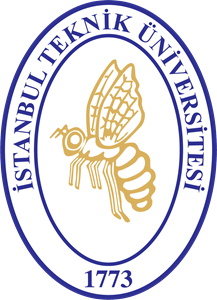
**DIGITAL SYSTEM DESIGN APPLICATIONS**

**(CRN: 11275)**

**THE REPORT OF EXPERIMENT – 3**



Faculty of Electrical and Electronics Engineering

Electronics and Communication Engineering

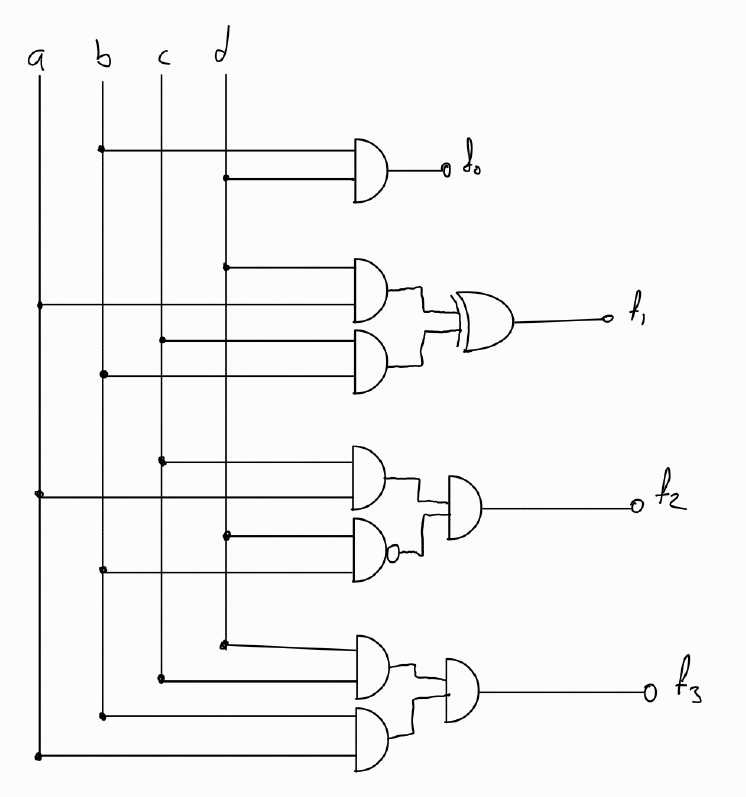
Yusuf Tekin – 040200043

1. **Realization with SSI Library**

**A group of math equations

Description automatically generated**

Karnaugh Maps and Boolean Expressions (with simplifications)



