A Project report on

8-Bit Modulo 7 Asynchronous up counter using D Flip Flop

Submitted By

SHREYAS SAUNSHI	4NM21EC140
SHRIRAM PAI	4NM21EC141
SHUBHAM KUMAR VATSA	4NM21EC142
PRAVEEN KUMAR K S	4NM22EC412

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

NITTE EDUCATION TRUST | N.M.A.M. INSTITUTE OF TECHNOLOGY
(An Autonomous Institution affiliated to Visvesvaraya Technological University, Belagavi)
Nitte – 574 110, Karnataka, India

August - 2023

NITTE EDUCATION TRUST | N.M.A.M. INSTITUTE OF TECHNOLOGY
(An Autonomous Institution affiliated to Visvesvaraya Technological University, Belagavi)
Nitte – 574 110, Karnataka, India
ISO 9001:2015 Certified, Accredited with 'A' Grade by NAAC
☎:08258 - 281039 – 281263, Fax: 08258 – 281265

Department of Electronics and Communication Engineering

CERTIFICATE

This is to certify that ***SHREYAS S SAUNSHI - 4NM21EC140, SHRIRAM PAI - 4NM21EC141, SHUBHAM KUMAR VATSA - 4NM21EC142, PRAVEEN KUMAR K S - 4NM22EC412,*** *Bonafede students of N.M.A.M. Institute of Technology, Nitte have submitted a project report entitled "****8-Bit Modulo 7 Asynchronous up counter using D Flip Flop"*** *as part of the Project based System Verilog Lab, in partial fulfillment of the requirements for the award of Bachelor of Engineering Degree in Electronics and Communication Engineering during the year 2022-2023.*

Name of the Examiner	**Signature with date**
……	……
……	…….

ABSTRACT

This report presents the design and implementation of an 8-bit modulo-7 asynchronous up counter using D flip-flops in System Verilog. The counter is designed to count from 0 to 6 in a loop, providing a modulo-7 sequence as specified in the inputs. The D flip-flop module is first defined, which serves as the basic building block for the counter. The counter module is then constructed by instantiating eight D flip-flops, interconnected to form the 8-bit counter.

A test bench is developed to verify the functionality of the counter compares the output with the expected results using a pkg file validating its accuracy of the modulo-7 sequence. The implemented design and test bench demonstrate the effectiveness of the D flip-flop-based 8-bit modulo-7 counter and its ability to produce the desired output sequence under different clock and reset conditions.

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

Counters are an essential component of many digital electronics applications, from clock generators to sequential circuits. This project shows how to create an 8-bit modulo-7 counter using D flip-flops, an essential component of digital circuit design. The goal is to build a counter that counts in a modulo-7 pattern, repeatedly cycling through the numbers 0 through 6. The structural design of the counter is created using System Verilog, a hardware description language, with D flip-flops serving as its main building blocks.

The key focus of this project extends beyond the counter's mere implementation; it encompasses a comprehensive testing strategy as well. To ensure the counter's accuracy and functionality, an object-oriented programming-based test bench is constructed. This test bench treats the Design Under Test (DUT), i.e., the counter, as a black box, and rigorously compares its output with the expected results. This verification process is essential to validate the counter's adherence to the specified modulo-7 sequence, especially under varying clock and reset conditions. By treating the counter as an isolated module, the test bench facilitates efficient testing and eases the integration process into larger digital systems. Overall, the project aims to showcase the efficiency and robustness of the 8-bit modulo-7 counter, which is shown in the figure ‘Figure 2.1’ and the effectiveness of the object-oriented test bench in verifying its functionality. [1]

