Experiment 10

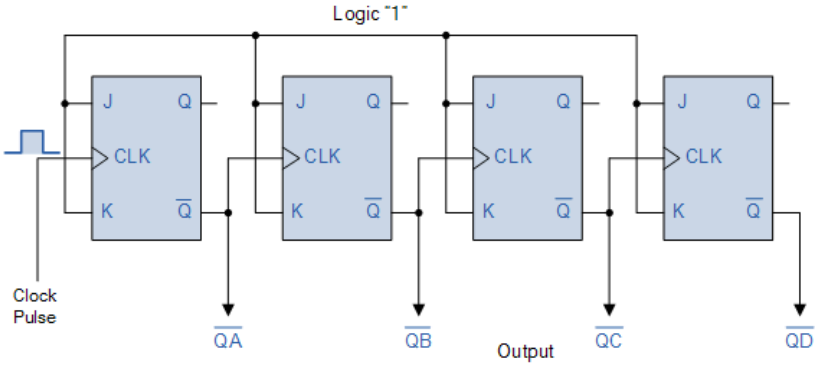
Aim: Write a Verilog code to implement 4 bit up down counter and verify the functionality using test bench.

Theory:

A 4-bit up-down counter consists of four flip-flops, one for each bit, and control logic that determines the direction of counting (up or down). The control logic takes input from two control signals: an Up signal and a Down signal. Depending on the values of these control signals, the counter will increment (up) or decrement (down) by one.

Here's how it works:

1. When the Up signal is asserted (usually logic high, '1'), the counter increments by one with each clock pulse. This means that the count goes from 0000 to 0001, 0010, 0011, and so on, until it reaches 1111, at which point it rolls over to 0000.
2. When the Down signal is asserted (usually logic high, '1'), the counter decrements by one with each clock pulse. This means that the count goes from 1111 to 1110, 1101, 1100, and so on, until it reaches 0000, at which point it rolls over to 1111.
3. If neither the Up nor the Down signal is asserted, the counter remains in its current state, effectively holding the count.



Code:

Behavioural

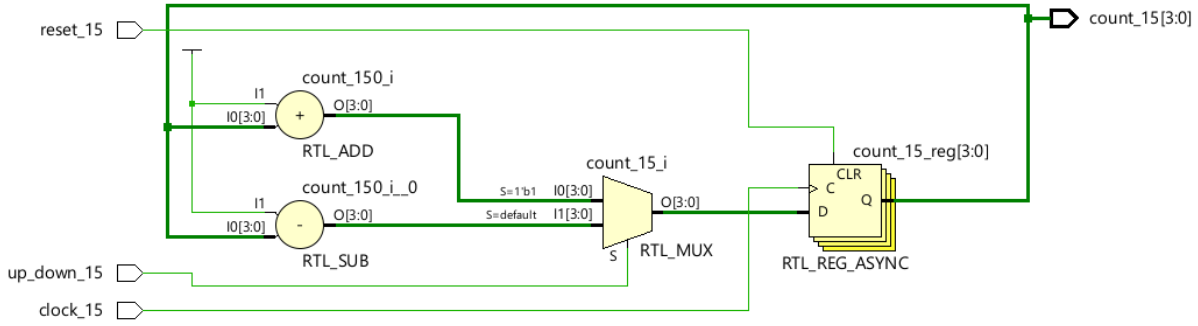
```
module up_down_counter (
  input wire clock,          // Clock signal
  input wire reset,          // Reset signal
  input wire up_down,        // Up/Down control input (1 for up, 0 for down)
  output wire [3:0] count    // 4-bit counter output
);

  reg [3:0] count; // 4-bit counter

  always @(posedge clock or posedge reset) begin
    if (reset) begin
      count <= 4'b0000; // Reset the counter
```

```
    end else if (up_down) begin
      // Increment the counter when up_down is high (up)
      count <= count + 1;
    end else begin
      // Decrement the counter when up_down is low (down)
      count <= count - 1;
    end
  end

endmodule
```

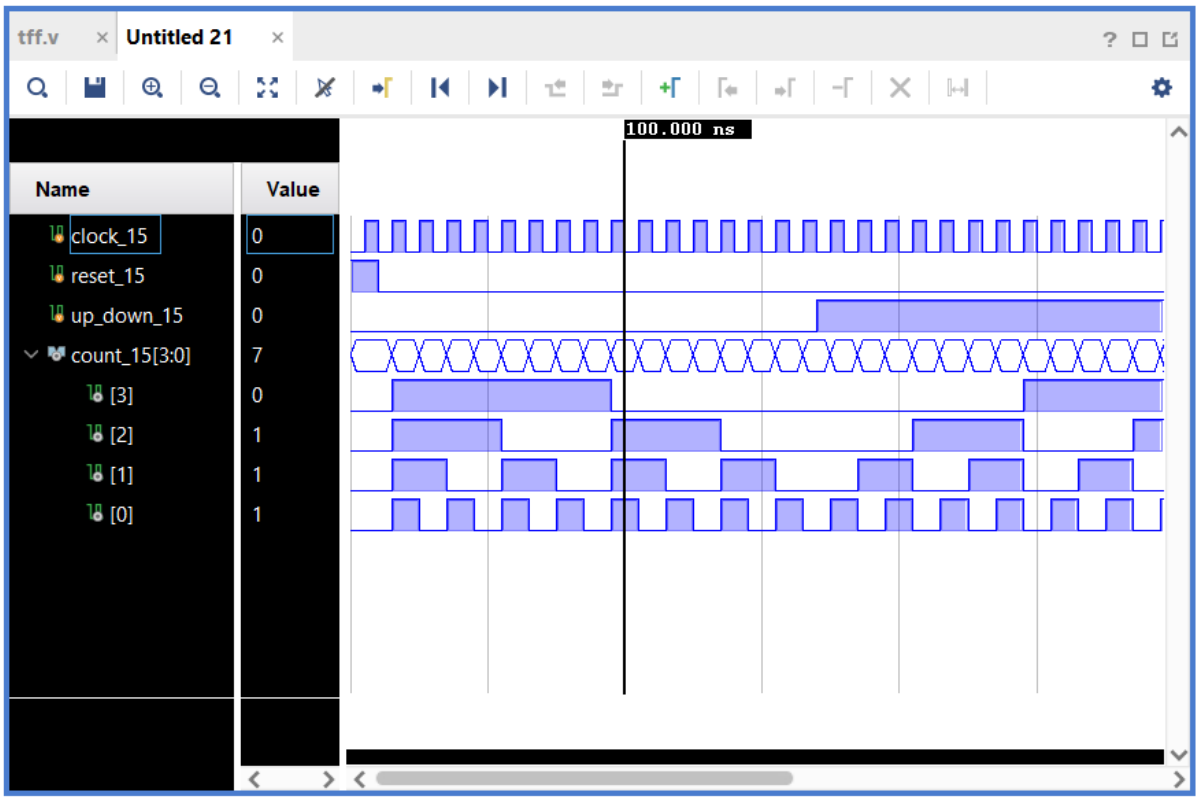
Test bench

```
module testbench_up_down_counter;

  reg clock;
  reg reset;
  reg up_down;
  wire [3:0] count;

  up_down_counter uut (
    .clock(clock),
    .reset(reset),
    .up_down(up_down),
    .count(count)
  );

  initial begin
    clock = 0;
    reset = 1;
    up_down = 0;
    #10 reset = 0;
    #160 up_down = 1; // Count up
    #160 up_down = 0; // Count down
    #80 up_down = 1; // Count up
    #80 up_down = 0; // Count down
    $finish;
  end

  always begin
    #5 clock = ~clock;
  end

endmodule
```

