DIGITAL SYSTEM DESIGN APPLICATION

EHB436E CRN: 11280

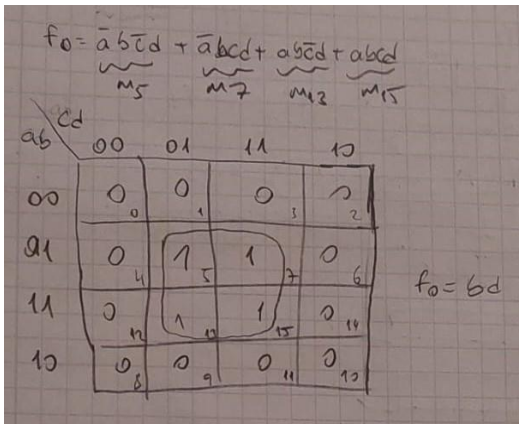
Salih Ömer Ongün
040220780

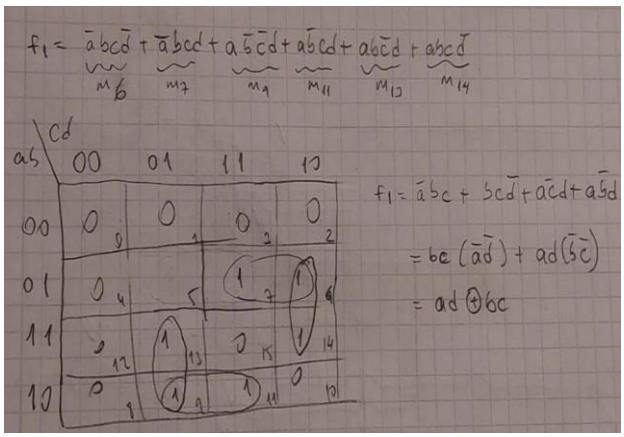
Experiment 3

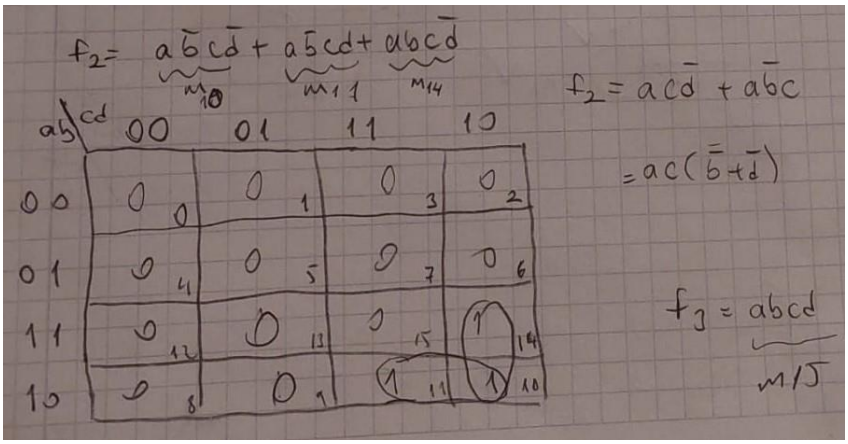
Salih Ömer Ongün
040220780

REALIZATION WITH SSI Library

Karnaugh Map of Outputs

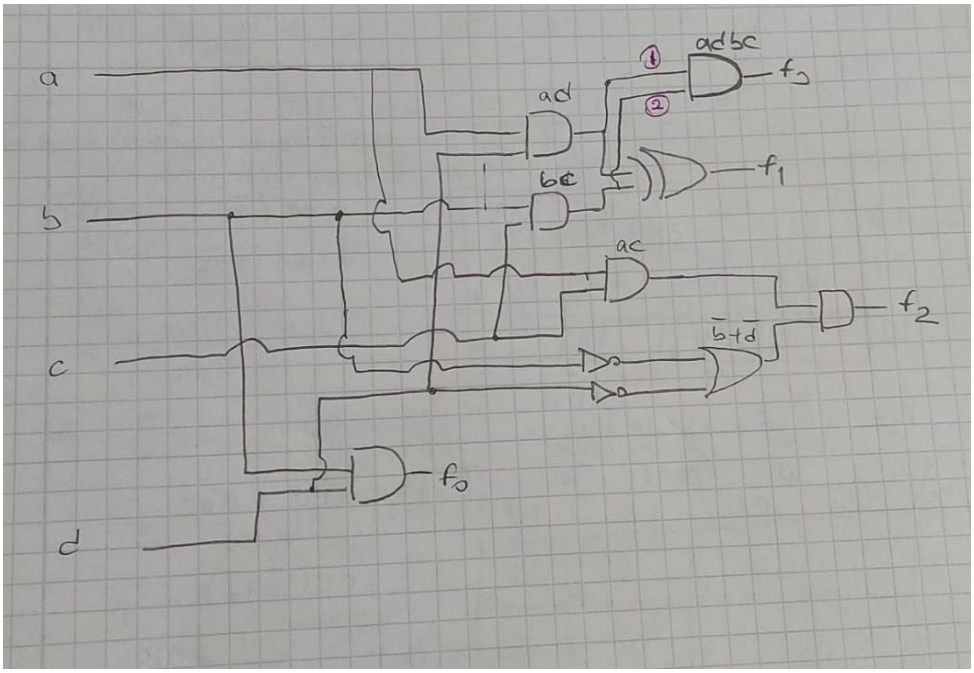






Salih Ömer Ongün
040220780

Gate Level Circuit Schematic



TCL Console Output

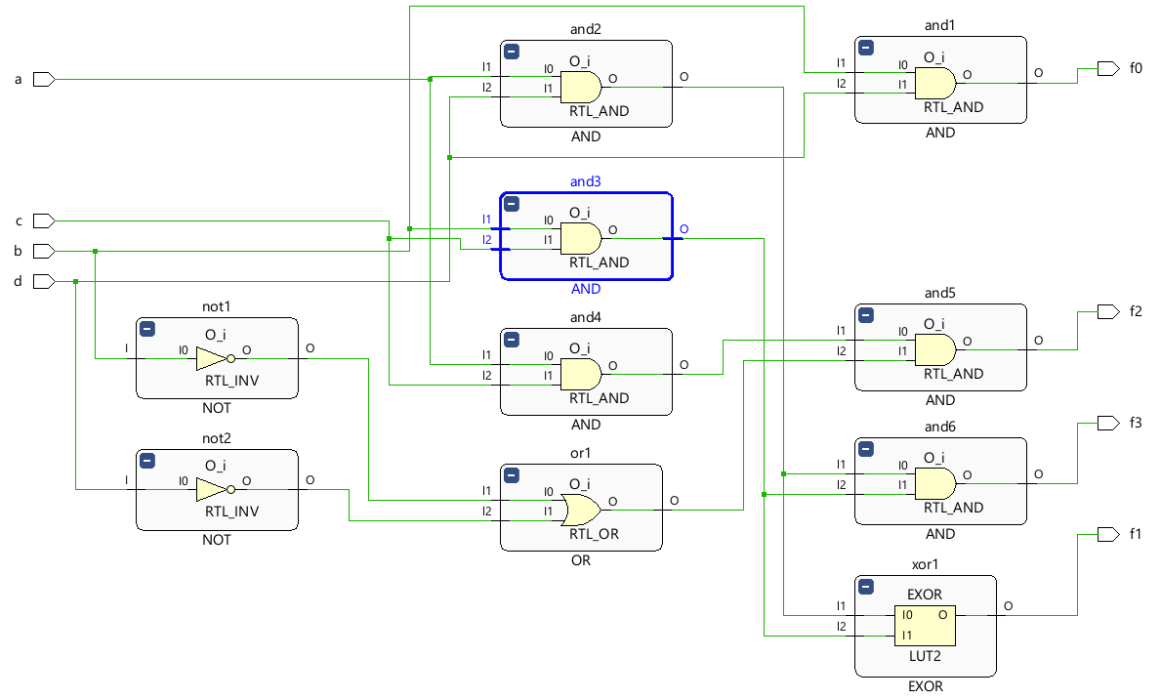
```
# run 1000ns
{a,b,c,d}=0000 => {f3,f2,f1,f0} = 0000 -- TRUE
{a,b,c,d}=0001 => {f3,f2,f1,f0} = 0000 -- TRUE
{a,b,c,d}=0010 => {f3,f2,f1,f0} = 0000 -- TRUE
{a,b,c,d}=0011 => {f3,f2,f1,f0} = 0000 -- TRUE
{a,b,c,d}=0100 => {f3,f2,f1,f0} = 0000 -- TRUE
{a,b,c,d}=0101 => {f3,f2,f1,f0} = 0001 -- TRUE
{a,b,c,d}=0110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

As you can see from the TCL console output, the design made with the SSI Library works correctly.

