EECT/CE 6325 VLSI Design - Fall 2022

Department of Electrical & Computer Engineering

The University of Texas at Dallas

Non-Overlapping 5 Bit Sequence Detector

Project 1: Verilog Design

Submitted By

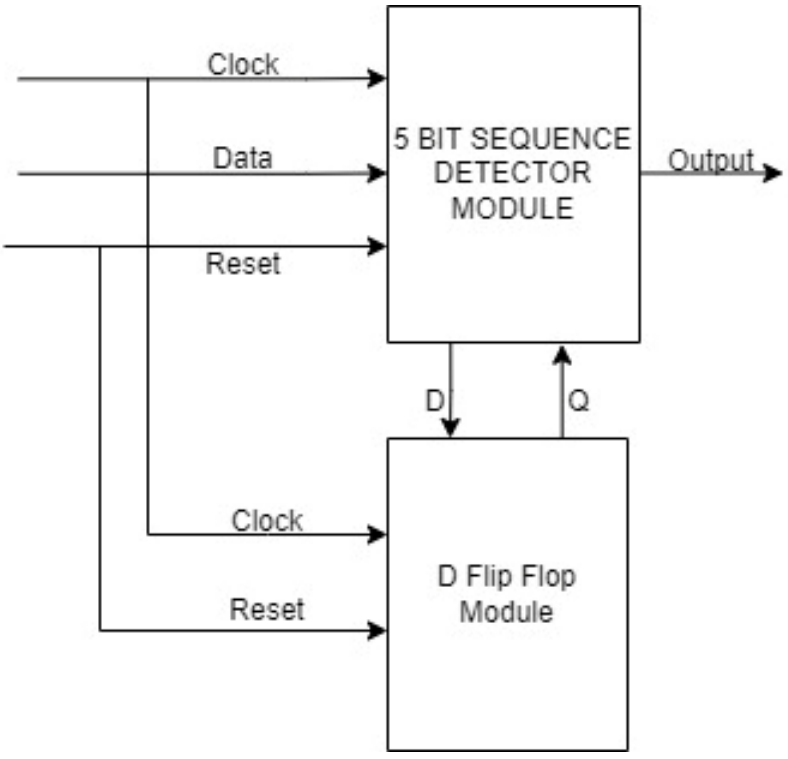
Sai Shantan Nagelli	SXN210083
Sanjay Sankarasubramanian	SXS220137
Surya Ramesh Kumar	SXR220066

Aim

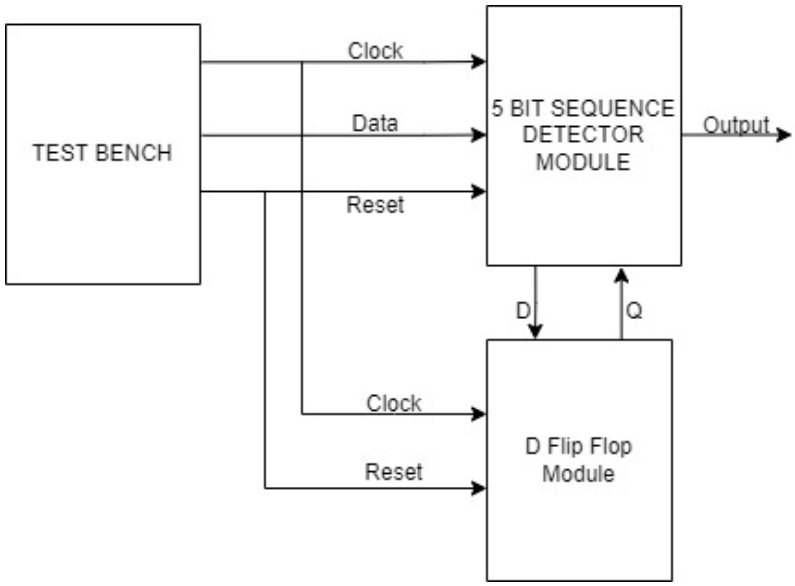
Our project aims to design a 5 bit non overlapping sequence detector. The circuit takes a stream of bits as input and generates a high output whenever the target sequence is detected. As a non-overlapping sequence detector, the last bit of the detected sequence is not considered as the starting bit of the next sequence.

