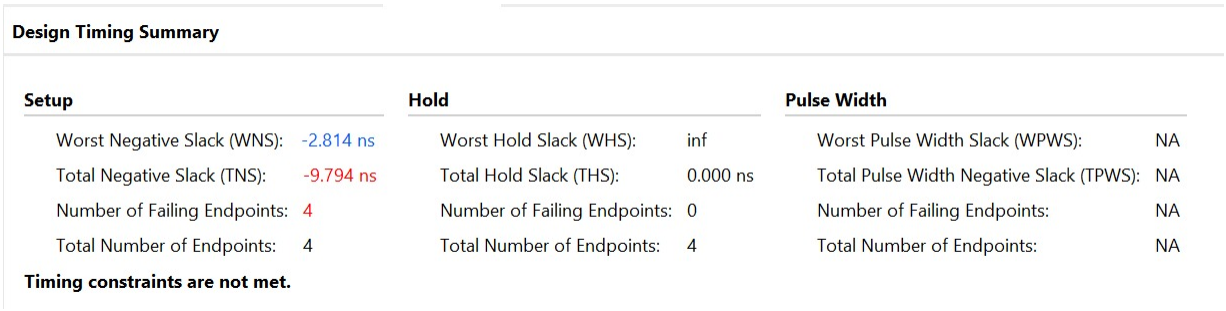


Figure 39: Negatif Slack After Forcing 6ns Max Delay

After adding max_delay to the constrain file, unfortunately, it was observed that the result decreased from 9,754 to 8,814, although not as much as we wanted (6 ns). Here, the distance to our desired 6 ns was revealed as 2.814 negative slacks.

4 Miscelaneous Q/W

4.1 1. Differences between Simulation Types

Behavioral Simulation: This is an abstract simulation that models the system's behavior based on high-level descriptions. It focuses on functionality and is not concerned with the low-level details of the hardware implementation.
Post-Synthesis Functional Simulation: This simulation occurs after the synthesis process, which translates a high-level hardware description into a netlist of gates. The simulation takes into account the synthesized netlist but still focuses on functionality.
Post-Implementation Functional Simulation: This simulation is conducted after the design has been implemented on the target FPGA or ASIC device. It considers the actual physical components used in the implementation and ensures that the design functions correctly.
Post-Implementation Timing Simulation: This simulation includes timing considerations and models the delays introduced by the physical components, such as gates and interconnections. It provides insights into the timing performance of the design.

4.2 2. Consider that the truth table

This truth table verifies a multiplication situation. If c and d are considered binary, they are observed as multipliers, and if a and b are considered binary, they are observed as multipliers. Therefore, in the outputs, it is observed as 0 (ab ==00) in the first 0.3 index, as a direct cd value between 4-7, twice the cd values between 8-11, and 3 times the cd values between 12-15.

4.3 Evaluation Traits:

Comparison of models in terms of time	
SSI realization	9.753
SSI realization with time constraint	9.008
SSI realization with loc constraint	12.048
SSI realization with time and loc constraint	10.938
Decoder realization	9.497
Decoder realization with time constraint	8.803
MUX realization	9.754
MUX realization with time constraint	8.814

Table 2: Comparison of all realizations

When we look at the table, we see different delays in different designs. Regarding these delays: When we do not intervene in SSI, 9.753 ns is updated to 9.008 when we give a time constraint, while negative slack becomes 0.008. It is observed that when we change the location ourselves, the delay increases. In other words, it has been observed that when we do not impose any restrictions on the vivado, its placement is more optimized than when we do. When we put a time constraint (9ns) with Location, the delay decreased to 10,938 and 1,938 ns negative slacks occurred.

In Decoder, with its own optimization, the delay appears to be 9,497 ns. The reason why this is lower than SSI and MSI is that this structure is more behavioral. This structure, in which we connected the SOPs directly with or structures, caused less delay. When we set the timing constraint as 6ns, a negative slack of 2,803 ns occurred, which did not fall to the desired rate.

In MUX, a delay of 9.754 ns was observed when there was no intervention. The higher delay from the logic gates and decoder may be due to the fact that we used 4 muxes, and after the timing constraint, it dropped to 8,814 ns, giving a negative slack of 2,814 ns.

Design Difficulty:

Primitive Gates: Simplest, but manual design can be error-prone. Decoder: Moderate difficulty, involves selecting outputs based on input combinations.

