Digital Design with FPGA-ELA

Lab Assignment 1

Name : Shreyas S Bagi **Roll No. : 21MVD0086**

Subject Code: ECE5017 **LAB Batch : L5+L6**

Objective:

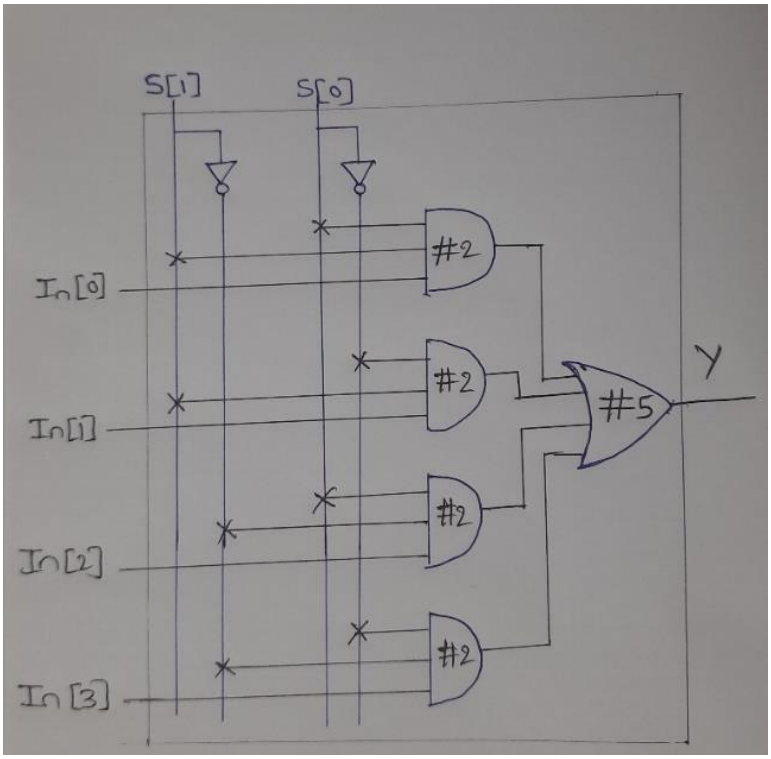
- To design the given digital circuit using Verilog HDL
- To write the test bench for generating stimulus
- To perform functional simulation
- Identify logic gate primitives provided in Verilog
- Understand instantiation of gates, gate symbols, and truth tables.
- Describe rise, fall and turn-off delays in the gate-level design
- Describe the continuous assignment (assign) statement.
- Define expressions, operators, and operands

Software Details:

For design Functional Simulation: Modelsim-INTEL FPGA STARTER EDITION

1. Write a Verilog code in Gate Level Modelling for the logic shown below:

The Circuit Diagram :



Gate Level Verilog Code for Multiplexer:

```
`timescale 10ns/1ns
module gate_da_1 (In,S,Y);
output Y;
input [3:0] In; // Mux input line
input [1:0] S; // 2 bit Select line
wire S1_bar,S0_bar,t1,t2,t3,t4;
not G1 (S0_bar,S[0]); // Generating S0_bar
not G2 (S1_bar,S[1]); // Generating S1_bar
and #(2) G3 (t1,S[0],S[1],In[0]);
// gate_type #(delay of gate) gate_inst (out, in1, in2, ……..);
// Since trise=tfall=turnoff delay all are same since only one type of delay is present.
and #(2) G4 (t2,S0_bar,S[1],In[1]);
and #(2) G5 (t3,S[0],S1_bar,In[2]);
and #(2) G6 (t4,S0_bar,S1_bar,In[3]);
or #(5) G7 (Y,t1,t2,t3,t4);
endmodule
```

