Ankit Kanwar – 2021AM10785

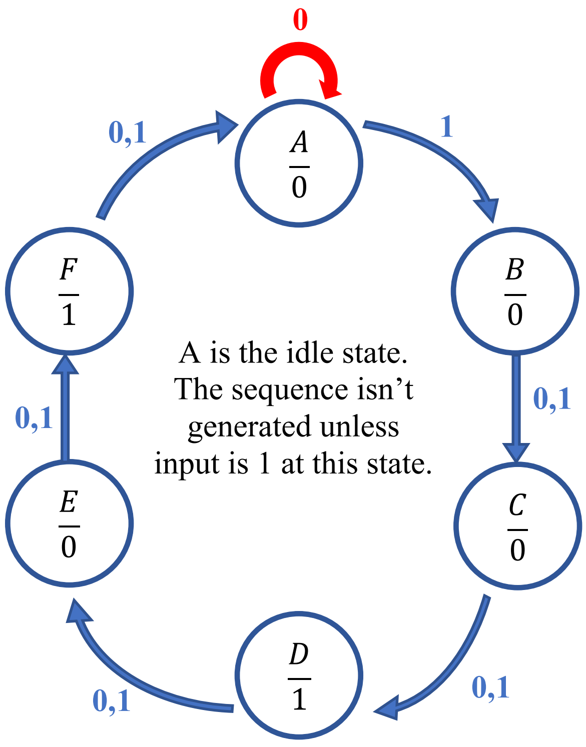
ELL201 Lab Project
Sequence Generator

As my entry number ends in 85, therefore,

$$X_3 = 8 \Rightarrow (X_3 \% 8) = (0)_{10} = (000)_2$$
$$X_4 = 5 \Rightarrow (X_4 \% 8) = (5)_{10} = (101)_2$$

Hence, the sequence to be generated is **000101** (left to right).

I would be implementing a Moore FSM, in which the output depends on the current state of the FSM and not on the input.
It can be shown that the number of states cannot be reduced further from the state table shown ahead.
Since the number of states cannot be reduced further, hence, this is the reduced state diagram of my FSM.


0
1
0,1
A
0
B
0
C
0
D
1
E
0
F
1
0,1
0,1
0,1
0,1
A is the idle state. The sequence isn't generated unless input is 1 at this state.

FSM Reduced State Diagram

D	Q	$\bar{Q}$
0	0	1
1	1	0

State	Value
A	000
B	001
C	010
D	011
E	100
F	101

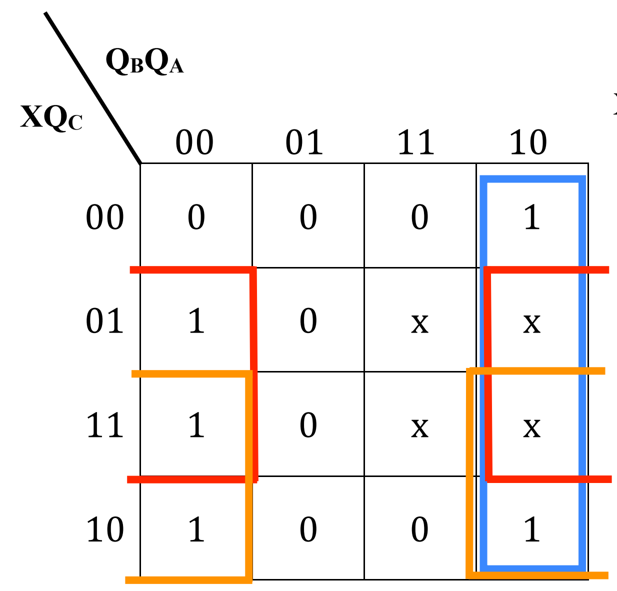
As there are 6 states in the state diagram, at least $\log_2 6$ flip flops are used.
Since $2^2 < 6 < 2^3$, I would be using **3 D flip flops**.
Let the flip flops be A, B, C. Their outputs (Q_C, Q_B, Q_A) determine the current state (most significant to least significant respectively).

Current State	**Next State**		**D_C**		**D_B**		**D_A**		**Output (Y)**
	X = 0	X = 1	X = 0	X = 1	X = 0	X = 1	X = 0	X = 1	
000	000	001	0	0	0	0	0	1	**0**
001	010	010	0	0	1	1	0	0	**0**
010	011	011	0	0	1	1	1	1	**0**
011	100	100	1	1	0	0	0	0	**1**
100	101	101	1	1	0	0	1	1	**0**
101	000	000	0	0	0	0	0	0	**1**
110	x	x	x	x	x	x	x	x	x
111	x	x	x	x	x	x	x	x	x