Mux: Moderate difficulty, requires selecting and routing inputs.

Coding Difficulty:

Primitive Gates: Can be complex due to manual design.

Decoder: Relatively straightforward, especially with higher-level hardware description languages.

Mux: Similar to the decoder, coding is relatively straightforward.

Since there are so many doors in SSI, coding has become more complex for me.

LUT Usage:

Primitive Gates: Directly correlates with the complexity of the logic; can be high.

Decoder: Depends on the number of outputs; moderate to high.

Mux: Depends on the number of inputs and select lines; can vary.

Respectively, 4 luts were used in 3 muxes in 5 decoders in SSI. As can be seen, although the functions and functions are the same, different lut numbers may occur in different designs.

Path Delays:
Without any forcing on constrain file, i took this delays: SSI : 9.753
Decoder: 9.497
MUX: 9.754
This result show that, when number of luts are increased, path delays increased too respectively.

5 Appendix: Testbench

Listing 11: Testhbench Code

```
`timescale 1ns / 1ps

module experiment3_tb();

  // ----- Inputs & Outputs
  reg a,b,c,d;
  wire f0,f1,f2,f3;

  // ----- Testbench Parameters ----- //
  parameter wait_time = 50;
  integer i;

  reg [3:0]correct_results[0:15];
  initial begin
    correct_results[0]  = 0; correct_results[1]  = 0;
       correct_results[2]  = 0 ;correct_results[3] = 0;
    correct_results[4]  = 0; correct_results[5]  = 1;
       correct_results[6]  = 2 ;correct_results[7] = 3;
    correct_results[8]  = 0; correct_results[9]  = 2;
       correct_results[10] = 4 ;correct_results[11] = 6;
    correct_results[12] = 0; correct_results[13] = 3;
       correct_results[14] = 6 ;correct_results[15] = 9;
  end
  // --------------------------------- //

  // ---------- UUT Instantiation -------------- //
  //with_SSI
  with_decoder
  //with_MUX
  UUT (.a(a),
    .b(b),
    .c(c),
    .d(d),
    .f0(f0),
```

```
    .f1(f1),
    .f2(f2),
    .f3(f3));

  // ------- Test Procedure ------- //
  initial
  begin

    for(i=0;i<16;i=i+1)
    begin

      {a,b,c,d} = i;
      #(wait_time);
      $write("{a,b,c,d}=%d%d%d%d => {f3,f2,f1,f0} = %d%d%d%d
          -- ",a,b,c,d,f3,f2,f1,f0);
      if({f3,f2,f1,f0} == correct_results[i])
        $display("TRUE");
      else
        $display("FALSE");

    end

    $finish();

  end
  // ------------------------------- //

endmodule
```

6 Constrain File

Listing 12: Constrain File

```
## This file is a general .xdc for the Nexys A7-100T
## To use it in a project:
## - uncomment the lines corresponding to used pins
## - rename the used ports (in each line, after get_ports) according to the top level


##Switches
set_property -dict {PACKAGE_PIN J15 IOSTANDARD LVCMOS33} [get_ports d]
set_property -dict {PACKAGE_PIN L16 IOSTANDARD LVCMOS33} [get_ports c]
set_property -dict {PACKAGE_PIN M13 IOSTANDARD LVCMOS33} [get_ports b]
set_property -dict {PACKAGE_PIN R15 IOSTANDARD LVCMOS33} [get_ports a]
```

```
## LEDs
set_property -dict {PACKAGE_PIN H17 IOSTANDARD LVCMOS33} [get_ports f0]
set_property -dict {PACKAGE_PIN K15 IOSTANDARD LVCMOS33} [get_ports f1]
set_property -dict {PACKAGE_PIN J13 IOSTANDARD LVCMOS33} [get_ports f2]
set_property -dict {PACKAGE_PIN N14 IOSTANDARD LVCMOS33} [get_ports f3]

###Set Timing Constrain
#set_max_delay 9 -from [all_inputs] -to [all_outputs]

###Set Location Constrain
#set_property LOC SLICE_X12Y67 [get_cells f0_OBUF_inst_i_1]
#set_property LOC SLICE_X12Y66 [get_cells f2_OBUF_inst_i_1]
#set_property LOC SLICE_X13Y67 [get_cells f1_exor/LUT2_inst]
#set_property LOC SLICE_X14Y64 [get_cells f1_exor/LUT2_inst_i_1]
```