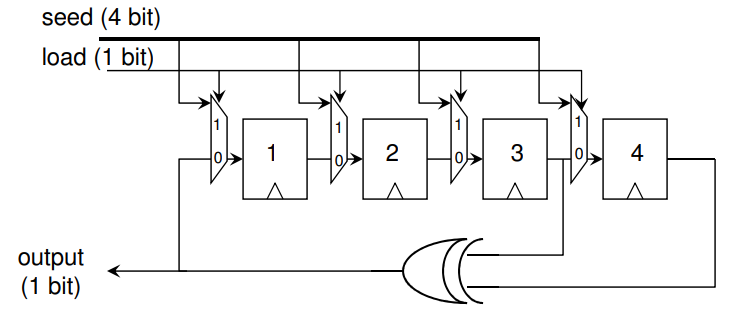
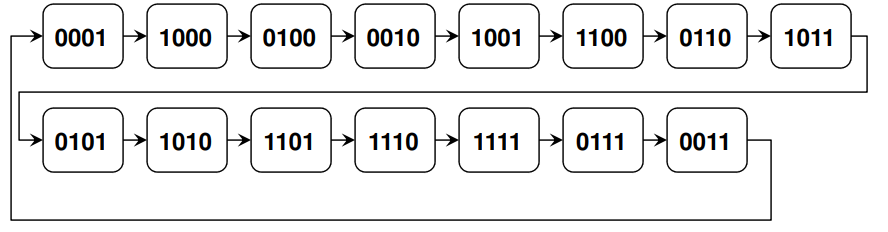
1. **報告**

**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)

1. 建立Verilog Module (2X1 MUX.v)
   * 若Sel = 0，則output = input\_A
   * 若Sel = 1，則output = input\_B
2. 加入TA提供的source file (d\_flipflop.v、lfsr.v)
3. Compile program
   * Systhesize
4. 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

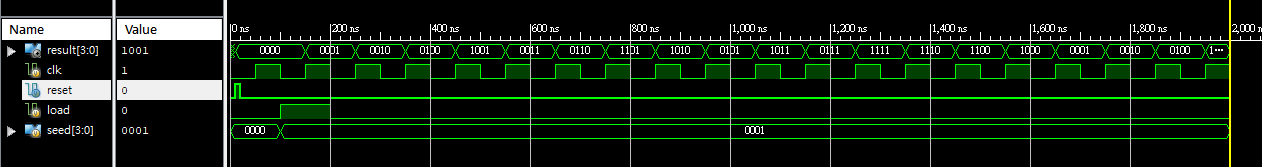
initial #2000 $finish;

initial $monitor($realtime,"ns %h %h %h\n", load, seed,

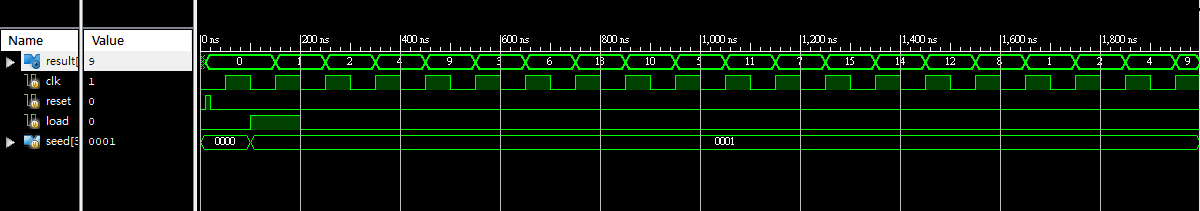
result);

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 |

1. **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

1. **遭遇困難及解決方法**
2. 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

1. 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

1. **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。