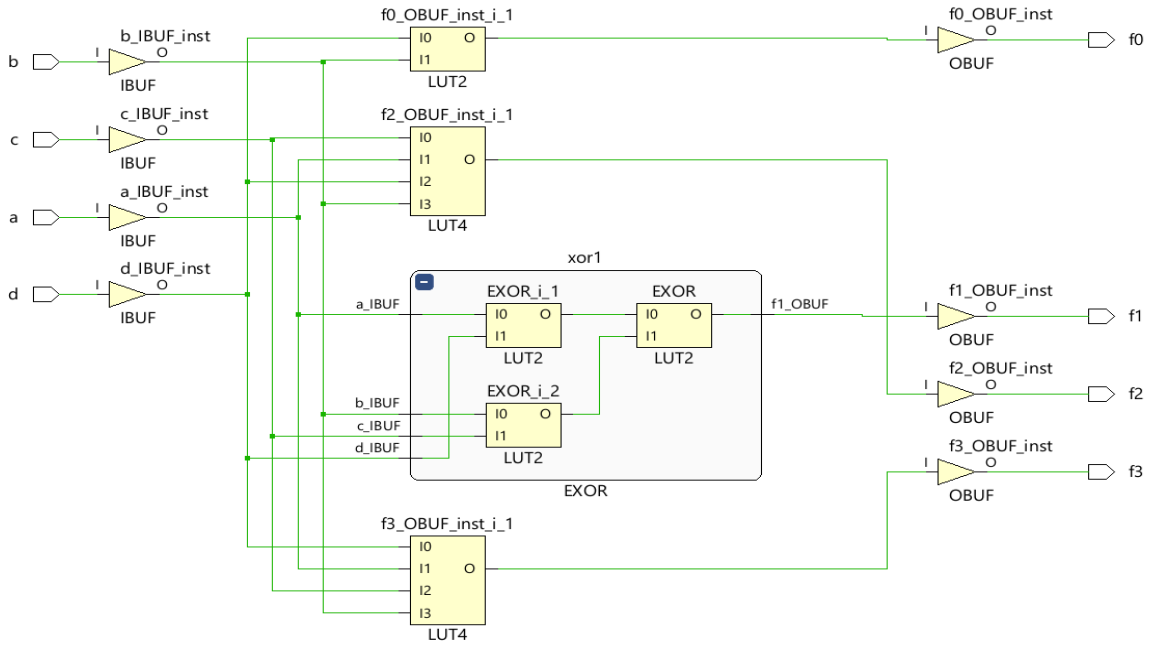
If there are 2-bit 2 numbers at the input of the function, 2-bit parallel operation can be performed in the function. These inputs can be used for two-bit parallel multipliers.

NO TIMING AND LOC CONSTRAINTS

RTL Schematic



Technology Schematic



Salih Ömer Ongün
040220780

Pad to Pad Delays

From Port	To Port	Max Delay	Max Process Corner	Min Delay	Min Process Corner
a	f1	9.523	SLOW	2.950	FAST
a	f2	9.725	SLOW	3.016	FAST
a	f3	9.229	SLOW	2.827	FAST
b	f0	8.972	SLOW	2.709	FAST
b	f1	9.228	SLOW	2.757	FAST
b	f2	8.877	SLOW	2.660	FAST
b	f3	8.382	SLOW	2.469	FAST
c	f1	9.378	SLOW	2.827	FAST
c	f2	9.102	SLOW	2.744	FAST
c	f3	8.607	SLOW	2.556	FAST
d	f0	8.109	SLOW	2.399	FAST
d	f1	8.954	SLOW	2.721	FAST
d	f2	9.715	SLOW	2.986	FAST
d	f3	9.218	SLOW	2.797	FAST

Resource Usage

Utilization Post-Synthesis | **Post-Implementation**

Graph | **Table**

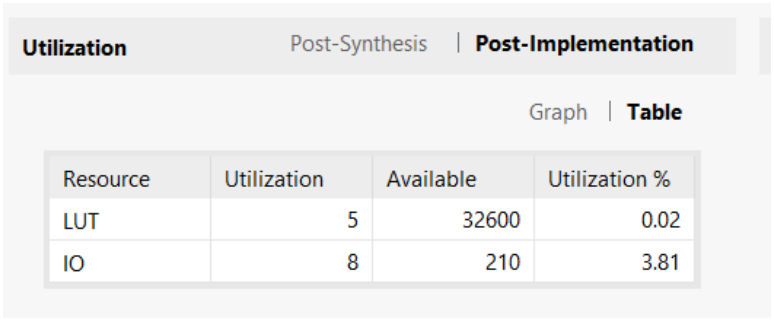
Resource	Utilization	Available	Utilization %
LUT	5	32600	0.02
IO	8	210	3.81

TIMING CONSTRAINED DESIGN WITHOUT LOC CONSTRAINS

From Port	To Port	Max Delay	Max Process Corner
a	f1	9.048	SLOW
a	f2	8.914	SLOW
a	f3	8.215	SLOW
b	f0	8.822	SLOW
b	f1	8.771	SLOW
b	f2	8.625	SLOW
b	f3	8.153	SLOW
c	f1	8.791	SLOW
c	f2	8.517	SLOW
c	f3	7.654	SLOW
d	f0	8.701	SLOW
d	f1	8.658	SLOW
d	f2	8.497	SLOW
d	f3	8.140	SLOW

I set constraints to adjust maximum delay to 9 ns. However, the delays from port "a" to port "f1" exceed 9 ns by 0.048 ns. Although the delays cannot achieve the 9ns target, there is a significant reduction in delays.

