Assignment - 3 Report

Member 1: Shekhar Shrivas
Member 2: Nagmani Kumar

Q1. Write Verilog code for a mod 16 synchronous counter along with the test bench.

CODE:

```
module jk_flipflop (
    input J, K, clk, reset,
    output reg Q
);
always @(posedge clk or posedge reset) begin
    if (reset)
        Q <= 1'b0;
    else begin
        case ({J, K})
            2'b00: Q <= Q;
            2'b01: Q <= 1'b0;
            2'b10: Q <= 1'b1;
            2'b11: Q <= ~Q;
        endcase
    end
end
endmodule
```

mod_16_counter.v ●

Question_1 > mod_16_counter.v

```
1   `include "jk_flipflop.v"
2   module mod_16_counter (
3       input wire clk,
4       input wire reset,
5       output reg [3:0] count
6   );
7   reg jk_in, jk_out;
8   reg [3:0] temp;
9   jk_flipflop jk_ff_0 (
10      .J(1'b1),
11      .K(1'b0),
12      .clk(clk),
13      .reset(reset),
14      .Q(jk_out)
15  );
16  assign jk_in = jk_out;
17  genvar i;
18  generate
19      for (i = 1; i < 4; i = i + 1) begin : jk_ff_loop
20          jk_flipflop jk_ff (
21              .J(jk_in),
22              .K(1'b0),
23              .clk(clk),
24              .reset(reset),
25              .Q(jk_out)
26          );
27          assign jk_in = jk_out;
28      end
```

mod_16_counter.v ●

Question_1 > mod_16_counter.v

```
18    generate
28        end
29    endgenerate
30    always @(*) begin
31        case (jk_out)
32            4'b0000: temp = 4'b0000;
33            4'b0001: temp = 4'b0001;
34            4'b0011: temp = 4'b0011;
35            4'b0010: temp = 4'b0010;
36            4'b0110: temp = 4'b0100;
37            4'b0111: temp = 4'b0101;
38            4'b0101: temp = 4'b0111;
39            4'b0100: temp = 4'b0110;
40            4'b1100: temp = 4'b1000;
41            4'b1101: temp = 4'b1001;
42            4'b1111: temp = 4'b1011;
43            4'b1110: temp = 4'b1010;
44            4'b1010: temp = 4'b1100;
45            4'b1011: temp = 4'b1101;
46            4'b1001: temp = 4'b1111;
47            4'b1000: temp = 4'b1110;
48            default: temp = 4'b0000;
49        endcase
50    end
51    always @(posedge clk or posedge reset) begin
52        if (reset)
53            count <= 4'b0000;
54        else
55            count <= temp + 1;
56    end
57
58    endmodule
```

```
`include "mod_16_counter.v"
module mod_16_counter_tb;
reg clk;
reg reset;
wire [3:0] count;
mod_16_counter mod_16_counter_inst(
    .clk(clk),
    .reset(reset),
    .count(count)
);
always #5 clk = ~clk;
initial begin
    clk = 0;
    reset = 1;
    #10 reset = 0;
end
always @(posedge clk)
begin
    $display("Count: %d", count);
end
initial #50 $finish;
endmodule
```

It is giving some error so we didn’t understand the error after a lot of effort.

Q2. Write Verilog code for a 4-bit universal shift register along with the test bench. Instead of
writing the entire code into a single file, you have to create modules for the flip-flops and call them
into your main file.

CODE:

```
module D_flipflop (
    input D, clk, reset,
    output reg Q
);
always @(posedge clk or posedge reset) begin
    if (reset)
        Q <= 1'b0;
    else
        Q <= D;
end
endmodule
```

mod_16_counter.v | shift_register.v ●

Question_2 > shift_register.v

```
1   `include "D_flipflop.v"
2   module shift_register (
3       input wire [3:0] data_in,
4       input wire shift_left,
5       input wire shift_right,
6       input wire clk,
7       input wire reset,
8       output reg [3:0] data_out
9   );
10  reg [3:0] temp;
11  D_flipflop DFF0 (
12      .D(data_in[0]),
13      .clk(clk),
14      .reset(reset),
15      .Q(data_out[0])
16  );
17  D_flipflop DFF1 (
18      .D(shift_left ? data_out[0] : data_in[1]),
19      .clk(clk),
20      .reset(reset),
21      .Q(data_out[1])
22  );
23  D_flipflop DFF2 (
24      .D(shift_left ? data_out[1] : data_in[2]),
25      .clk(clk),
26      .reset(reset),
27      .Q(data_out[2])
28  );
29  D_flipflop DFF3 (
30      .D(shift_left ? data_out[2] : data_in[3]),
31      .clk(clk),
32      .reset(reset),
```

```
29  D_flipflop DFF3 (
30      .D(shift_left ? data_out[2] : data_in[3]),
31      .clk(clk),
32      .reset(reset),
33      .Q(data_out[3])
34  );
35  always @(*) begin
36      if (shift_right)
37          temp = {data_in[3], data_in[0], data_in[1], da
38      else
39          temp = data_out;
40  end
41  always @(posedge clk or posedge reset) begin
42      if (reset)
43          data_out <= 4'b0000;
44      else if (shift_right || shift_left)
45          data_out <= temp;
46  end
47  endmodule
48
49
50
```

mod_16_counter.v | shift_register.v | shift_register_tb.v

Question_2 > shift_register_tb.v

```v
`include "shift_register.v"
module shift_register_tb;
reg [3:0] data_in;
reg shift_left;
reg shift_right;
reg clk;
reg reset;
wire [3:0] data_out;
shift_register shift_register_inst (
    .data_in(data_in),
    .shift_left(shift_left),
    .shift_right(shift_right),
    .clk(clk),
    .reset(reset),
    .data_out(data_out)
);
always #5 clk = ~clk;
initial begin
    data_in = 4'b0000;
    shift_left = 0;
    shift_right = 0;
    clk = 0;
    reset = 1;
    #10 reset = 0;
    #20 shift_left = 1;
    #30 shift_right = 1;
end
always @(posedge clk)
begin
    $display("data_out: %b", data_out);
end
initial #50 $finish;
endmodule
```