Gate Level Circuit Schematic

*A screenshot of a computer program

Description automatically generated*

RTL Schematics of The Functions

Behavioral Simulation Console Message

Time resolution is 1 ps

source experiment3\_tb.tcl

{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

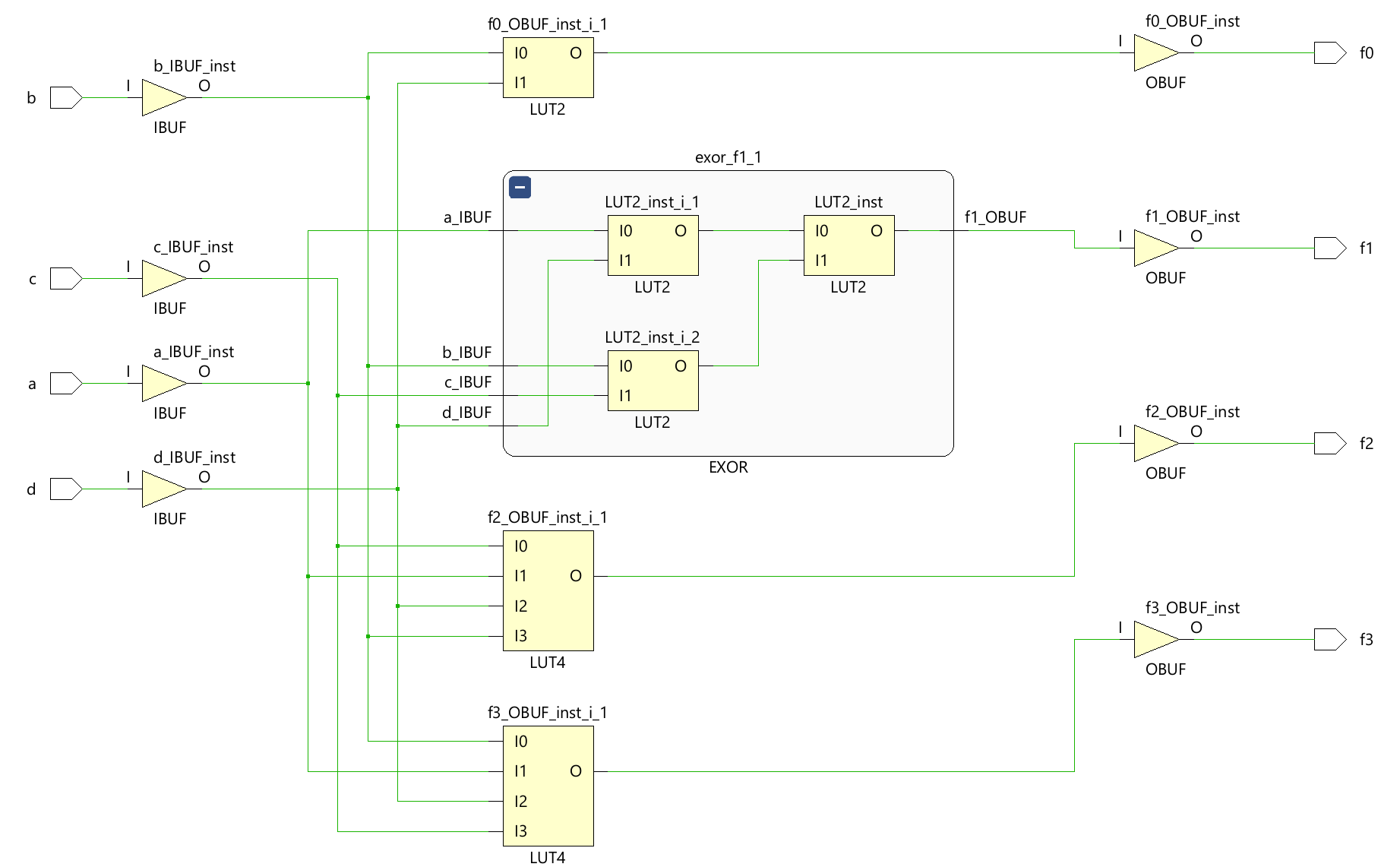
$finish called at time : 800 ns : File "C:/Users/jsphtkn/Vivado Projects/sstu\_experiment\_3/sstu\_experiment\_3.srcs/sim\_1/imports/Experiment\_3/experiment3\_tb.v" Line 52

INFO: [USF-XSim-96] XSim completed. Design snapshot 'experiment3\_tb\_behav' loaded.