Test Bench Verilog Code:

```
module gate_da1_test();
reg [3:0] Input;
reg [1:0] Sel;
wire Out;
gate_da_1 G8 (.In(Input),.S(Sel),.Y(Out)); // Port by name Instantiation
// Main Module instance
initial
begin
$monitor($time,"Y=%b,IN[3:0]=%b,Sel[1:0]=%b",Out,Input,Sel);
// $monitor is event-based print statement. It continuously monitors the values of the variables
//or signals specified in the parameter list and displays all parameters in the list.
// Providing Test inputs to the Main module
#2 Sel=2'bxx; Input=4'b1111;
#2 Input = 4'b1000; Sel=2'b00;
#2 Input = 4'b0111; Sel=2'b00;
#2 Input = 4'b0100; Sel =2'b01;
#2 Input = 4'b0000; Sel =2'b10;
#2 Input = 4'b1111; Sel =2'b11;
#2 Input = 4'b0010; Sel =2'b10;
```

```
#2 Input = 4'b0100; Sel =2'b01;
#2 Input = 4'b1000; Sel =2'b00;
#5 Input = 4'b0111; Sel =2'b01;
#5 Input = 4'b0110; Sel =2'b11;
#7 Input = 4'b0110; Sel =2'b10;
#8 $stop; // Stop simulation at this point
end
endmodule
```

Simulation Wave form:

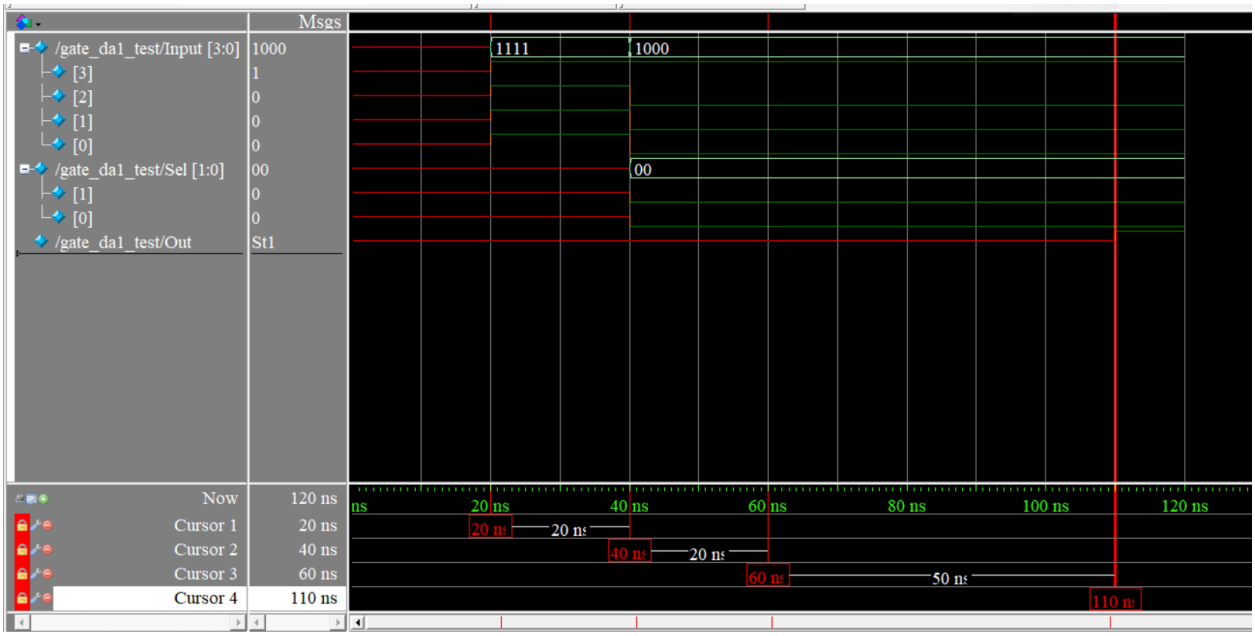


Figure 1.1 Output of Mux for Input =4 'b1000 and Select = 2'b00. The output is delayed by 70ns.