The sequence detector was designed by developing the State Machines and their corresponding State Tables. From the State Tables, the Boolean expressions for the final circuit are derived using Karnaugh Maps. The sequence detector is realized using three D flipflops and various combinational gates.

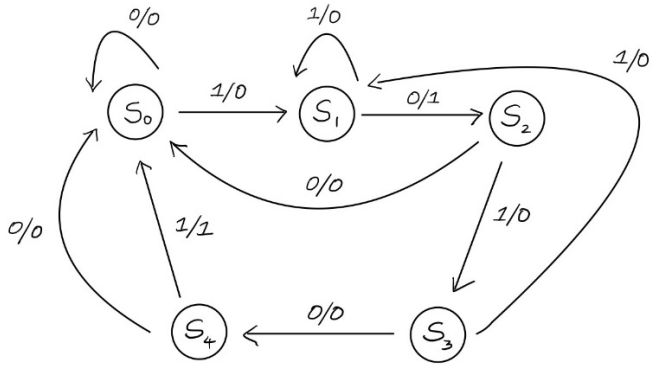
Block Diagram



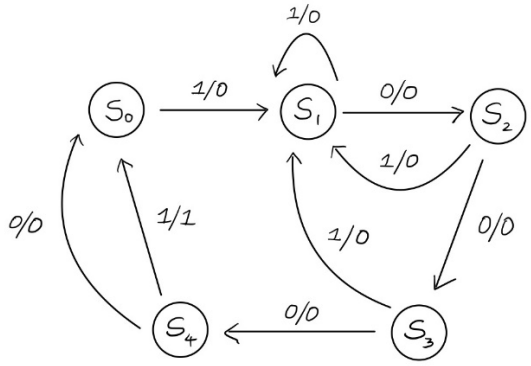
Test Bench



State Machines



State Machine for sequence 10101



State Machine for sequence 10101

State Tables

For Sequence 10101

State	Qa	Qb	Qc	X	Next State	Qa_{t+1}	Qb_{t+1}	Qc_{t+1}	Y
S_0	0	0	0	0	S_0	0	0	0	0
S_0	0	0	0	1	S_1	0	0	1	0
S_1	0	0	1	0	S_2	0	1	0	0
S_1	0	0	1	1	S_1	0	0	1	0
S_2	0	1	0	0	S_0	0	0	0	0
S_2	0	1	0	1	S_3	0	1	1	0
S_3	0	1	1	0	S_4	1	0	0	0
S_3	0	1	1	1	S_1	0	0	1	0
S_4	1	0	0	0	S_0	0	0	0	0
S_4	1	0	0	1	S_0	0	0	0	1

For Sequence 10001

State	Qa	Qb	Qc	X	Next State	Qa_{t+1}	Qb_{t+1}	Qc_{t+1}	Y
S_0	0	0	0	0	S_0	0	0	0	0
S_0	0	0	0	1	S_1	0	0	1	0
S_1	0	0	1	0	S_2	0	1	0	0
S_1	0	0	1	1	S_1	0	0	1	0
S_2	0	1	0	0	S_3	0	1	1	0
S_2	0	1	0	1	S_1	0	0	1	0
S_3	0	1	1	0	S_4	1	0	0	0
S_3	0	1	1	1	S_1	0	0	1	0
S_4	1	0	0	0	S_0	0	0	0	0
S_4	1	0	0	1	S_0	0	0	0	1

$$Y = Q_a \overline{Q}_b \overline{Q}_c X$$

Karnaugh Map Simplification

For sequence 10101

D_a

Q_cX \ Q_aQ_b				
	1			

$$D_a = \overline{Q}_a Q_b Q_c \overline{X}$$

D_b

Q_cX \ Q_aQ_b				
	1			
1				

$$D_b = \overline{Q}_a Q_b \overline{Q}_c X + \overline{Q}_a \overline{Q}_b Q_c \overline{X}$$

