**KIIT Deemed to be University**

**School of Electronics Engineering**
**Digital System Design**

# Verilog and Testbench codes for combinational digital circuits

## *1.* A] Verilog code for AND gate:*(using gate level modeling)*

```
module and2(Y, A, B);
        input A, B;
        output Y;
        and (Y, A, B);
endmodule
```

## 1.B] Testbench code for AND gate:

```
module test_and2;
        reg A, B;
        wire Y;
        and2 a2(Y, A, B);
        initial
        begin
                                $dumpfile("dump.vcd");
                                $
                                d
                                u
                                m
                                p
                                v
                                a
                                r
                                s
                                ;
                                A
                                =
                                0
                                ;
                                B
                                =
                                0
        end endmodule
```

;
#
1
0
A
=
0
;
B
=
1
;
#10 A=1; B=0;
#10 A=1; B=1;
#10 $finish;

***NOTE: For other gates, change the module name (both in design file and testbench file) and replace the gate name according to the gate choosen. Only for NOT gate there will be one input and one output.***

## *2.* A] Verilog code for Half Adder:*(using gate level modeling)*

```
module halfadder(output sum, carry, input a,b);
  xor (sum, a, b);
  and (carry, a, b);
endmodule
```

## 2.B] Testbench code for Half Adder:

```
module test_halfadder;
  reg a,b;
  wire sum, carry;
  halfadder ha(sum, carry, a, b);
  initial begin
    $dumpfile("dump.vcd");
    $dumpvars;
    a=0; b=0; #20
    a=0; b=1;
    #20 a=1; b=0;
    #20 a=1; b=1;
    #20 $finish;
end
endmodule
```

### 3.A] Verilog code for 2 to 4 line Decoder:*(using dataflow modeling)*

```
module decoder2_4 (Q0,Q1,Q2,Q3,A,B);
input A, B;
output Q0,Q1,Q2,Q3;
assign Q0= (~A) & (~B);
assign Q1= (~A) & B;
assign Q2= A & (~B);
assign Q3= A & B;
endmodule
```

### 3.B] Testbench code for 2 to 4 line Decoder:

```
module test_decoder2_4;
        reg A, B;
        wire Q0,Q1,Q2,Q3;
        decoder2_4 d24(Q0,Q1,Q2,Q3,A,B);
        initial
        begin
                                        $dumpfile("dump.vcd");
                                        $
                                        d
                                        u
                                        m
                                        p
                                        v
        end endmodule                   a
                                        r
                                        s
                                        ;
                                        A
                                        =
                                        0
                                        ;
                                        B
```

=
0
;
#
1
0
A
=
0

;
B
=
1
;
#10 A=1; B=0;
#10 A=1; B=1;
#10 $finish;

***NOTE:*** ***Symbol of operator used in verilog for AND logic:*** ***&***
***Symbol of operator used in verilog for OR logic:*** ***|***
***Symbol of operator used in verilog for NOT logic:*** ***~***
***Symbol of operator used in verilog for XOR logic:*** ***^***

**4.A] Verilog code for 2x1 Multiplexer:*(using gate level modeling)***

```
module mux2_1(output F, input I0,I1,S);
  wire w1, w2, w3;
  nand (F, w2, w3);
  nand (w2, I1, S);
  nand (w3, w1, I0);
  nand (w1, S, S);
endmodule
```

**4.B] Testbench code for 2x1 Multiplexer:**

```
module test_mux2_1;
  reg I0,I1,S;
  wire F;
  mux2_1 m21(F,I0,I1,S);
  initial begin
    $dumpfile("dump.vcd");
    $dumpvars;
    S=0;I1=0;I0=0; #20
    S=0;I1=0;I0=1;
    #20 S=0;I1=1;I0=0;
    #20 S=1;I1=0;I0=1;
    #20 S=1;I1=1;I0=0;
    #20 $finish;
end
endmodule
```