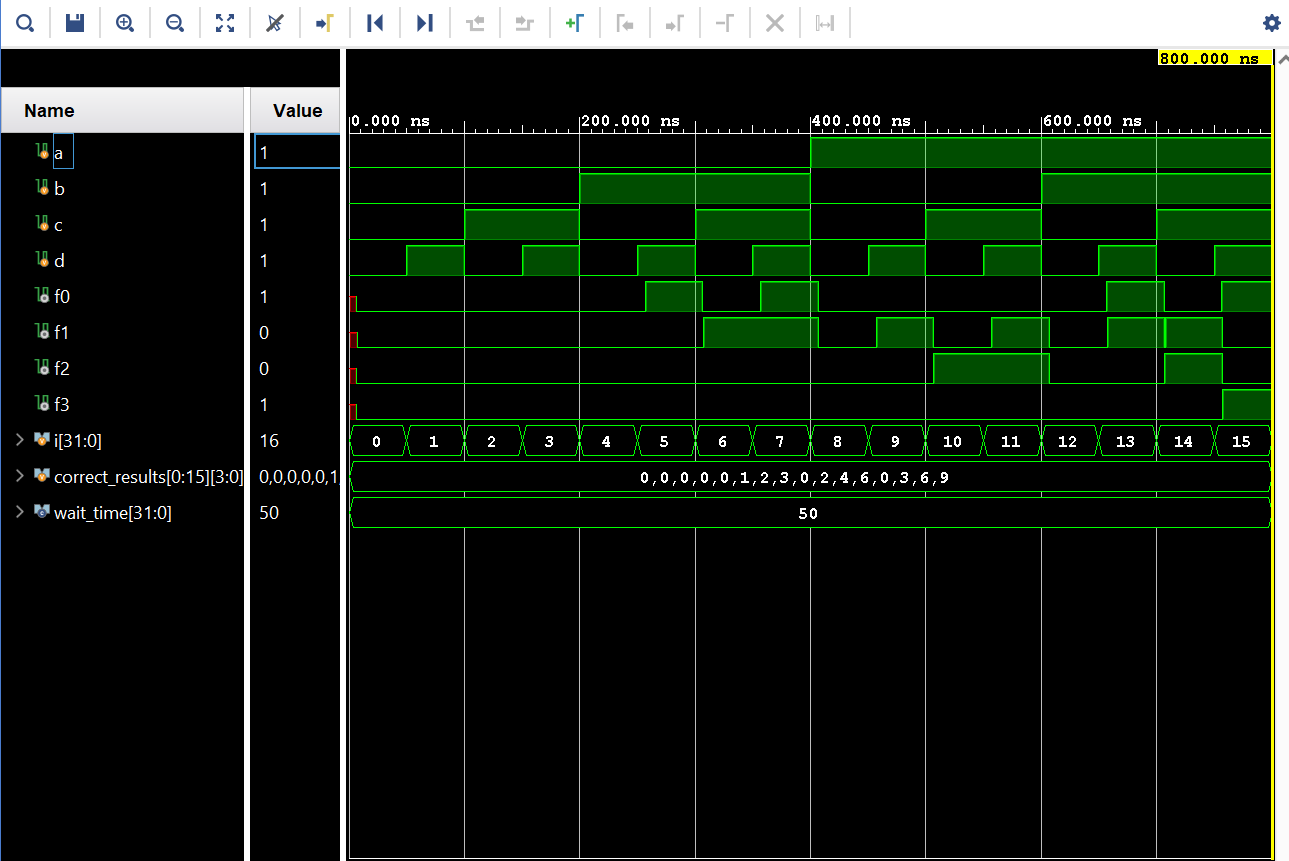
INFO: [USF-XSim-97] XSim simulation ran for 1000ns

launch\_simulation: Time (s): cpu = 00:00:01 ; elapsed = 00:00:08 . Memory (MB): peak = 2459.781 ; gain = 40.254

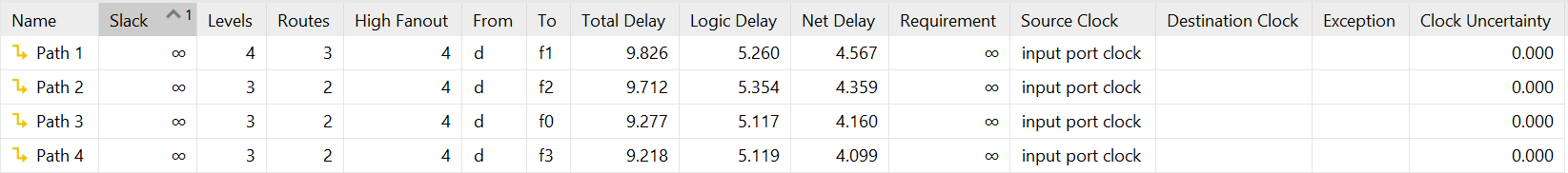
If it considered that the 4-bit output as a decimal number and two 2-bit input {a,b} and {c,d} as decimal numbers, the output is zero whenever either one of the input value is zero. Such behavior can be seen in a multiplication function. Where the inputs are named as {a,b}= X and {c,d}= Y, the multiplication function can be expressed as . Therefore this function is a multiplication function.



