DCCDL LAB1

matlab

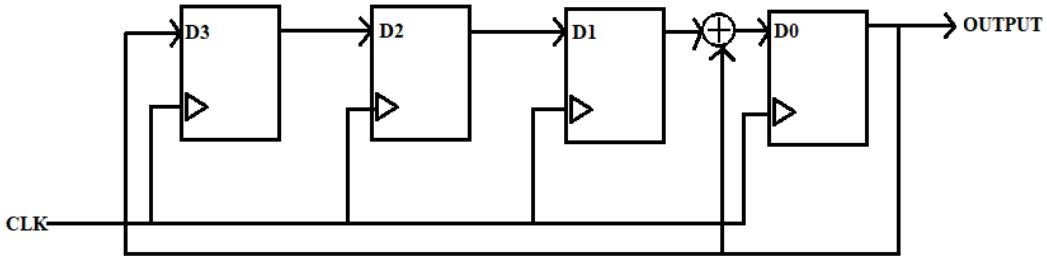
電機碩一　111521035　林豪澤

8.

Draw the block diagram of the LFSR with coefficients 23_{oct}. (10%)

$23_{oct} = 10011_{bin}$

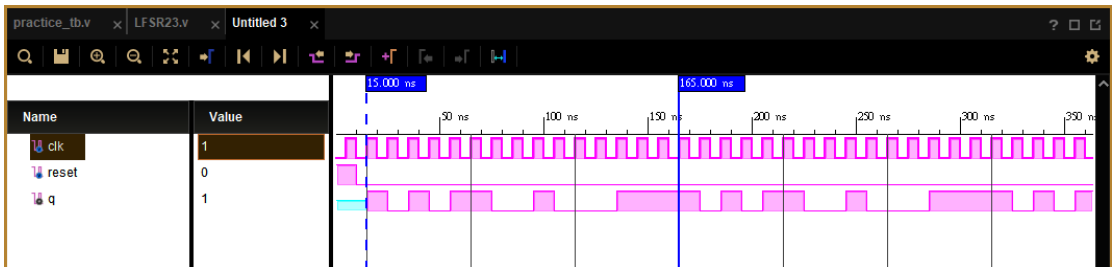
$g(D) = D^3+D+1$



Implement block diagram by Verilog to see if the RTL simulation results are the same as the Matlab results.Print out the Verilog codes. (20%)

```
1  `timescale 1ns / 1ps
2
3  module LFSR_23(clk, reset, Q);
4      input clk;
5      input reset;
6      output Q;
7      reg Q;
8      reg [3:0] q;
9
10 always @(posedge clk)
11     begin
12         if(reset)
13         begin
14             q <= 4'b1111;
15         end
16         else
17
18 /* use 23(oct) to generate m-sequence.
19     D^3+D+1                                                    */
20         begin
21             q <= { q[0], q[3:2], q[0]^q[1]};
22             Q <=  q[0];
23         end
24     end
25 endmodule
26
```

Print out the timing diagram of RTL simulation.