Resource Usage

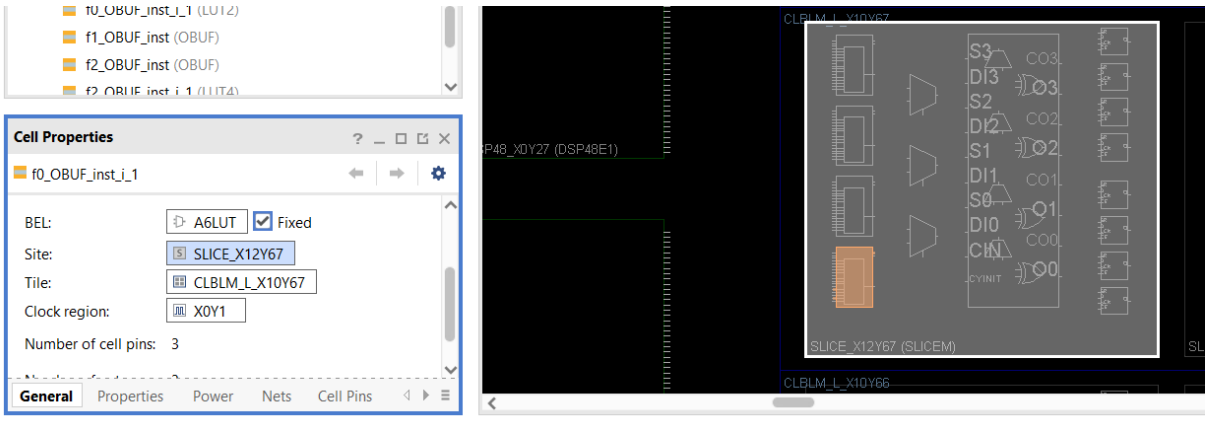


Resource	Utilization	Available	Utilization %
LUT	5	32600	0.02
IO	8	210	3.81

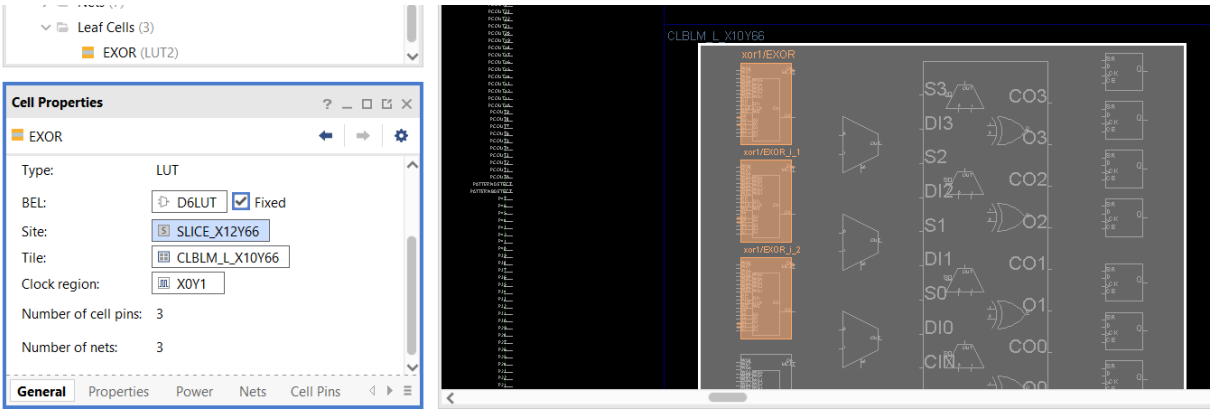
There are not any differences in resource usage.

LOC CONSTRAINED DESIGN WITHOUT TIMING CONSTRAINS

Placement

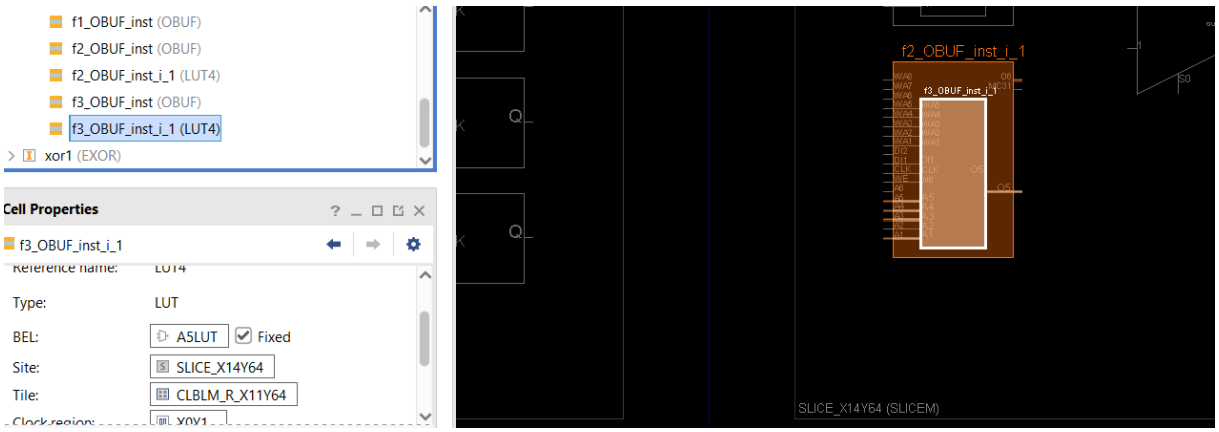


f0 placement



f1 Placement

f1 output is was saved as xor1 not f1_OBUF_inst_i_1.



I placed f2 and f3 at X13Y67 and X14Y64 respectively. However, f2 and f3 are placed in the same "Slice". They cannot be separated.

