Digital Design and Computer Organization Laboratory

UE22CS251A

3rd Semester, Academic Year 2023

Date:

Name of team members:	SRN of team members:	Section
Arushi Katta	PES1UG22CS109	B
Archisha Janawade	PES1UG22CS105	B
Arasada Lakshmi Sai Haneela	PES1UG22CS104	B

Title of mini project:

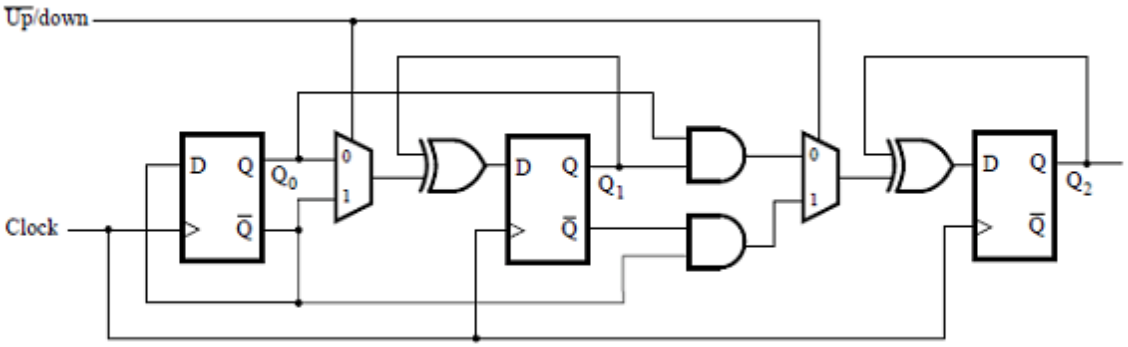
3 BIT UP AND DOWN COUNTER

Abstract:

This paper presents the design and implementation of a 3 bit up counter and down counter using Verilog. Counters are essential components in digital systems, used for various applications such as frequency division, sequencing, and control. A synchronous design approach is employed, ensuring reliable and predictable operation.

Circuit Diagram:

The provided Verilog code does not explicitly specify the flip flops used. Instead, it uses the procedural block ‘always @(posedge clk or negedge rst)’ to describe the behavior at the positive edge of the clock and negative edge of reset. The default assumption in digital design is that edge triggered D flip flops are used.



Screenshot of code (Design File):

UP COUNTER:

```
module counter(cnt,clk,rst);
input clk,rst;
output [2:0]cnt;

reg [2:0]cnt;
wire [2:0]next_cnt;

assign next_cnt = cnt + 1'b1;  //Just increment by 1

always @ (posedge clk or negedge rst)
begin
 if(!rst)
 begin
 cnt <= 3'b0;
 end
 else
 begin
 cnt <= next_cnt;
 end
end

endmodule
```

DOWN COUNTER:

```
module downcounter (
    input clk,
    input rst,
    output reg [2:0] cnt
);

    wire [2:0] next_cnt;

    assign next_cnt = cnt - 1'b1;  // Decrement by 1

    always @(posedge clk or negedge rst) begin
        if (!rst) begin
            cnt <= 3'b0;
        end else begin
            cnt <= next_cnt;
        end
    end

endmodule
```

Screenshot of code (Test bench):

UP COUNTER TESTBENCH:

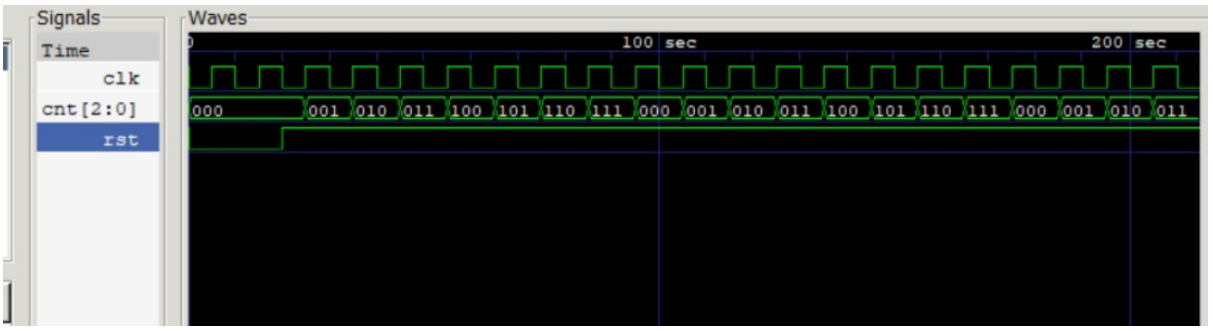
```
module counter_tb;
 reg clk;
 reg rst;
 wire [2:0] cnt;

 counter uut (
  .cnt(cnt),
  .clk(clk),
  .rst(rst)
 );

 always #5 clk = ~clk;

 initial begin
  clk = 1'b0;
  rst = 1'b0;
  #20 rst = 1'b1;
  #200 $finish;
 end

initial begin
  $dumpfile("counter.vcd");
  $dumpvars(0, counter_tb);
end

endmodule
```

DOWN COUNTER TESTBENCH:

```
module downcountertb;
 reg clk;
 reg rst;
 wire [2:0] cnt;

 downcounter uut (
  .cnt(cnt),
  .clk(clk),
  .rst(rst)
 );

 always #5 clk = ~clk;

 initial begin
  clk = 1'b0;
  rst = 1'b0;
  #20 rst = 1'b1;
  #200 $finish;
 end

initial begin
  $dumpfile("downcounter.vcd");
  $dumpvars(0, downcountertb);
end

endmodule
```

GTKWAVE OUTPUT:

UP COUNTER:



DOWN COUNTER: