

# <N26112437>\_<劉兆軒> AIAS 2022 Lab 6 HW Submission

---

- <N26112437>\_<劉兆軒> AIAS 2022 Lab 6 HW Submission
  - 注意事項
  - Gitlab 連結
  - Hw6-1 TrafficLight with Pedestrian button
    - Scala Code
    - Waveform
  - Hw6-2-1 Negative Integer Generator
    - Scala Code
    - Test Result
  - Hw6-2-2 N operands N-1 operators(+ 、 -)
  - Hw6-2-3 Order of Operation (+ 、 - 、 \* 、 ( 、 ))
    - Scala Code
    - Test Result
  - Hw6-3-1 Pseudo Random Number Generator
    - Scala Code
    - Test Result
  - Hw6-3-2 1A2B game quiz
    - Scala Code
    - Test Result
  - Bonus : 1A2B hardware solver
    - Scala Code
    - Test Result
  - 文件中的問答題
  - 意見回饋和心得(可填可不填)

## 注意事項

---

1. 請不要用這份template 交作業, 建立一個新的codimd 檔案, 然後copy & paste 這個template 到你創建的文件做修改。
2. 在上面的title修改你的學號與姓名, 避免TA修改作業時把檔案跟人弄錯。
  - Ex: E24062133\_徐韋凱 AIAS 2022 Lab5 HW Submission

3. 在Playlab 作業中心繳交作業時, 請用你創建的檔案鏈結繳交, 其他相關的資料與鏈結請依照 Template 規定的格式記載於codimd 上。

## Gitlab 連結

---

Please paste the link to your private Gitlab repository for this homework submission here.

- Gitlab link (<https://playlab.computing.ncku.edu.tw:4001/kevin1217/Lab06>) <- 將連結填入括號中。