(X: input, x: don't care terms) **FSM Reduced State Transition Table**

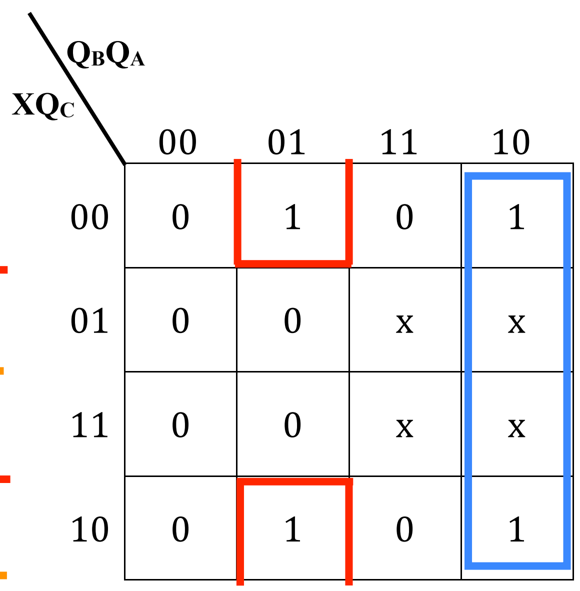
As can be seen, states cannot be reduced further.

D_A



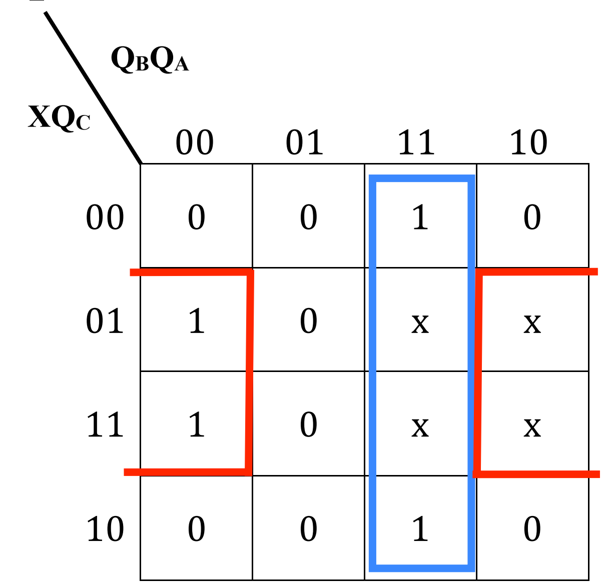
XQ_C \ Q_BQ_A	00	01	11	10
00	0	0	0	1
01	1	0	x	x
11	1	0	x	x
10	1	0	0	1

D_B



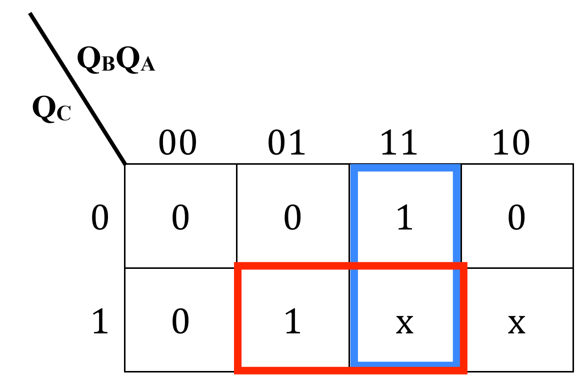
XQ_C \ Q_BQ_A	00	01	11	10
00	0	1	0	1
01	0	0	x	x
11	0	0	x	x
10	0	1	0	1

D_C



XQ_C \ Q_BQ_A	00	01	11	10
00	0	0	1	0
01	1	0	x	x
11	1	0	x	x
10	0	0	1	0

Y



Q_C \ Q_BQ_A	00	01	11	10
0	0	0	1	0
1	0	1	x	x

$$D_A = X\overline{Q_A} + Q_B\overline{Q_A} + Q_C\overline{Q_A}$$
$$D_B = \overline{Q_C}\,\overline{Q_B}Q_A + Q_B\overline{Q_A}$$
$$D_C = Q_BQ_A + Q_C\overline{Q_A}$$
$$Y = Q_CQ_A + Q_BQ_A$$

Minimized SoP Expressions

D flip flop implementation in Verilog

```verilog
module d_flip_flop (input d, input clk, output reg q, output reg qn);
always @(posedge clk) begin
    q <= d;
    qn <= ~d;
end
endmodule
```

Note: The D flip flop is positive edge-triggered.

