COL215 HW Assignment 1 - 2 x 1 and 4 x 1 Multiplexers

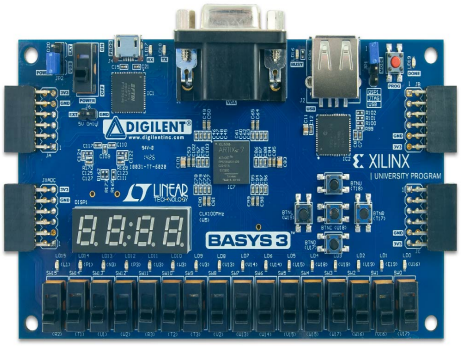
Submission By :

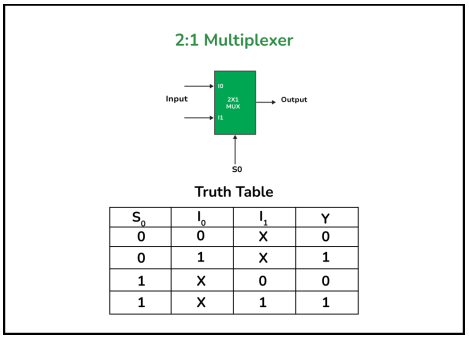
Yash Rawat **2023CS50334**

Priyanshi Gupta **2023CS10106**

Department of Computer Science and Engineering
Indian Institute of Technology, Delhi

August 11, 2024





Truth Table

S_0	I_0	I_1	Y
0	0	X	0
0	1	X	1
1	X	0	0
1	X	1	1

1 Working Principle of Multiplexers

1.1 What is a Multiplexer ?

A multiplexer is a device that selects one of several input signals and forwards the chosen input into a single output signal. An $NX1$ mux (abbreviation for multiplexer) has the ability to choose between N distinct inputs, for which it requires atleast $\lceil \log_2 N \rceil$ selection bits. Hence a $2X1$ mux requires 1 input bit whereas $4X1$ mux requires 2 input bits.

The truth table for a 2x1 Mux and a 4x1 Mux look as following (Note that X/Y/Z means that output is invariant of the respective bit, be it 0 or 1) :

D_0	D_1	S	Out
1	X	0	1
0	X	0	0
X	1	1	1
X	0	1	0

Table 1: Truth Table of a $2X1$ Mux

D_0	D_1	D_2	D_3	S_0	S_1	Out
1	X	Y	Z	0	0	1
0	X	Y	Z	0	0	0
X	1	Y	Z	1	0	1
X	0	Y	Z	1	0	0
X	Y	1	Z	0	1	1
X	Y	0	Z	0	1	0
X	Y	Z	1	1	1	1
X	Y	Z	0	1	1	0

Table 2: Truth Table of a $4X1$ Mux

1.2 Expression of 2X1 , 4X1 Mux and Recursive Definition of 4X1 Mux using 2X1 Mux

From the above truth table we get the formula for the $2X1$ Mux as:

$$MUX_2(D_0, D_1, S) = S^{'} \cdot D_0 + S \cdot D_1$$

whereas the formula for $4X1$ Mux is :

$$MUX_4(D_0, D_1, D_2, D_3, S_0, S_1) = S_0^{'} \cdot (S_1^{'} \cdot D_0 + S_1 \cdot D_2) + S_0 \cdot (S_1^{'} \cdot D_1 + S_1 \cdot D_3)$$