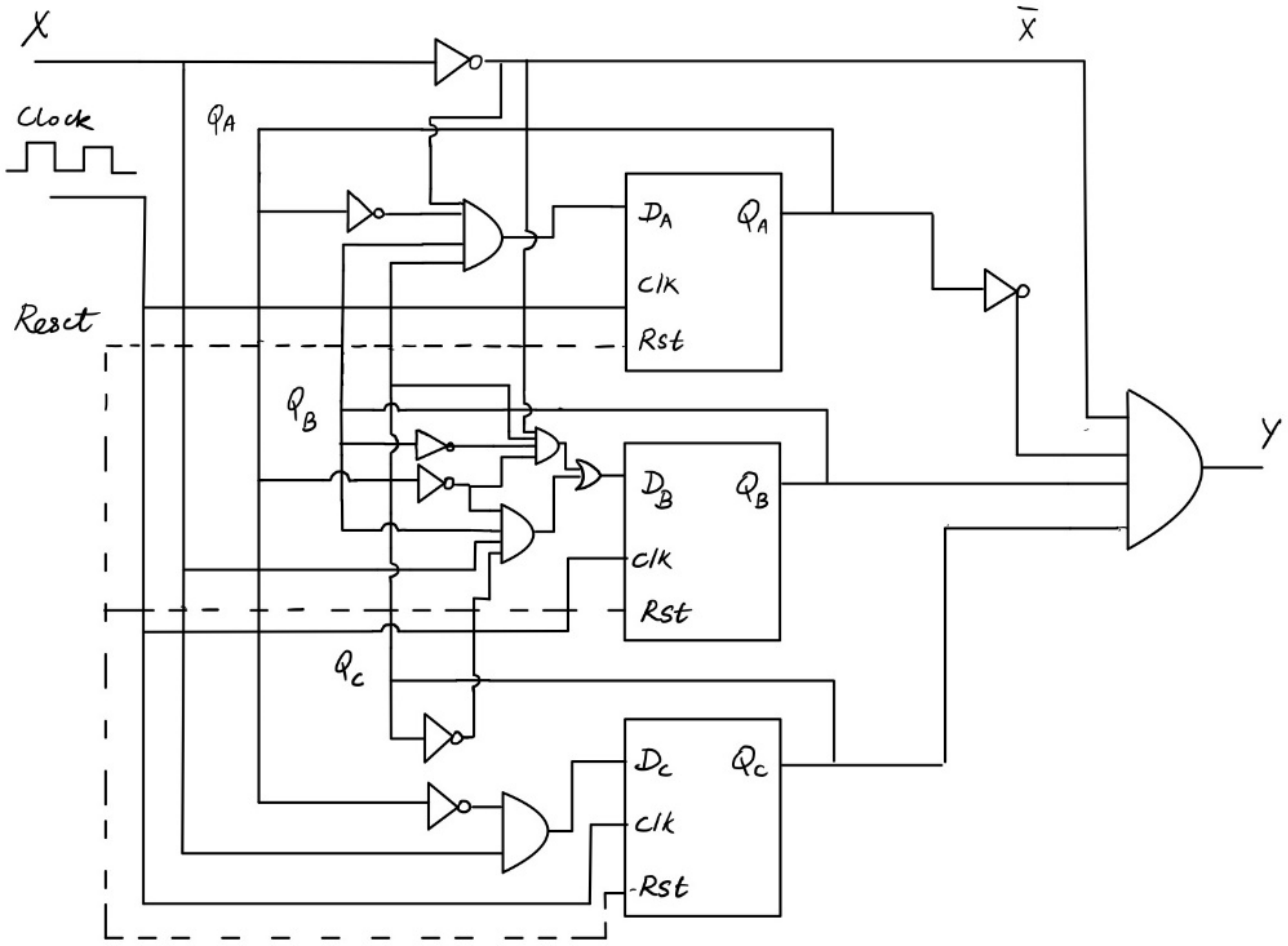
D_c

Q_cX \ Q_aQ_b				
1	1			
1	1			

$$D_c = \overline{Q}_a X$$

Logic Gate Diagram

For Sequence 10101



Logic gate diagram for sequence 10101

Our logic gate circuit for the detection of sequence 10101 utilizes three D flipflops (D_A, D_B and D_C) and combinational gates AND, OR and Inverters. The same K-Map simplification and Logic gate circuit design seen above has been followed for the '10001' sequence detector as well.

