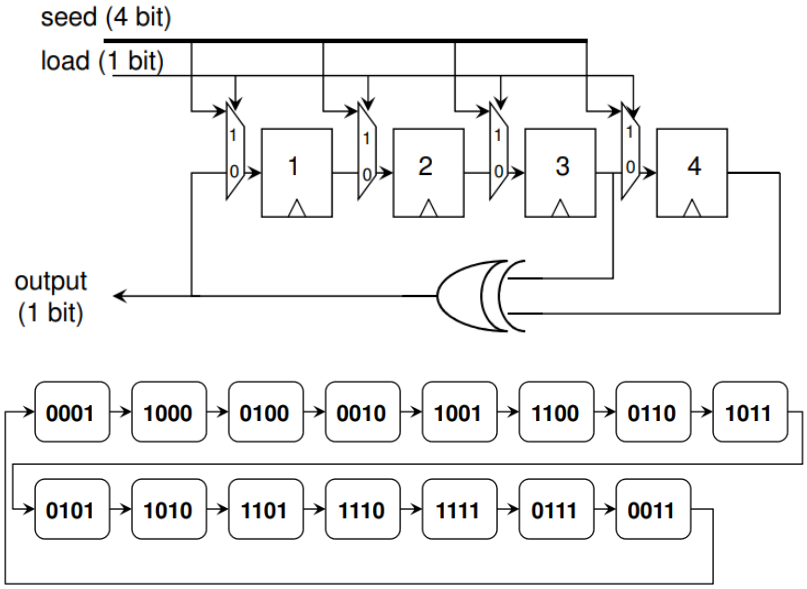
一、　報告

PART 1：Simulate the behavior of Verilog code

【STEP】

1. 安裝 ISE 14.7
2. 建立一個 new project
3. 這次的 code 是要用 XOR、D-Flip flop、2X1 MUX 實作一個 Linear Feedback Shift Register (如下圖)

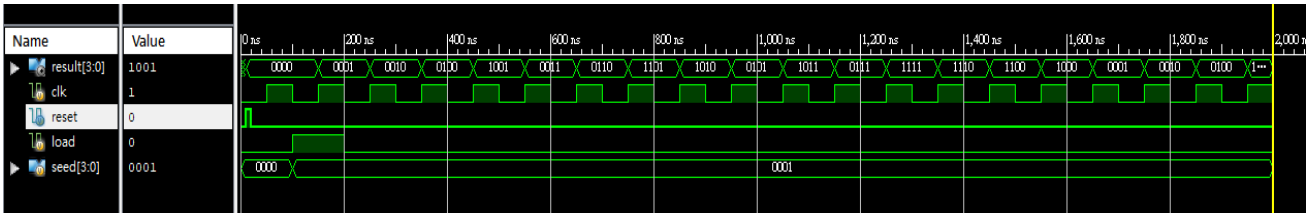


4. 建立 Verilog Module (2X1 MUX.v)
   - 若 Sel = 0，則 output = input_A
   - 若 Sel = 1，則 output = input_B
5. 加入 TA 提供的 source file (d_flipflop.v、lfsr.v)
6. Compile program
   - Systhesize
7. 建立 test bench 拿來測試 lfsr.v (lfsr_tb.v)
   - Verilog Test Fixture

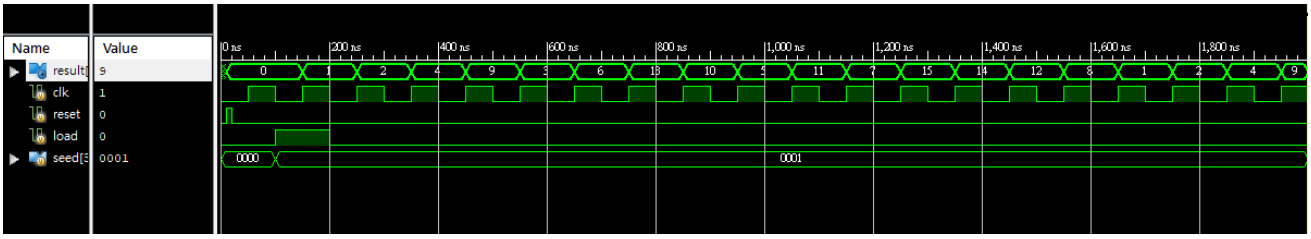
```
// Instantiate the Unit Under Test (UUT)
    lfsr uut (
        .clk(clk),
        .reset(reset),
        .load(load),
        .seed(seed),
        .result(result)
    );
```

```
        initial begin
        reset = 0;
        #10 reset = 1; // reset pulse
        #10 reset = 0;
        end

        initial clk = 0;
        always #50 clk = !clk; // generate clock

        initial begin
             load = 0;
             seed = 0;
             #100 seed = 4'b0001;
             load = 1;
             #100 load = 0;
        end
        initial #2000 $finish;
        initial $monitor($realtime,"ns %h %h %h\n", load, seed,
       result);
    endmodule
```