## Hw6-1 TrafficLight with Pedestrian button

---

### Scala Code

請放上你的程式碼並加上註解(中英文不限)，讓 TA明白你是如何完成的。



```
1 package aias_lab6.Hw1
2
3 import chisel3._
4 import chisel3.util._
5
6 class TrafficLight_p(Ytime:Int, Gtime:Int, Ptime:Int) extends Module{
7   val io = IO(new Bundle{
8     val P_button = Input(Bool())
9     val H_traffic = Output(UInt(2.W))
10    val V_traffic = Output(UInt(2.W))
11    val P_traffic = Output(UInt(2.W))
12    val timer      = Output(UInt(5.W))
13  })
14
15  //please implement your code below...
16  //parameter declaration
17  val Off = 0.U
18  val Red = 1.U
19  val Yellow = 2.U
20  val Green = 3.U
21  val PG = 1.U
22
23
24
25  val sIdle :: sHGV :: sHYVR :: sHRVG :: sHRVY :: sPG :: Nil = Enum(6)
26
27  //State register
28  val state = RegInit(sIdle)
29  val ori_state = RegInit(sIdle)
30
31  //Counter=====
32  val cntMode = WireDefault(0.U(2.W))
33  val cntReg = RegInit(0.U(4.W))
34  val cntDone = Wire(Bool())
35  cntDone := cntReg === 0.U
36
37  when(cntDone){
38    //重置
39    when(cntMode === 0.U){
40      //綠燈
41      cntReg := (Gtime-1).U
42    }.elsewhen(cntMode === 1.U){
43      //黃燈
44      cntReg := (Ytime-1).U
45    }.elsewhen(cntMode === 2.U){
46      //行人
47      cntReg := (Ptime-1).U
48    }
49  }.otherwise{
50    //cntDone不等於0時，不斷的將cntReg-1做倒數
51    cntReg := cntReg - 1.U
52  }
53  //Counter end=====
```

```
54 //Next State Decoder
55 //按下行人通行，初始化cntReg
56 //ori_state := state
57 when(io.P_button) {
58     state := sPG
59     cntReg := 4.U
60 }
61 .otherwise{
62     switch(state){
63         is(sIdle){
64             state := sHGVR
65         }
66         //行人開始倒數，完成切回原狀態，並且原時間重新計時
67         is(sPG){
68             when(cntDone) {
69                 state := ori_state
70                 when(ori_state === sHGVR || ori_state === sHRVG)
71                 {
72                     cntReg:= 6.U
73                 }
74                 .elsewhen(ori_state === sHYVR || ori_state === sHRVY)
75                 {
76                     cntReg:= 2.U
77                 }.otherwise{cntReg:= 10.U}
78             }
79         }
80         is(sHGVR){
81             ori_state := state
82             when(cntDone) {
83                 state := sHYVR
84                 ori_state := sHYVR
85             }
86         }
87         is(sHYVR){
88             ori_state := state
89             when(cntDone) {
90                 state := sHRVG
91                 ori_state := sHRVG
92             }
93         }
94         is(sHRVG){
95             ori_state := state
96             when(cntDone) {
97                 state := sHRVY
98                 ori_state := sHRVY
99             }
100         }
101     }
102     is(sHRVY){
103         ori_state := state
104         when(cntDone) {
105             state := sHGVR
106             ori_state := sHGVR
```

```
107         }
108
109     }
110 }
111 }
112
113
114 //Output Decoder
115 //Default statement
116 cntMode := 0.U
117 io.H_traffic := Off
118 io.V_traffic := Off
119 io.P_traffic := Off
120
121 switch(state){
122     is(sHGVR){
123         cntMode := 1.U
124         io.H_traffic := Green
125         io.V_traffic := Red
126     }
127     is(sHYVR){
128         cntMode := 0.U
129         io.H_traffic := Yellow
130         io.V_traffic := Red
131     }
132     is(sHRVG){
133         cntMode := 1.U
134         io.H_traffic := Red
135         io.V_traffic := Green
136     }
137     is(sHRVY){
138         cntMode := 0.U
139         io.H_traffic := Red
140         io.V_traffic := Yellow
141     }
142     is(sPG){
143         cntMode := 2.U
144         io.H_traffic := Red
145         io.V_traffic := Red
146         io.P_traffic := PG
147     }
148 }
149
150 io.timer := cntReg
151 }
```

The figure displays two timing diagrams for digital signals. The top diagram covers the time range from 0 ns to 60 ns, and the bottom diagram covers the time range from 80 ns to 140 ns. Both diagrams show a clock signal (clock=1) and four data signals: io\_timer[4:0], io\_H\_traffic[1:0], io\_V\_traffic[1:0], and io\_P\_button.

**Top Timing Diagram (0 ns to 60 ns):**

- clock=1:** A periodic square wave with a period of approximately 10 ns.
- io\_timer[4:0]:** A 5-bit counter that increments from 00000 to 00001 at approximately 10 ns, then continues to increment in steps of 1 until it reaches 00010 at approximately 60 ns.
- io\_H\_traffic[1:0]:** A 2-bit signal that is 11 from 0 ns to approximately 10 ns, then 10 from approximately 10 ns to approximately 40 ns, and finally 11 from approximately 40 ns to 60 ns.
- io\_V\_traffic[1:0]:** A 2-bit signal that is 01 from 0 ns to approximately 30 ns, then 11 from approximately 30 ns to approximately 50 ns, and finally 01 from approximately 50 ns to 60 ns.
- io\_P\_button:** A signal that is 00 from 0 ns to approximately 10 ns, then 01 from approximately 10 ns to approximately 60 ns.