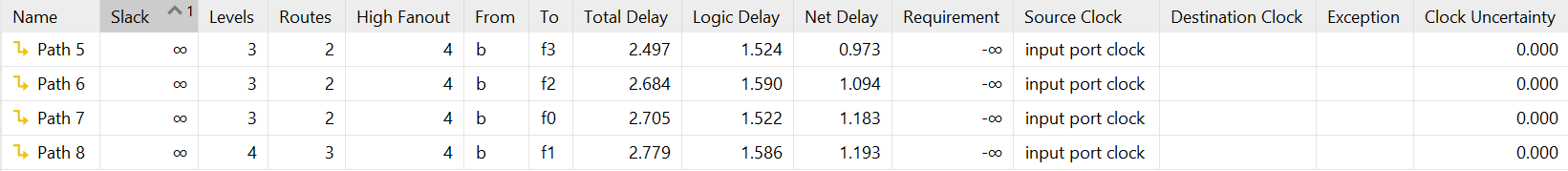
SSI Realization Technology Schematic



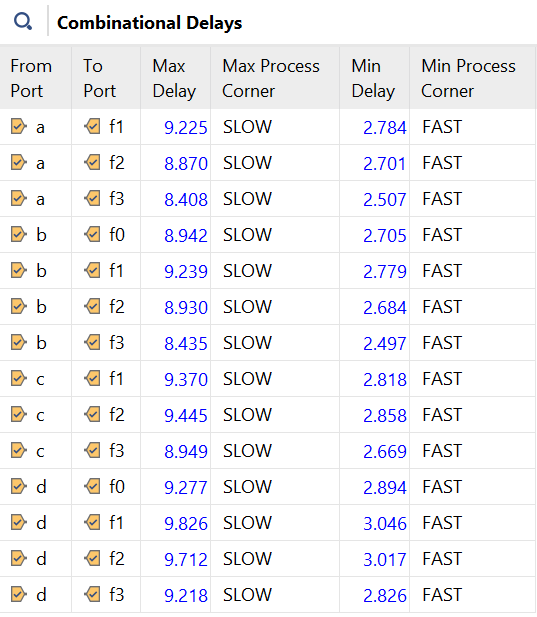
SSI Realization Post-Synthesis Timing Simulation



SSI Realization Path Delays - Setup



SSI Realization Path Delays - Hold



SSI Realization Combinational Path Delays

A screenshot of a graph

Description automatically generated

SSI Realization Utilization Summary

**Timing Constraint Design:**

**A screenshot of a computer

Description automatically generated**

SSI Realization Path Delays - Setup (9 ns max delay constraint)

A screenshot of a computer

Description automatically generated

SSI Realization Path Delays - Hold (9 ns max delay constraint)

A screenshot of a computer

Description automatically generated

SSI Realization Partial Input Delay (9 ns max delay constraint)

A screenshot of a graph

Description automatically generated

SSI Realization Utilization Summary (9 ns max delay constraint)

**set\_max\_delay** constraint mainly focuses on setting the delays of the paths on the setup section to 9 ns. It can be seen in the constrained path delays figure above that the objective is achieved. By doing so is increased the input delays yet increased the LUTs it uses by one. Thats because while relocating the LUTs to be able to reduce the delays between them, the input delays rises due to the higher path lengths.

**LOC Constraint Design:**

**A screenshot of a computer

Description automatically generated**

SSI Realization Path Delays - Setup (LOC Constraint)

A screenshot of a computer

Description automatically generated

SSI Realization Path Delays - Hold (LOC Constraint)

A screenshot of a computer screen

Description automatically generated

SSI Realization - Combinational Delays (LOC Constraint)

A screenshot of a computer

Description automatically generated

f0 relocated LUT

A screenshot of a computer

Description automatically generated

f1 LUT

Output *f1* is the output function which reduced into a Boolean expression that includes a exor gate. As it turns out that in SSI\_Library the design of exor gate uses three LUT2 blocks instead of a LUT4. Therefore, it cannot be relocated to SLICE\_X12Y67 position.

A computer screen shot of a circuit board

Description automatically generated

f2 relocated LUT

A screenshot of a computer

Description automatically generated

f3 relocated LUT

In the figures above, both f2 and f3 use the same LUTs. That’s because of the similar circuitry they have. There is only a gate difference between the outputs *f2* and *f3.* If the *Gate Level Circuit Schematic (p.2)*  is checked, it can be seen that the difference between these outputs is AND/NAND gates. After the implementation, Vivado (2024.1) decides that there is no different LUTs need for that little difference and just puts a NOT gate to create such difference. This occurrence might be different for the other versions of the program.