【圖一】未改 result[3:0]



【圖二】已改成 unsigned decimal



PART 2：Learn how to explain RISC-V code

1. Reference

ABI name	Register	Description
Ra	x1	Return address
Sp	x2	Stack pointer
s0	x8	Saved register
a0-1	x10-11	Function arguments Return values
a2-3	x12-13	Function arguments

2. Explanation of RISC-V code

```
multiplication(int, int):          # @multiplication(int, int)
        addi    sp, sp, -32        set stack pointer to -32
                                   create 32 bytes-space
        sd      ra, 24(sp)         store double word return address(8 bytes)
                                   mem[-7 -> 0]
        sd      s0, 16(sp)         store double word saved register(8 bytes)
                                   mem[-15 -> -8]
        addi    s0, sp, 32         saved register 指向原點(0)
        add     a2, zero, a1       a2 = a1 + 0 = 3
        add     a3, zero, a0       a3 = a0 + 0 = 2
        sw      a0, -20(s0)        store word(4 bytes) mem[-19 -> -16] to a0
        sw      a1, -24(s0)        store word(4 bytes) mem[-23 -> -20] to a1
        lw      a0, -20(s0)        load mem[-19 -> -16] to a0 (a0 = 2)
        lw      a1, -24(s0)        load mem[-23 -> -20] to a1 (a1 = 3)
        mulw    a0, a0, a1         a0 = a0 * a1 (a0 = 2*3 = 6)
        ld      s0, 16(sp)         load doubleword mem[-15 -> -8] to s0
        ld      ra, 24(sp)         load doubleword mem[-7 -> 0] to ra
        addi    sp, sp, 32         set stack pointer back to 0
        ret
main:                                   # @main
        addi    sp, sp, -32        set stack pointer to -32
                                   create 32 bytes-space
        sd      ra, 24(sp)         store double word return address(8 bytes)
                                   mem[-7 -> 0]
        sd      s0, 16(sp)         store double word saved register(8 bytes)
                                   mem[-15 -> -8]
        addi    s0, sp, 32         set saved register to point to 0
```

```
addi    a0, zero, 2         a0 = 2
sw      a0, -20(s0)         store word a0(4 bytes) to mem[-19 -> -16]
addi    a0, zero, 3         a0 = 3
sw      a0, -24(s0)         store word a0(4 bytes) to mem[-23 -> -20]
lw      a0, -20(s0)         load word mem[-19 -> -16] to a0
lw      a1, -24(s0)         load word mem[-23 -> -20] to a1
call    multiplication(int, int)
sw      a0, -28(s0)         store word a0(result) to mem[-27 -> -24]
mv      a0, zero            set a0 to zero
ld      s0, 16(sp)          load doubleword mem[-15 -> -8] to s0
ld      ra, 24(sp)          load doubleword mem[-7 -> 0] to ra
addi    sp, sp, 32          set stack pointer back to 0
ret
```

二、 遭遇困難及解決方法

1. ISE 一直不能 run，一直說無法找到檔案

【解】自己 debug 出來了

檔名路徑不能用中文他會讀成亂碼，導致無法編譯

2. 因為所有的波形都是 0 和 1 組成，要怎麼用十進制表示？

【解】有詢問 TA

舉個例子 reg [3:0] Test;

可由 4 個 register 所組成每個都只能表示 0 及 1

那他的值 rang: 0000~1111 換算成 10 進制就是 0~15

同理這次作業也是一樣 只要用 tool 中的選項

將 bus 轉換成 10 進制的表示方法即可.

再用一個例子 Test [3:0] = 4'b1100

表示時進制 12 但 bus 上各個 element 的表示為 1/0

Test[3]=1, Test[2]=1, Test[1]=0 , Test[0]=0

三、 作業心得討論

這是我第一次學習打 verilog，之前數位電路設計課上老師只有考紙本的講解 code，我還以為自己以前學過「看 verilog」可能輕鬆許多，結果第一次 compile 就遇到這麼多問題，也問了 TA 很多次，很感謝 TA 們都超細心地回答我的問題，我這次 lab 也學到了不少。