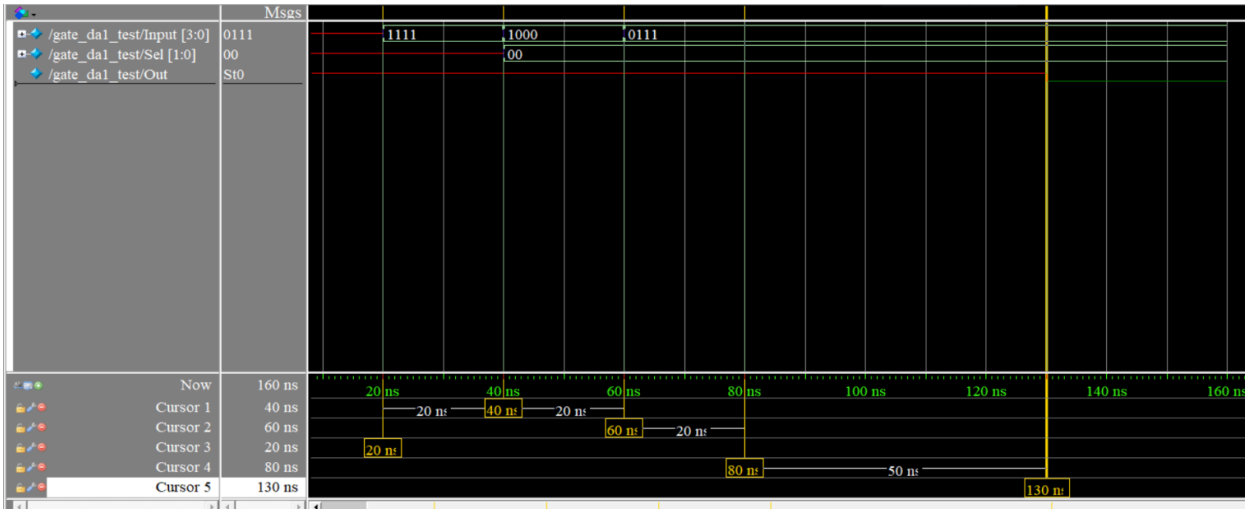


Figure 1.2 Output of Mux for Input =4 'b1000 and Select = 2'b00. The output is delayed by 70ns.

If there is any change in the select line then output line gets updated. Since, and gate has 2 units of time delay and or gate has 5 units of time delay. The output gets updated after 7 units of time delay. In Figure 1.2 we can observe at time=40ns, input=4'b1000 and select line=2'b00 and output should be HIGH at 110ns but, the input is getting changed after 20ns due to this output is getting LOW and gets updated after a delay of 70ns.

