Brac University

EEE 412/ ECE 412/ CSE 460

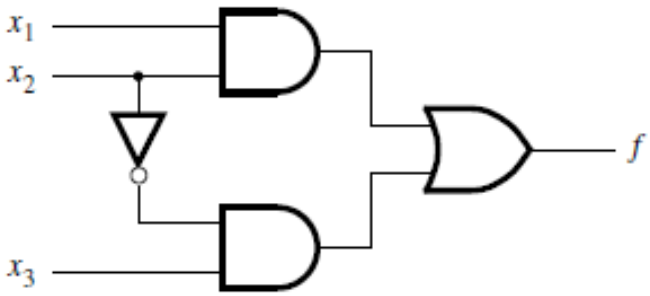
Verilog Design Laboratory

Experiment Title: Introduction to Verilog HDL and familiarization with Altera Quartus as simulation tool for Verilog HDL codes

Verilog: Verilog was originally intended for simulation and verification of digital circuits. Subsequently, with the addition of synthesis capability, Verilog has also become popular for use in design entry in CAD systems. The CAD tools are used to synthesize the Verilog code into a hardware implementation of the described circuit.

Verilog allows the designer to represent circuits in two fundamentally different ways. One possibility is to use Verilog constructs that represent simple circuit elements such as logic gates or even transistors. A larger circuit is defined by writing code that connects such elements together. This is referred to as the **structural representation** of logic circuits. The second possibility is to describe a circuit by using logic expressions and programming constructs that define the behavior of the circuit but not its actual structure in terms of gates. This is called the **behavioral representation**.

Example 1: Construct the following circuit using Verilog HDL and verify the output through timing diagram.



Truth Table

x3	x2	x1	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

Structural Representation:

```
module expt1(f,x1,x2,x3);
input x1,x2,x3;
output f;
and (g,x1,x2);
not (y,x2);
and (h,y,x3);
or (f,g,h);
endmodule
```

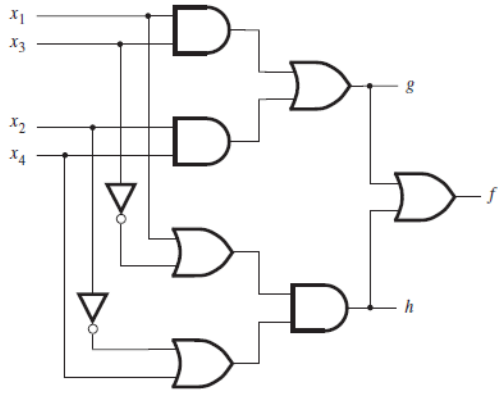
Behavioral Representation1:

```
module expt1(f,x1,x2,x3);
input x1,x2,x3;
output f;
assign f=(x1 & x2) |
(~x2 & x3);
endmodule
```

Behavioral Representation2:

```
module expt1 (f, x1,
x2, x3);
input x1, x2, x3;
output f;
reg f;
always @(x1 or x2 or
x3)
if (x2 == 1)
f = x1;
else
f = x3;
endmodule
```

Example 2: Construct the following circuit using Verilog HDL and verify the output through timing diagram.



Truth Table:

x4	x3	x2	x1	f
0	0	0	0	1
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Structural Representation:

```
module
expt2(f,x1,x2,x3,x4);
input x1,x2,x3,x4;
output f;
and (w1,x1,x3);
and (w2,x2,x4);
or (g,w1,w2);
or (w3,x1,~x3);
or (w4,x4,~x2);
and (h,w3,w4);
or (f, g, h);
endmodule
```

Behavioral Representation:

```
module expt2(f,x1,x2,x3,x4);
input x1,x2,x3,x4;
output f;
assign f= ((x1 & x3) | (x2 & x4)) | ((x1
| ~x3) & (~x2 | x4));
endmodule
```

Class Work: Design a 1-bit full adder and using the 1-bit full adder modules, design a 4-bit full adder using Verilog HDL.

```
module fulladd4(S0,S1,S2,S3,Cout, A0, A1, A2, A3, B0, B1, B2,
B3, Cin);
input A0, A1, A2, A3, B0, B1, B2, B3, Cin;
output S0,S1,S2,S3,Cout;
fulladd stage0 (S0,Cout0,A0,B0,Cin);
```

```
fulladd stage1 (S1,Cout1,A1,B1,Cout0);
fulladd stage2 (S2,Cout2,A2,B2,Cout1);
fulladd stage3 (S3,Cout,A3,B3,Cout2);
endmodule

module fulladd(S,Cout,A,B,Cin);
input A, B, Cin;
output S, Cout;
assign S = A ^ B ^ Cin;
assign Cout = (A & B) | (Cin & (A ^ B));
endmodule
```

Another approach:

```
module fadd (S, A, B, Cin);
input Cin;
input [3:0] A,B;
output [4:0] S;
assign S = A+B+Cin;
endmodule
```

Working with Altera Quartus:

Procedure:

1. Open Quartus.

2. Click File->New Project Wizard->Next

3. Fill out the following:

What is the working directory for this project? – Browse->Create a folder in desktop->Select the folder

What is the name of the project->Type a name and **remember it**(Say “expt1”). Same name is going to be copied to the next box automatically.

Press next.

4. Press next

5. Select Device Family: FLEX10KE and press next.

6. Fill out the following in all three pair of boxes:

Tool name: Custom

Format: Verilog HDL

Press Next->Finish

7. File->New->Verilog HDL File->OK

8. Write the code and save it with the same name as that given in expt1 3 with extension of .v (Example: expt1.v)

9. File->New->Vector Waveform File

10. Right click on Name->Insert->Insert Node or BUS

11. Click Node Finder.

Filter: Pins: all

Look in: Filename (Example expt1.v)

Click list->Click ">>" -> OK -> OK

12. Right click on each input->Value->Clock and set up the clocks.

13. Save with the same filename as the .v file (Example: expt1.vwf)

14. Assignment->Settings->Simulator Settings

Simulation Mode: Functional

Click OK.

15. Processing-> Generate Functional Simulation Netlist

16. Processing-> Start Simulation

Home Work:

1. Design a 4 to 1 MUX using Verilog HDL and verify using timing diagram.

2. Design a priority encoder (3>1>0>2) using Verilog HDL and verify using timing diagram.

3. Design a 4 to 2 Decoder using Verilog HDL and verify using timing diagram.

4. Design a 4 bit adder-subtractor using Verilog HDL and verify using timing diagram.