Experiment 1: Combinational Circuit 1

Prateek Garg
20D070060
EE-214, WEL, IIT Bombay
August 11, 2021

Overview of the experiment:

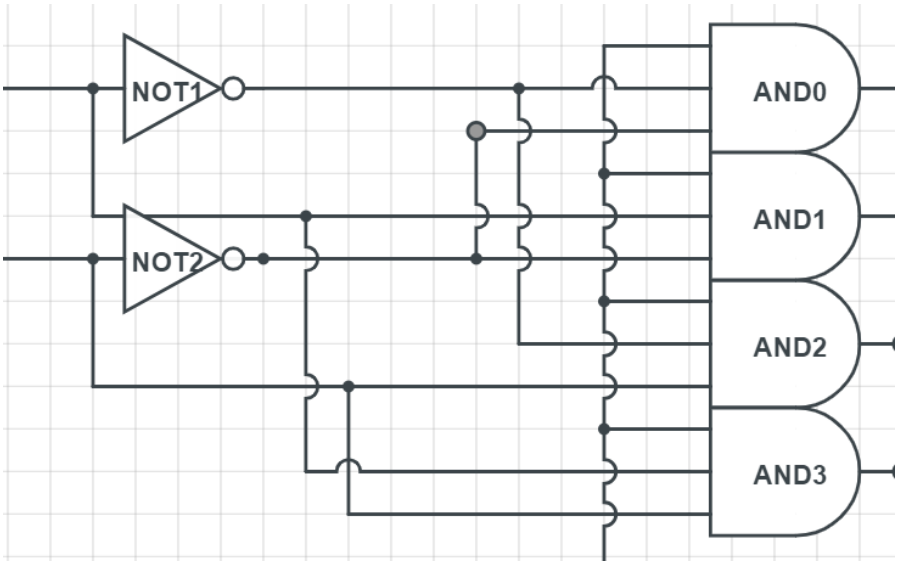
Objective:

- To design a 2 to 4 decoder with enable input using only 2 or 3 input AND / NAND gate and inverters.
- To design a 3 to 8 decoder with enable input using 2 to 4 decoder.
- To design a full adder using a 3 to 8 decoder.

All of the mentioned tasks are done using VHDL with the software Quartus.We used structural modelling which means components were instantiated and port mapping is used to describe connections .Implemented some additional gates in the Gates.vhdl file which were used in designs .All the designs were then simulated using ModelSim checked against every possible testcase.

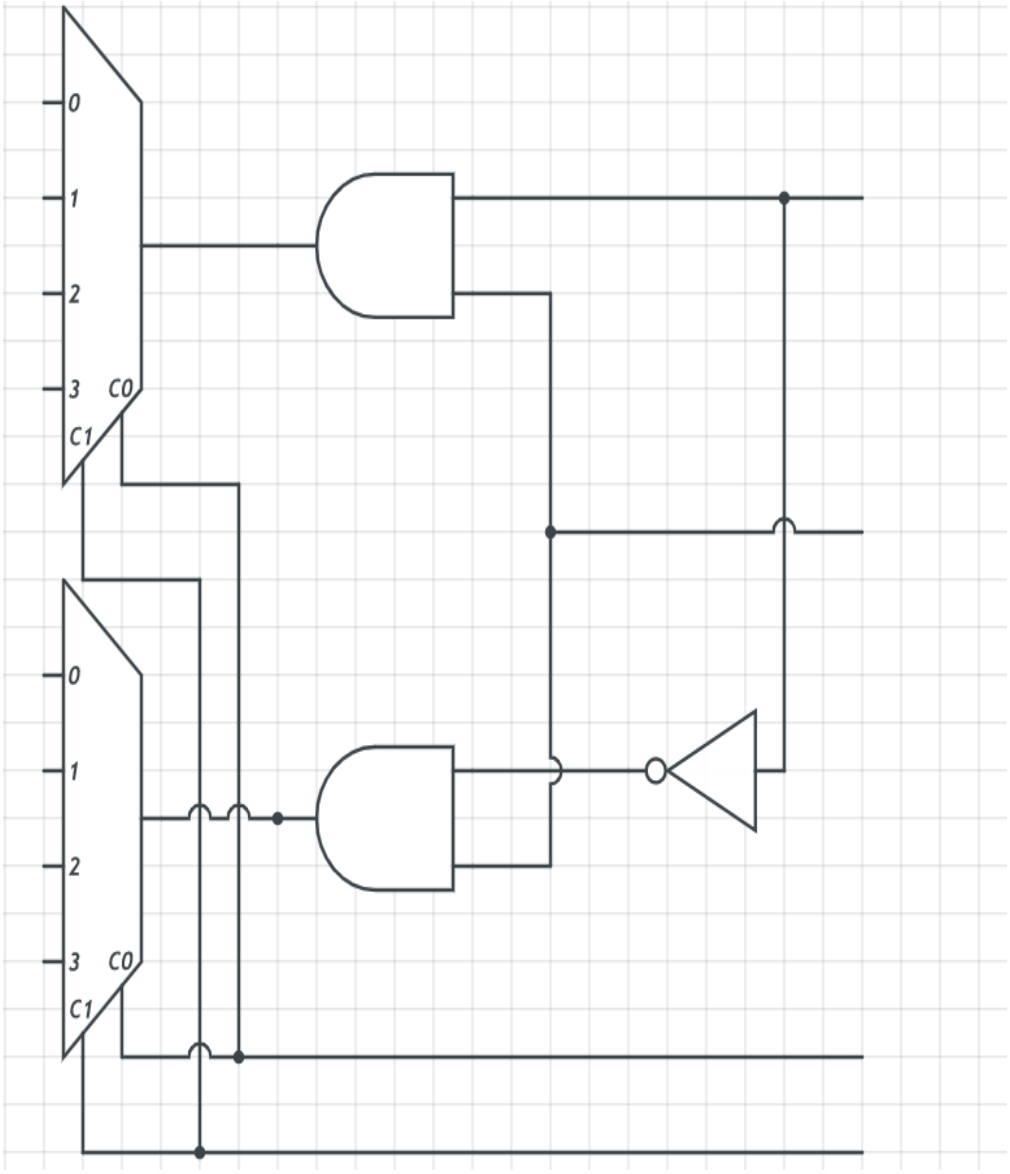
Approach to the experiment:

2 to 4 decoder:



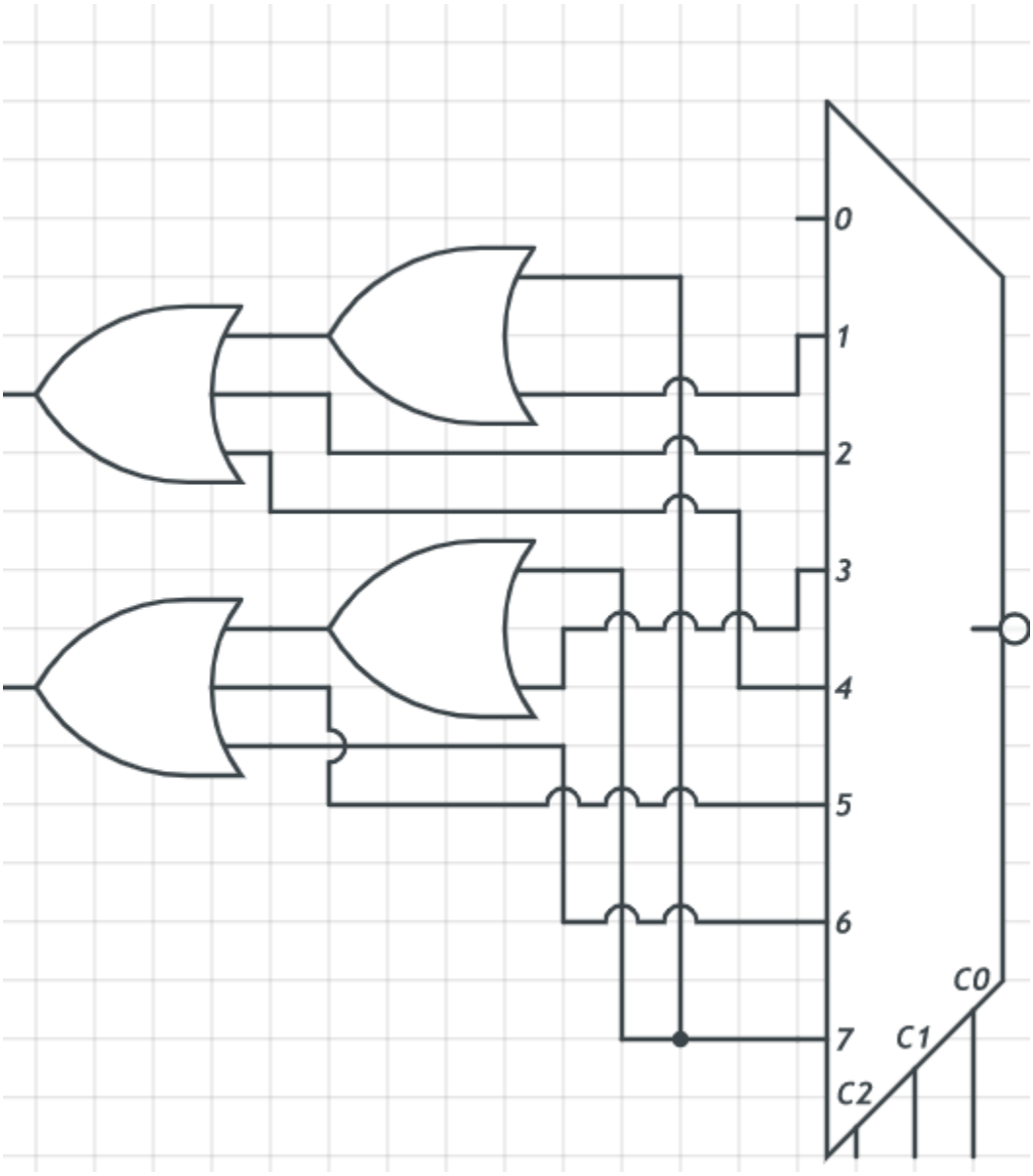
3-input AND gate is defined in gates.vhdl and used to 2 to 4 decoder.

3 to 8 Decoder:



3 to 8 decoder is created using two 2 to 4 decoders utilising the enable input pin of both of the decoders. The enable put is fed by a and gate which performs AND operation on Enable input 3 to 8 and third address input. Thus successfully retaining enable functionality as well creating a third address input for 3 to 8 decoder

Full Adder using 3 to 8 Decoder:



4-inputs OR gate was defined inside in the gates.vhdl file which was then subsequently used to implement Full-Adder using 3 to 8 decoder. The enable input of 3 to decoder is default set to 1. As indicated in diagram using Active Low sign.

All the circuits are drawn using online tool: www.circuitlab.com

Design document and VHDL code if relevant:

Architecture of 2 to 4 decoder:

```
architecture Struct of Decoder_2x4 is
   component AND_3 is
   port (A, B, C: in std_logic; Y: out std_logic);
  end component;

  component INVERTER is
    port (A: in std_logic; Y: out std_logic);
   end component;

  signal tI0, tI1, tA0, tA1: std_logic;
begin
  n0: INVERTER port map (A=> A0, Y=> tA0);
  n1: INVERTER port map (A=> A1, Y=> tA1);

  ag0: AND_3 port map (A => tA0, B => tA1 ,c => E, Y => Y0);
  ag1: AND_3 port map (A => A0, B => tA1 ,c => E, Y => Y1);
  ag2: AND_3 port map (A => tA0, B => A1,c => E, Y => Y2);
  ag3: AND_3 port map (A => A0 , B => A1 ,c => E, Y => Y3);

end Struct;
```

Architecture of 3 to 8 decoder:

```
architecture Struct of Decoder_3x8 is
   component AND_2 is
   port (A, B: in std_logic; Y: out std_logic);
  end component ;

  component INVERTER is
    port (A: in std_logic; Y: out std_logic);
   end component ;

  component Decoder_2x4  is
    port (A0,A1,E: in std_logic; Y0,Y1,Y2,Y3: out std_logic);
  end component ;

  signal tA2,tE0,tE1: std_logic;
```

```
begin
  ng: INVERTER port map (A=> A2, Y=> tA2);
   ag0: AND_2 port map (A => tA2, B => E, Y => tE0);
   ag1: AND_2 port map (A => A2, B => E, Y => tE1);
   dec2x4_0: Decoder_2x4 port map (A0 => A0, A1 =>A1, E => tE0, Y0 =>
Y0, Y1 => Y1, Y2 => Y2, Y3 => Y3);
  dec2x4_1: Decoder_2x4 port map (A0 => A0, A1 =>A1, E => tE1, Y0 => Y4,
Y1 => Y5, Y2 => Y6, Y3 => Y7);
end Struct;
```