Chapter 2

DESIGN AND IMPLEMENTATION

2.1 CIRCUIT DIAGRAM/ BLOCK DIAGRAM

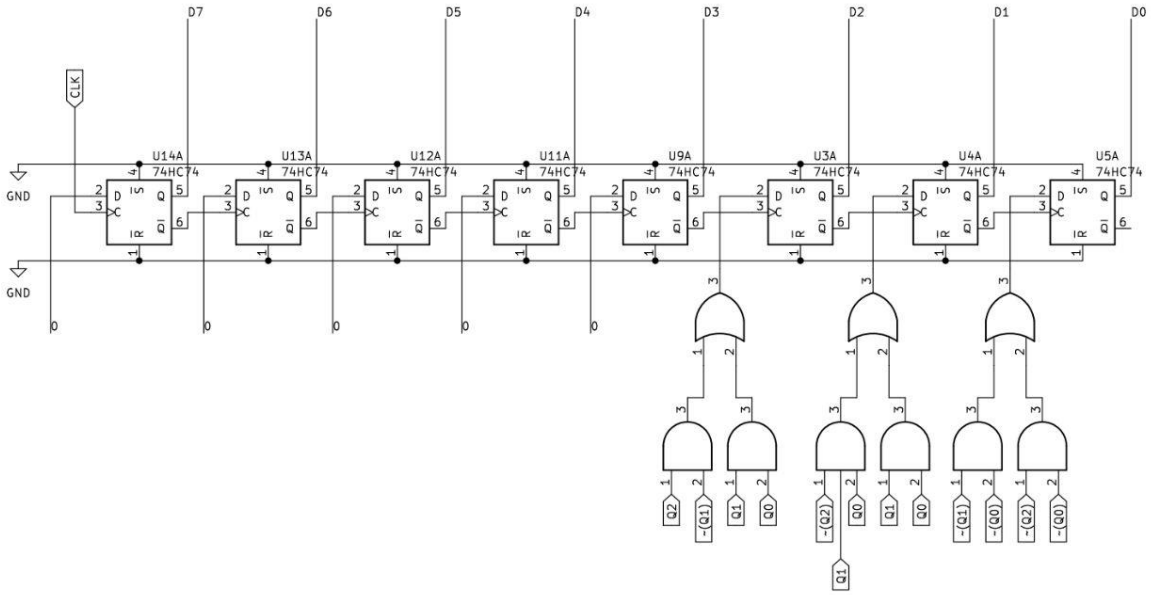

Figure 2.1 Counter Circuit Diagram

2.2 COMPONENTS USED

2.2.1 SOFTWARE

1. **Xilinx ISE:** Xilinx ISE (Integrated Software Environment) is a popular development environment for designing, implementing, and debugging digital circuits using Xilinx FPGAs (Field-Programmable Gate Arrays) and CPLDs (Complex Programmable Logic Devices).

2. **ISIM:** ISIM is the logic simulator provided by Xilinx as part of their ISE (Integrated Software Environment) tool suite. It is used for simulating and testing HDL (Hardware Description Language) designs targeted for

Xilinx FPGAs. ISIM is integrated into the Xilinx ISE design flow and is commonly used by FPGA designers working with Xilinx devices.

2.2.2 HARDWARE

1. **D Flip-Flop**: The D flip-flop is a fundamental sequential logic element used to store and propagate data in asynchronous systems. It acts as a building block for the 8-bit modulo-7 counter, enabling the storage and transfer of data bits based on the clock signal.

2. **8-bit Modulo-7 Counter:** The 8-bit modulo-7 counter is the main component of the project, responsible for counting from 0 to 6 and repeating the sequence. It is constructed using eight D flip-flops interconnected to form the counter, which produces the expected output sequence given clock and reset inputs.

3. **Clock Source:** The clock source provides regular clock pulses, crucial for the synchronous operation of the 8-bit modulo-7 counter. It ensures that the counter updates its state at each positive edge of the clock signal, enabling precise counting.

4. **Reset Signal:** The reset signal is an input to the counter, allowing the system to be initialized to a known state. When asserted, it sets the counter to the initial value (00000000) to ensure proper functionality

2.3 WORKING

1. **D Flip-Flop Module:** The first step is to implement the D flip-flop module, which takes inputs for clock (clk), reset (reset). It has a single output Q, which holds the value of d when the positive edge of the clock occurs. If the reset signal is asserted (high), the output q is set to 0; otherwise, it takes the value of d on the positive edge of the clock.