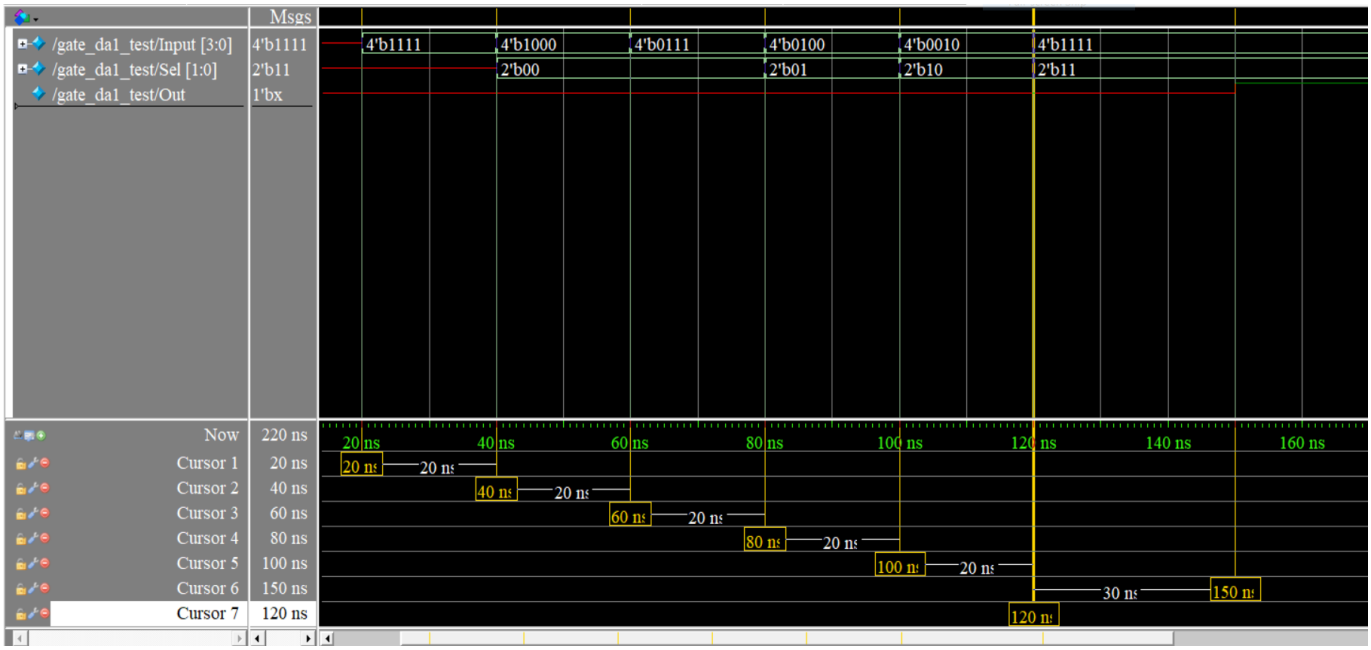


Figure 1.3 Output of Mux for different combination of input and select line.

In the Figure 1.3 we observe that at time=80ns,100ns,120ns in all these the select line selects I[2], I[1] and I[0] which are held HIGH. Since, there is no updating of the output hence output is at HIGH. There is 30ns delay because it is the delay of OR gate. If we observe the ckt-diagram and gates all are in parallel and output of these and gate is connected to OR gate with 5 units of time delay.

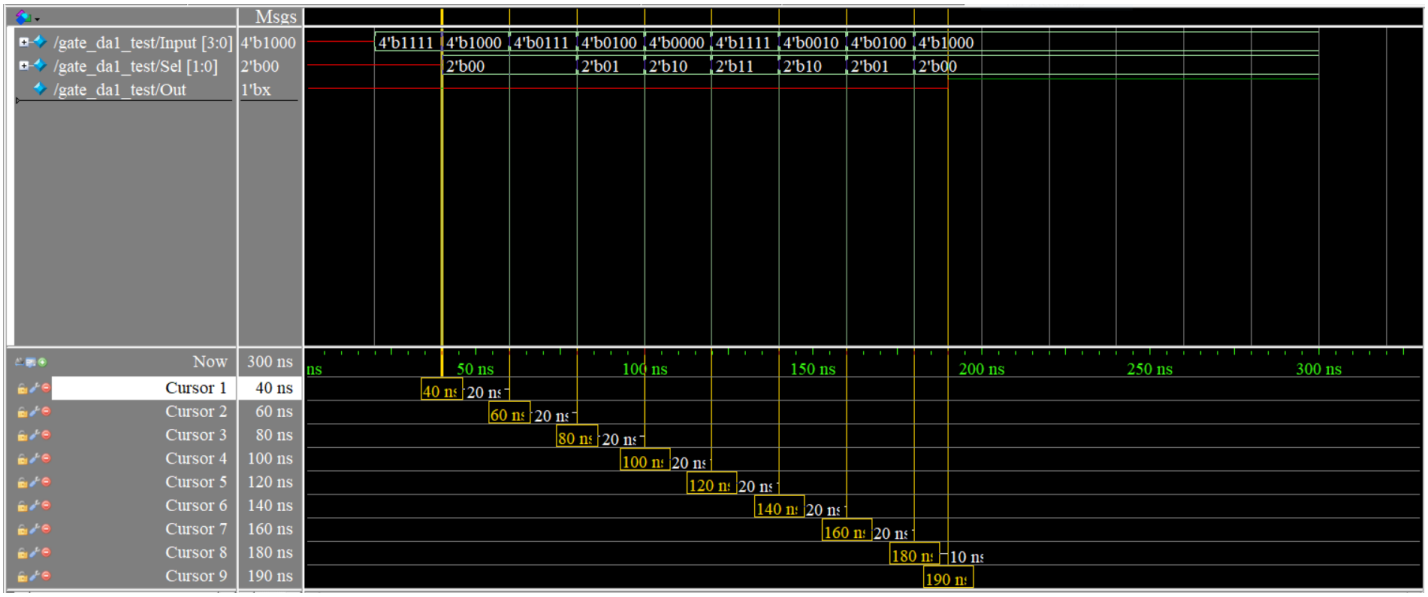


Figure 1.4 Output of Mux for different combination of input and select line.