Verilog Code

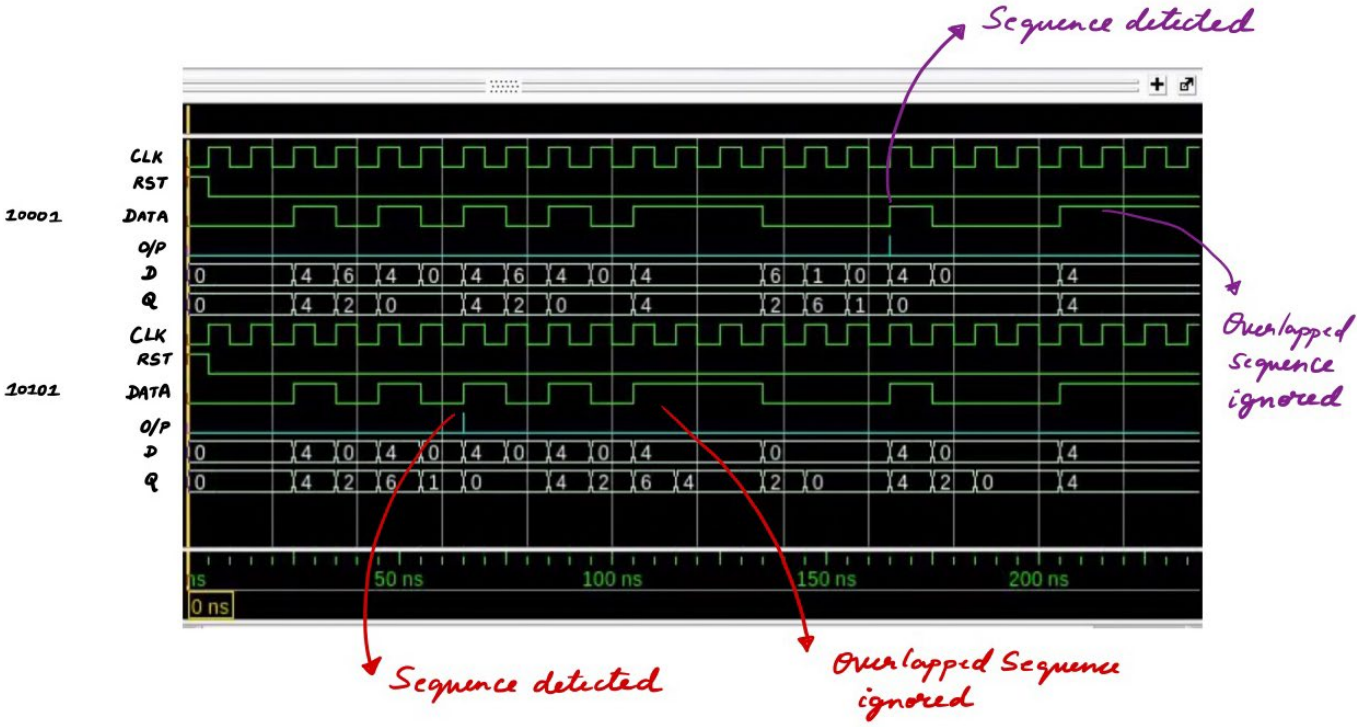
```
//EECT/CE 6325 VLSI Design -Fall 2022
//PROJECT - 1: Verilog/VHD

//Members: Sai Shantan Nagelli (sxn210083), Sanjay Sankarasubramanian (sxs220137), Surya Ramesh Kumar (sxr220066)

//5 Bit Non Overlapping Sequence Detector (10101 & 10001)

//D Flip Flop Module

module dff (
  input clock,          // Clock input to drive the flip flop
  input reset,          // Reset input to reset flip flop state
  input d,              // Data input to flip flop
  output reg q          // Output of flip flop - Register Q
  );

  always @ (posedge clock or posedge reset)             //Flip flop works on either postive edge of clock or reset inputs
  begin
    if(reset)           //When Reset pin set to 1, output of flipflop 'q' is reset to 0
      q <= 0;
    else
      q <= d;           //On positve edge of clock, output of flipflop 'q' is set to Data input 'd'
  end
endmodule

// Module to detect 5 bit sequence 10101

module seq_det_10101 (
  input clock,          // Clock input to drive the flip flop
  input reset,          // Reset input to reset the flip flop state
  input data,           // Data input to flip flop
  output detected_10101         //Output - 0 when sequence is not detected, 1 when sequence is detected
  );

  wire [2:0] D1;                //3-bit wire for flipflop data inputs
  wire [2:0] Q1;                //3-bit wire for flipflop output

//Boolean Expressions from reducing State Table using K-Map

  assign D1[0] = (-Q1[0] & Q1[1] & Q1[2] & -data);              //Input to flipflop A

  assign D1[1] = (-Q1[0] & Q1[1] & -Q1[2] & data) | (-Q1[0] & -Q1[1] & Q1[2] & -data);          //Input to flipflop B

  assign D1[2] = (-Q1[0] & data);               //Input to flipflop C

  assign detected_10101 = -Q1[1] & data & Q1[0] & -Q1[2];       //Sequence detection output for 10101

//Passing input and output parameters to the flipflops (A, B and C)

  dff DA (clock, reset, D1[0], Q1[0]);
  dff DB (clock, reset, D1[1], Q1[1]);
  dff DC (clock, reset, D1[2], Q1[2]);

endmodule

// Module to detect 5 bit sequence 10001

module seq_det_10001 (
  input clock,          // Clock input to drive the flip flop
  input reset,          // Reset input to reset the flip flop state
  input data,           // Data input to flip flop
  output detected_10001         //Output - 0 when sequence is not detected, 1 when sequence is detected
  );

  wire [2:0] D2;        //3-bit wire for flipflop data inputs
  wire [2:0] Q2;        //3-bit wire for flipflop output

//Boolean Expressions from reducing State Table using K-Map

  assign D2[0] = (-Q2[0] & Q2[1] & Q2[2] & -data);              //Input to flipflop D

  assign D2[1] = (-Q2[0] & Q2[1] & -Q2[2] & -data) | (-Q2[0] & -Q2[1] & Q2[2] & -data);         //Input to flipflop E

  assign D2[2] = (-Q2[0] & data) + (-Q2[0] & Q2[1] & -Q2[2]);           //Input to flipflop F

  assign detected_10001 = -Q2[1] & data & Q2[0] & -Q2[2];               //Sequence detection output for 10001
```

```
//Passing input and output parameters to the flipflops (D, E and F)

  dff DD (clock, reset, D2[0], Q2[0]);
  dff DE (clock, reset, D2[1], Q2[1]);
  dff DF (clock, reset, D2[2], Q2[2]);

endmodule
```