Architecture of Full Adder:

```
architecture Struct of Full_Adder is
  component Decoder_3x8  is
    port (A0,A1,A2,E: in std_logic; Y0,Y1,Y2,Y3,Y4,Y5,Y6,Y7: out
std_logic);
  end component;

  component OR_4 is
   port (A, B, C, D: in std_logic; Y: out std_logic);
  end component ;

  signal tY1,tY2,tY3,tY4,tY5,tY6,tY7: std_logic;
begin

  dec3x8: Decoder_3x8 port map(A0 => A, A1 => B, A2 =>C, E => '1', Y1 =>
tY1 ,Y2 => tY2,Y3 => tY3,Y4 => tY4,Y5 => tY5,Y6 => tY6 ,Y7 => tY7);

  oSum: OR_4 port map (A => tY1, B => tY2 , C => tY4, D => tY7,Y => S );
  oCout: OR_4 port map (A => tY3, B => tY5, C => tY6, D => tY7,Y =>
Cout);
end Struct;
```

RTL View:

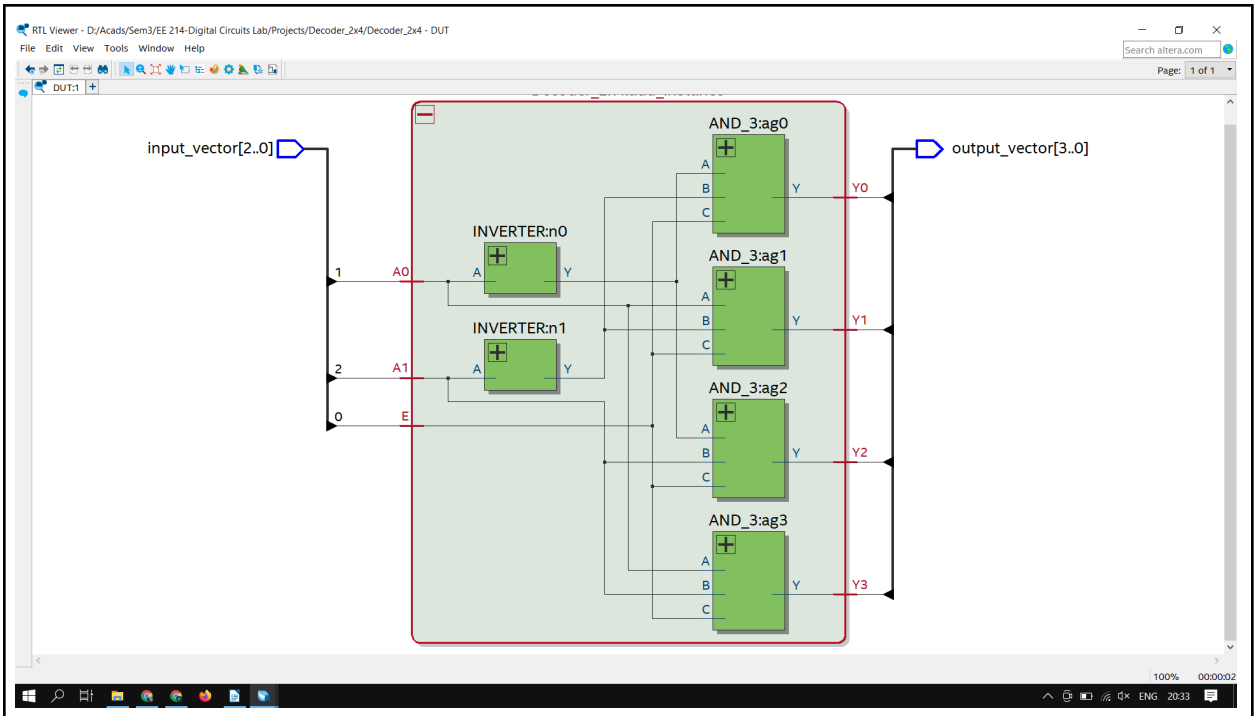


Fig: 2 to 4 Decoder RTL View.