In the Figure 1.4 we observe that at time=80ns,100ns,120ns,140ns,160ns and 180ns in all these the select line selects I[2], I[1], I[0], I[1], I[2] and I[3] some are HIGH and some are LOW. Since, there is updating of the output at 120ns and all the input lines remain HIGH hence output becomes HIGH at 190ns.

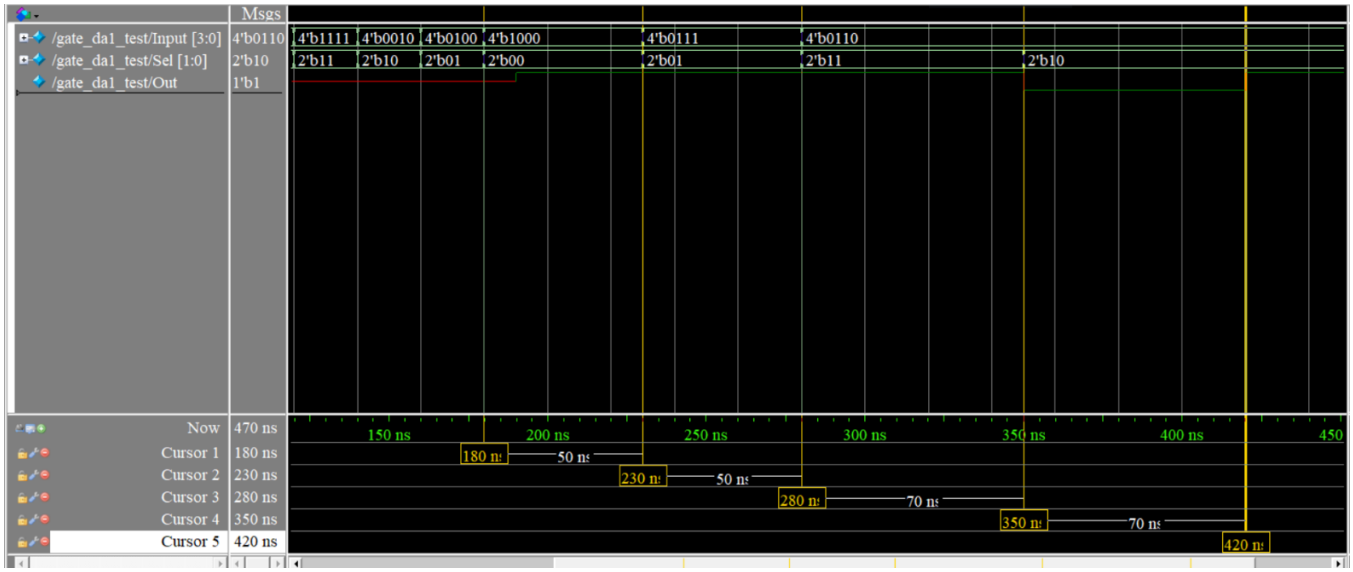


Figure 1.5 Output of Mux for different combination of input and select line.

In the Figure 1.5 we observe that if there is any update in output line then there is delay of 7unit of time (70ns). In the waveform from 280ns to 350ns we see the output goes from HIGH to LOW.

