EECS 2070 02 Digital Design Labs 2019 Lab 2
學號：104021215　姓名：熊磊

1. 實作過程

- Lab2_1 是要做一個 4bit 的 positive-edge-triggered，因此我在設計時先做一個 DFF，接著在 module lab2_1 中使用 DFF 一個就可以做 counter。

```
22  module DFF (clk, init_out, next_out, out, rst_n);
23      parameter n = 1;
24      input clk,rst_n;
25      input [n-1:0] next_out,init_out;
26      output reg [n-1:0] out;
27      always@(posedge clk, negedge rst_n) begin
28          if(rst_n==0)begin
29              out <= init_out;
30          end
31          else begin
32              out <= next_out;
33          end
34      end
35  endmodule
```

DFF 會把 out 在 posedge 時設成 next_out，而在 reset = 0 時，也就是 negedge rst_n 時，out 設成初始值 0000。

```
37  module lab2_1(clk, rst_n, en, dir, in, data, out);
38      input clk, rst_n, en, dir, in;
39      input [3:0] data;
40      output [3:0] out;
41
42      wire [3:0]next_out;
43
44      DFF #(4) DFF1(clk, 4'b0, next_out, out, rst_n);
45      assign next_out = (en == 0) ? out : (in == 0) ? (dir == 1) ? out + 1'b1 : out - 1'b1 : data;
46  endmodule
```

[Testbench]

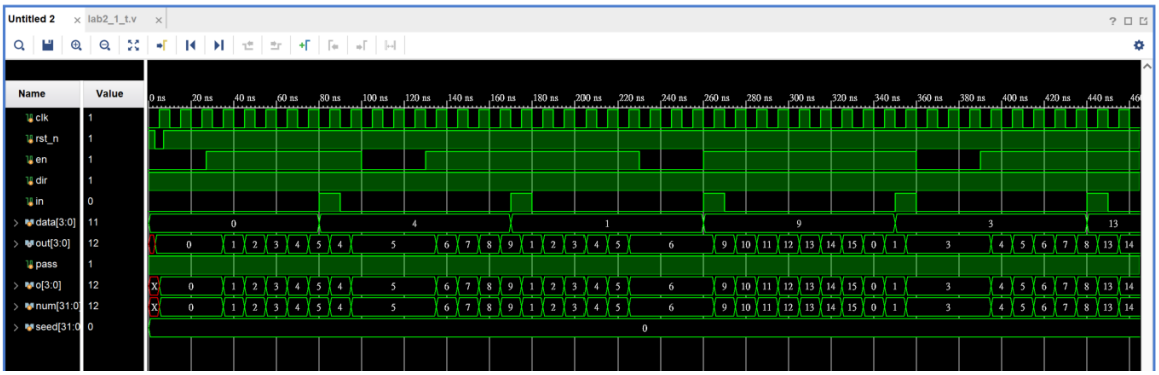
在設計 Testbench 的時候我利用 random 來隨機產生 in data 的值，而 dir 在 3000ns 後會換一個方向。Pass 一開始設成 1，其他 initial value 如圖

```
30      lab2_1 counter (.clk(clk), .rst_n(rst_n), .en(en), .dir(dir), .in(in), .data(data), .out(out));
31      integer num, seed;
32      initial begin
33          seed = 0;
34          clk = 0;
35          rst_n = 1;
36          en = 0;
37          dir = 1;
38          in = 0;
39          data[3:0] = 4'b0;
40          pass = 1;
41          #3
42          rst_n = 0;
43          #4
44          rst_n = 1;
45          #20
46          en = 1;
47          #3000
48          dir = 0;
49          #3000
50          dir = 1;
51          #10
52          if (pass == 1)
53              $display("-------------------------\n     -----[PASS]-----     \n-------------------------");
54          $finish;
55      end
```

而用來檢測是否正確的方式，在 testbench 中他每隔 2 個 ns 就會 check 一次，如果 lab2_1 個 module 出來的 out 和 testbench 值不一樣，就會把 pass 設成 0，也就會 Fail。

```
76      always @ (posedge clk or posedge rst_n) begin
77          if (!rst_n) begin
78              num <= 0;
79          end else begin
80              if (en == 1'b1 && in == 1'b0 && dir == 1'b1) begin
81                  num <= (num == 15) ? 4'b0 : num + 4'b1;
82              end else if (en == 1'b1 && in == 1'b0 && dir == 1'b0)begin
83                  num <= (num == 0) ? 4'b1111 : num - 4'b1;
84              end else if (en == 1'b1 && in == 1'b1) begin
85                  num <= data;
86              end
87          end
88      end
89
90      always@(*)begin
91          o = num;
92      end
93
94      always @ (out or o) begin
95          #2
96              if(en==1)
97              if (out !== o) begin
98                  pass = 0;
99                  $display("[NOT_PASS_1] : OUT : %d, num : %d", out[3:0], num);
100             end
101     end
```