Pad to Pad Path Delays

From Port	To Port	Max Delay	Max Process Corner	Min Delay	Min Process Corner
a	f1	11.656	SLOW	3.862	FAST
a	f2	10.182	SLOW	3.310	FAST
a	f3	11.289	SLOW	3.770	FAST
b	f0	9.636	SLOW	3.094	FAST
b	f1	10.484	SLOW	3.458	FAST
b	f2	9.952	SLOW	3.219	FAST
b	f3	11.061	SLOW	3.682	FAST
c	f1	10.275	SLOW	3.386	FAST
c	f2	9.528	SLOW	3.048	FAST
c	f3	10.603	SLOW	3.513	FAST
d	f0	9.992	SLOW	3.195	FAST
d	f1	10.635	SLOW	3.429	FAST
d	f2	9.530	SLOW	3.020	FAST
d	f3	10.638	SLOW	3.481	FAST

Delays are more than no constraints and timing constraints.

Salih Ömer Ongün
040220780

TIMING AND LOC CONSTRAINED DESIGN

Pad to Pad Path Delays

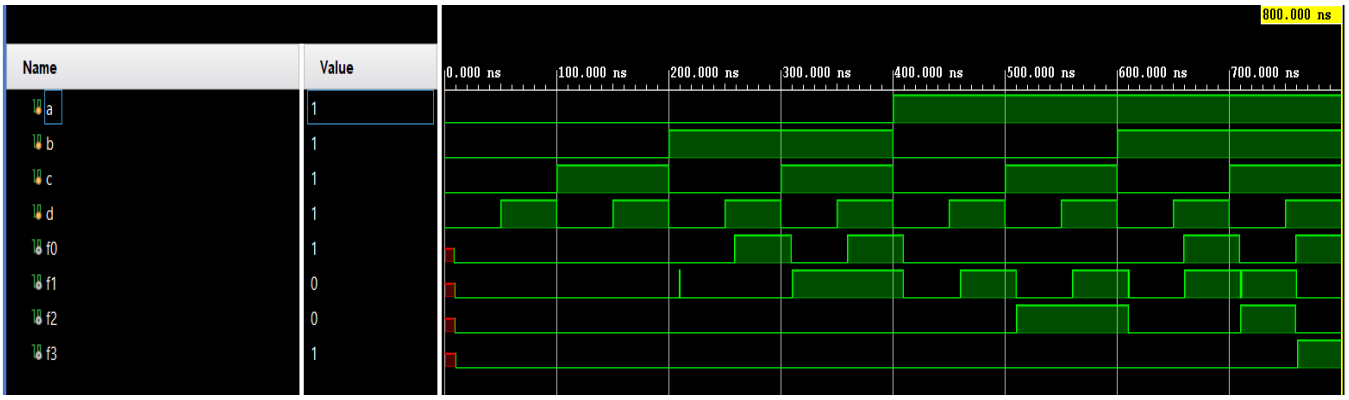
From Port	To Port	Max Delay	Max Process Corner
a	f1	10.387	SLOW
a	f2	9.625	SLOW
a	f3	10.577	SLOW
b	f0	9.188	SLOW
b	f1	9.742	SLOW
b	f2	9.666	SLOW
b	f3	10.652	SLOW
c	f1	10.030	SLOW
c	f2	9.564	SLOW
c	f3	10.550	SLOW
d	f0	9.065	SLOW
d	f1	10.116	SLOW
d	f2	9.242	SLOW
d	f3	10.226	SLOW

I set constraints to adjust maximum delay to 9 ns. However, some delays exceed 9 ns. Although the delays cannot achieve the 9ns target, there is a significant reduction in delays.

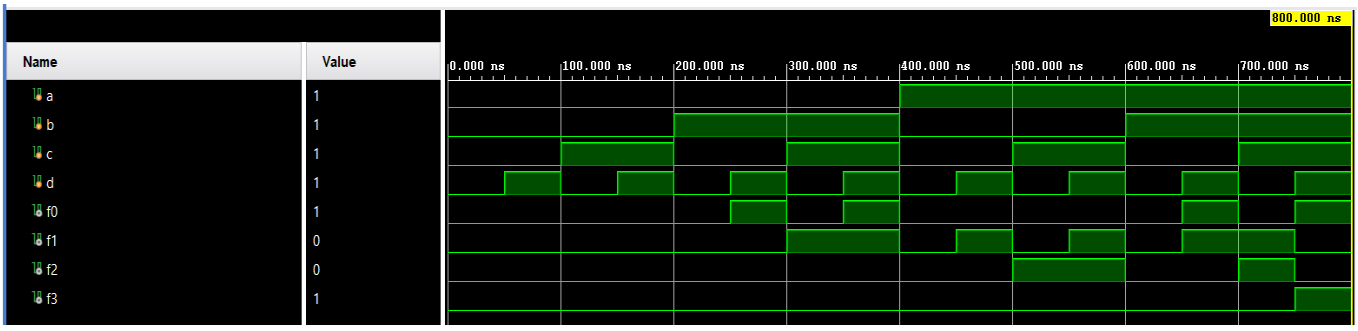
The design where timing is constrained has the least delay design. However, it has not reached the 9ns goal. Placement increases pad to pad delays. I set constraints to adjust maximum delay to 9 ns, but design has not reached this goal. Vivado adjusts placement according to optimum design for time and power consumption. Therefore, different placements increase delays.