In the case of $4X1$ Mux ,by collecting the sum of minterms in the manner mentioned above, it provides insight into the recursive nature of $4X1$ Mux and how we can make it from $3 : 2X1$ Mux as shown below :

$$MUX_4(D_0, D_1, D_2, D_3, S_0, S_1) = S_0' \cdot (S_1' \cdot D_0 + S_1 \cdot D_2) + S_0 \cdot (S_1' \cdot D_1 + S_1 \cdot D_3)$$

$$MUX_4(D_0, D_1, D_2, D_3, S_0, S_1) = S_0' \cdot MUX_2(D_0, D_2, S_1) + S_0 \cdot MUX_2(D_1, D_3, S_1)$$

$$\mathbf{MUX_4(D_0, D_1, D_2, D_3, S_0, S_1) = MUX_2(MUX_2(D_0, D_2, S_1), MUX_2(D_1, D_3, S_1), S_0)}$$

We will be implementing the $4X1$ Mux based on the above principle instead of implementing the actual formula. Below attached are the circuit diagrams of our implementation of $2X1$ and $4X1$ Multiplexers (order of variables/ signal naming may vary) :

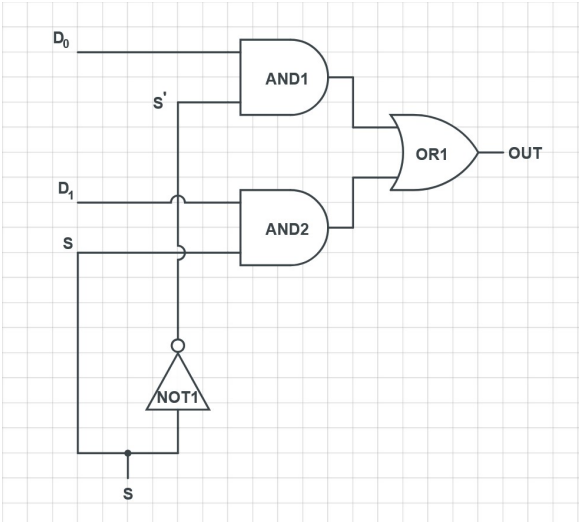


Figure 1: Implementation of $2X1$ Mux using AND, NOT and OR Gates

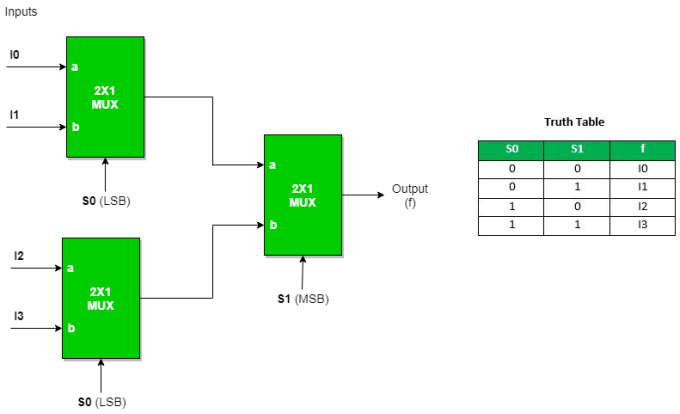
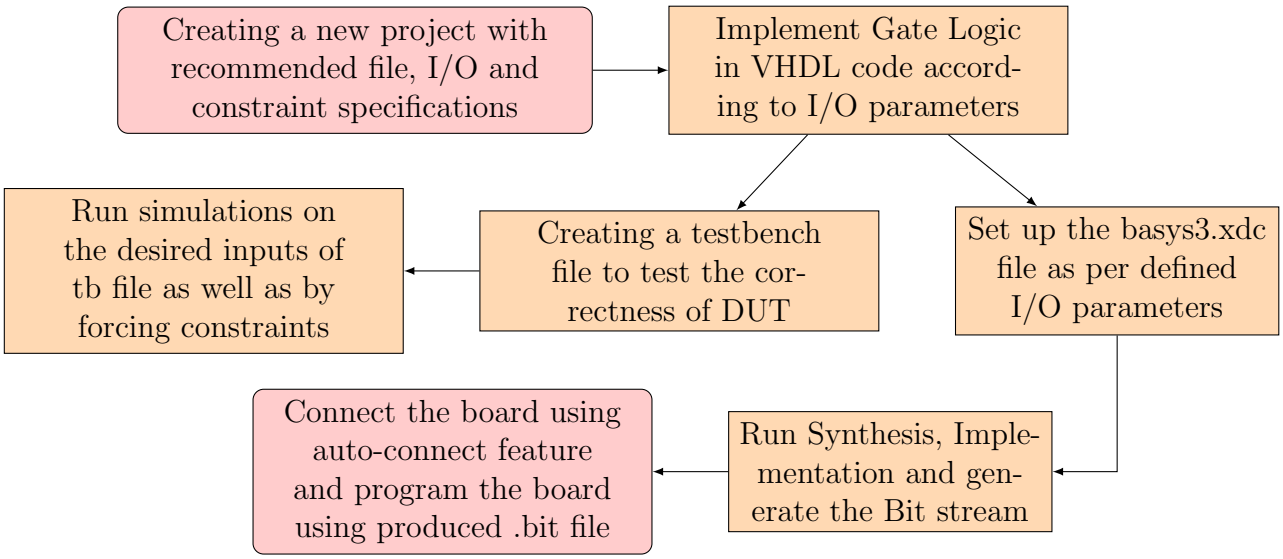


