KIIT Deemed to be University

School of Electronics Engineering
Digital System Design

Verilog and Testbench codes for sequential digital circuits

1. A] Verilog code for JK Flipflop:*(using behavioral modeling)*

```
module jkff(input reset, input clk, input j, input k, output reg q, output qnot);
assign qnot=~q;
  always @(negedge clk)
 begin
  if (reset) q<=1'b0;
  else
  case ({j, k})
   2'b00: q<=q;
   2'b01: q<=1'b0;
   2'b10: q<=1'b1;
   2'b11: q<=~q;
  endcase
 end
endmodule
```

1.B] Testbench code for JK Flipflop:

```
module test;
reg clk=0;
reg j=0;
reg k=0;
reg
reset;
wire q, qnot;
jkff dut(reset, clk,j,k,q,qnot);
initial
  begin
    $dumpfile("dump.vcd");
    $dumpvars;
        reset=1'b1;
#10 reset=1'b0;
        j=1'b0;k=1'b1;
#20 reset=1'b0;
        j=1'b0; k=1'b0;
#20 reset=1'b0;
        j=1'b1; k=1'b0;
#20 reset=1'b0;
        j=1'b1; k=1'b1;
#40 $finish;
  end
always #5 clk=~clk;
```

```
endmodule
```

2. A] Verilog code for D Flipflop:*(using behavioral modeling)*

```
module dff(input reset, input clk, input d, output reg q, output qnot);
assign qnot=~q;
  always @(posedge clk)
  begin
    if (reset) begin
      q <= 1'b0;
    end
    else begin
    q <= d;
    end
  end
endmodule
```

2.B] Testbench code for D Flipflop:

```
module test_dff;
reg clk=0;
reg d=0;
reg reset;
wire q, qnot;
dff dut(reset, clk,d,q,qnot);
initial
  begin
    $dumpfile("dump.vcd");
    $dumpvars;
     reset=1'b1;
#10 reset=1'b0;
    d=1'b0;
#10 reset=1'b0;
    d=1'b1;
#20 $finish;
  end
always #5 clk=~clk;
endmodule
```

3.A] Verilog code for 2-bit synchronous counter:*(using structural modeling similar to gate level modeling)*

Note: Write the code of JK Flipflop first, then write the following code in design window

```
module twobitcounter(j,k, clk, reset, q_out, qbar_out);
   input [1:0] j; input [1:0] k; input clk; input reset;
   output wire [1:0] q_out;
   output wire [1:0] qbar_out;
   assign qbar_out[0] = ~q_out[0];
   assign j[1] = q_out[0];
   assign k[1] = q_out[0];
  jkff M1(reset,clk, j[0], k[0], q_out[0], qbar_out[0]);
  jkff M2(reset,clk, j[1], k[1], q_out[1], qbar_out[1]);
endmodule
```

3.B] Testbench code for 2-bit synchronous counter:

```
module test;
reg clk=0;
reg j=1;
reg k=1;
reg reset;
wire [1:0] q_out, qbar_out;
twobitcounter dut(j,k, clk,reset, q_out, qbar_out);
initial
  begin
    $dumpfile("dump.vcd");
    $dumpvars;
     reset=1'b1;
#15 reset=1'b0;
#60 $finish;
  end
always #5 clk=~clk;
endmodule
```

4.A] Verilog code for 2-bit asynchronous counter:*(using structural modeling)*

Note: Write the code of JK Flipflop first, then write the following code in design window

```
module twobitasyncounter(j,k, clk, reset, q_out, qbar_out);
   input [1:0] j; input [1:0] k; input clk; input reset;
   output wire [1:0] q_out;
   output wire [1:0] qbar_out;
  jkff M1(reset,clk, j[0], k[0], q_out[0], qbar_out[0]);
  jkff M2(reset,q_out[0], j[1], k[1], q_out[1], qbar_out[1]);
endmodule
```

4.B] Testbench code for 2-bit asynchronous counter:

```
module test;
reg clk=0;
reg [1:0] j, k;
reg reset;
wire [1:0] q_out, qbar_out;
twobitasyncounter dut(j,k, clk,reset, q_out, qbar_out);
initial
  begin
   $dumpfile("dump.vcd");
   $dumpvars;
   reset=1'b1;
#25 reset=1'b0;
   j[0]=1;
   k[0]=1;
   j[1]=1;
   k[1]=1;
#90 reset = 1;
#20 $finish;
  end
always #5 clk=~clk;
endmodule
```

5.A] Verilog code for 4-bit SIPO Register:*(using structural modeling)*

Note: Write the code of D Flipflop first, then write the following code in design window

```
module siporeg(din, clk, reset, q0, q1, q2, q3, qb0, qb1, qb2, qb3);
  input din; input clk; input reset;
  output wire q0, q1, q2, q3, qb0, qb1, qb2, qb3;
  dff M1(reset,clk, din, q0, qb0);
  dff M2(reset,clk, q0, q1, qb1);
dff  M3(reset,clk,  q1,  q2,  qb2);
dff  M4(reset,clk,  q2,  q3,  qb3);
endmodule
```

5.B] Testbench code for 4-bit SIPO Register:

```
module test;
reg clk=0;
reg din;
reg reset;
wire q0, q1, q2, q3, qb0, qb1, qb2, qb3;
siporeg dut(din, clk,reset, q0, q1, q2, q3, qb0, qb1, qb2, qb3);
initial
  begin
    $dumpfile("dump.vcd");
    $dumpvars;
    reset=1'b1;din=0;
#25 reset=1'b0;
    din=0;
    #10 din=1;
    #10 din=0;
    #50 $finish;
  end
always #5 clk=~clk;
endmodule
```