2. **Modulo-7 Counter Module:** Next, we create the 8-bit modulo-7 counter module using the D flip-flop module. The counter module takes inputs for clock (clk) and reset (reset) and has an 8-bit output count_out representing the counter's current value. We then connected them in series to form the counter. Each D flip-flop holds one bit of the counter's value, and the next_count signal interconnects them. The next_count signal stores the value to be loaded into the counter on the next clock edge. By updating the counter_reg with the next_count on each positive edge of the clock, the counter increments in a modulo-7 sequence.

3. **Test Bench:** To test the counter's functionality, we develop an test bench. The test bench defines the parameters, signals, and modules required for the test environment. The DUT (Design Under Test) is instantiated as the modulo-7 counter module. The test bench generates the clock signal and sets the reset signal to simulate different test scenarios. After waiting for a few clock cycles to stabilize the system, the test bench monitors the counter's output and compares it with the expected output sequence (00000000 to 00000110). The test bench treats the counter as a black box, meaning it does not access the internal details of the counter module, ensuring a realistic verification approach. The output are shown in the below table named ‘Table 1’.

4. **Verification:** During the simulation, the test bench checks if the counter produces the expected output sequence using a simulation file. If the counter's output matches the expected values for each clock cycle and reset condition, it indicates that the 8-bit modulo-7 counter is functioning correctly. The test bench ensures that the counter counts up to 6 (00000110) and then resets back to 0 (00000000), as shown in the table 2.1, repeating the modulo-7 sequence as specified in the expected

output. The test bench provides confidence in the counter's correctness and adherence to the desired functionality.

Present State (Q7 Q6 Q5 Q6 Q4 Q3 Q2 Q1 Q0)	Next State (Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0)	D7	D6	D5	D4	D3	D2	D1	D0
00000111	00000000	0	0	0	0	0	0	0	0
00000000	00000001	0	0	0	0	0	0	0	1
00000001	00000010	0	0	0	0	0	0	1	0
00000010	00000011	0	0	0	0	0	0	1	1
00000011	00000100	0	0	0	0	0	1	0	0
00000100	00000101	0	0	0	0	0	1	0	1
00000101	00000110	0	0	0	0	0	1	1	0
00000110	00000111	X	X	X	X	X	X	X	X

Tabel 2.1 Expected Output

2.4 CODE

2.4.1 D Flipflop:

```
module DFlipFlop(D, reset, clk, Q, Qb);
 input D, reset, clk;
 output Q, Qb;
 reg Q, Qb;

 always @(posedge clk,posedge reset) //asynchronous reset and
posedge clk
 begin
 if (reset)begin
 Q = 0;
 Qb=1;
 end
 else
 Q = D;
 Qb= ~Q;
 end
endmodule
```

2.4.2 8bit-Modulo7Counter:

```
module Modulo7Counter8Bit(clk,reset,Q);
input clk,reset;
output [7:0]Q;

wire [7:0]Qb;
wire d2,d1,d0;
wire w0,w1,w2,w3,w4,w5,w6;

and a0(w0,Qb[2],Q[1],Qb[0]);
and a1(w1,Qb[1],Q[0]);
or o0(d1,w0,w1);
and a2(w3,Q[2],Qb[1]);
and a3(w4,Q[1],Q[0]);
or o1(d2,w3,w4);
and a4(w5,Qb[1],Qb[0]);
and a5(w6,Qb[2],Qb[0]);
or o2(d0,w5,w6);

DFlipFlop dff7(.D(Q[7]), .reset(reset), .clk(clk), .Q(Q[7]), .Qb(Qb[7]));
DFlipFlop dff6(.D(Q[6]), .reset(reset), .clk(clk), .Q(Q[6]), .Qb(Qb[6]));
DFlipFlop dff5(.D(Q[6]), .reset(reset), .clk(clk), .Q(Q[5]), .Qb(Qb[5]));
DFlipFlop dff4(.D(Q[4]), .reset(reset), .clk(clk), .Q(Q[4]), .Qb(Qb[4]));
DFlipFlop dff3(.D(Q[3]), .reset(reset), .clk(clk), .Q(Q[3]), .Qb(Qb[3]));
DFlipFlop dff2(.D(d2), .reset(reset), .clk(clk), .Q(Q[2]), .Qb(Qb[2]));
DFlipFlop dff1(.D(d1), .reset(reset), .clk(clk), .Q(Q[1]), .Qb(Qb[1]));
DFlipFlop dff0(.D(d0), .reset(reset), .clk(clk), .Q(Q[0]),.Qb(Qb[0]));

Endmodule
```

