DIGITAL SYSTEM DESIGN LAB CSE-308L

SEMESTER:6TH

LAB REPORT # 1

Submitted By: *Zainab Khalid*
Registration No:*19PWCSE1743*
Section: A
Submitted to: *Mam Madeeha Sher*

DEPARTMENT OF COMPUTER SYSTEMS ENGINEERING
UNIVERSITY OF ENGINEERING AND TECHNOLOGY PESHAWAR

LAB No 1

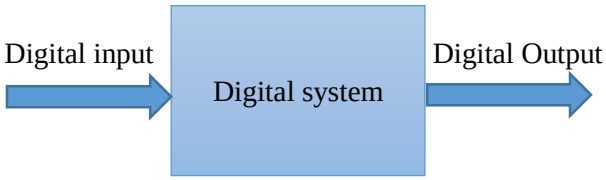
INTRODUCTION TO MODELSIM AND GATE LEVEL MODELING

Objectives:

- Introduction to HDL
- Introduction to MODELSIM
- Introduction to different modeling levels

Digital systems:

A digital circuit is a circuit that operates within two discrete levels (true, false). HDL language helps to describe the functionalities of digital systems. a software language helps to write programs for various applications such as web, mobile, enterprise applications, etc.



Software languages:

Software language is a computer language used to write a set of instructions to allow the CPU to perform a specific task. We can convert these programs into machine language using a compiler or an interpreter. Finally, the CPU can directly execute the machine codes to perform the defined task. Software languages are ***sequential***.

Examples: C, C++, Java, Perl, and Ruby etc.

Hardware languages:

HDL is a specialized computer language used to describe the structure and behavior of electronic circuits, most commonly, digital logic circuits. HDL languages consist of programming structures such as expressions, statements, control structures, etc. HDL is more complex than a software language.

Examples: Verilog and VHDL are common HDL. HDL are ***non-sequential***.

Verilog:

This language is similar to C. It is a case sensitive language that allows real-time simulations. The basic building block of the language is called a module.

VHDL:

Base languages for this language are Ada and Pascal. It is not case sensitive. A hardware module in VHDL is called an entity. Usually, VHDL is complex than Verilog.

Software used:

Software used for working on Verilog are:

- **MODELSIM**
- **XILINX**

MODELSIM:

MODELSIM is a simulator which can be used for the simulations of both VHDL and Verilog HDL. It is used for system-level testing.

XILINX:

The Xilinx is primarily used for circuit synthesis and design. There is no need to write a test bench.

Hardware used:

FPGA

*FPGA stands for **field-programmable gate array**. An FPGA is a hardware circuit that a user can program to carry out logical operations. It is a semiconductor device composed of logic blocks interconnected via programmable connections.*

Stages of Modeling:

- Switch level
- Gate level
- Data flow level
- Behavioral level

Switch level modeling:

Switch level modeling forms the basic level of modeling digital circuits. At the switch level, transistors behave as on-off switches. This is the lowest level of abstraction. The design is described in terms of switching (modeling a transistor).

Gate level modeling:

Designing circuits using basic logic gates is known as gate-level modeling. A digital circuit is implemented using logic gates and interconnections between these gates. Gate level modeling also takes care of gate delays, which is not possible in a higher level of abstraction like behavioral modeling.

Syntax:

```
module gates (  input a, b,  output c, d);
buf (c, a, b);  // c is the output, a and b are inputs
not (d, a, b);  // d is the output, a and b are inputs
endmodule
```

Gate level modeling works best for circuits having a limited number of gates. It allows the designer to instantiate and connect each gate individually

Data flow level modeling:

Dataflow modeling provides the means of describing combinational circuits by their function rather than by their gate structure. Dataflow modeling uses a number of operators that act on operands to produce the desired results. Verilog HDL provides about 30 operator types like &, |, ^, ==, >, <,? etc. Dataflow modeling uses continuous assignments and the keyword ***assign.***

Dataflow modeling describes hardware in terms of the flow of data from input to output. For example, to describe an AND gate using dataflow, the code will look something like this:

Syntax:

```
module and_gate(a,b,out);
input a,b;
output out;
assign out = a&b;
endmodule
```

Behavioral level modeling:

It is higher level of modeling where behavior of logic is modeled. It is used mostly to describe sequential circuits, but can be used to describe combinational circuits.

During simulation of behavioral model, all the flows defined by the ***'always'*** and ***'initial'*** statements start together at simulation time 'zero'. The initial statements are executed once, and the always statements are executed repetitively.