**Bottom Timing Diagram (80 ns to 140 ns):**

- clock=1:** A periodic square wave with a period of approximately 10 ns.
- io\_timer[4:0]:** A 5-bit counter that continues from the previous diagram, incrementing from 00010 to 00011 at approximately 80 ns, then continues to increment in steps of 1 until it reaches 00010 at approximately 140 ns.
- io\_H\_traffic[1:0]:** A 2-bit signal that is 01 from 80 ns to approximately 90 ns, then 10 from approximately 90 ns to approximately 110 ns, and finally 01 from approximately 110 ns to 140 ns.
- io\_V\_traffic[1:0]:** A 2-bit signal that is 11 from 80 ns to approximately 100 ns, then 01 from approximately 100 ns to approximately 120 ns, and finally 11 from approximately 120 ns to 140 ns.
- io\_P\_button:** A signal that is 00 from 80 ns to approximately 100 ns, then 01 from approximately 100 ns to approximately 120 ns, and finally 00 from approximately 120 ns to 140 ns.

## Scala Code

請放上你的程式碼並加上註解(中英文不限)，讓 TA明白你是如何完成的。





```
1 package aias_lab6.Hw2
2
3 import chisel3._
4 import chisel3.util._
5
6 class NegIntGen extends Module{
7     val io = IO(new Bundle{
8         val key_in = Input(UInt(4.W))
9         val value = Output(Valid(UInt(32.W)))
10    })
11
12    //please implement your code below
13    val equal = WireDefault(false.B)
14    equal := io.key_in === 15.U
15    val num = WireDefault(false.B)
16    num := io.key_in < 10.U
17
18
19
20    val sIdle :: sAccept :: sEqual :: Nil = Enum(3)
21    val state = RegInit(sIdle)
22    //Next State Decoder
23    switch(state){
24        is(sIdle){
25            state := sAccept
26        }
27        is(sAccept){
28            when(equal) {state := sEqual}
29        }
30        is(sEqual){
31            state := sAccept
32        }
33    }
34
35    val in_buffer = RegNext(io.key_in)
36    val number = RegInit(0.U(32.W))
37    val neg = RegInit(0.U(1.W))
38
39    when(state === sAccept){
40        when(in_buffer < 10.U){
41            number := (number<<3.U) + (number<<1.U) + in_buffer
42            //number := (number<<6.U) + (number<<5.U) + (number<<2.U) + in_buf
43            .elsewhen(in_buffer === 11.U){neg := 1.U}
44
45        }.elsewhen(state === sEqual){
46            in_buffer := 0.U
47            number := 0.U
48            neg := 0.U
49        }
50
51    io.value.valid := Mux(state === sEqual,true.B,false.B)
52        when(neg===1.U){io.value.bits := -number}
53        .otherwise{io.value.bits := number}
```

54 | }

## Test Result

```
Elaborating design...
Done elaborating.
[info] [0.004] SEED 1681797729019
[info] [0.009] Test 1 : pass!
[info] [0.011] Test 2 : pass!
[info] [0.012] Test 3 : pass!
test NegIntGen Success: 0 tests passed in 29 cycles in 0.041764 seconds 694.38 Hz
[info] [0.013] RAN 24 CYCLES PASSED
```