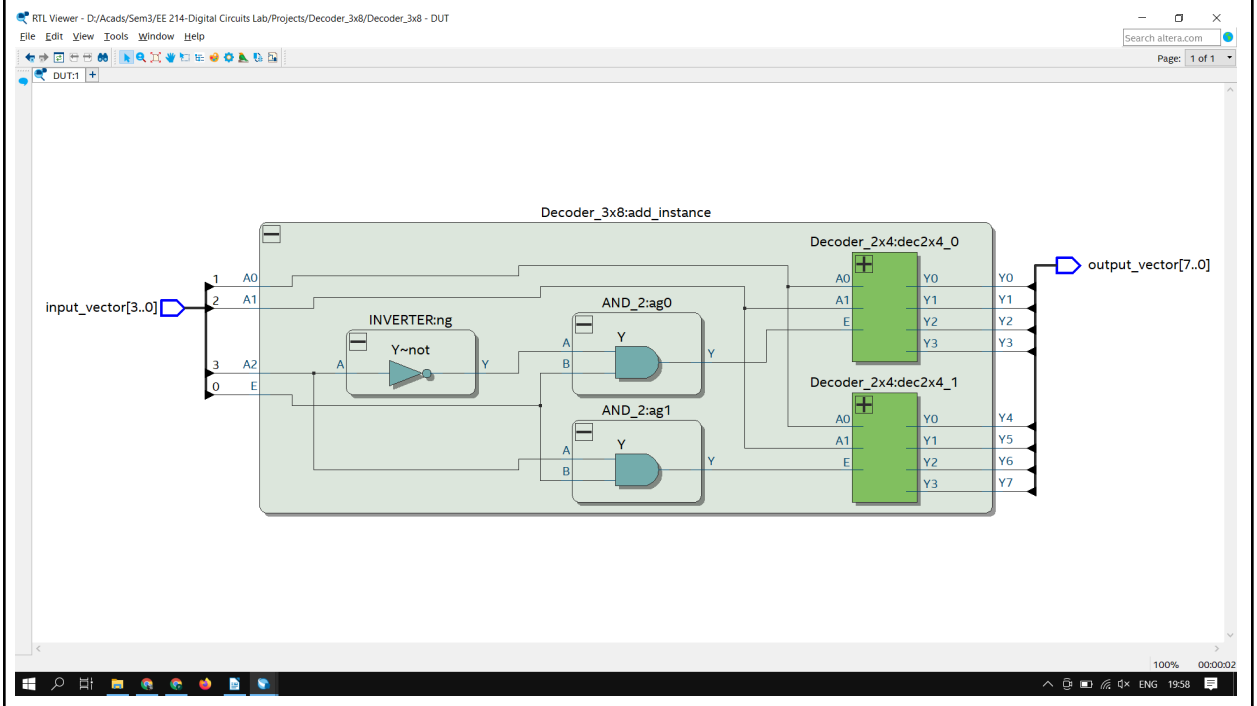


Fig: 3 to 8 Decoder RTL View.

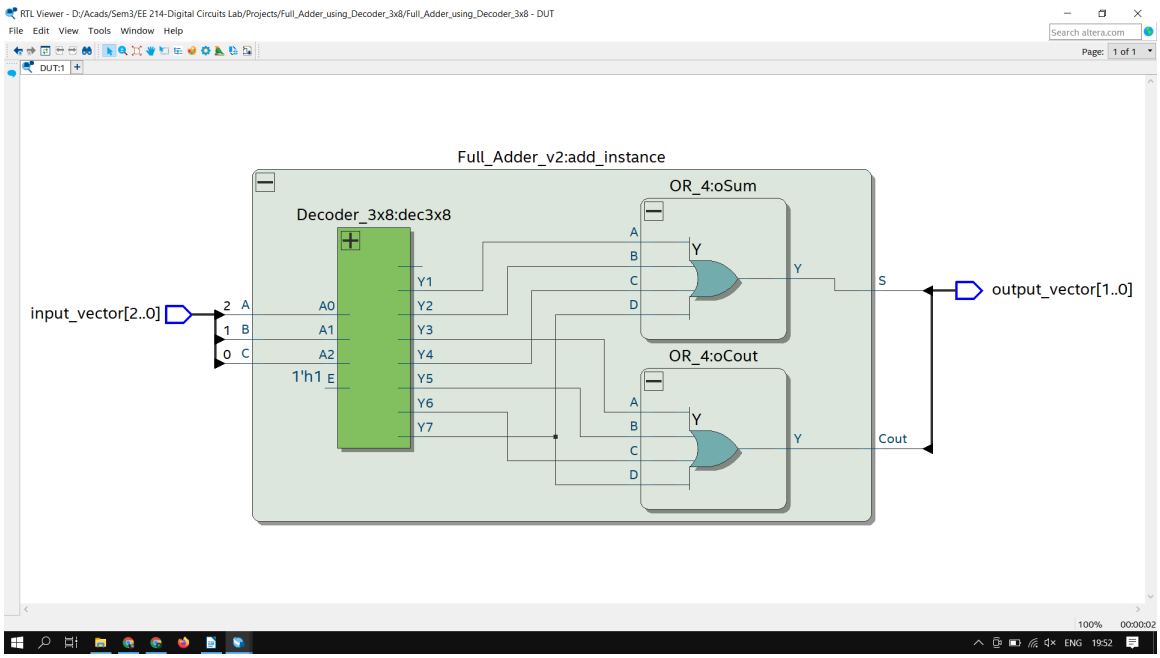


Fig: Full Adder using 3 to 8 Decoder RTL View.

DUT Input/Output Format:

2 to 4 Decoder:

```
                        --input map
                        A1 => input_vector(2),
                        A0 => input_vector(1),
                        E => input_vector(0),
                        --output map
                        Y3 => output_vector(3),
                        Y2 => output_vector(2),
                        Y1 => output_vector(1),
                        Y0 => output_vector(0)
Some test cases from TRACEFILE:
TRACEFILE format
<A1 A0><E> <Y3 Y2 Y1 Y0> 1111

000 0000 1111
001 0001 1111
010 0000 1111
011 0010 1111
100 0000 1111
101 0100 1111
110 0000 1111
111 1000 1111
```

3 to 8 Decoder:

```
--input map
A2 => input_vector(3),
A1 => input_vector(2),
A0 => input_vector(1),
E => input_vector(0),
--output map
Y7 => output_vector(7),
Y6 => output_vector(6),
Y5 => output_vector(5),
Y4 => output_vector(4),
Y3 => output_vector(3),
Y2 => output_vector(2),
Y1 => output_vector(1),
Y0 => output_vector(0)
```

Some test cases from TRACEFILE:
TRACEFILE format:
<A2 A1 A0><E> <Y7 Y6 Y5 Y4 Y3 Y2 Y1 Y0> 11111111

```
1001 00010000 11111111
1010 00000000 11111111
1011 00100000 11111111
1100 00000000 11111111
1101 01000000 11111111
1110 00000000 11111111
1111 10000000 11111111
```

Full Adder:

```
--input map
A => input_vector(2),
B => input_vector(1),
C => input_vector(0),
--output map
S => output_vector(1),
Cout => output_vector(0)
```

Some test cases from TRACEFILE:
TRACEFILE format:
<A B Cin> <S Cout> 11

```
100 10 11
101 01 11
110 01 11
111 11 11
```

Truth Table for 2 to 4 decoder:

A1	A0	E	Y3	Y2	Y1	Y0
0	0	0	0	0	0	0
0	0	1	0	0	0	1
0	1	0	0	0	0	0
0	1	1	0	0	1	0
1	0	0	0	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	0	0
1	1	1	1	0	0	0

Truth Table for 3 to 8 decoder:

A2	A1	A0	E	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	o	0	0	0	1
0	0	1	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	1	0	0	0	o
1	0	1	0	0	0	0	0	0	0	0	0
1	0	1	1	0	0	1	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0
1	1	0	1	0	1	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0

Truth Table for Full Adder:

A	B	Cin	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

RTL Simulation:

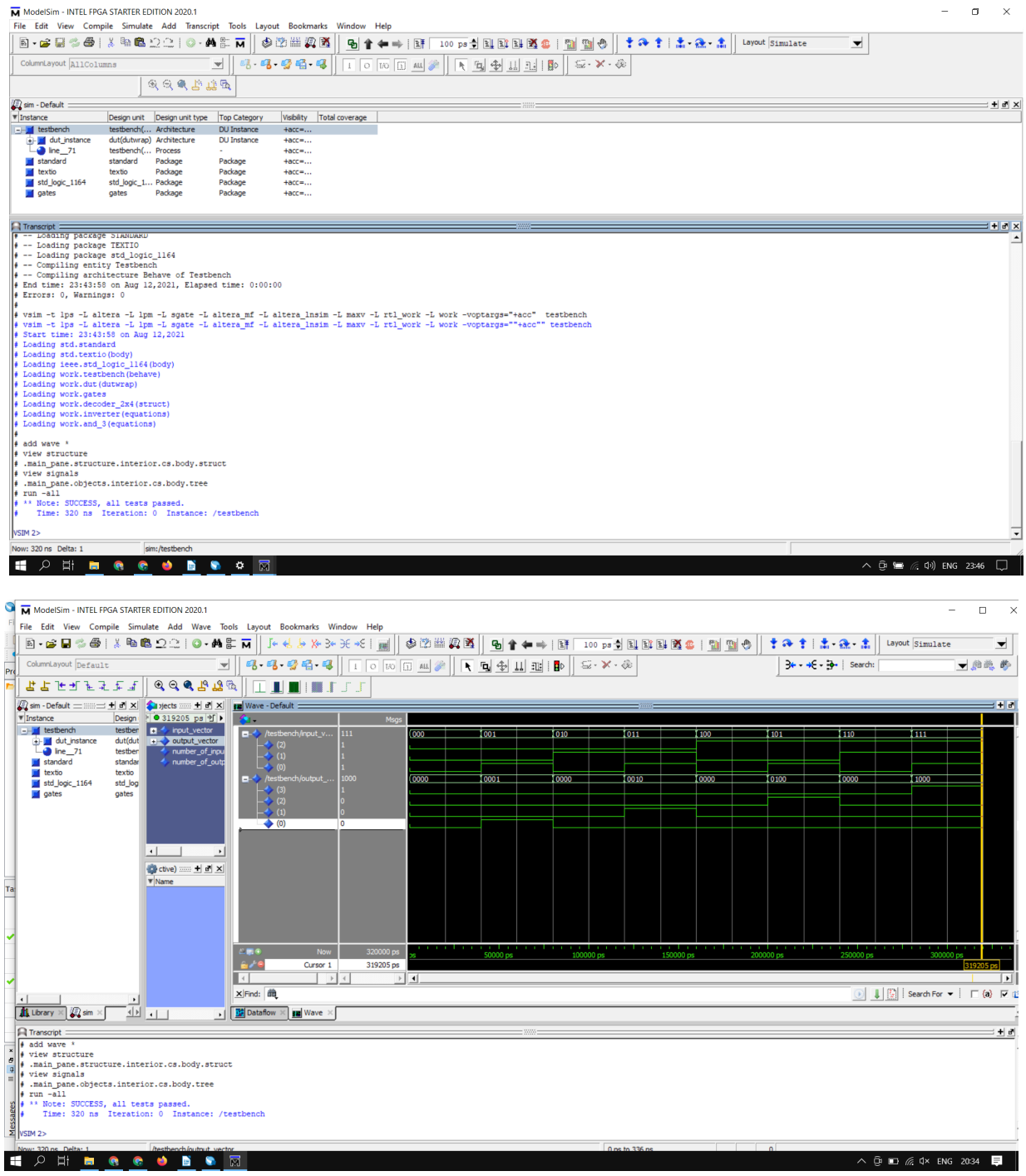
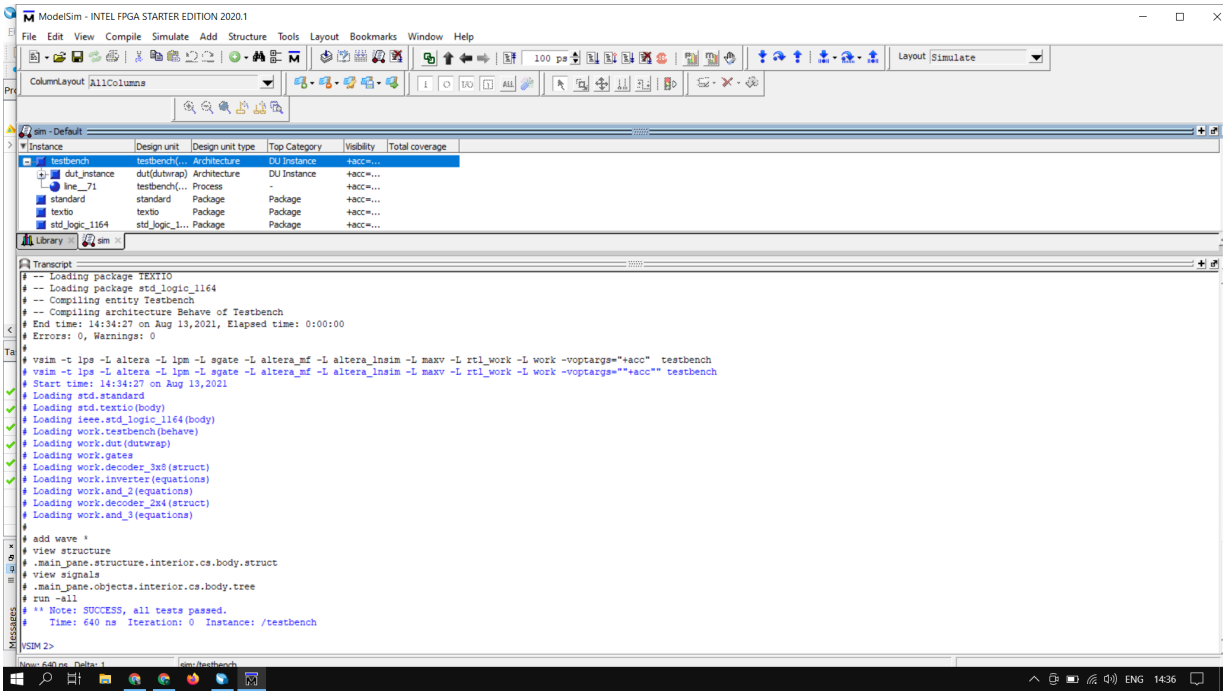