2.4.3 Test Bench:

```
module testbench;
import mod7Counter_pkg::*;
bit clk,reset;
bit [7:0]Q; //Expected output
bit [7:0]q; //Got Output
modcounter counter=new(clk,reset);
Modulo7Counter8Bit dut(.clk(clk),.reset(reset),.Q(Q));
initial begin
 clk=1'b0;
 reset=1'b1;
#8 reset=1'b0;
end
always@(posedge clk,posedge reset)begin
counter.performCount(reset);
q=counter.out;
$display("The Expected and the got outputs are %b and %b",q,Q);
end
always #5 clk=~clk;
endmodule
```

2.4.4 Verification Pkg:

```
package mod7Counter_pkg;
 class modcounter;
 bit clk;
 bit reset;
 bit [7:0] out;

function new(bit clk1, bit reset1);
 clk = clk1;
 reset = reset1;
endfunction
```

```
function void performCount(reset );
if (reset)
 out = 8'b0;
 else begin
 if (out<8'b00000110)
 out = out + 1;
 else
 out = 8'b00000000;
 end
endfunction
 endclass
endpackage
```

Chapter 3

RESULT AND DISCUSSION

The implementation of an 8-bit modulo-7 counter using D flip-flops in Verilog has been successfully completed. The structural design of the counter is based on eight interconnected D flip-flop module as shown is the figure below Figure 3.1 & Figure 3.2.