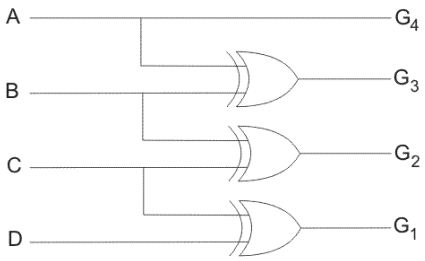
成功 Pass 的 Waveform



- lab2_2 則是要設計 Gray Code Counter，且要求要用 negative trigger。在設計 2 digits 之前，我先做 1 digit 的 counter.

```
37  module gc_1_counter (clk, rst_n, en, dir, gray, cout);
38      input clk, rst_n, dir, en;
39      output [3:0] gray;
40      output cout;
41
42      wire [3:0] next_bin, bin;
43
44      DFF #(4) DFF1(clk, 4'b0, next_bin, bin, rst_n);
45
46      assign next_bin = (en == 0) ? bin : (dir == 1) ? bin + 4'b1 : bin - 4'b1;
47      assign cout = (en == 1'b1 && dir == 1'b1 && bin == 4'b1111) ? 1'b1:
48                    (en == 1'b1 && dir == 1'b0 && bin == 4'b0) ? 1'b1 : 1'b0 ;
49      assign gray = {bin[3], bin[2:0] ^ bin[3:1]};
50  endmodule
```

Binary 轉 Gray Code 的方法其實很簡單，如下圖 MSB 的 Binary 和 Gray Code 相同，剩下的三個 Digit 只要平移做 XOR 就能得到 Gray Code。

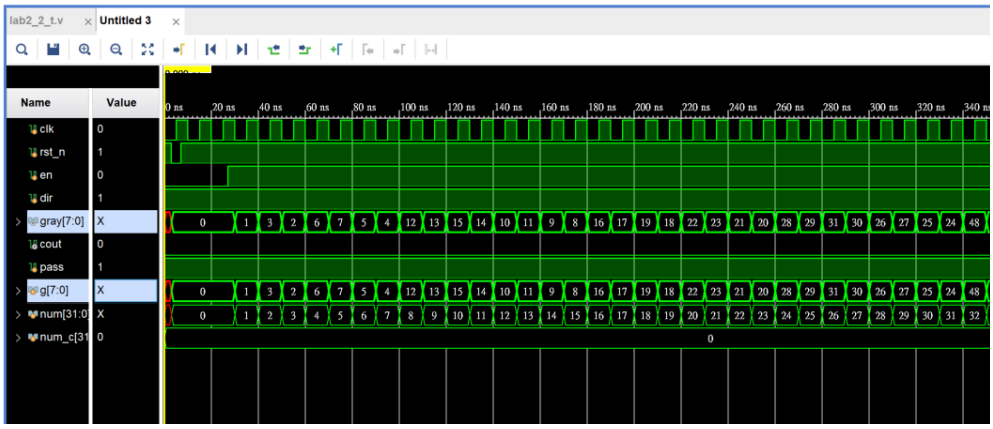


Logic Circuit for Binary to Gray Code Converter

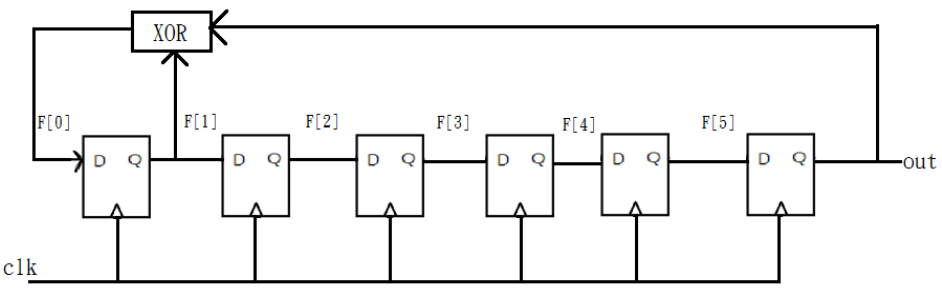
接著就可以做 2 digits Gray Code Counter。我們可以用 LSB 的 Carry out 來當作 MSB 的 enable，這樣就能組出 2 digits Gray Code Counter。

```
52  module gc_2_counter (clk, rst_n, en, dir, gray, cout);
53      input clk, rst_n, dir, en;
54      output [7:0] gray;
55      output cout;
56
57      wire cout_1, cout_2;
58      gc_1_counter g1(clk, rst_n, en, dir, gray[3:0], cout_1);
59      gc_1_counter g2(clk, rst_n, cout_1, dir, gray[7:4], cout_2);
60      assign cout = cout_1&&cout_2;
61  endmodule
```