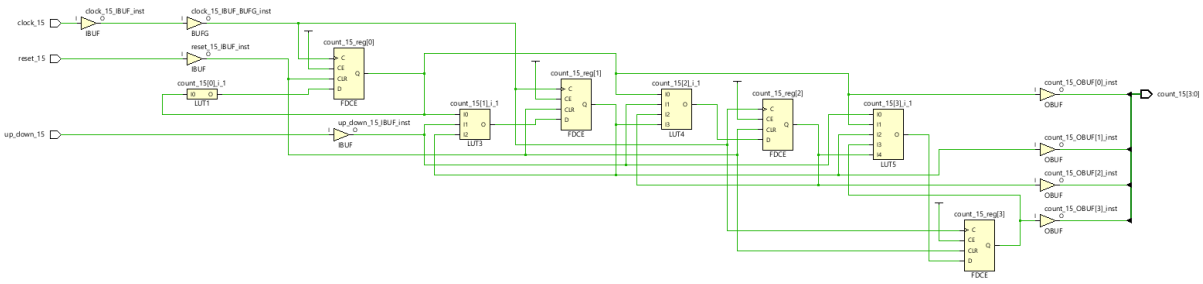
RTL Schematic:



Function Verification:



Post synthesis schematic:



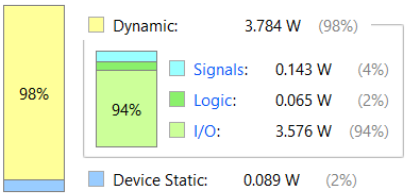
Post power report:

Summary

Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

Total On-Chip Power:	**3.874 W**
Design Power Budget:	**Not Specified**
Process:	typical
Power Budget Margin:	**N/A**
Junction Temperature:	**32.3°C**
Thermal Margin:	52.7°C (27.8 W)
Ambient Temperature:	25.0 °C
Effective ϑJA:	1.9°C/W

On-Chip Power

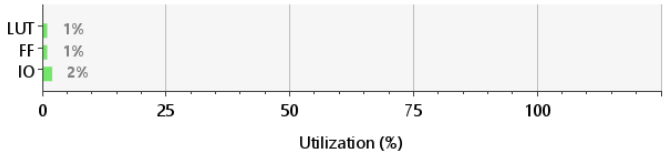


Post synthesis timing summary:

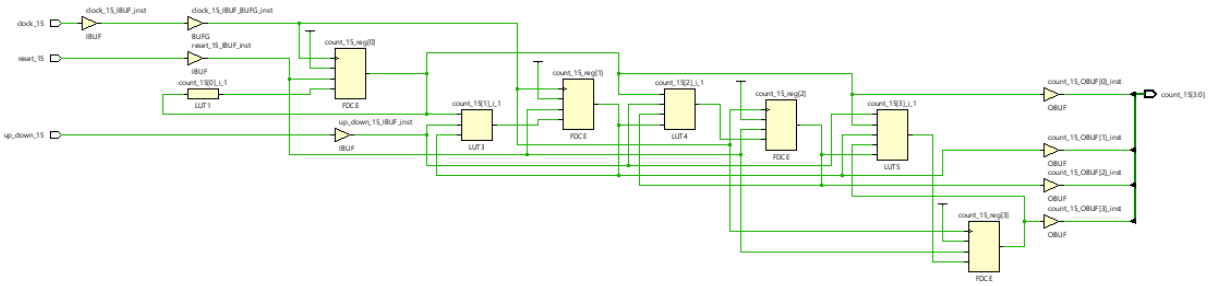
Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock
Path 1	∞	2	2	5	count_15_reg[0]/C	count_15[0]	3.419	2.836	0.584	∞	
Path 2	∞	2	2	4	count_15_reg[1]/C	count_15[1]	3.419	2.836	0.584	∞	
Path 3	∞	2	2	3	count_15_reg[2]/C	count_15[2]	3.419	2.836	0.584	∞	
Path 4	∞	2	2	2	count_15_reg[3]/C	count_15[3]	3.419	2.836	0.584	∞	
Path 5	∞	2	3	3	up_down_15	count_15_reg[1]/D	1.478	0.895	0.584	∞	input port clock
Path 6	∞	2	3	3	up_down_15	count_15_reg[3]/D	1.476	0.893	0.584	∞	input port clock
Path 7	∞	2	3	3	up_down_15	count_15_reg[2]/D	1.466	0.883	0.584	∞	input port clock
Path 8	∞	1	2	4	reset_15	count_15_reg[0]/CLR	1.413	0.830	0.584	∞	input port clock
Path 9	∞	1	2	4	reset_15	count_15_reg[1]/CLR	1.413	0.830	0.584	∞	input port clock
Path 10	∞	1	2	4	reset_15	count_15_reg[2]/CLR	1.413	0.830	0.584	∞	input port clock