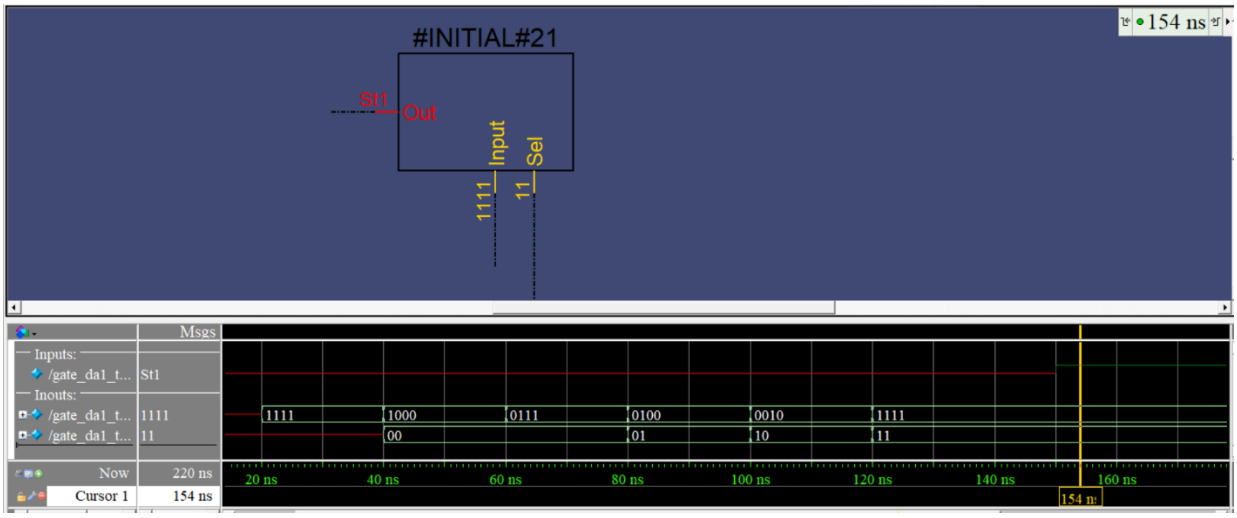


Figure 1.6 When we add the signal to Dataflow window we could see Mux type of realization.

Conclusion

Gate level modelling is a low level of abstraction. The module is implemented in terms of logic gates and interconnection between these gates. Design at this level is like describing a design in terms of a gate-level logic diagram.

2. Design a 3-bit Magnitude comparator in Dataflow Modelling

Dataflow Modelling of 3-bit Magnitude comparator

Verilog Code:

```
`timescale 10ns/1ns
module gate2_da2(A,B,L,G,E);
input wire [2:0] A,B;
output wire L,G,E;
assign E =( (A[2]==B[2])&(A[1]==B[1])&(A[0]==B[0])); // Comparing A is equal to B
// Comparing A is less than B
assign L =( (A[2]<B[2])|((A[2]==B[2])&(A[1]<B[1]))|((A[1]==B[1])&(A[0]<B[0])));
// Comparing A is greater than B
assign G = ((A[2]>B[2])|((A[2]==B[2])&(A[1]>B[1]))|((A[1]==B[1])&(A[0]>B[0])));
/* Multiple line comment Other way to define the same logic using
assign E = ( (A[2]~^B[2])&(A[1] ~^B[1])&(A[0] ~^B[0]));
assign L =( ((!A[2])&B[2])|((A[2]~^B[2])&((!A[1])&B[1]))|((A[1]~^B[1])&((!A[0])&B[0]));
assign G =( (A[2]&(!B[2]))|((A[2]~^B[2])&(A[1]&(!B[1])))|((A[1]~^B[1])&(A[0])&(!B[0])));
*/
endmodule
```

Testbench Module

```
module gate2_da2_test();
reg [2:0] A,B;
wire L,G,E;
gate2_da2 G4 (A,B,L,G,E);
```

```
initial
begin
$monitor($time, “A=%b,B=%b,L=%b,G=%b,E=%b",A,B,L,G,E);
#2 A=4'bx;B=4'bx;
#2 A=4;B=4;
#2 A=3;B=3;
#2 A=7;B=7;
#2 A=3'bx; B=3'bx;
#2
#2 A=7;B=4;
#2 A=4;B=3;
#2 A=5;B=2;
#2 A=3'bx; B=3'bx;
#2
#2 A=4;B=7;
#2 A=1;B=6;
#2 A=2;B=5;
#5
#5 $stop;
end
endmodule
```

