VLSI LABORATORY [EC39004]
LAB REPORT - 1

Kundrapu N R Sai Akash [20EC10043]
Nagisetti Rithihas [20EC10049]

OBJECTIVE:

1) Design a 3:8 decoder(with enable input) using 2:4 decoders (with enable input).Write using both Behavioral and Structural styles. Demonstrate through test bench waveform, that each of the combination of inputs gives the correct output.

2) Implement a full adder using 2 half adders. Demonstrate the output of the full adder using test bench waveform.

CODES:

1) *Behavioural Code for 3 to 8 Decoder.*

```
module decoder2to4B(Y,A,OE);

    input[1:0] A;
    input OE;
    output[3:0] Y;

    assign Y[0] = OE & ~A[1] & ~A[0];
    assign Y[1] = OE & ~A[1] &  A[0];
    assign Y[2] = OE &  A[1] & ~A[0];
    assign Y[3] = OE &  A[1] &  A[0];

endmodule

module decoder3to8B(Y,A);

    input[2:0] A;
    output[7:0] Y;

    decoder2to4B dec1 (Y[3:0],A[1:0],~A[2]);
    decoder2to4B dec2 (Y[7:4],A[1:0], A[2]);

endmodule
```

2) *Structural Code for 3 to 8 Decoder.*

```
module decoder2to4S(Y,A,OE);

    input[1:0] A;
    input OE;
    output[3:0] Y;
    wire[1:0] nA;

    not (nA[0],A[0]);
    not (nA[1],A[1]);

    and (Y[0],OE,nA[0],nA[1]);
    and (Y[1],OE, A[0],nA[1]);
    and (Y[2],OE,nA[0], A[1]);
    and (Y[3],OE, A[0], A[1]);

endmodule
```

```
module decoder3to8S(Y,A);

    input[2:0] A;
    output[7:0] Y;
    wire nA;

    not (nA,A[2]);

    decoder2to4S dec1 (Y[3:0],A[1:0],nA);
    decoder2to4S dec2 (Y[7:4],A[1:0],A[2]);

endmodule
```

3) Behavioural Code for Full Adder.

```
module half_adder(S,C,A,B);

    input A,B;
    output S,C;

    assign S = A^B;
    assign C = A&B;

endmodule

module full_adderB(S,Cout,Cin,A,B);

    input Cin,A,B;
    output S,Cout;

    wire t1,t2,t3;

    half_adder ha1 (t1,t2,A,B);
    half_adder ha2 (S,t3,Cin,t1);
    assign Cout = t2|t3;

endmodule
```

4) Structural Code for Full Adder.

```
module half_adder(S,C,A,B);

    input A,B;
    output S,C;
```

```
    xor (S,A,B);
    and (C,A,B);

endmodule

module full_adderS(S,Cout,Cin,A,B);

    input Cin,A,B;
    output S,Cout;

    wire t1,t2,t3;

    half_adder ha1 (t1,t2,A,B);
    half_adder ha2 (S,t3,Cin,t1);
    or (Cout,t2,t3);

endmodule
```

5) Testbench for 3 to 8 decoder.

```
module decoder3to8B_tb;

    reg[2:0] A;
    wire[7:0] Y;
    decoder3to8B dut(Y,A);

    initial
       begin
              $dumpfile("decoder3to8B.vcd");
              $dumpvars;
       end

    initial
       begin
              $monitor("Time=%5d,%b,%b,%b,%b", $time, A[2], A[1], A[0],
Y);
              #2 A=3'b000;
              #2 A=3'b001;
              #2 A=3'b010;
              #2 A=3'b011;
              #2 A=3'b100;
              #2 A=3'b101;
              #2 A=3'b110;
              #2 A=3'b111;
       end

endmodule
```