Simulation Wave

Post-implementation timing simulation waveform



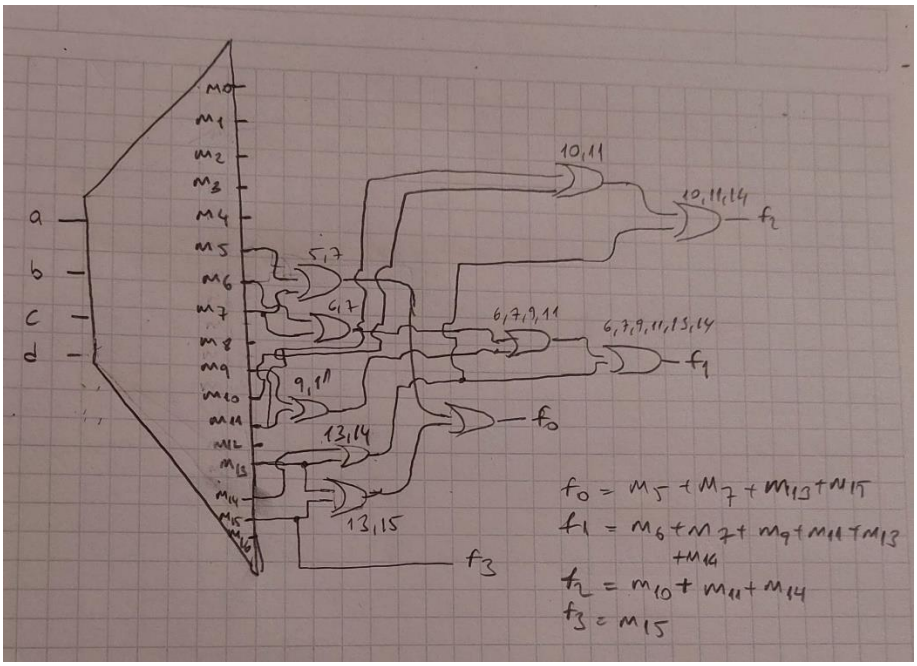
Behavioral Simulation Waveform



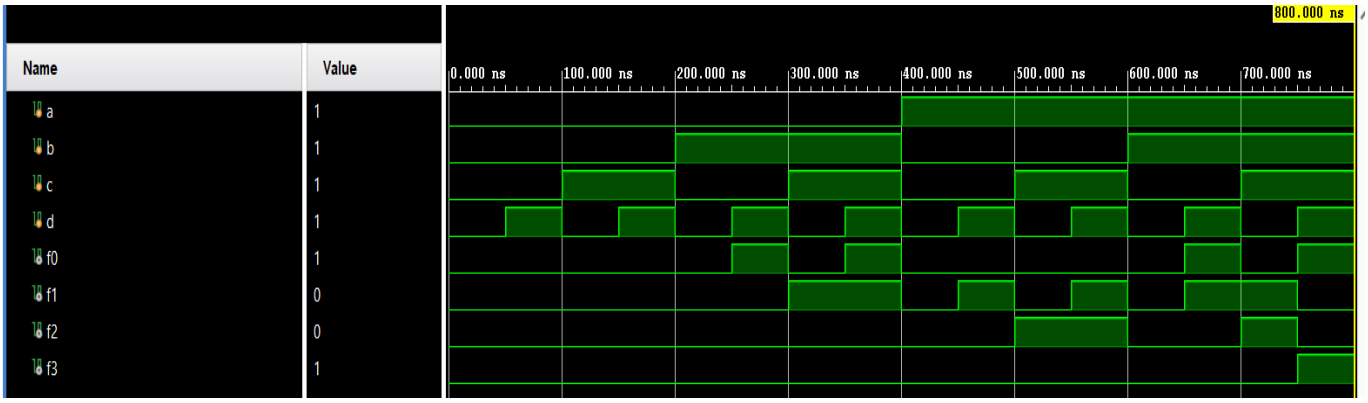
Behavioral simulation waveforms have clear waveforms. However, post-implementation timing simulation waveform shifted waveforms. Redline delays may have caused this shift.

REALIZATION WITH DECODER

4X16 Decoder

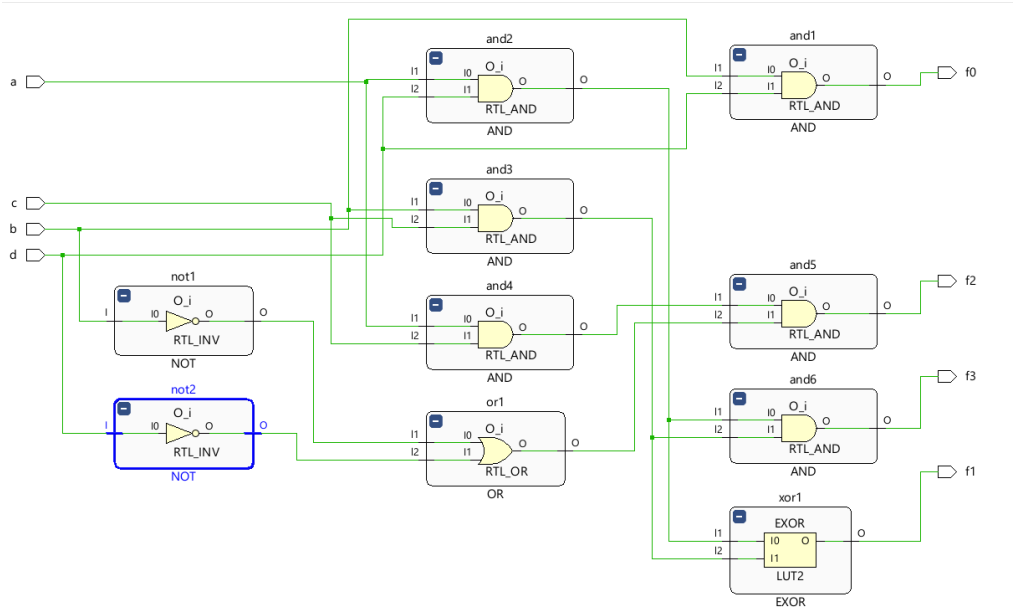


Behavioral Simulation Waveform

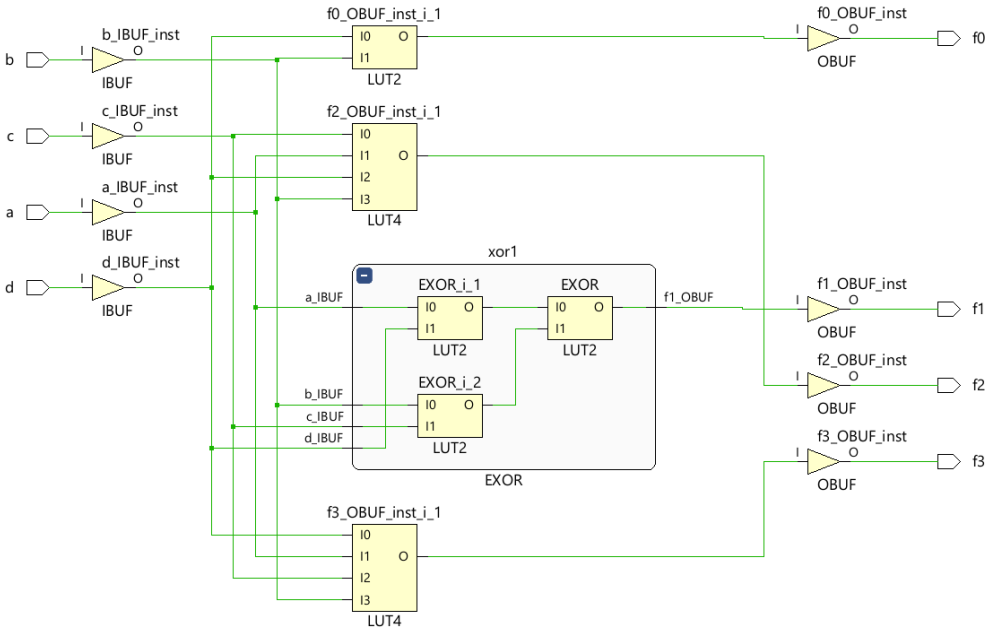


According to a,b,c,d inputs from m0 to m15 outputs have logic 1 value. Our f0,f1,f2,f3 outputs have SOP forms. In this form outputs have or structure. For a=1,b=1,c=0, d=1 values m13 have logic 1 value. F0 and f1 have m13 outputs. Therefore, f0 and f1 have logic 1 value.