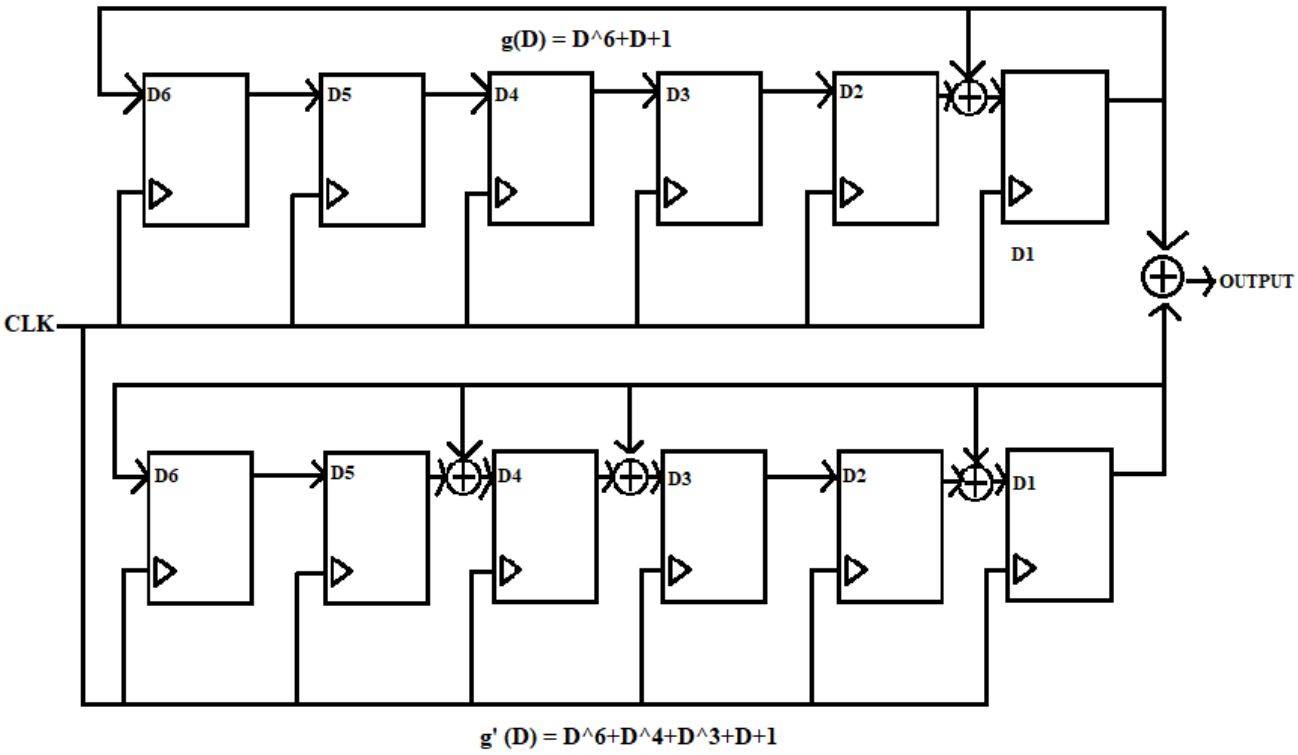


The result is q: [1 0 1 0 1 1 0 0 1 0 0 0 1 1 1], same as the result of Matlab.(start at 15ns, end at 165ns.)

9.

Draw the block diagram of your Gold code generator. (10%)

Upper LFSR's coefficient is 103_{oct} , g(D) = D^6+D+1.
lower LFSR 's coefficient is 133_{oct} , g'(D) = $D^6+D^4+D^3+D+1$



Print out Verilog code including test bench. (25%)

Use module ‘LFSR103’ to implement upper LFSR circuit with 103_{oct}.

```
1  `timescale 1ns / 1ps
2
3  module LFSR103(clk, reset, Q);
4      parameter degree = 6;
5      input clk;
6      input reset;
7      output [degree-1:0] Q;
8      reg [degree-1:0] Q;
9
10 always @(posedge clk)
11     begin
12         if(reset)
13         begin
14             Q <= 6'b110011;
15         end
16         else
17
18 // use 103(oct) to generate m-sequence.
19 // D^6+D+1
20         begin
21             Q <= { Q[0], Q[degree-1:2], Q[0]^Q[1]};
22         end
23     end
24 endmodule
25
```

Use ‘module LFSR133’ to implement lower LFSR circuit with 133_{oct}.

```
1  `timescale 1ns / 1ps
2
3  module LFSR133(clk, reset, Q);
4      parameter degree = 6;
5      input clk;
6      input reset;
7      output [degree-1:0] Q;
8      reg [degree-1:0] Q;
9
10 always @(posedge clk)
11     begin
12         if(reset)
13         begin
14             Q <= 6'b110110;
15         end
16         else
17
18 // use 133(oct) to generate m-sequence.
19 //  D^6+D^4+D^3+D+1
20         begin
21             Q <= {  Q[0],
22                     Q[degree-1],
23                     Q[4]^Q[0],
24                     Q[3]^Q[0],
25                     Q[2],
26                     Q[0]^Q[1]};
27         end
28     end
29 endmodule
30
```

Use module ‘Gold_top’ to implement all circuit in one circuit.

Gold_top.v | LFSR133.v | LFSR103.v | gold_tb.v

```
1  `timescale 1ns / 1ps
2
3  module Gold_top(clk, reset, Q1, Q2, Q3);
4      // Inputs
5          input clk;
6          input reset;
7      // Outputs;
8          output Q1;
9          output Q2;
10         output Q3;
11
12     wire[5:0] q1;
13     wire[5:0] q2;
14     reg Q1,Q2,Q3; // Q1 = b , Q2 = b' , Q3 = b+b'
15
16     LFSR103 lfsr103( .clk(clk), .reset(reset), .Q(q1));
17     LFSR133 lfsr133( .clk(clk), .reset(reset), .Q(q2));
18
19     always @(posedge clk)
20     begin
21         Q1 <= q1[0];
22         Q2 <= q2[0];
23         Q3 <= q1[0] ^ q2[0];
24     end
25
26 endmodule
27
```

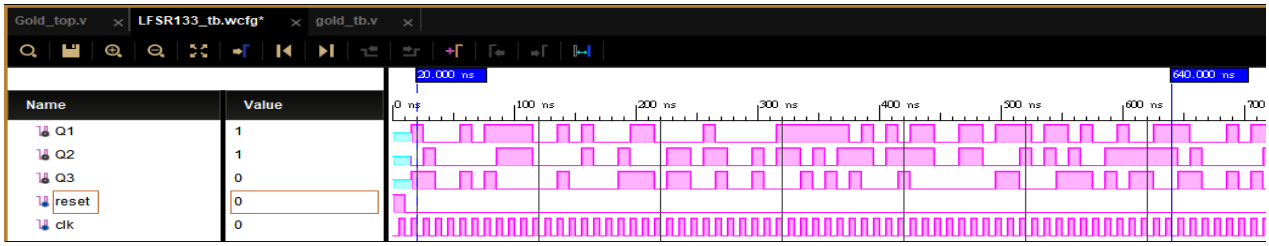
Testbench of ‘gold_top’