Testbench Code

```
module seqDetect_tb;

  reg clock;
  reg reset;
  reg data;

  wire detected_10001;
  wire detected_10101;

  //Link main modules with testbench

  seq_det_10001 uut1 (.clock(clock),.reset(reset),.data(data),.detected_10001(detected_10001));
  seq_det_10101 uut2 (.clock(clock),.reset(reset),.data(data),.detected_10101(detected_10101));

  initial begin

    //Initialize Inputs

    clock = 1'b0;
    reset = 1'b1;
    data = 0;

    #5 reset = 1'b0;    //Unset reset pin

    //Input sequence stream

    #10 data = 0;
    #10 data = 1;
    #10 data = 0;
    #10 data = 1;
    #10 data = 0;
    #10 data = 1;       //Overlap = Not counted as first bit of sequence 10101
    #10 data = 0;
    #10 data = 1;
    #10 data = 0;
    #10 data = 1;
    #10 data = 1;
    #10 data = 1;
    #10 data = 0;
    #10 data = 0;
    #10 data = 0;
    #10 data = 1;       //Overlap = Not counted as first bit of sequence 10001
    #10 data = 0;
    #10 data = 0;
    #10 data = 0;
    #10 data = 1;
  end

  initial #250 $finish;         //Finish simulation
  always #5 clock = ~clock;     //Switch clock from 0 to 1 for every 5 nanoseconds

endmodule
```

Output Wave Form



Results

From the output waveform, it is observed that the output goes high only when the 5-bit sequences ‘10101’ or ‘10001’ are detected. Based on our testbench input, we receive the sequences ‘10101’ and ‘10001’ at times 65 and 175 nanoseconds respectively. During these times, we get a high output which corresponds to the peaks observed in the graph (Cyan O/P). During these times, we also have continuing overlapping sequences, and these are not detected.