Figure 2: Recursive implementation of $4X1$ Mux using $2X1$ Mux

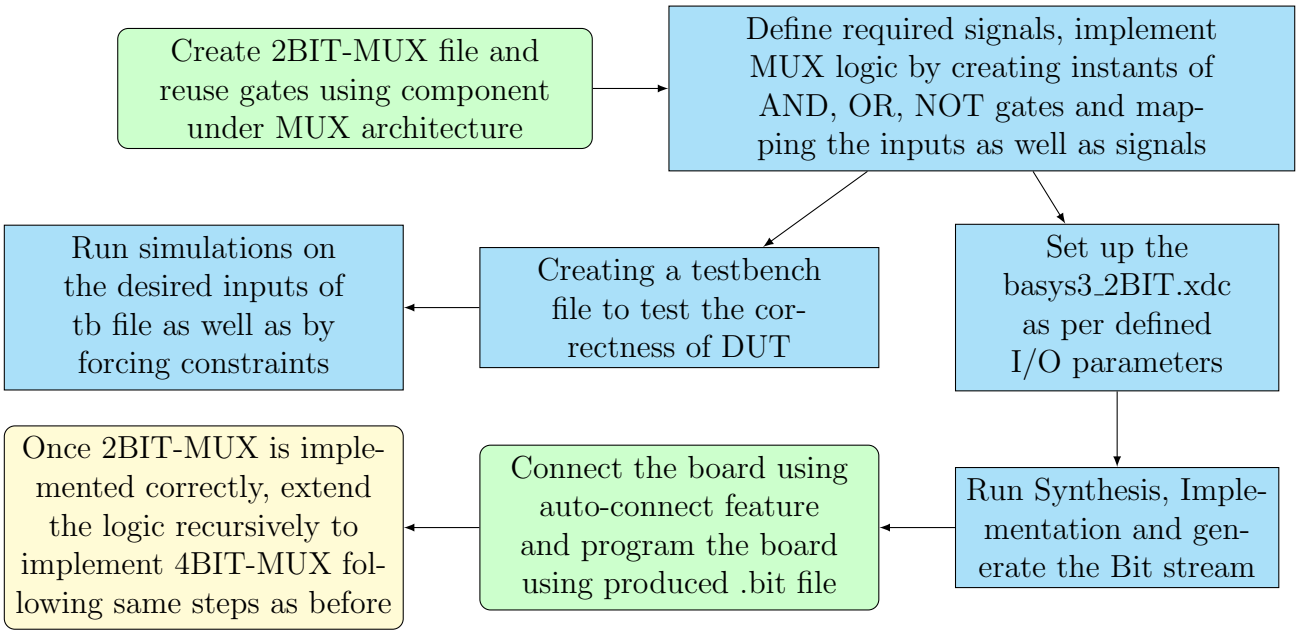
2 Design Philosophy

2.1 Design Process Flow Chart

2.1.1 Week 1 - Implementing AND, OR, NOT Gate Logic



2.1.2 Week 2 - Implementing 2x1 MUX and by extension 4x1 MUX



2.2 I/O Terminals

2.2.1 Week 1 - Basic Setup

Implementing a 2 input AND gate, 2 input OR gate and 1 input NOT gate required us to define five inputs and three outputs. The mapping of ports to the constraint file are as given below :

