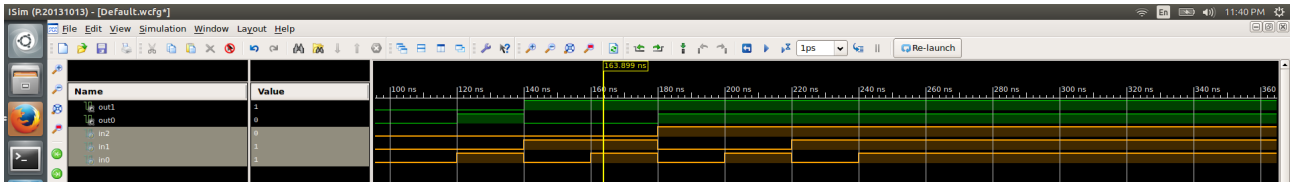
Embedded Logic Design
Assignment – Priority Encoder
Group 08

1. Truth Table for three input priority encoder..

In2	In1	In0	out1	out0
0	0	0	0	0
0	0	1	0	1
0	1	0	1	0
0	1	1	1	0
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1



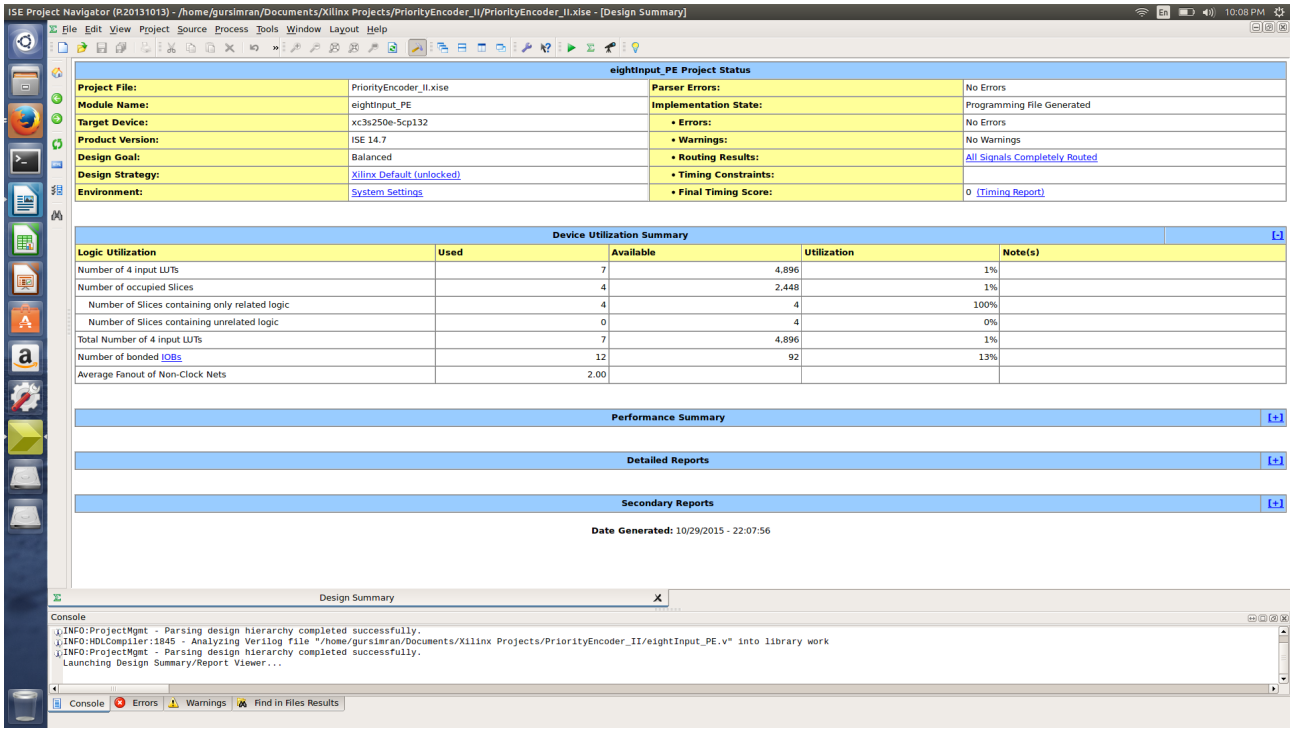
5. What test cases are useful and why ?

Test cases like 11, 101, 110, 111 are useful because they check the functionality of priority encoder instantly ie., they check if the encoder is considering only the set MSB or not.