Fig: 2 to 4 Decoder RTL Simulation



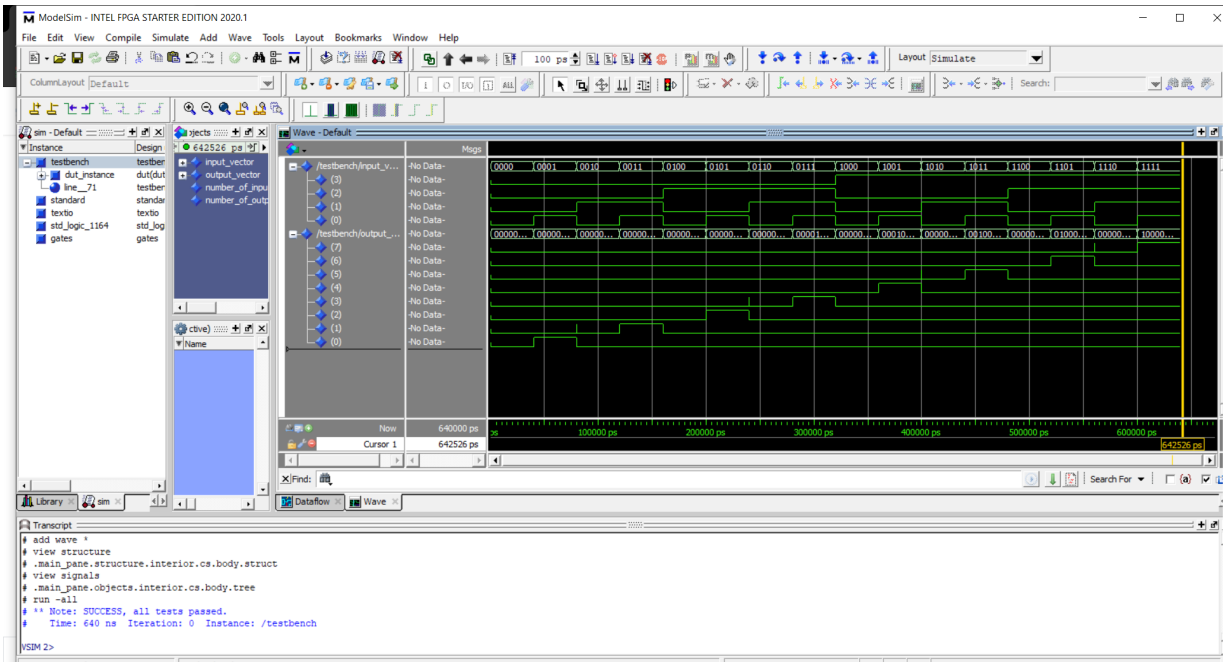


Fig: 3 to 8 Decoder RTL Simulation

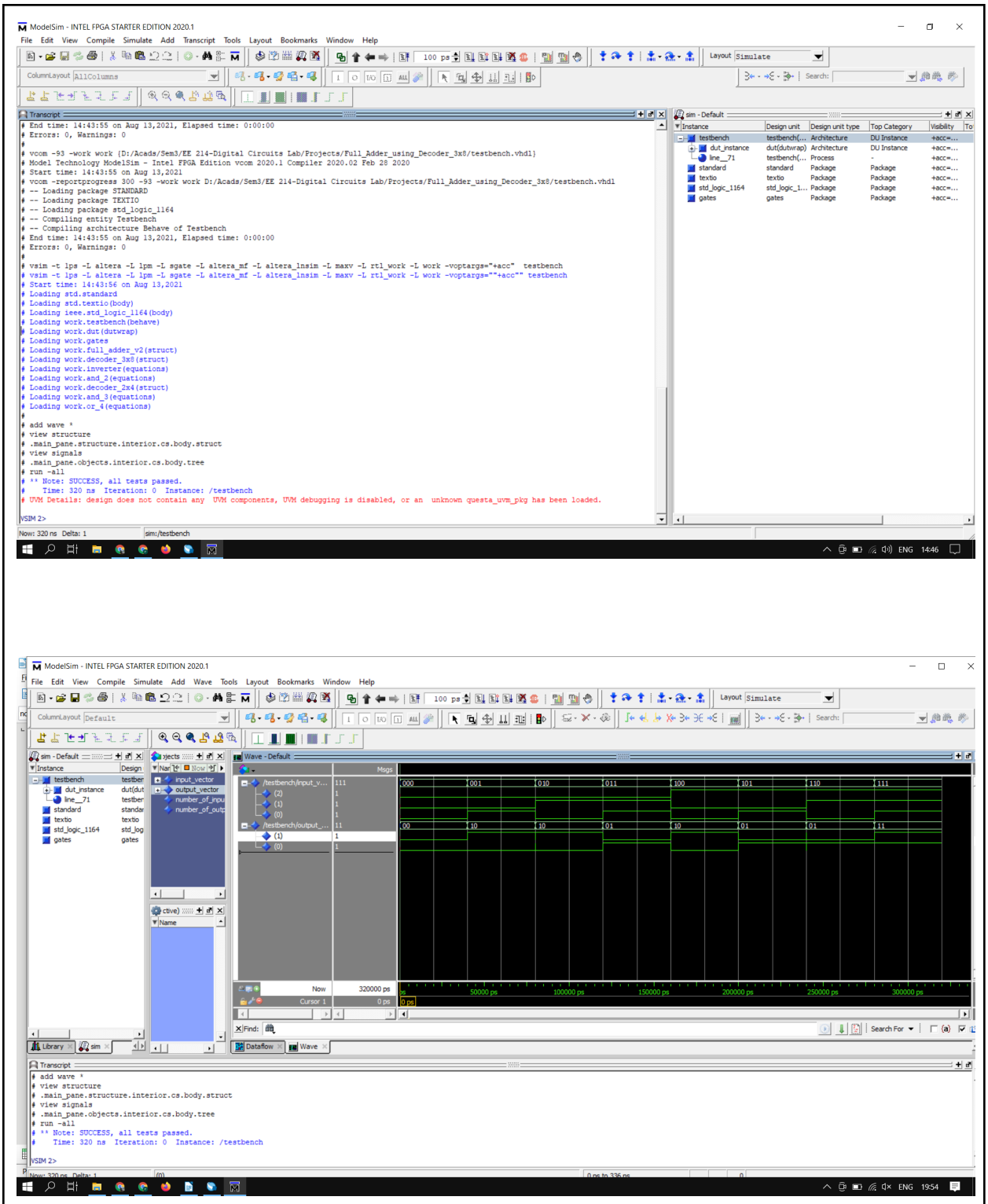


Fig: Full Adder using 3 to 8 Decoder RTL Simulation.