Simulation Waveform

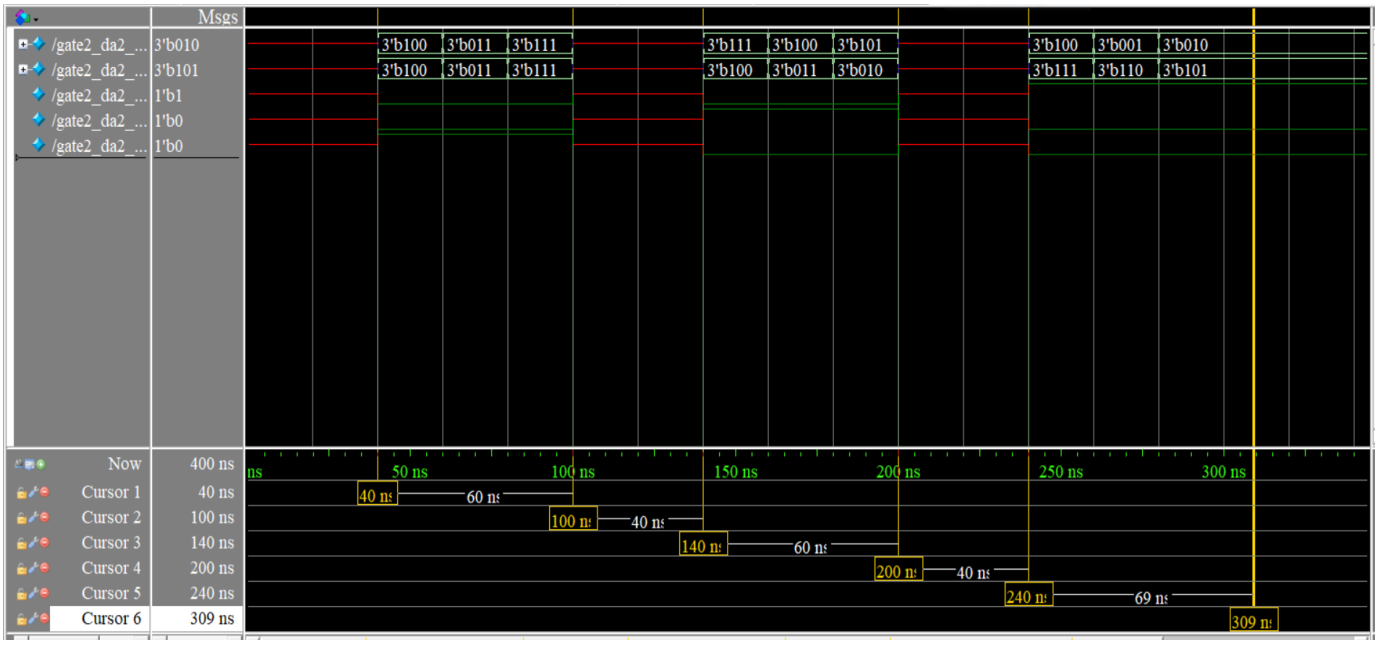
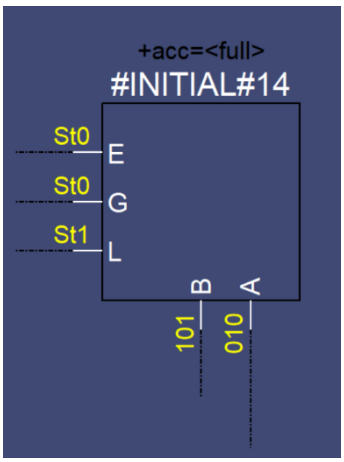


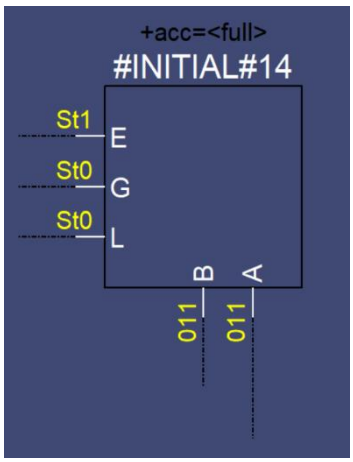
Figure 1.7 The output lines are Less(L), Greater(G) and Equality(E).