Post utilization/ area summary:

Resource	Utilization	Available	Utilization %
LUT	2	41000	0.00
FF	4	82000	0.00
IO	7	300	2.33



Post implementation schematic:

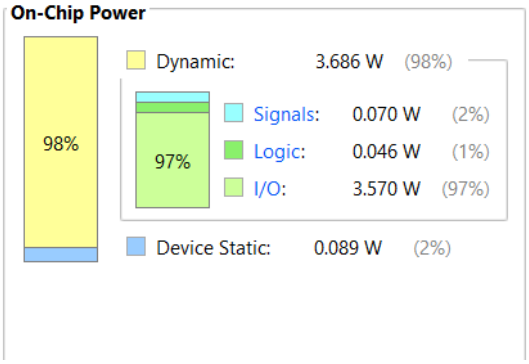


Post implementation power report:

Summary

Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**3.775 W**
Design Power Budget:	**Not Specified**
Process:	typical
Power Budget Margin:	**N/A**
Junction Temperature:	**32.1°C**
Thermal Margin:	52.9°C (27.9 W)
Ambient Temperature:	25.0 °C

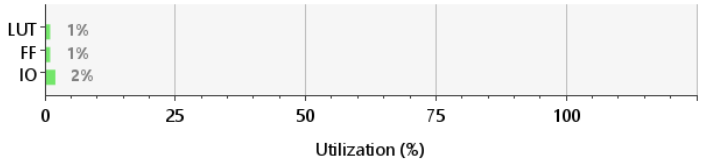


Post implementation timing summary:

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock
Path 1	∞	2	1	4	count_15_reg[1]/C	count_15[1]	4.062	2.802	1.261	∞	
Path 2	∞	2	1	5	count_15_reg[0]/C	count_15[0]	4.018	2.703	1.315	∞	
Path 3	∞	2	1	2	count_15_reg[3]/C	count_15[3]	4.014	2.765	1.249	∞	
Path 4	∞	2	1	3	count_15_reg[2]/C	count_15[2]	3.976	2.715	1.261	∞	
Path 5	∞	2	1	3	up_down_15	count_15_reg[3]/D	1.891	0.899	0.992	∞	input port clock
Path 6	∞	2	1	3	up_down_15	count_15_reg[2]/D	1.874	0.882	0.992	∞	input port clock
Path 7	∞	2	1	3	up_down_15	count_15_reg[1]/D	1.733	0.885	0.849	∞	input port clock
Path 8	∞	1	1	4	reset_15	count_15_reg[0]/CLR	1.653	0.785	0.868	∞	input port clock
Path 9	∞	1	1	4	reset_15	count_15_reg[1]/CLR	1.653	0.785	0.868	∞	input port clock
Path 10	∞	1	1	4	reset_15	count_15_reg[2]/CLR	1.653	0.785	0.868	∞	input port clock

Post implementation utilization/ area summary:

Resource	Utilization	Available	Utilization %
LUT	2	41000	0.00
FF	4	82000	0.00
IO	7	300	2.33



Conclusion:

- In this experiment, we written Verilog code for a circuit of 4 bit up down counter and verified its output and Schematic using its test bench and we got desired output.