SNo.	Port Name	Board Contstraint	I/O
1	a_and_in	V17	IN
2	b_and_in	V16	IN
3	c_and_out	U16	OUT
4	d_or_in	W16	IN
5	e_or_in	W17	IN
6	f_or_out	E19	OUT
7	g_not_in	W15	IN
8	h_not_out	U19	OUT

Table 3: Mapping of Ports and Constraints in Basys3.xdc for BASIC-gates

2.2.2 Week 2 - 2x1-MUX and 4x1-MUX

Implementing a 2x1 MUX using 2 AND gates, 1 OR gate and 1 NOT gate required us to define three inputs and one output. The mapping of ports to the constraint file are as given below :

SNo.	Port Name	Board Contstraint	I/O
1	D1	V17	IN
2	D2	V16	IN
3	S	W16	IN
4	O	U16	OUT

Table 4: Mapping of Ports and Constraints in Basys3_2BIT.xdc for 2x1 MUX

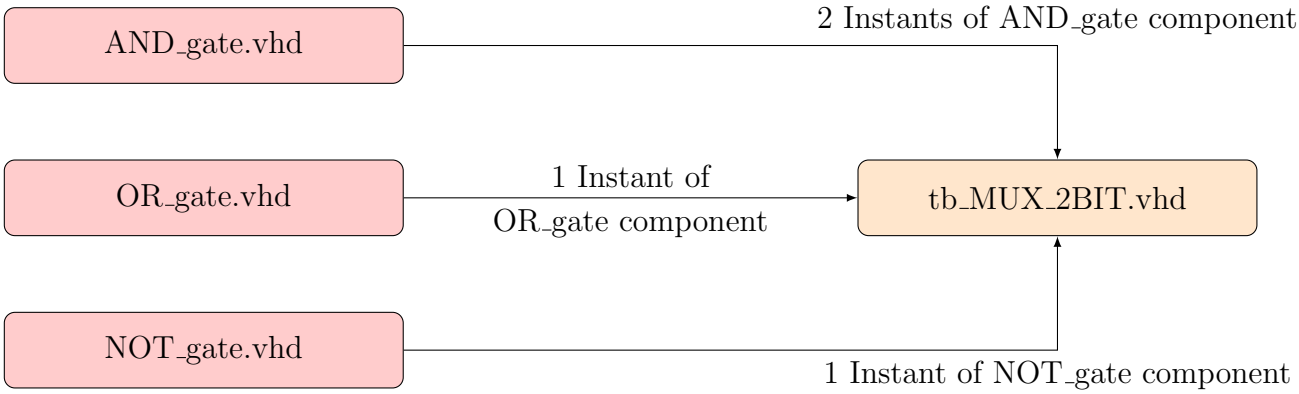
Implementing a 4x1 MUX using 4 AND gates, 3 OR gates and 2 NOT gates required us to define six inputs and one output.It is done by using two 2x1 MUX controlled by same switch and then using their outputs as inputs to another 2x1 MUX to get final output. The mapping of ports to the constraint file are as given below :

SNo.	Port Name	Board Contstraint	I/O
1	d1	V17	IN
2	d2	V16	IN
3	d3	W16	IN
4	d4	W17	IN
5	s1	W15	IN
6	s2	V16	IN
7	o	U16	OUT

