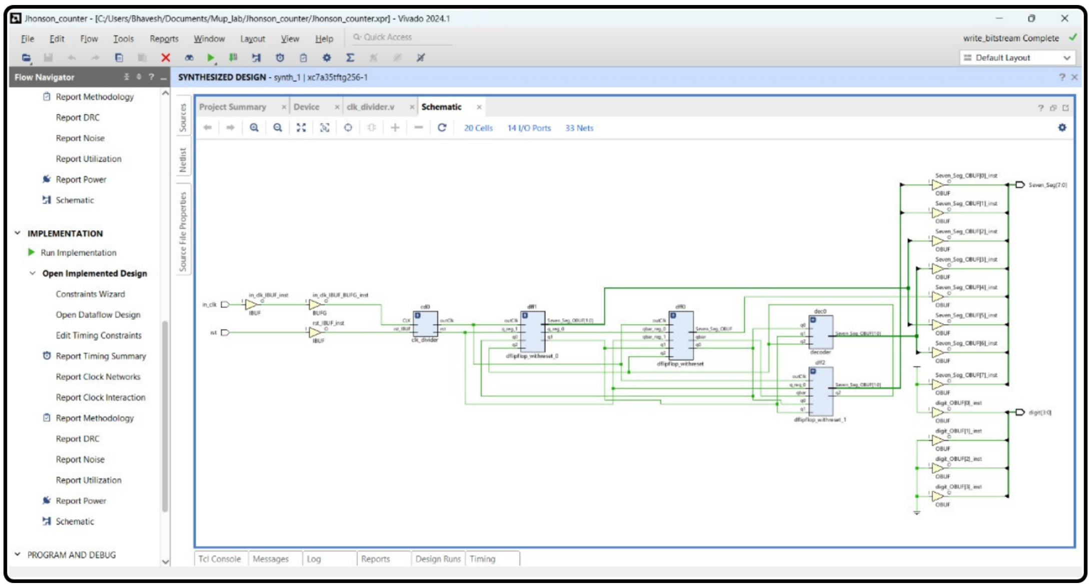


EE2016: Assignment-2

By Siva(EE23B151),
Bhavesh(EE23B017)
and Naveen(EE23B113)

Group 28

Date: 8/17/2024

Indian Institute of Technology Madras

Contents

1 Experiment Objective

To simulate and implement a 3-bit Johnson counter using Verilog in Vivado, and display the output on a seven-segment display on an FPGA board.

Steps:

1. **Simulating a D Flip-Flop:**
   - Create a Verilog module for a basic D flip-flop.
   - Simulate the module in Vivado to verify its functionality.
2. **Adding a Reset Signal:**
   - Modify the D flip-flop design to include an asynchronous reset signal.
   - Simulate the modified D flip-flop to ensure it resets correctly when the reset signal is active.
3. **Designing a 3-bit Johnson Counter:**
   - Connect three modified D flip-flops in a shift register configuration to create a 3-bit Johnson counter.
   - Simulate the Johnson counter to verify the correct sequence of states.
4. **Creating a Clock Divider:**
   - Design a clock divider module in Verilog to reduce the FPGA board's 50 MHz clock to a lower frequency suitable for observing the output on the seven-segment display.
   - Simulate the clock divider to confirm the correct output frequency.
5. **Seven-Segment Display Decoder:**
   - Develop a Verilog module to decode the Johnson counter's output and drive the seven-segment display.
   - Integrate this module with the Johnson counter and clock divider.
6. **FPGA Implementation:**
   - Synthesize the design in Vivado and implement it on the FPGA board.
   - Verify that the Johnson counter's output is correctly displayed on the seven-segment display.

2 D Flip-Flop simulation in Xilinx Vivado

2.1 D Flip-Flop

Takes in two inputs: **Data** (**D**) and *Clock*. We will be using *positive edge* of the clock. The outputs Q and Q' are *complementary* to each other and related to **D** as follows:

$$Q(t+1) = D(t)$$

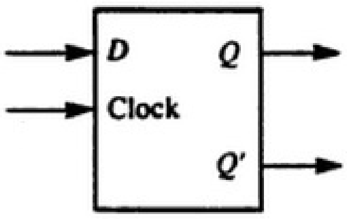


Figure 1: D Flip-Flop Diagram

2.2 Verilog Codes

Click on the blue hyperlinks:

- **D Flip-Flop Module**: contains the main module.
- **Testbench**: used for testing the D Flip-Flop module.

2.3 Outputs

CLICK HERE to see full utilization report.

3 Adding a Reset functionality for D Flip-Flop

3.1 Reset button in D Flip-Flop

Same as the basic D Flip-Flop, but with an extra input, **reset**. If reset is low, whatever the change in **D**, Q will be zero.

$$Q(t+1) = R.D$$

Note, this 'reset' is *asynchronous*, ie, when R is low, at the same instant Q turns 0 irrespective of the clock edge!

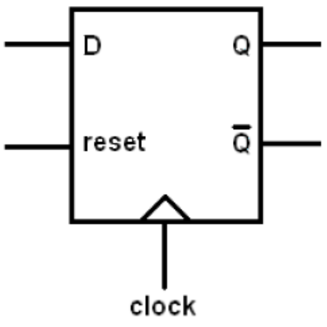


Figure 2: D Flip-Flop with Reset

3.2 Verilog Codes

Click on the blue hyperlinks:

- **Modified D Flip-Flop Module**: contains the main module.
- **Testbench**: used for testing this modified flip-flop module.

3.3 Outputs

CLICK HERE to see full utilization report.