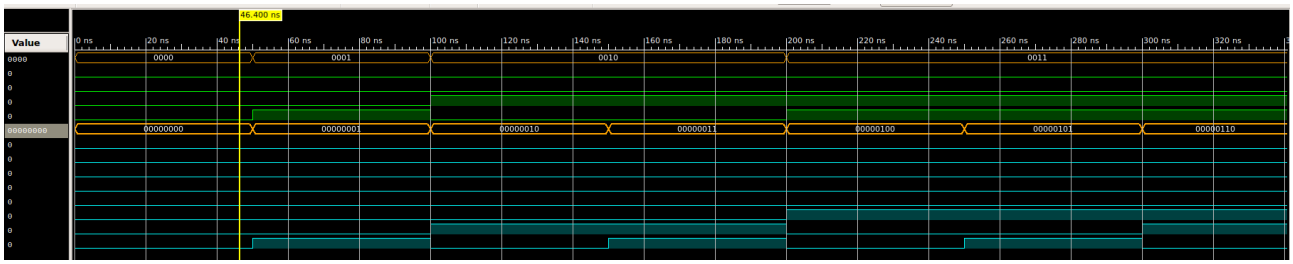
6. Truth table for 8 input priority encoder..

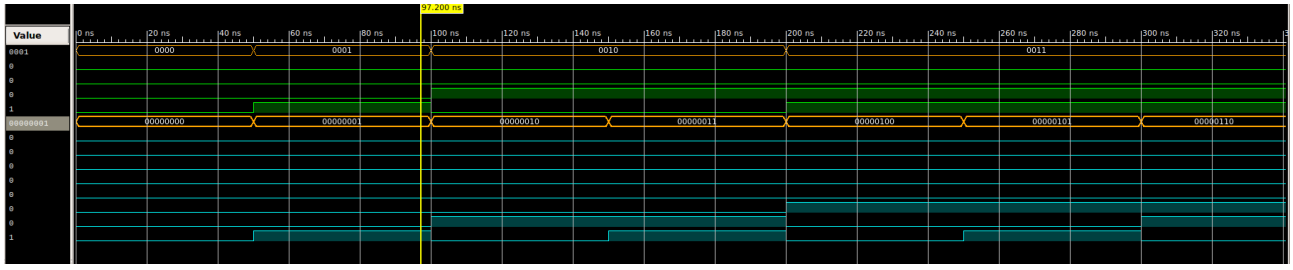
In7	In6	In5	In4	In3	In2	In1	In0	Out3	Out2	Out1	Out0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	0	1	X	0	0	1	0
0	0	0	0	0	1	X	X	0	0	1	1
0	0	0	0	1	X	X	X	0	1	0	0
0	0	0	1	X	X	X	X	0	1	0	1
0	0	1	X	X	X	X	X	0	1	1	0
0	1	X	X	X	X	X	X	0	1	1	1
1	X	X	X	X	X	X	X	1	0	0	0

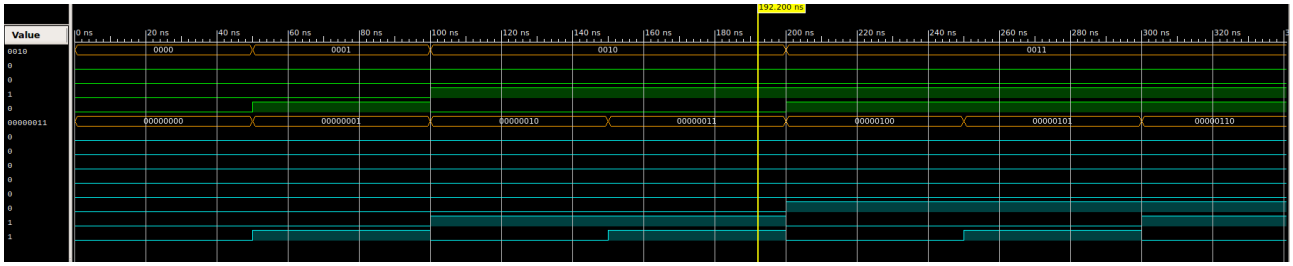
The following screenshot shows resource requirement for 8-input priority encoder..