Syntax:

```
// Behavioral description of 2-to-1 line multiplexer
module mux2x1_bh (A,B,select,OUT);
input A,B,select;
output OUT;
reg OUT;
always @(select or A or B)
if (select==1) OUT=A;
else OUT=B;
endmodule
```

Lab tasks:

Task1:

Implement a buffer at the gate level.

Code:

```
 1 module task1(x,y);
 2 input x;
 3 output y;
 4 buf b(y,x);
 5 endmodule
 6
 7 module test();
 8 reg x;
 9 wire y;
10 task1 a1(x,y);
11 initial
12 begin
13 $display("X     Y");
14 x=0;
15 #1 $display("%b  %b",x,y);
16 x=1;
17 #1 $display("%b  %b",x,y);
18 end
19 endmodule
```

Output:

```
# Compile of task1.v was successful.
vsim work.test
# vsim work.test
# Loading work.test
# Loading work.task1
run
# X Y
# 0 0
# 1 1
```

Task2:

Implement an inverter at the gate level.

Code:

```
 1 module inverter(x,y);
 2 input x;
 3 output y;
 4 not i(y,x);
 5 endmodule
 6
 7 module stim_inverter();
 8 reg x;
 9 wire y;
10 inverter i1(x,y);
11 initial
12 begin
13 $display("X     Y");
14 x=0;
15 #1 $display("%b  %b",x,y);
16 x=1;
17 #1 $display("%b  %b",x,y);
18 end
19 endmodule
```

Output:

```
# Compile of task2.v was successful.
vsim work.stim_inverter
# vsim work.stim_inverter
# Loading work.stim_inverter
# Loading work.inverter
run
# X Y
# 0 1
# 1 0
```

Task3:

Implement an OR gate using a NAND gates.

Code:

```
 1 module orusingnand(a,b,out);
 2 input a,b;
 3 output out;
 4 nand(x,a,a);
 5 nand(y,b,b);
 6 nand(out,x,y);
 7 endmodule
 8
 9 module stim_or();
10 reg a,b;
11 wire out;
12 orusingnand n1(a,b,out);
13 initial
14 begin
15 $display("a     b       out");
16 a=0;b=0;
17 #1 $display("%b %b %b",a,b,out);
18 a=0;b=1;
19 #1 $display("%b %b %b",a,b,out);
20 a=1;b=0;
21 #1 $display("%b %b %b",a,b,out);
22 a=1;b=1;
23 #1 $display("%b %b %b",a,b,out);
24 end
25 endmodule
```

Output:

```
# Compile of task3.v was successful.
vsim work.stim_or
# vsim work.stim_or
# Loading work.stim_or
# Loading work.orusingnand
run
# a  b out
# 0 0 0
# 0 1 1
# 1 0 1
# 1 1 1
```

Task4:

Implement the following equation where z is output and x1, x2, x3, x4, and x5 are inputs of the circuit.

z = (y1 + y2)'

y1 = x1.x2

y2 = (x3.x4.x5)

Code:

```
 1 module equations(x1,x2,x3,x4,x5,z);
 2 input x1,x2,x3,x4,x5;
 3 output z;
 4 wire y1,y2,y3;
 5 and (y1,x1,x2);
 6 and (y2,x3,x4,x5);
 7 or (y3,y1,y2);
 8 not (z,y3);
 9 endmodule
10
11 module stim_equations();
12 reg x1,x2,x3,x4,x5;
13 wire z;
14 equations n1(x1,x2,x3,x4,x5,z);
15 initial
16 begin
17 $display("x1 x2 x3 x4 x5 z");
18 x1=0;x2=0;x3=0;x4=0;x5=0;
19 #1 $display("%b %b %b %b %b %b",x1,x2,x3,x4,x5,z);
20 x1=0;x2=0;x3=0;x4=0;x5=1;
21 #1 $display("%b %b %b %b %b %b",x1,x2,x3,x4,x5,z);
22 x1=0;x2=0;x3=0;x4=1;x5=0;
23 #1 $display("%b %b %b %b %b %b",x1,x2,x3,x4,x5,z);
24 x1=0;x2=1;x3=1;x4=1;x5=1;
25 #1 $display("%b %b %b %b %b %b",x1,x2,x3,x4,x5,z);
26
27 end
28 endmodule
```

Output:

```
# Compile of task4.v was successful.
vsim work.stim_equations
# vsim work.stim_equations
# Loading work.stim_equations
# Loading work.equations
run
# x1 x2 x3 x4 x5 z
# 0 0 0 0 0 1
# 0 0 0 0 1 1
# 0 0 0 1 0 1
# 0 1 1 1 1 0
```