## Hw6-2-2 N operands N-1 operators(+ 、 -)

---

### Scala Code

請放上你的程式碼並加上註解(中英文不限)，讓 TA明白你是如何完成的。



```
1 package aias_lab6.Hw2
2
3 import chisel3._
4 import chisel3.util._
5
6 class LongCal extends Module{
7     val io = IO(new Bundle{
8         val key_in = Input(UInt(4.W))
9         val value = Output(Valid(UInt(32.W)))
10    })
11
12    //please implement your code below
13    val operator = WireDefault(false.B)
14    operator := io.key_in >= 10.U && io.key_in <= 12.U
15
16    val num = WireDefault(false.B)
17    num := io.key_in < 10.U
18
19    val equal = WireDefault(false.B)
20    equal := io.key_in === 15.U
21
22    val L_brack = WireDefault(false.B)
23    L_brack := io.key_in === 13.U
24
25    val R_brack = WireDefault(false.B)
26    R_brack := io.key_in === 14.U
27
28    //Reg Declaration=====
29    val in_buffer = RegNext(io.key_in)
30    val src1 = RegInit(0.U(32.W))
31    val op = RegInit(0.U(2.W))
32    val src2 = RegInit(0.U(32.W))
33    val not_op = RegInit(0.U(1.W))
34
35
36    //State and Constant Declaration=====
37    val sIdle :: sSrc1 :: sOp :: sSrc2 :: sEqual :: Nil = Enum(5)
38    val add = 0.U
39    val sub = 1.U
40    val mul = 2.U
41    val state = RegInit(sIdle)
42
43    //Next State Decoder
44    switch(state){
45        is(sIdle){
46            when(L_brack){not_op := 1.U}
47            state := sSrc1
48        }
49
50        is(sSrc1){
51            when(L_brack){not_op := 1.U}
52            when(R_brack){not_op := 0.U}
53            when(equal) {state := sEqual}
```

```

54
55         when(not_op ==0.U){
56             when(operator) {state := sOp}
57         }
58     }
59
60     is(sOp){
61         when(num) {state := sSrc2}
62             when(L_brack){
63                 state := sSrc2
64                 not_op := 1.U
65             }
66     }
67
68     is(sSrc2){
69         when(L_brack){not_op := 1.U}
70         when(R_brack){not_op := 0.U}
71         when(equal) {state := sEqual}
72         when(not_op ==0.U){
73             when(operator) {state := sOp}
74         }
75     }
76
77     is(sEqual){
78         state := sSrc1
79     }
80 }
81 //=====
82 when(state == sSrc1){
83     when(in_buffer < 10.U){src1 := (src1<<3.U) + (src1<<1.U) + in_bu
84     when(in_buffer ==14.U){src1 := -src1}
85 }
86 when(state == sSrc2){
87     when(in_buffer < 10.U){src2 := (src2<<3.U) + (src2<<1.U) + in_bu
88     when(in_buffer ==14.U){src2 := -src2}
89 }
90 when(state == sOp){
91     op := in_buffer - 10.U
92     src1 := MuxLookup((op),0.U,Seq(
93         add -> (src1 + src2),
94         sub -> (src1 - src2),
95         mul -> (src1 * src2)
96     ))
97     src2 :=0.U
98 }
99
100 when(state == sEqual){
101     src1 := 0.U
102     src2 := 0.U
103     op := 0.U
104     in_buffer := 0.U
105     not_op := 0.U
106 }

```

```
107
108     io.value.valid := Mux(state === sEqual,true.B,false.B)
109     io.value.bits := MuxLookup(op,0.U,Seq(
110         add -> (src1 + src2),
111         sub -> (src1 - src2),
112         mul -> (src1 * src2)
113     ))
114 }
```

## Test Result

```
Elaborating design...
Done elaborating.
[info] [0.004] SEED 1681797875604
[info] [0.012] Test 1 : pass
[info] [0.017] Test 2 : pass
[info] [0.019] Test 3 : pass
[info] [0.020] Test 4 : pass
test LongCal Success: 0 tests passed in 66 cycles in 0.049990 seconds 1320.27 Hz
[info] [0.021] RAN 61 CYCLES PASSED
```