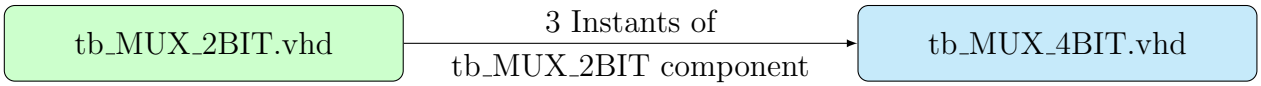
Table 5: Mapping of Ports and Constraints in Basys3_4BIT.xdc for 4x1 MUX

2.3 Hierarchy of VHD Files

2.3.1 For tb_MUX_2BIT.vhd



2.3.2 For tb_MUX_4BIT.vhd



The instants of entities from other files are defined under the architecture of the current file's entity by using *component* block. *port map* is used to map the signals between different components and make the system work as a unit. Any additional signal used (which was not defined during entity definition) also needs to be defined under architecture such as $m1$ and $m2$ for tb_MUX_4BIT (Which store the intermittent states of recursive calculation and act as inputs to $MUXF$)

The Final expression of the implemented 2x1 and 4x1 MUX respectively are :

$$MUX_2(D_1, D_2, S) = S \cdot D_1 + S^{'} \cdot D_2 = O \quad (1)$$

$$MUX_4(d_1, d_2, d_3, d_4, s_1, s_2) = s_2 \cdot (s_1 \cdot d_1 + s_1^{'} \cdot d_2) + s_2^{'} \cdot (s_1 \cdot d_3 + s_1^{'} \cdot d_4) = o \quad (2)$$

