Worksheet of Experiment

Date of Experiment	15 oct, 2024
Name	Akshada Jotiram Ghorpade.
Registration Number	12302956.

Title of Experiment: Develop RTL designs in verilog to implement Counter and verify their functionality using cadence NCSIM.

Name of Language: Verilog.

Name of Software: Cadence NCSIM, simvision.

Theory of Experiment:

COUNTERS:

- A counter in Verilog is a sequential circuit that counts events, typically based on clock pulses. Counters are widely used in digital systems to keep track of occurrences such as time intervals, events, or steps in a process.
- **Key Aspects of Counters in Verilog:**
  1. **Clock Signal:** Counters are driven by a clock signal. On each clock cycle (typically on the rising or falling edge of the clock), the counter either increments (in an up-counter) or decrements (in a down-counter).
  2. **Reset Signal:** A reset signal is often included to initialize the counter to a specific value (usually zero). This reset can be synchronous (occurs with the clock edge) or asynchronous (independent of the clock).
  3. **Counter Type:**

     - Up-Counter: Increments the counter value with each clock pulse.

     - Down-Counter: Decrements the counter value with each clock pulse.

     - Up/Down Counter: Can increment or decrement based on control inputs.

  4. **Width of the Counter:** The counter can have different bit-widths, determining the range of values it can represent. For example, a 4-bit counter can count from 0 to 15.

  5. **Overflow/Underflow:** When a counter reaches its maximum value (e.g., for a 4-bit counter, the value is `1111`), it will overflow and wrap around to zero (in the case of an up-counter). For a down-counter, underflow occurs when it reaches zero and wraps around to its maximum value.

Program and Stimulus (Testbench) program:

PROGRAM: (UPDOWN COUNTER)

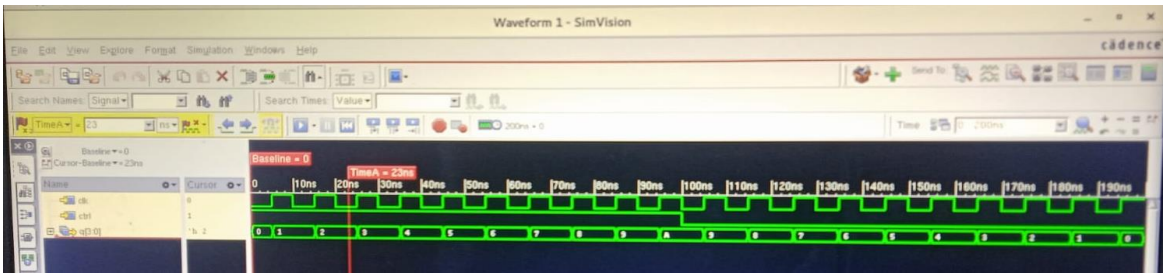
```
module updown(clk,ctrl,q);
input clk,ctrl;
output reg[3:0]q = 0;
always@(posedge clk)
begin
case(ctrl)
1'b1:q=q+1;
1'b0:q=q-1;
endcase
end
endmodule

module updown_tb();
reg clk,ctrl;
wire [3:0]q;
updown dut(clk,ctrl,q);
initial
begin
clk=0;
ctrl=1;
$monitor("time=%d,q=%d",$tim
#200 $stop;
end
always #5 clk=~clk;
always #100 ctrl=~ctrl;
endmodule
```

- UP COUNTER

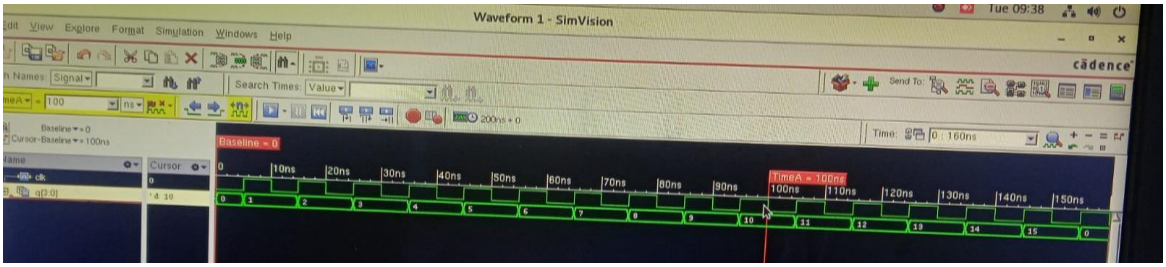
```
module counter(clk,q);
input clk;
output reg [3:0]q =0;
always @(posedge clk)
q=q+1;
endmodule

module counter_tb;
reg clk;
wire [3:0]q;
counter dut(clk,q);
initial
begin
clk=0;
#200 $stop;
end
always #5 clk=~clk;
endmodule
```

Simulation waveform:

UPDOWN COUNTER

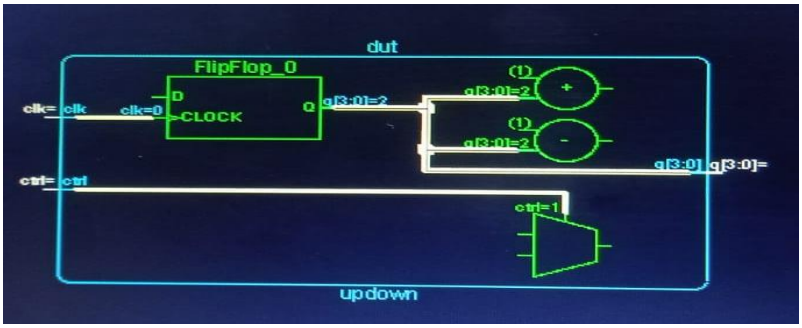


UP COUNTER

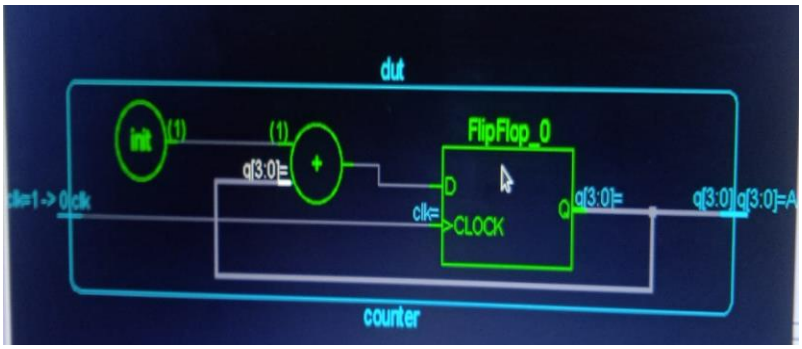


RTL Schematic:

UPDOWN COUNTER



UP COUNTER



Learning Outcome:

1. Learn to write syntactically correct Verilog code for Counter.

2. Create testbenches to simulate and verify the functionality of Counter.

3. Apply techniques to optimize logic circuits for speed, area, and power efficiency.

4. Perform timing analysis to ensure circuits meet required timing constraints.

5. Analyze the performance of Counter and circuits in terms of timing, power, and area using simulation results.