**Both LOC and Timing Constrained Design:**

A screenshot of a computer

Description automatically generated

SSI Realization Timing Summary (with all constraints)

A screenshot of a computer

Description automatically generated

SSI Realization Path Delays - Setup (with all constraints)

Even though the *Post-Implementation Simulation* gives the same results as pre-constrained design, the tool could not manage to meet the desired design. For *Path 1, Path 2* and *Path 3,* the required max delays are not achieved.

In terms of combinational delay, out of all four designs, the best one is the design with the max delay constraint. Relocating the LUTs has resulted worse and combining both relocated and max delay constrained design has resulted the worst and even the requirements have not met.

To conclude, it is not advisable to use placement and routing manually. The default algorithm of Vivado does the job better. But that’s not the case for the delay and timing modulations. For the right conditions and the reasons, the max delay constraint could be very useful.

A screenshot of a computer

Description automatically generated

SSI Realization Post-Implementation Simulation (with all constraints)

There are significant output delays and glitches throughout the *Post-Implementation Simulation* which have not been observed in *Behavioral Simulation*. These glitches are mostly caused by the output nets which do not meet the requirements of the constraints and non-clocked design.

1. **Realization with Decoder**

A yellow and black math equations and numbers

Description automatically generated with medium confidence

Decoder Minterms, Schematic and Logical Functions

Decoder Realization TCL Console - Simulation

{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:/Users/jsphtkn/Vivado Projects/sstu\_experiment\_3/sstu\_experiment\_3.srcs/sim\_1/imports/Experiment\_3/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 = 00:00:01 ; elapsed = 00:00:05 . Memory (MB): peak = 2673.414 ; gain = 0.000

A screenshot of a computer

Description automatically generated

Decoder Realization Behavioral Simulation

As can be both seen in the *TCL Console* and *Behavioral Simulation* graph, the design works as intended which is a multiplication block.

A diagram of a computer

Description automatically generated

Decoder Realization RTL Schematic

A diagram of a computer

Description automatically generated

Decoder Realization Technology Schematic

A screenshot of a computer screen

Description automatically generated

Decoder Realization Path Delays - Combinational Delays

A screenshot of a computer

Description automatically generated

Decoder Realization Path Delays - Setup

A screenshot of a computer

Description automatically generated

Decoder Realization Path Delays - Hold

A screenshot of a computer

Description automatically generated

FPGA Device Overview of the Placed Design

Between this decoder design and previous SSI design there is no difference in terms of the used cell type but in this design the LUTs are not distributed as previous ones.

A screenshot of a graph

Description automatically generated

Decoder Realization - Utilization Summary

As it can be seen in the figures, the amount of LUTs in-use is drop to 3 from 5 (or 6 depending on the constraints). The number of I/O ports does not change as expected. Because there are still four inputs and four outputs as the first design.

A screenshot of a test

Description automatically generated

Combination Delays (after 6ns max delay)

A screenshot of a computer

Description automatically generated

Path Delays - Setup (after 6ns max delay)

1. **Realization with MUX**

A screenshot of a whiteboard with text and diagrams

Description automatically generated

MUX Based Schematic and It's Minterms

MUX Realization TCL Console (after behavioral sim)

{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:/Users/jsphtkn/Vivado Projects/sstu\_experiment\_3/sstu\_experiment\_3.srcs/sim\_1/imports/Experiment\_3/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 = 00:00:01 ; elapsed = 00:00:05 . Memory (MB): peak = 3185.719 ; gain = 0.000

A screenshot of a computer

Description automatically generated

MUX Realization Behavioral Simulation

A diagram of a computer

Description automatically generated

MUX Realization RTL Schematic

A diagram of a computer network

Description automatically generated

MUX Realization Technology Schematic

A screenshot of a computer screen

Description automatically generated

MUX Realization Combinational Delays

A screenshot of a computer

Description automatically generated

MUX Realization Path Delays – Setup

A screenshot of a computer

Description automatically generated

MUX Realization Path Delays – Hold

A screenshot of a computer

Description automatically generated

MUX Realization FPGA Device Overview

It is observable that in this design a LUT2 primitive and four LUT4 primitives are in use. There is not any LUT2 used for the decoder design nor SSI design.