3 Simulation and Schematic Snapshots

3.1 Week 1 - AND, OR, NOT Gates

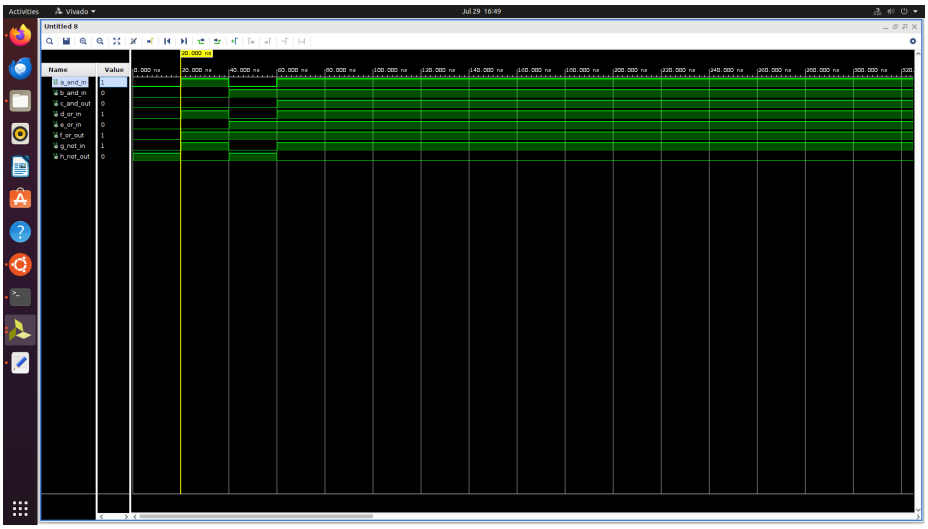


Figure 3: Simulation for AND,OR and NOT gates

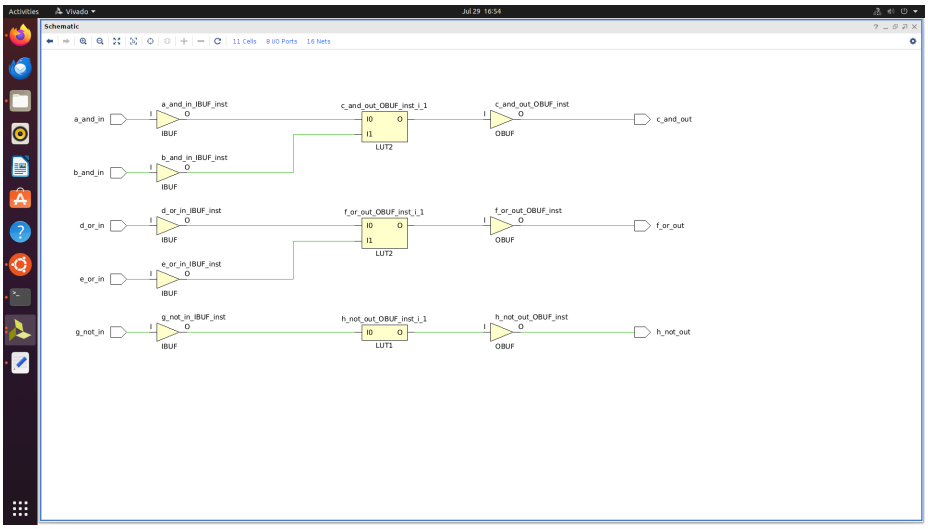


Figure 4: Schematic of AND,OR and NOT gates

3.2 Week 2 - $2X1$ Multiplexer

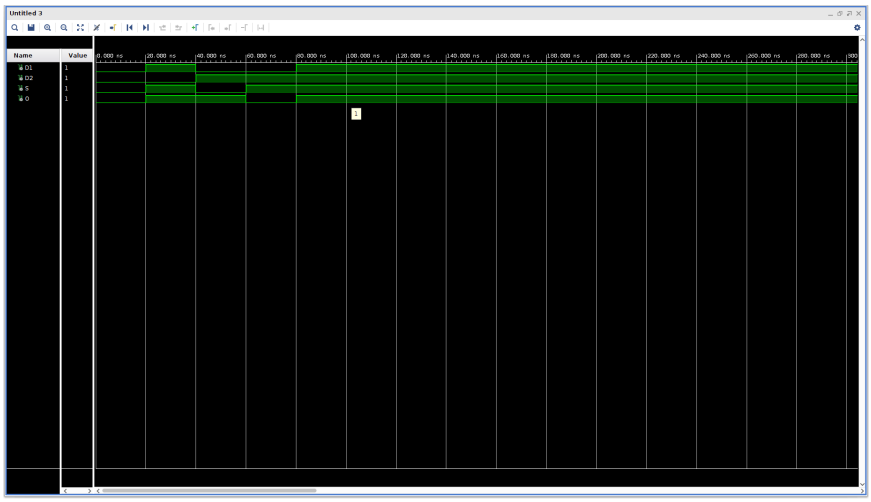


Figure 5: Simulation for 2x1 MUX

1. Slice Logic

Site Type	Used	Fixed	Prohibited	Available	Util%
Slice LUTs*	1	0	0	20800	<0.01
LUT as Logic	1	0	0	20800	<0.01
LUT as Memory	0	0	0	9600	0.00
Slice Registers	0	0	0	41600	0.00
Register as Flip Flop	0	0	0	41600	0.00
Register as Latch	0	0	0	41600	0.00
F7 Muxes	0	0	0	16300	0.00
F8 Muxes	0	0	0	8150	0.00

* Warning! The Final LUT count, after physical optimizations and full implementation, is typically lower. Run opt_design after synthesis, if not already completed, for a more realistic count.

2. Memory

Site Type	Used	Fixed	Prohibited	Available	Util%
Block RAM Tile	0	0	0	50	0.00
RAMB36/FIFO*	0	0	0	50	0.00
RAMB18	0	0	0	100	0.00

* Note: Each Block RAM Tile only has one FIFO logic available and therefore can accommodate only one FIFO36E1 or one FIFO18E1. However, if a FIFO18E1 occupies a Block RAM Tile, that tile can still accommodate a RAMB18E1