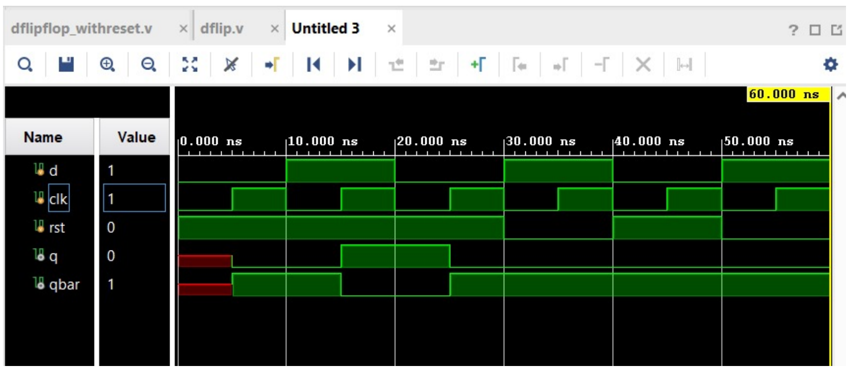


Figure 3: Simulation Graph

4 Creating the Clock Divider and SSD Decoder

4.1 Frequency division by incrementing a variable

By incrementing a variable (initially, 0), say *clock_ divider*, per positive edge of the clock until it reaches a number N, then re-assigning the variable to '0' and considering this drop as one clock edge, we have created a new clock with frequency:

$$f' = \frac{f_0}{N}$$

The result is a clock signal with a lower frequency that can be used for the Johnson counter circuit.

4.2 Decoder for converting Binary to 7 bit data for 7 segment display

The SSD is composed of 7 LEDs (+ an extra LED for decimal point) which are named a,b,c,d,e,f and g in *clockwise* order. Using these 7 'segments' we can show different numbers as shown in fig.4. For $val = (gfedcba)_2 = (0111111)_2$ we get '0' as output.

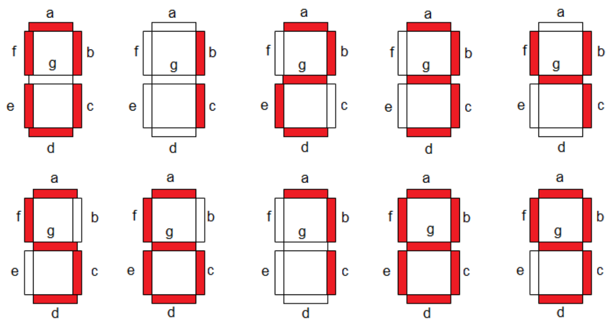


Figure 4: Seven Segment Display

To light the LEDs, we need to give LOGIC LOW (0). Hence, we need to take the compliment of the variable val.

4.3 Verilog Codes

Click on the blue hyperlinks:

- **Clock divider Module**: it contains module for clock divider.
- **Seven Seg. Decoder Module**: contains module for 7 segment display decoder.

5 Simulating Johnson Counter and implementing in FPGA

5.1 Johnson Counter

As seen from fig.5, the $\bar{Q}$ of LSB flip-flop is connected to the 'Data' of MSB flip-flop. This causes the following pattern for (Q_2, Q_1, Q_0):

$$(000) \to (100) \to (110) \to (111) \to (011) \to (001) \to (000) \to \dots$$

We also use an *asynchronous* reset which, if LOW, causes $(Q_2, Q_1, Q_0) = (0, 0, 0)$.

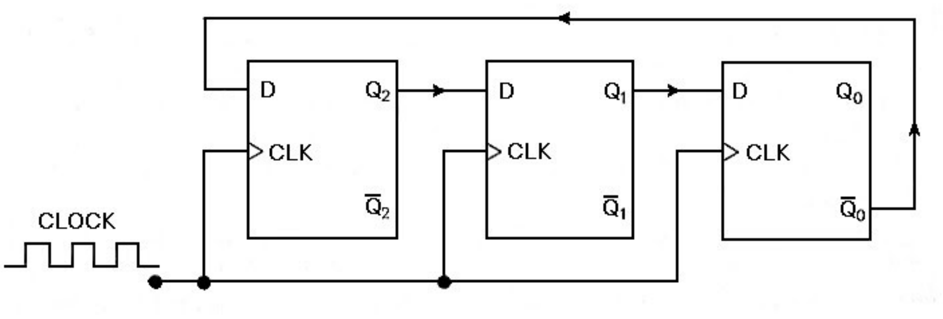


Figure 5: 3-bit Johnson Counter using three D Flip-Flops

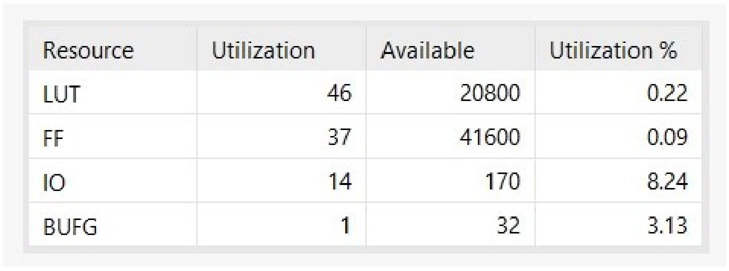
5.2 Verilog Codes and Constraint files

Click on the blue hyperlinks:

- **Johnson Counter Module**: contains the main module.
- **Testbench**: used for testing the module.
- **Constraint file**: used for mapping the variables in main module to respective i/o ports in FGPA board.

5.3 Outputs

CLICK HERE to see full utilization report.
CLICK HERE to see video of circuit testing.



Resource	Utilization	Available	Utilization %
LUT	46	20800	0.22
FF	37	41600	0.09
IO	14	170	8.24
BUFG	1	32	3.13

Figure 6: No. of LUTs used