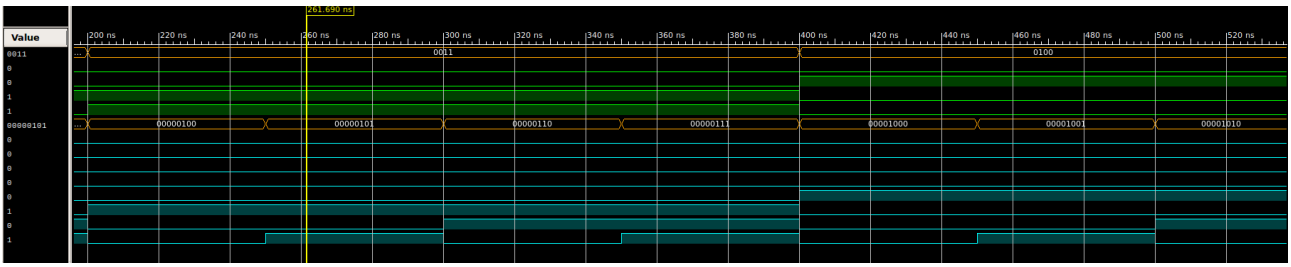


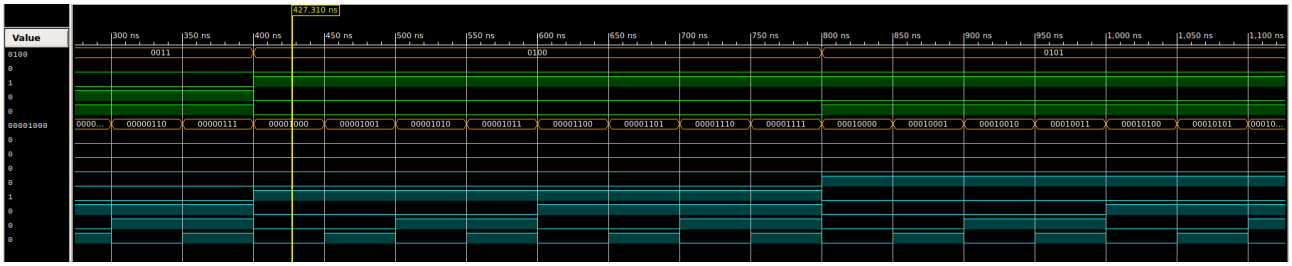
The following are the simulation waveforms for 8-input priority encoder..