3. DSP

Site Type	Used	Fixed	Prohibited	Available	Util%
DSPs	0	0	0	90	0.00

Figure 6: Resource Utilization for 2x1 MUX

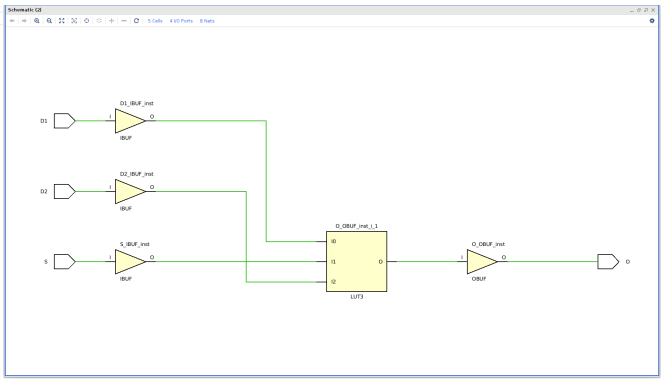


Figure 7: Schematic for 2x1 MUX

3.3 Week 2 - $4X1$ Multiplexer

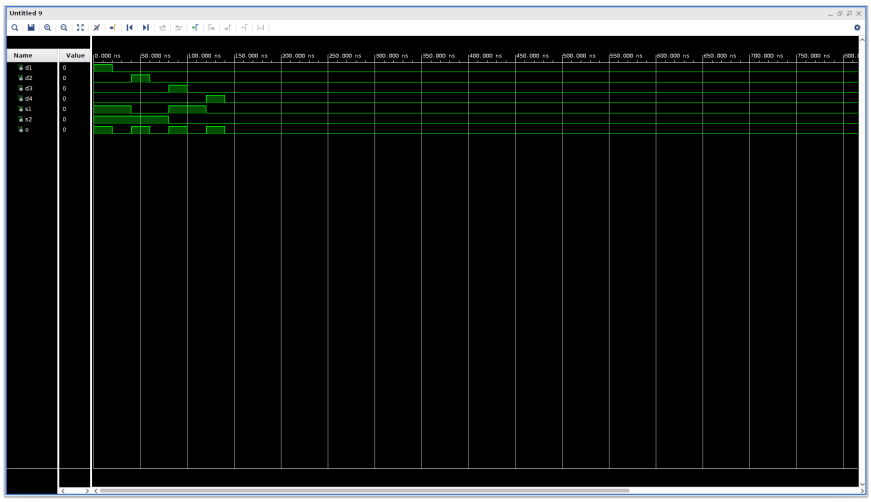


Figure 8: Simulation for 4x1 MUX

1. Slice Logic

Site Type	Used	Fixed	Prohibited	Available	Util%
Slice LUTs*	1	0	0	20800	<0.01
LUT as Logic	1	0	0	20800	<0.01
LUT as Memory	0	0	0	9600	0.00
Slice Registers	0	0	0	41600	0.00
Register as Flip Flop	0	0	0	41600	0.00
Register as Latch	0	0	0	41600	0.00
F7 Muxes	0	0	0	16300	0.00
F8 Muxes	0	0	0	8150	0.00

* Warning! The Final LUT count, after physical optimizations and full implementation, is typically lower. Run opt_design after synthesis, if not already completed, for a more realistic count.

2. Memory

Site Type	Used	Fixed	Prohibited	Available	Util%
Block RAM Tile	0	0	0	50	0.00
RAMB36/FIFO*	0	0	0	50	0.00
RAMB18	0	0	0	100	0.00

* Note: Each Block RAM Tile only has one FIFO logic available and therefore can accommodate only one FIFO36E1 or one FIFO18E1. However, if a FIFO18E1 occupies a Block RAM Tile, that tile can still accommodate a RAMB18E1