Gold_top.v | LFSR133.v | LFSR103.v | gold_tb.v

```
1  `timescale 1ns / 1ps
2
3  module gold_tb;
4
5      // Inputs
6          reg clk = 0;
7          reg reset = 0;
8      // Outputs
9          wire Q1; // b
10         wire Q2; // b'
11         wire Q3; // b+b'
12
13     Gold_top uut3  (.clk(clk),
14                     .reset(reset),
15                     .Q1(Q1),
16                     .Q2(Q2),
17                     .Q3(Q3));
18
19
20 always #5 clk=~clk;
21
22 initial begin
23     reset <=1;
24     #10 reset <=0;
25     #1890;$stop;
26 end
27
28
29 endmodule
```

Print out the timing diagram of three code sequences. (10%)

(start at 15ns, stop at 635ns)

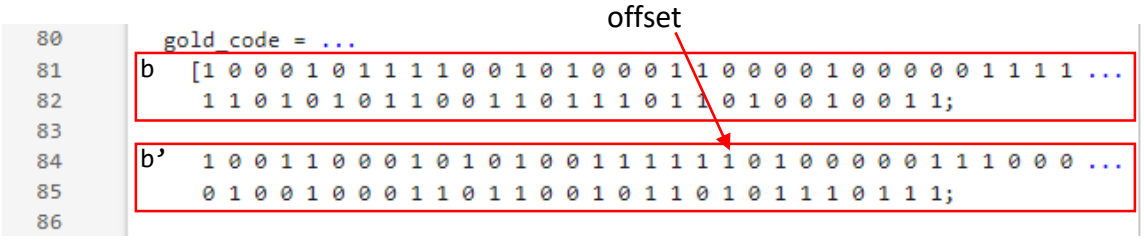


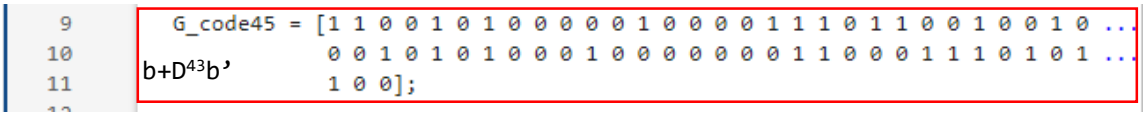
Q1:[1 0 0 0 1 0 1 1 1 1 0 0 1 0 1 0 0 0 1 1 0 0 0 0 1 0 0 0 0 0 1 1 1 1
1 1 0 1 0 1 0 1 1 0 0 1 1 0 1 1 1 0 1 1 0 1 0 0 1 0 0 1 1]

Q2:[1 0 1 0 0 0 0 0 1 1 1 0 0 0 0 1 0 0 1 0 0 0 1 1 0 1 1 0 0 1 0 1 1 0
1 0 1 1 1 0 1 1 1 1 0 0 1 1 0 0 0 1 0 1 0 1 0 0 1 1 1 1 1]

Q3:[1 1 0 0 1 0 1 0 0 0 0 0 1 0 0 0 0 1 1 1 0 1 1 0 0 1 0 0 1 0 0 0 1 0
1 0 1 0 0 0 1 0 0 0 0 0 0 0 1 1 0 0 0 1 1 1 0 1 0 1 1 0 0]

Show that three code sequences generated by Verilog are part of the sequences in Q5. (10%)





如圖所示，b、b’、b+b’為 Q5 所產生的前三個 gold code sequences。

而 Verilog 所產生的 Q1 與 matlab 中的 gold code 第一個 sequence 一模一樣。

Q2 則是與 b’相差了 offset 21 個 bits，為相同 sequence。

Q3 由 Q2 與 Q1 運算而來，為 Q5 所產生的第 45 個 gold code sequence。

由此可得知 Verilog 所產生的 code sequence 是 Q5 所產生的 gold code sequences 的一部分。

10. Why do Q9 and Q5 generate the same Gold code sequences? (5%)

使用同一組 initial state 生成 upper sequence 與 Q5 第一個生成的 gold code sequences 為同一個。lower sequence 與 Q5 中生成的 b’為相同 sequence，只是相差了 21 個 bits 的 offset。同理位移過 21 次的 b’與 b 做互斥或運算與 Q5 中第 45 個 gold code sequence 一樣。因此 Q9 與 Q5 生成相同的 gold code sequences。