Gate-level Simulation:

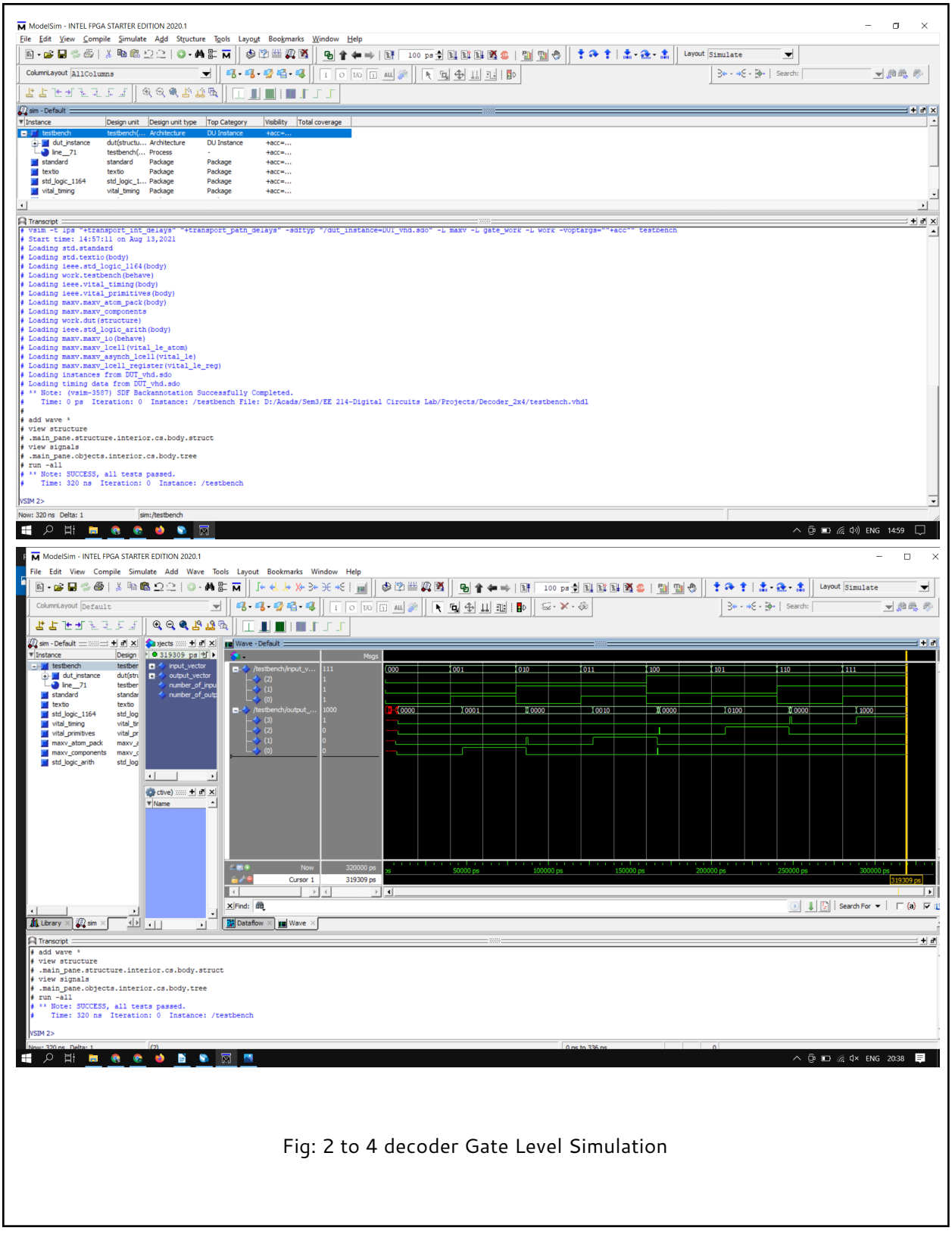


Fig: 2 to 4 decoder Gate Level Simulation

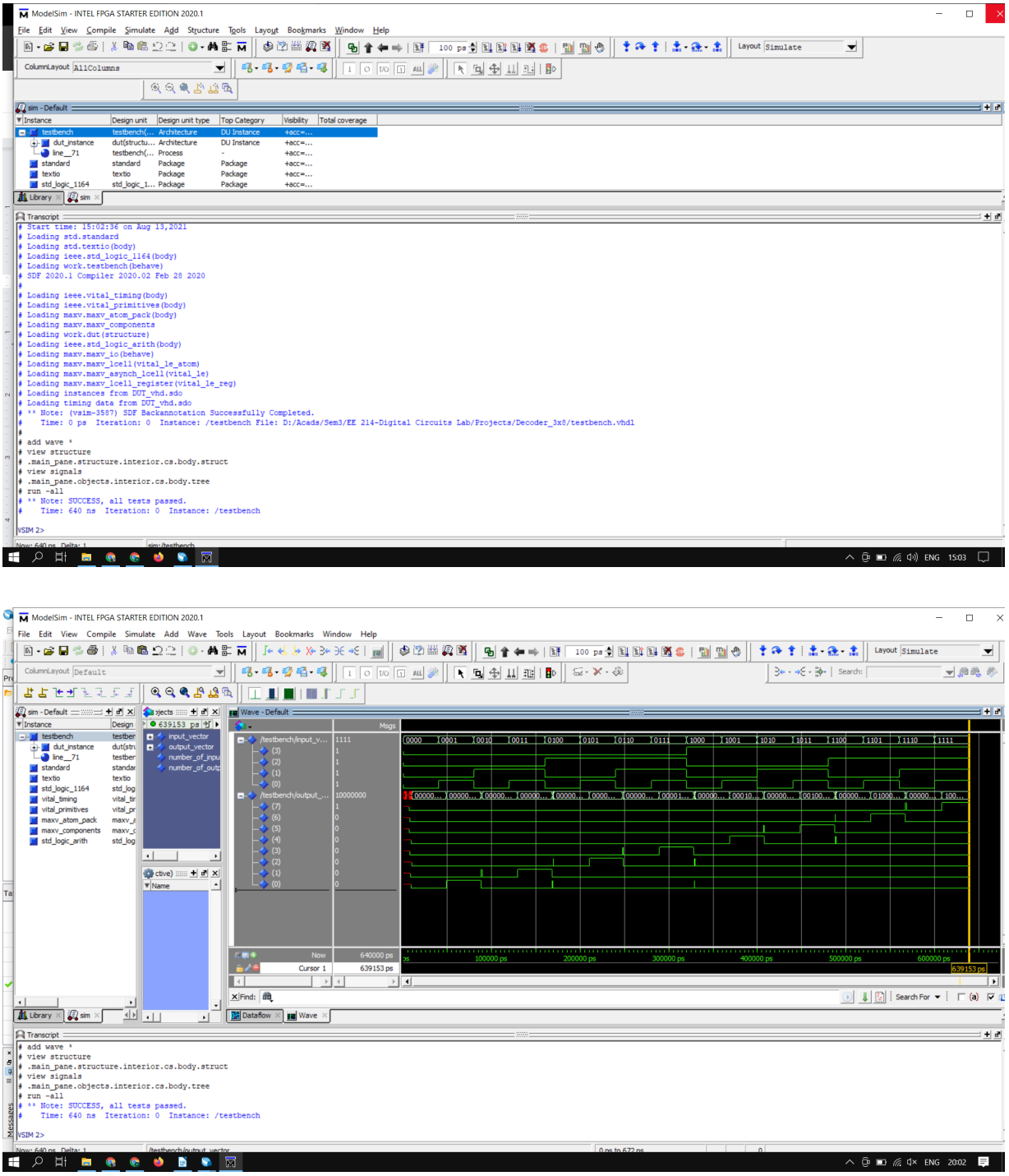


Fig: 3 to 8 decoder Gate Level Simulation