成功 Pass 的 Waveform。



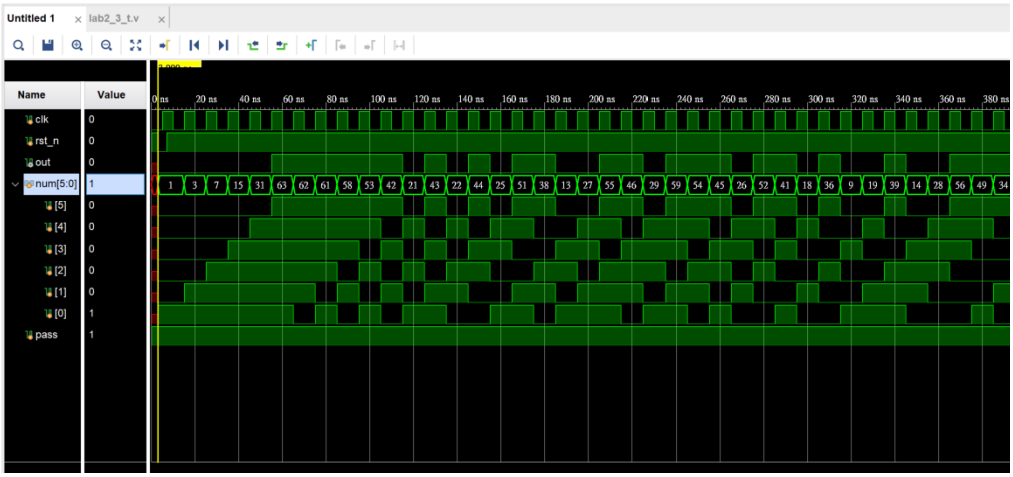
- lab2_3 是要設計一個 6-bit Linear feedback shift register (LFSR)，根據助教提供的 Block Diagram，我們 F[5]會是我們的 out，且每次的 F[0]，都是 F[5]跟 F[0]做 XOR。



因此我們可以將 lab2_3 做如下的設計，只要每次 clock positive-edge，就會把 F 的 bit 設為前一個位元的 bit。而 out 直接 assign 成 F[5]即可。

```
37  module lab2_3 (clk, rst_n, out);
38      input clk, rst_n;
39      output out;
40      wire [5:0] F;
41
42      DFF #(1) DFF0(clk, 1'b1, F[5]^F[0], F[0] ,rst_n);
43      DFF #(1) DFF1(clk, 1'b0, F[0], F[1] ,rst_n);
44      DFF #(1) DFF2(clk, 1'b0, F[1], F[2] ,rst_n);
45      DFF #(1) DFF3(clk, 1'b0, F[2], F[3] ,rst_n);
46      DFF #(1) DFF4(clk, 1'b0, F[3], F[4] ,rst_n);
47      DFF #(1) DFF5(clk, 1'b0, F[4], F[5] ,rst_n);
48
49      assign out = F[5];
50  endmodule
```

正確 Pass 的 waveform



[觀察] 在做完 LFSR 後，觀察發現，其實 LFSR 產生的 Pattern 是固定的順序，跟 initial value 有關，而且只會有 63 個值。

2. 學到的東西與遇到的困難

   這次的 Lab 蠻有趣的，而在做 Gray Code 時有點看不懂 Spec 的 2 digit 是什麼意思，因為一般來說 Gray Code 應該是一直延續下去，才發現原來是要成 1 digit for 4bits binary。

3. 想對老師或助教說的話

   謝謝助教/老師花時間幫我們 Demo，下一次要開始 synthesis，終於要開始做跟電有關的，希望能順利做出有去的遊戲。每個禮拜能做 Lab 時間只有 4 天，希望能提早公布 Lab 的 Spec:)