6) *Testbench for Full Adder.*

```
module full_adderB_tb;

    reg A,B,Cin;
    wire S,Cout;
    full_adderB dut(S,Cout,Cin,A,B);

       initial
       begin
              $dumpfile("full_adderB.vcd");
              $dumpvars;
       end

    initial
    begin
              $monitor("Time=%5d,%b,%b,%b,%b,%b", $time, A, B, Cin, S,
Cout);
              #2 A=0;B=0;Cin=0;
              #2 A=0;B=1;Cin=0;
              #2 A=1;B=0;Cin=0;
              #2 A=1;B=1;Cin=0;
              #2 A=0;B=0;Cin=1;
              #2 A=0;B=1;Cin=1;
              #2 A=1;B=0;Cin=1;
              #2 A=1;B=1;Cin=1;
    end

endmodule
```

RUNTIME OUTPUTS:

Time= $time, A[2], A[1], A[0], Y

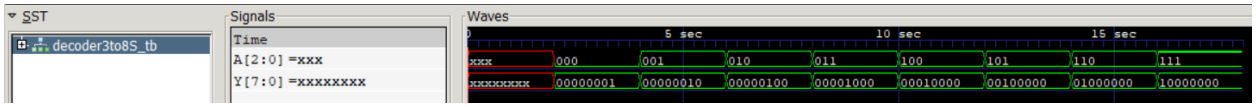
```
PS C:\IIT KGP\SEM 6\Verilog\Assignment_1> iverilog -o decoder3to8B decoder3to8B.v decoder3to8B_tb.v
PS C:\IIT KGP\SEM 6\Verilog\Assignment_1> vvp decoder3to8B
VCD info: dumpfile decoder3to8B.vcd opened for output.
Time=    0,x,x,x,xxxxxxxx
Time=    2,0,0,0,00000001
Time=    4,0,0,1,00000010
Time=    6,0,1,0,00000100
Time=    8,0,1,1,00001000
Time=   10,1,0,0,00010000
Time=   12,1,0,1,00100000
Time=   14,1,1,0,01000000
Time=   16,1,1,1,10000000
```

Time= $time, A, B, Cin, S, Cout

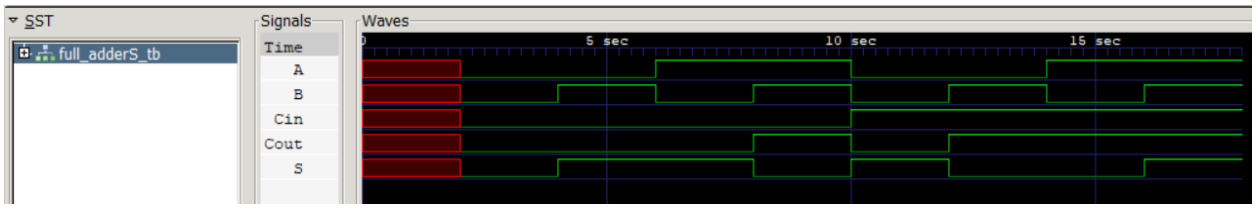
```
PS C:\IIT KGP\SEM 6\Verilog\Assignment_1> iverilog -o full_adderS full_adderS.v full_adderS_tb.v
PS C:\IIT KGP\SEM 6\Verilog\Assignment_1> vvp full_adderS
VCD info: dumpfile full_adderS.vcd opened for output.
Time=    0,x,x,x,x,x
Time=    2,0,0,0,0,0
Time=    4,0,1,0,1,0
Time=    6,1,0,0,1,0
Time=    8,1,1,0,0,1
Time=   10,0,0,1,1,0
Time=   12,0,1,1,0,1
Time=   14,1,0,1,0,1
Time=   16,1,1,1,1,1
```

GTK Waveform:

For 3to8 decoder.



For Full Adder.



Discussion:

- In this first class of the lab, we learnt how to use verilog, writing codes and generating testbench files to check the written code.
- All verilog codes can be written in atleast two ways, behavioural and structural level of implementation.
- Behavioural code gives a brief idea of what is happening in the given circuitry using logical operations.
- Structural code is used to give an idea of how the major elements(Register level) are connected within the circuit.
- In first question, we are asked to generate a code of 3 to 8 decoder using 2 to 4 decoders using enable input. We will need a total of two 2 to 4 decoders for this.
- In the second question, we need to create a full adder using half adders. Even here, we will need two half adders along with some combinational circuit to completely generate a full adder.