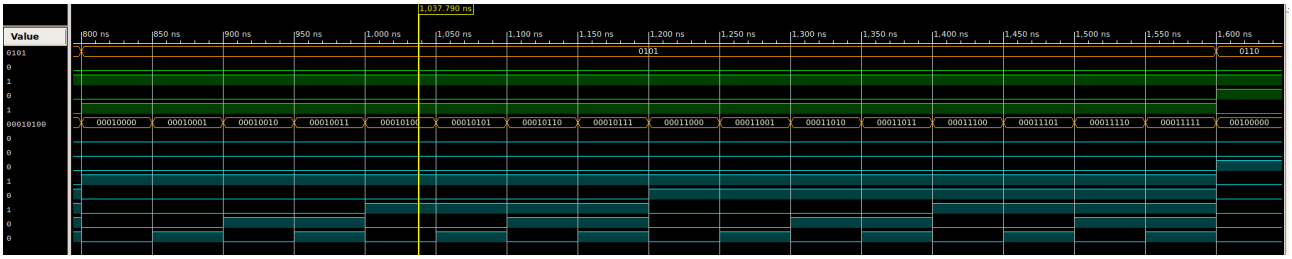


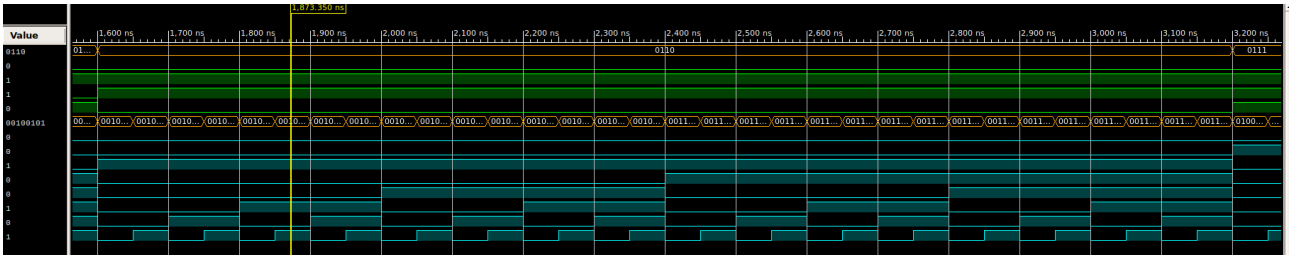


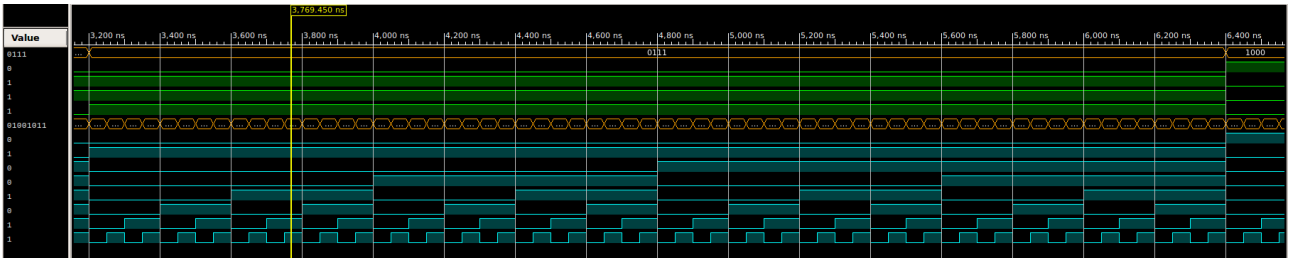


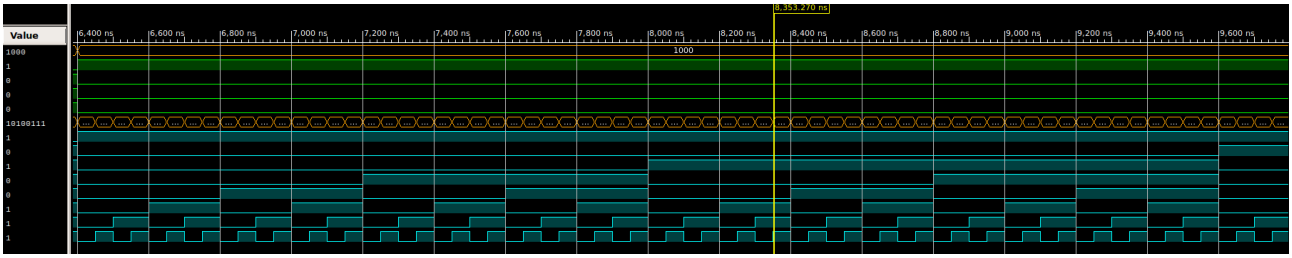

261.690 ns
Value
0011
00000100
00000101
00000110
00000111
00001000
00001001
00001010
0100

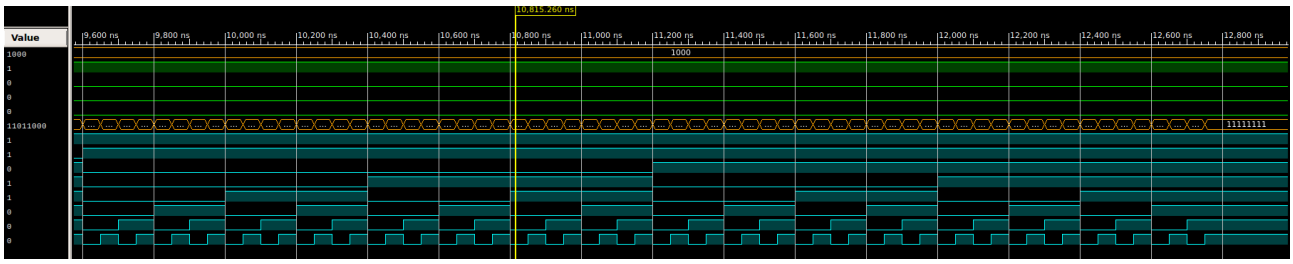

427.310 ns
Value
0100
00001000
0011
0100
0101
00000110
00000111
00001000
00001001
00001010
00001011
00001100
00001101
00001110
00001111
00010000
00010001
00010010
00010011
00010100
00010101


1,037.790 ns
Value
0101
00010100
0101
0110
00010000
00010001
00010010
00010011
00010100
00010101
00010110
00010111
00011000
00011001
00011010
00011011
00011100
00011101
00011110
00011111
00100000


1,875.350 ns
Value
0110
00100101
0110
0111
0010...
0011...
0100...


3,769.450 ns
Value
0111
01001011
0111
1000


8,353.270 ns
Value
1000
10100111
1000



LUT:

A **LUT**, which stands for **LookUp Table**, in general terms is basically a table that determines what the output is for any given input(s). In the context of combinational logic, it is the **truth table**. This truth table effectively defines how the combinatorial logic behaves.
In other words, whatever behavior we get by interconnecting any number of gates (like AND, NOR, etc.), can be implemented by a LUT.

The way FPGAs typically implement combinatorial logic is with LUTs, and when the FPGA gets configured, it just fills in the table output values, which are called the "LUT-Mask", and is physically composed of SRAM bits. So the same physical LUT can implement Y=AB and Y=AB', but the LUT-Mask is different, since the truth table is different.

IOB:

Stands for **Input/Output Blocks**. These are special blocks at periphery for external connaections.

Slice:

Logic resources are resources on the FPGA that can perform logic functions. Logic resources are grouped in slices to create configurable logic blocks. A slice contains a set number of LUTs, flip-flops and multiplexers. Different FPGA families implement slices and LUTs differently. For example, a slice on a Virtex-II FPGA has two LUTs and two flip-flops. In addition, the number of inputs to a LUT, commonly two to six, depend on the FPGA family.