RTL Schematic



Technology Schematic



Pad to Pad Delays

From Port	To Port	Max Delay	Max Process Corner	Min Delay	Min Process Corner
a	f1	9.523	SLOW	2.950	FAST
a	f2	9.725	SLOW	3.016	FAST
a	f3	9.229	SLOW	2.827	FAST
b	f0	8.972	SLOW	2.709	FAST
b	f1	9.228	SLOW	2.757	FAST
b	f2	8.877	SLOW	2.660	FAST
b	f3	8.382	SLOW	2.469	FAST
c	f1	9.378	SLOW	2.827	FAST
c	f2	9.102	SLOW	2.744	FAST
c	f3	8.607	SLOW	2.556	FAST
d	f0	8.109	SLOW	2.399	FAST
d	f1	8.954	SLOW	2.721	FAST
d	f2	9.715	SLOW	2.986	FAST
d	f3	9.218	SLOW	2.797	FAST

Resource Usage

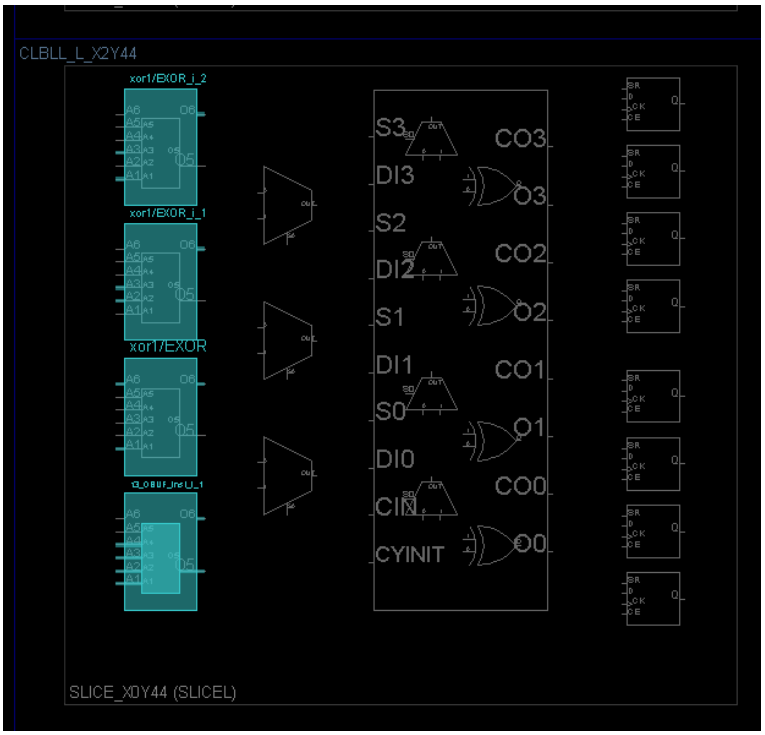
Utilization Post-Synthesis | **Post-Implementation**

Graph | **Table**

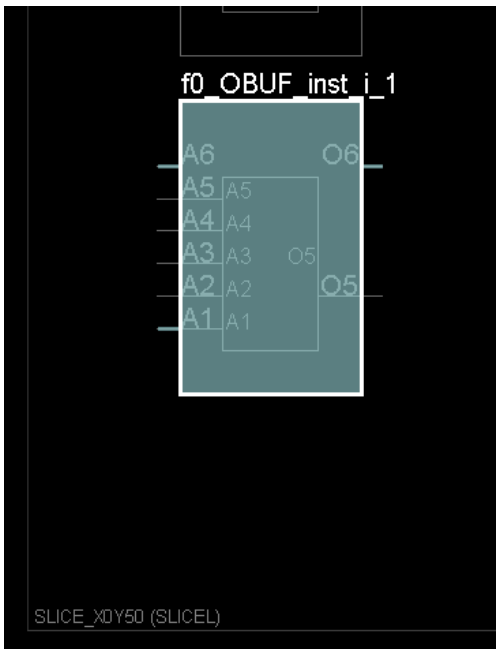
Resource	Utilization	Available	Utilization %
LUT	5	32600	0.02
IO	8	210	3.81

Resource usage is the same as the SSI library design.

Placement



f2, f 3, xor1(f1) placement



f0 placement

The cell used is the same as the SSI library design.

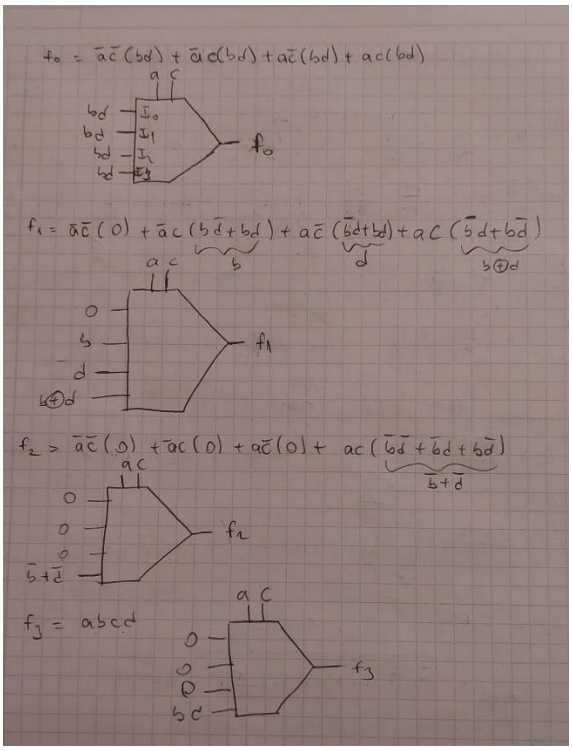
DESIGN WITH TIMING CONSTRAINTS

From Port	To Port	Max Delay	Max Process Corner
a	f1	8.985	SLOW
a	f2	8.905	SLOW
a	f3	7.945	SLOW
b	f0	8.179	SLOW
b	f1	8.450	SLOW
b	f2	8.690	SLOW
b	f3	7.923	SLOW
c	f1	8.614	SLOW
c	f2	8.517	SLOW
c	f3	7.908	SLOW
d	f0	8.032	SLOW
d	f1	8.643	SLOW
d	f2	8.480	SLOW
d	f3	8.447	SLOW

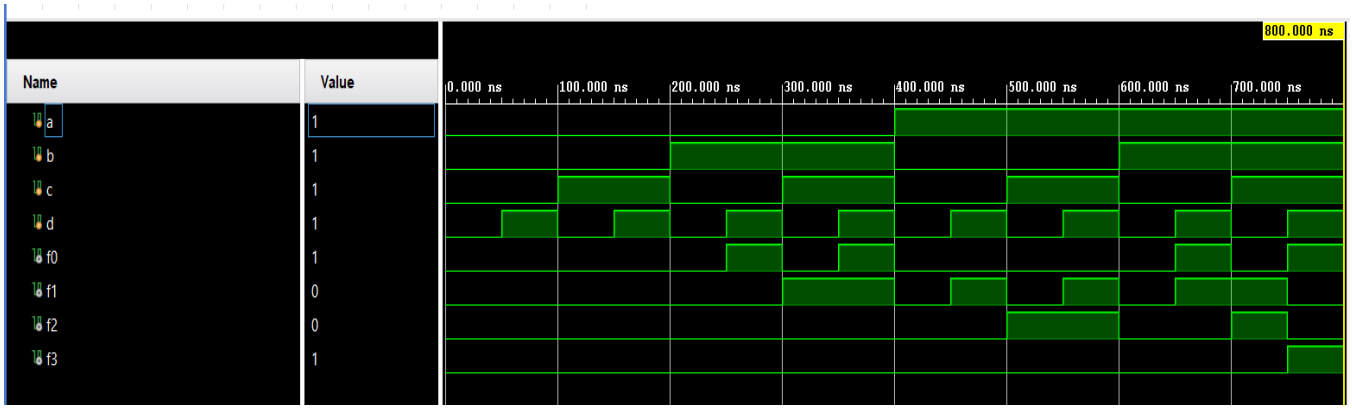
I set constraints to adjust maximum delay to 6 ns. However, all paths exceed 6 ns. Although the delays cannot achieve the 6ns target, there is a significant reduction in delays.