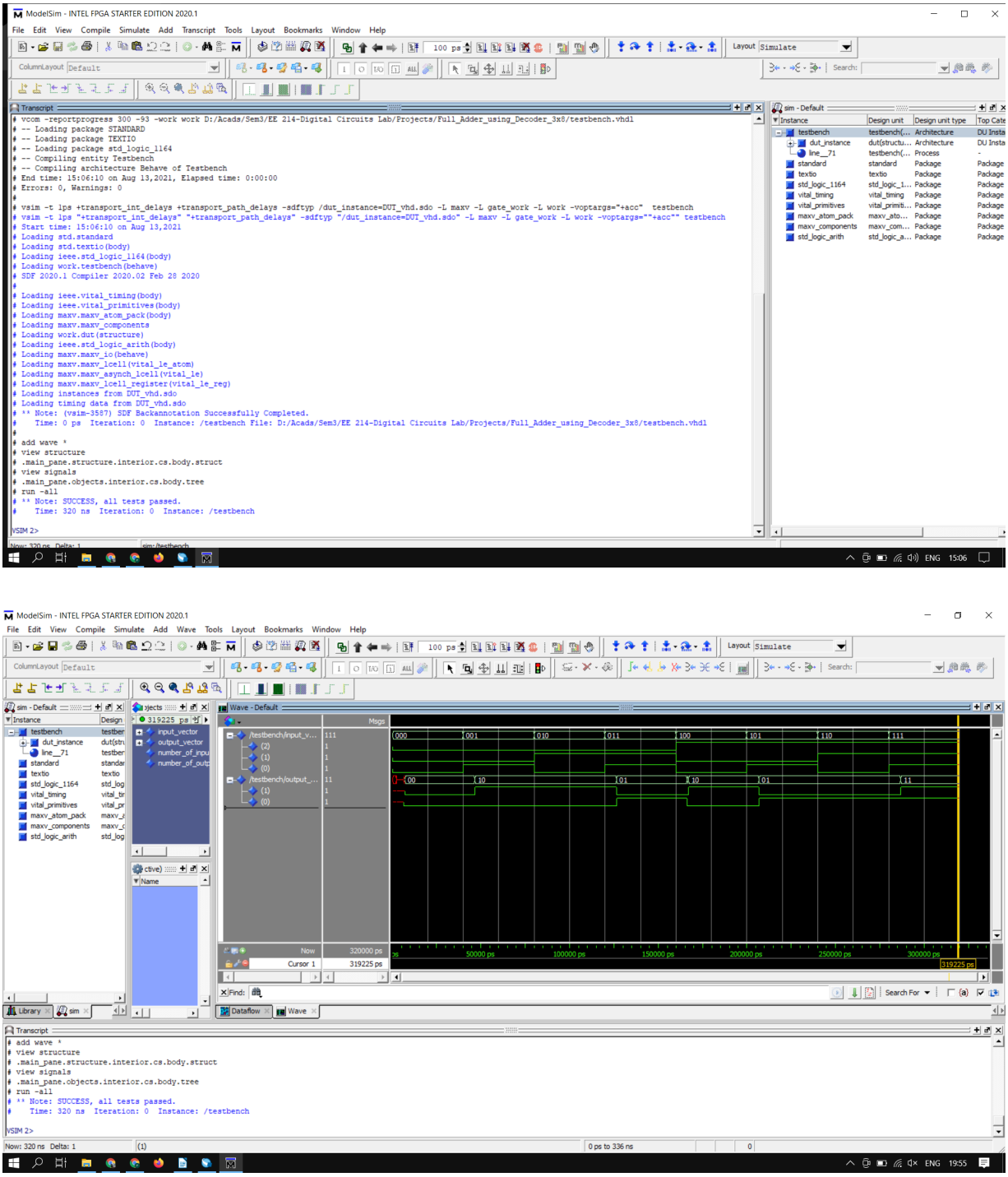


Fig: Full Adder using 3 to 8 Decoder Gate Level Simulation

References:

1. J.F.Wakerly: Digital Design, Principles and Practices,4th Edition,Pearson Education, 2005