Sequence generator implementation in Verilog

```verilog
module proj (input x, input clk, output reg y, output reg lck);
//initialising wires connected to D flip flops
//names ending in n are wires connected to complemented outputs of D flip flops
wire qa;
wire qan;
wire qb;
wire qbn;
wire qc;
wire qcn;
always @(clk) begin
    lck <= clk; //shows clock state as output through an LED
end
//initialising D flip flops
//notice minimized SoP expressions given as input to D flip flop modules
    d_flip_flop d0 (.d((qb & qan) | (x & qan) | (qc & qan)), .clk(clk), .q(qa),
.qn(qan));
    d_flip_flop d1 (.d((qa & qbn & qcn) | (qan & qb)), .clk(clk), .q(qb),
.qn(qbn));
    d_flip_flop d2 (.d((qc & qan) | (qa & qb)), .clk(clk), .q(qc), .qn(qcn));
always @(posedge clk) begin
y = ((qc & qa) | (qb & qa)); //reassign output at positive edge of clock
end
endmodule
```

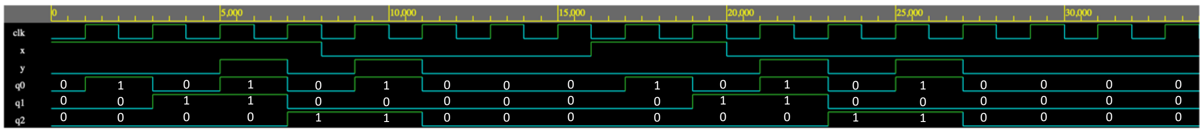
The code was implemented on CPLD board and demonstrated in lab. It was shown to be working as expected.

Verilog Testbench Code

```
`timescale 1ns/1ps
module project_tb;
  reg clk, x;
  wire y, z;
  proj uut (
    .clk(clk),
    .x(x),
    .y(y),
    .lck(z)
  );

  initial begin
    $dumpfile("dump.vcd");
    $dumpvars(0);
    clk = 0;
  end
  always #1 clk = ~clk;
  initial begin
    x = 1;
    #8
    x = 0;
    #8
    x = 1;
    #4;
    x = 0;
    #6;
    #8
    $finish;
  end
endmodule
```

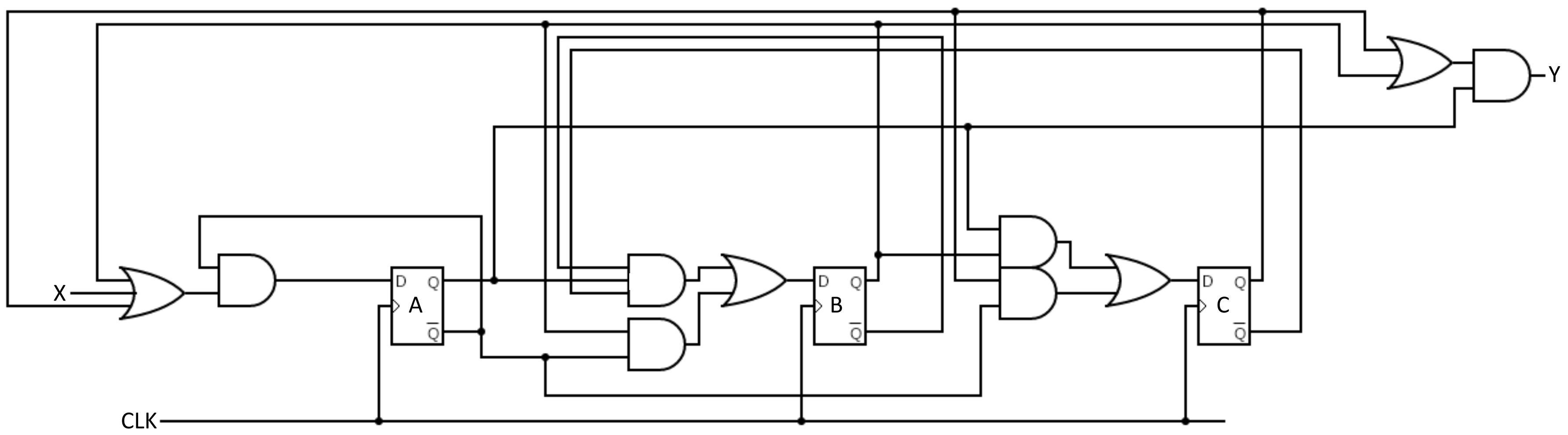
Simulated Waveforms



The FSM states Q_C to Q_A are labelled from bottom, moving upwards.
It is noticed that on reaching the idle state 000, the FSM stays in the idle state as long as the input X is 0 at positive edge transitions of clock input. When the input X is 1 at a positive edge transition of clock input, then the FSM moves to the next state 001 and continues to generate the sequence until the idle state, where it may stay or may regenerate the sequence, depending on the input X.
Observing the waveform, we may conclude that the Verilog implementation works as expected.

Circuit Diagram

Black points denote junctions.



$$D_A = X\overline{Q_A} + Q_B\overline{Q_A} + Q_C\overline{Q_A} = \overline{Q_A}(X + Q_B + Q_C)$$
$$D_B = \overline{Q_C}\,\overline{Q_B}Q_A + Q_B\overline{Q_A}$$
$$D_C = Q_BQ_A + Q_C\overline{Q_A}$$
$$Y = Q_CQ_A + Q_BQ_A = Q_A(Q_C + Q_B)$$

The expressions on the rightmost end are implemented and shown in the circuit diagram.