REALIZATION WITH MUX

Handwriting

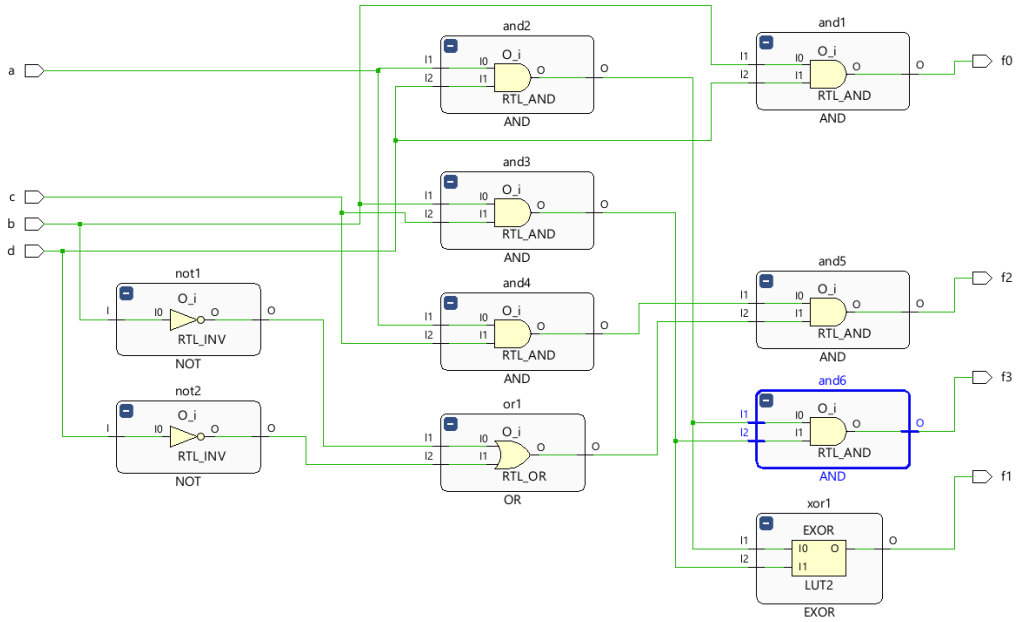


Behavioral Simulation Waveform

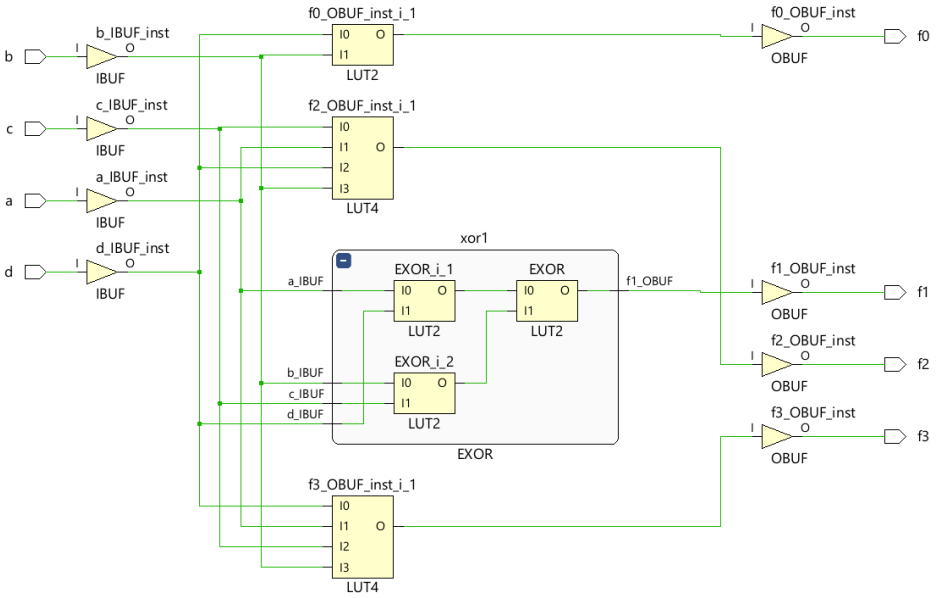


For f1 output, a=0 and c=0, f1= 0. a=0 and c=1, f1= b. c=1 and a=0, f1= b. c=1 and a=1, f1= (b xor d). For all outputs waveform provide exact true value.

RTL Schematic



Technology Schematic



Resource Usage

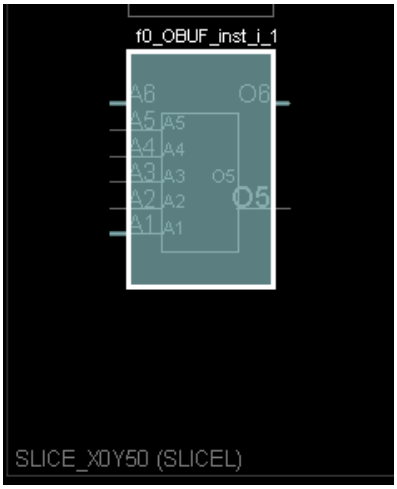
Name	Slice LUTs (32600)	Slice (8150)	LUT as Logic (32600)	Bonded IOB (210)
N with_SSI	5	2	5	8

Resource usage is the same as the SSI library design and Decoder design.