3. DSP

Site Type	Used	Fixed	Prohibited	Available	Util%
DSPs	0	0	0	90	0.00

Figure 9: Resource Utilization for 4x1 MUX

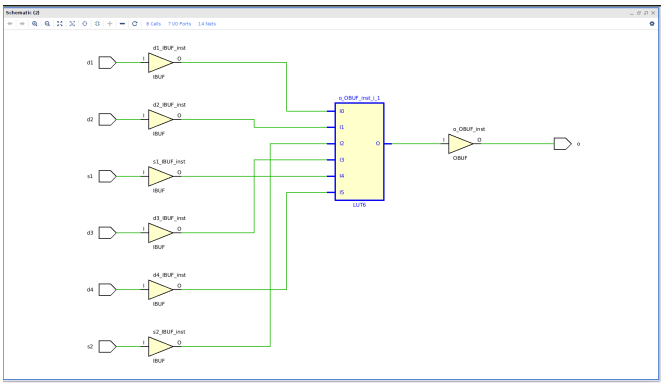


Figure 10: Schematic for 4x1 MUX

4 Working of FPGA Board

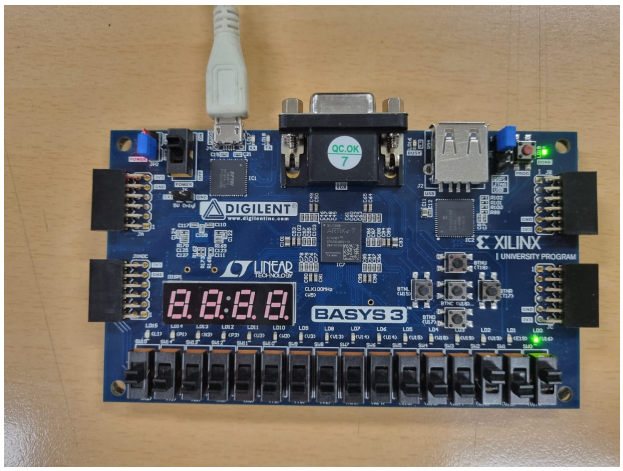


Figure 11: 2x1 MUX

In the board above, the input configuration is : $D_1 = 1$ (right most switch / V17), $D_2 = 0$ (left of D_1 / V16). The selection configuration is : S = 1 (left of D_2 / W16). Hence by the formula mentioned before [(1)], $O = 1$ as is visible by the ON state of LED U16.

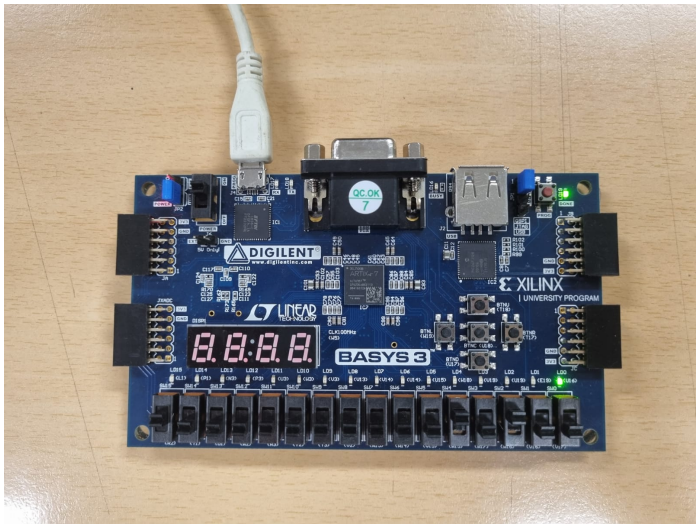


Figure 12: 4x1 MUX

In the board above, the input configuration is : $d_1 = 0$ (right most switch / V17), $d_2 = 0$ (left of d_1 / V16), $d_3 = 1$ (left of d_2 / W16) and $d_4 = 0$ (left of d_3 / W17). The selection configuration is : $s_1 = 1$ (left of d_4 / V16) and $s_2 = 0$ (left of s_1 / U16). Hence by the formula mentioned before [(2)], $o = 1$ as is visible by the ON state of LED U16.