A screenshot of a graph

Description automatically generated

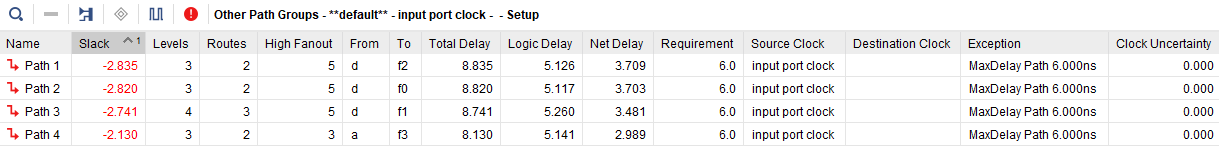
MUX Realization Utilization Summary

Comparing the utilizations of both three designs, the MUX design is the least LUT used design of all, but the I/O usage is the same for both designs. Although, the design used the least amount of LUTs, that does not mean it has the least delay. In terms of delay the ranking is actually backwards. Without any constraints applied, the MUX design has the greatest delay in the setup.

A screenshot of a graph

Description automatically generated

MUX Realization Combinational Delays (after 6ns max delay)



MUX Realization Path Delays - Setup (after 6ns max delay)

**Conclusion:**

For the non-constrained designs, if the desired design requires less resources, the best choice would be the MUX design due to the 5 LUTs it uses yet it has the longest delays. Therefore, if the design is required to be precise in terms of delays and timing the best choice would be the decoder design.

Even though the rankings don’t change, the delays and utilization change whenever a constraint is added. Most paths in the designs cannot meet the required max delay value but their delay drops remarkably. If the desire is the least delay the max-delay-constrained decoder design is slightly faster than the MUX design. Besides, all the time constrained designs have shorter delays.

1. **Research**

**Simulation types and their differences:**

**Behavioral Simulation** is the fastest and simplest simulation, but it only verifies the high-level logic. It’s easy to code via Verilog or VHDL which makes it ideal for the early stages of design. There is no timing or netlist included.1

**Post-Synthesis Functional Simulation** confirms that the design logic is still correct after synthesis, using the synthesized netlist without timing information. This is useful to notice that synthesis hasn't changed the functionality.1

**Post-Implementation Functional Simulation** verifies that the placed-and-routed design (after physical layout on the FPGA) maintains the correct logical behavior. This is closer to the final design but still ignores timing.1

**Post-Implementation Timing Simulation** is the most accurate and realistic simulation, as it includes timing delays from the physical layout. This simulation checks both logic and timing, confirming that the design meets timing constraints before deploying it on the FPGA.1

**FPGA Design Constraints:**

Timing constraints, location constraints, I/O constraints, clock constraints and area constraints are the main types of constraints. By using these constraints, designers can use Vivado to produce a design that not only functions correctly but also meet the desired performance, power, and layout requirements.2

**Synthesis Attributes:**

In FPGA design there are special directives or properties that provide instructions to the synthesis tool on how to handle specific parts of the design during the synthesis process. They do not alter the functionality of the design but control how the design is implemented on the FPGA hardware.3

The **DONT\_TOUCH** attribute is used to prevent the synthesis tool from optimizing or modifying a specific signal, register, or module. In Verilog or VHDL, you can apply the DONT\_TOUCH attribute to a signal, register, or module instance as:

(\* DONT\_TOUCH = "true" \*) reg [3:0] signal\_name;

This way, this line of code is not optimized during the synthesis. Can lead to increased resource usage because the synthesis tool cannot optimize the protected elements.3

The **RAM\_STYLE** attribute is used to control how the synthesis tool implements memory structures (like arrays or inferred RAM) in the FPGA. In FPGAs, memory can be implemented either using block RAM (BRAM), distributed RAM (using LUTs), or register-based memory. In Verilog or VHDL, you apply the RAM\_STYLE attribute to arrays or inferred memory as:

(\* ram\_style = "block" \*) reg [7:0] memory \_name[0:15];

Choosing the wrong RAM style can lead to inefficient resource usage. For example, using block RAM for a very small memory structure wastes block RAM resources, while using distributed RAM for a large memory can consume too many LUTs. Using RAM\_STYLE for such situations to avoid using unnecessary resources might result in a more efficient design.3

**References:**

1. Vivado Design Suite User Guide, Logic Simulation, Simulating with Vivado Simulator (UG900)
2. Vivado Design Suite User Guide, Using Constraints, UG903
3. Vivado Design Suite User Guide: Synthesis, Synthesis Attributes (UG901)