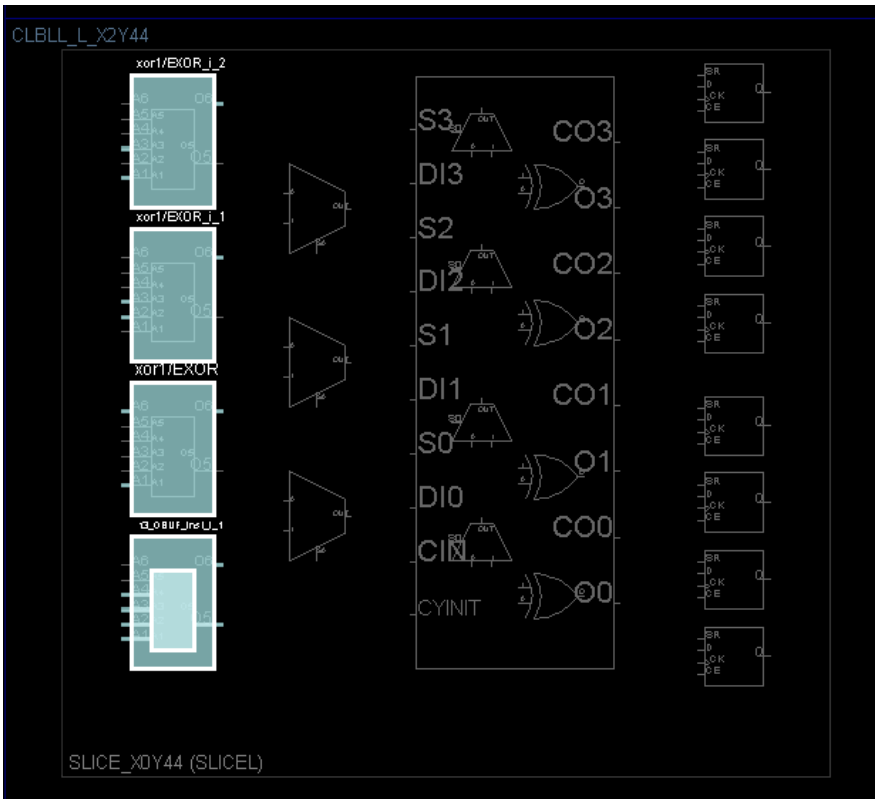
Pad to Pad Delays

From Port	To Port	Max Delay	Max Process Corner	Min Delay	Min Process Corner
a	f1	9.523	SLOW	2.950	FAST
a	f2	9.725	SLOW	3.016	FAST
a	f3	9.229	SLOW	2.827	FAST
b	f0	8.972	SLOW	2.709	FAST
b	f1	9.228	SLOW	2.757	FAST
b	f2	8.877	SLOW	2.660	FAST
b	f3	8.382	SLOW	2.469	FAST
c	f1	9.378	SLOW	2.827	FAST
c	f2	9.102	SLOW	2.744	FAST
c	f3	8.607	SLOW	2.556	FAST
d	f0	8.109	SLOW	2.399	FAST
d	f1	8.954	SLOW	2.721	FAST
d	f2	9.715	SLOW	2.986	FAST
d	f3	9.218	SLOW	2.797	FAST

Placement



f0 placement



f2, f 3, xor1(f1) placement

The cell used is the same as the SSI library design.

DESIGN WITH TIMING CONSTRAINTS

From Port	To Port	Max Delay	Max Process Corner
a	f1	8.985	SLOW
a	f2	8.905	SLOW
a	f3	7.945	SLOW
b	f0	8.179	SLOW
b	f1	8.450	SLOW
b	f2	8.690	SLOW
b	f3	7.923	SLOW
c	f1	8.614	SLOW
c	f2	8.517	SLOW
c	f3	7.908	SLOW
d	f0	8.032	SLOW
d	f1	8.643	SLOW
d	f2	8.480	SLOW
d	f3	8.447	SLOW

I set constraints to adjust maximum delay to 6 ns. However, all paths exceed 6 ns. Although the delays cannot achieve the 6ns target, there is a significant reduction in delays.

All realizations have almost the same no timing constraints design delays. SSI_Library design cannot achieve the 9ns target but there is a significant reduction in delays. Likewise, decoder and mux design cannot achieve the 6ns target, but there is a significant reduction in these delays. Decoder and mux have less delay than SSI_Library design with timing constraints. Decoder and Mux designs have less logic gates. Therefore, it is natural that their delays are less than SSI_Library.

All three realizations have same resource usage.

RESEARCH

1) There are different types of simulations in Vivado. There are fundamental differences for these simulations. There are different types of simulations in Vivado. There are fundamental differences for these simulations. Behavioral Simulation simulates design by ignoring the physical structure. They show the basic result that shows whether the design is correct or not. RTL analysis represents a digital system with logic gates. It does not show the structure of Fpga. Synthesis represents the digital system with FPGA resources. Therefore, Post-Synthesis Functional Simulation has more reliable simulation results compared to Behavioral Simulation. The synthesis phase does not include device placement. Therefore, it does not show absolute simulation and timing results. The implementation phase computes the simulation and timing results according to the placement on the FPGA board. Therefore, Post-Implementation Functional Simulation has more reliable results than Post-Synthesis Functional Simulation. There are different types of constraints. Timing constraint is one of these types. The design is put into practice according to the designer's timing constraint. Therefore, Post-Implementation Timing Simulation gives the most reliable design if there are any timing constraints.

2) Vivado implements the design onto the FPGA board based on optimal conditions. The design provides optimal results for timing, power consumption, placement, and other constraints. However, designers may want a design with less delay. The system may sacrifice less power consumption to meet the timing constraint. For such reasons, designers create different constraints to fulfill their demands.
Synthesis constraints are constraints that include how the synthesis of HDL code into RTL occurs. They have different synthesis constraints that have different format and using.
I/O constraints are used assign input/ outputs pins to specific locations. I/O constraints is important especially ASIC and fast Fpga applications.
Timing constraints are used for specific timing assignments. Designers can assign specific time value for specific paths or all paths. Vivado implements design according to these constraints.
Placement constraints are used for assignment Fpga resources to specific locations. Designers can want place LUTs or other resources to specific locations.

3) Synthesis attributes allow the design to be shaped as the designer wishes. Vivado wants to optimize the design for the most favorable conditions, sometimes optimizations are not the best solutions for specific designs. Therefore, synthesis attributes help designers for these implementations.
"DONT_TOUCH" prevents Vivado from changing or deleting a signal for optimization purposes.

```
(* dont_touch = "yes" *) wire sig1;
assign sig1 = in1 & in2;
assign out1 = sig1 & in2;
```

Verilog example of DONT_TOUCH

There are different memory RAM types in Vivado. RAM_STYLE allows the designer to use any type of RAM they want.

```
(* ram_style = "distributed" *) reg
[data_size-1:0] myram [2**addr_size-
1:0];
```

Verilog example of RAM_STYLE