Based on the 3-bit input lines the 3-bit comparator compares between A and B. If the two input are equal, then output E is set to HIGH. If input A is less than B then L is set to HIGH and, A is greater than B then G is set to HIGH.

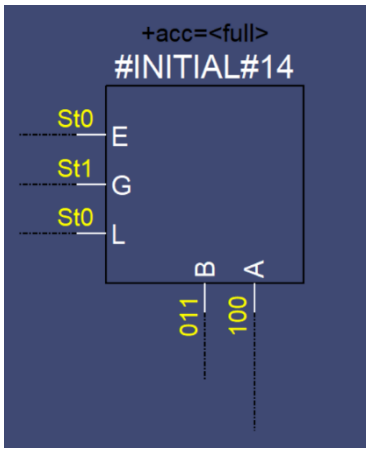
In the below figures we can observe output in Dataflow window



A < B



A=B



A>B

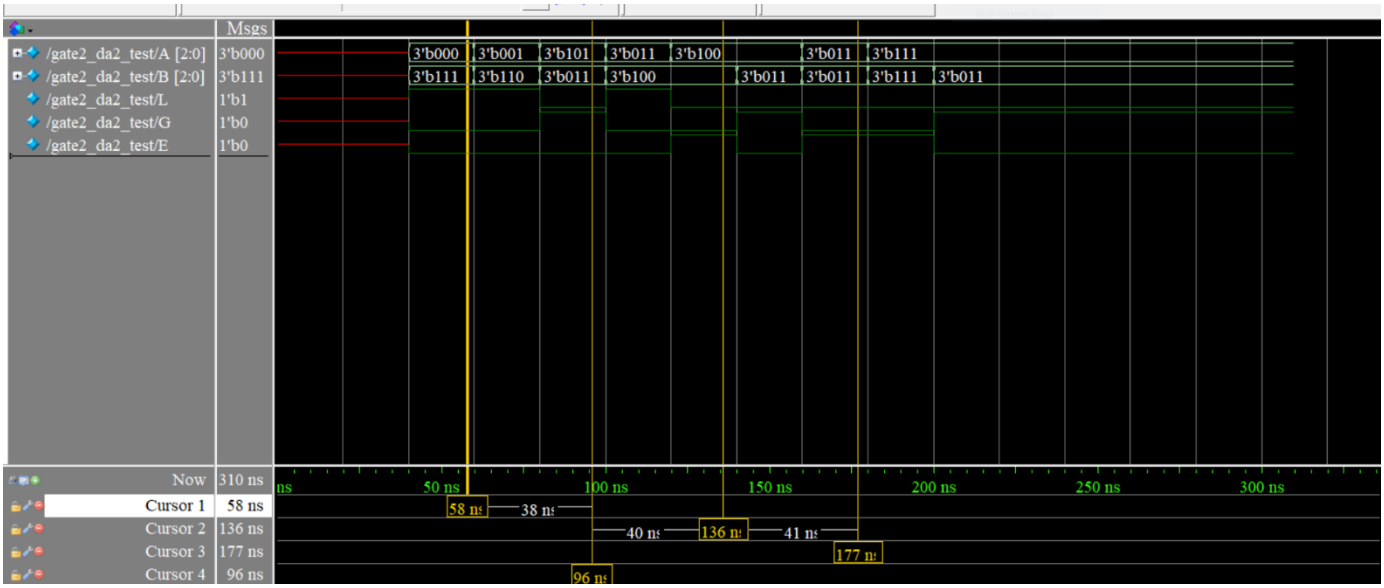


Figure 1.8 The output lines Less(L), Greater(G) and Equality(E) for different combination of input.

Conclusion

Dataflow modelling is second highest level of abstraction. The module is implementing the function at a level of abstraction higher than gate level. It provides a powerful way to implement a design. Verilog allows a circuit to be designed in terms of data flow between registers and how a design processes data rather than instantiation of individual gates.