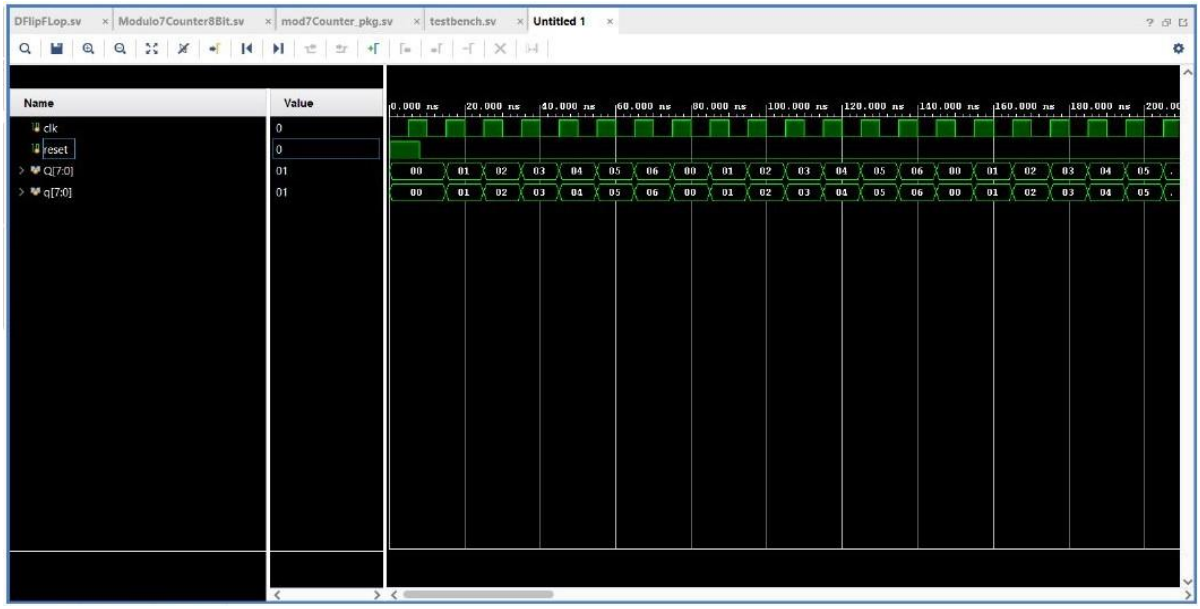

Figure 3.1 Output in decimals

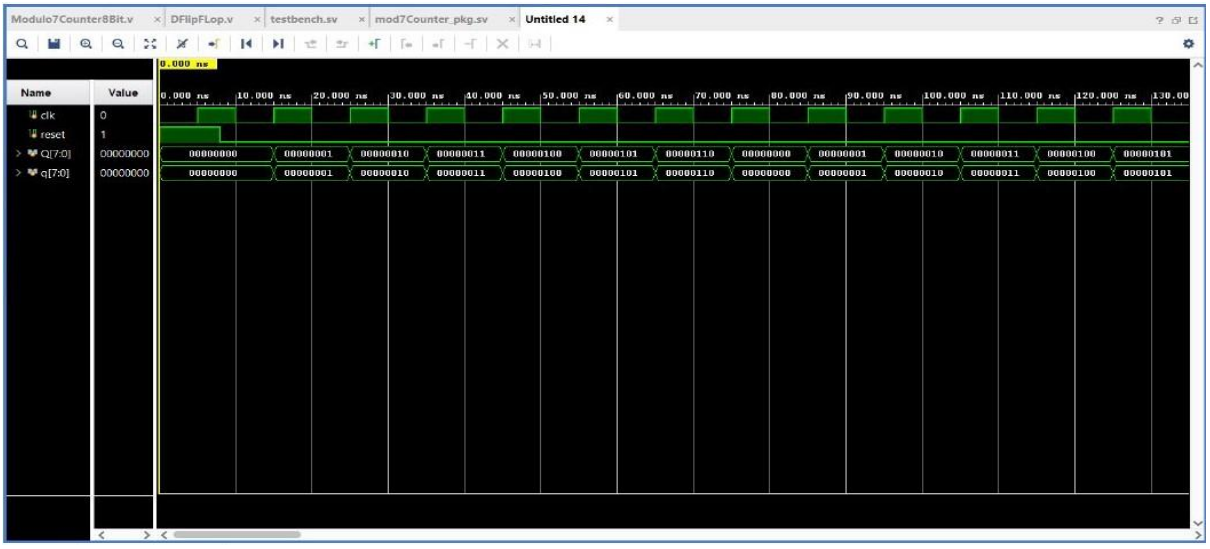

Figure 3.2 Output in Binary's

Upon running the test bench, the counter demonstrated correct behavior as per the expected output sequence. The counter started from 00000000 and incremented to 00000110. Then, the counter reset back to 00000000 and continued the sequence, repeating the modulo-7 pattern.

Chapter 4

CONCLUSION

D flip-flops have been successfully used to create an 8-bit modulo-7 counter, and testing has been done to confirm the counter's operation. D flip-flops made it possible for a simple, effective design that created an accurate counter that delivered the appropriate output sequence. The test bench and verification file is useful tool for verification, confirming the accuracy and dependability of the counter. Overall, the project shows off the efficient use of Verilog to create an 8-bit modulo-7 counter and rigorously tests its functioning.

REFERENCES

[1] https://circuitdigest.com/tutorial/synchronous-counter

[2] https://electronics.stackexchange.com/questions/268138/8-bit-counter-verilog