## Hw6-2-3 Order of Operation (+ 、 - 、 \* 、 ( 、 ))

- 如果你有完成**Bonus**部分，請在此註明。

## Scala Code

請放上你的程式碼並加上註解(中英文不限)，讓 TA明白你是如何完成的。



```
1 package aias_lab6.Hw2
2
3 import chisel3._
4 import chisel3.util._
5
6
7
8
9 class CpxCal extends Module{
10     val io = IO(new Bundle{
11         val key_in = Input(UInt(4.W))
12         val value = Output(Valid(UInt(32.W)))
13     })
14
15     //please implement your code below
16     val operator = WireDefault(false.B)
17     operator := io.key_in >= 10.U && io.key_in <= 12.U
18
19     val num = WireDefault(false.B)
20     num := io.key_in < 10.U
21
22     val equal = WireDefault(false.B)
23     equal := io.key_in === 15.U
24
25     val L_brack = WireDefault(false.B)
26     L_brack := io.key_in === 13.U
27
28     val R_brack = WireDefault(false.B)
29     R_brack := io.key_in === 14.U
30
31     //Reg Declaration=====
32     val in_buffer = RegNext(io.key_in)
33     val src1 = RegInit(0.U(32.W))
34     val op = RegInit(0.U(2.W))
35     val src2 = RegInit(0.U(32.W))
36     val not_op = RegInit(0.U(1.W))
37
38
39     //State and Constant Declaration=====
40     val sIdle :: sSrc1 :: sOp :: sSrc2 :: sEqual :: Nil = Enum(5)
41     val add = 0.U
42     val sub = 1.U
43     val mul = 2.U
44     val state = RegInit(sIdle)
45
46     //Next State Decoder
47     switch(state){
48         is(sIdle){
49             when(L_brack){not_op := 1.U}
50             state := sSrc1
51         }
52
53         is(sSrc1){
```



```

54         when(L_brack){not_op := 1.U}
55         when(R_brack){not_op := 0.U}
56         when(equal) {state := sEqual}
57
58         when(not_op ===0.U){
59             when(operator) {state := sOp}
60         }
61     }
62
63     is(sOp){
64         when(num) {state := sSrc2}
65         when(L_brack){
66             state := sSrc2
67             not_op := 1.U
68         }
69     }
70
71     is(sSrc2){
72         when(L_brack){not_op := 1.U}
73         when(R_brack){not_op := 0.U}
74         when(equal) {state := sEqual}
75         when(not_op ===0.U){
76             when(operator) {state := sOp}
77         }
78     }
79
80     is(sEqual){
81         state := sSrc1
82     }
83 }
84 //=====
85 when(state === sSrc1){
86     when(in_buffer < 10.U){src1 := (src1<<3.U) + (src1<<1.U) + in_bu
87     when(in_buffer ===14.U){src1 := -src1}
88 }
89 when(state === sSrc2){
90     when(in_buffer < 10.U){src2 := (src2<<3.U) + (src2<<1.U) + in_bu
91     when(in_buffer ===14.U){src2 := -src2}
92 }
93 when(state === sOp){
94     op := in_buffer - 10.U
95     src1 := MuxLookup((op),0.U,Seq(
96         add -> (src1 + src2),
97         sub -> (src1 - src2),
98         mul -> (src1 * src2)
99     ))
100     src2 :=0.U
101 }
102
103 when(state === sEqual){
104     src1 := 0.U
105     src2 := 0.U
106     op := 0.U

```

```
107         in_buffer := 0.U
108         not_op := 0.U
109     }
110
111     io.value.valid := Mux(state === sEqual,true.B,false.B)
112     io.value.bits := MuxLookup(op,0.U,Seq(
113         add -> (src1 + src2),
114         sub -> (src1 - src2),
115         mul -> (src1 * src2)
116     ))
117 }
```

## Test Result

```
Elaborating design...
Done elaborating.
[info] [0.003] SEED 1681798209175
[info] [0.009] Question1: 5+3*4=
[info] [0.009] the output of module is :32
[info] [0.009] the correct answer is :17
[info] [0.009] wrong
[info] [0.010] =====
[info] [0.012] Question2: 30+40=
[info] [0.012] the output of module is :70
[info] [0.012] the correct answer is :70
[info] [0.012] Correct
[info] [0.012] =====
[info] [0.014] Question3: 30-40=
[info] [0.014] the output of module is :-10
[info] [0.014] the correct answer is :-10
[info] [0.015] Correct
[info] [0.015] =====
[info] [0.016] Question4: 20*20=
[info] [0.016] the output of module is :400
[info] [0.016] the correct answer is :400
[info] [0.017] Correct
[info] [0.017] =====
[info] [0.019] Question5: (-123)=
[info] [0.019] the output of module is :-123
[info] [0.019] the correct answer is :-123
[info] [0.019] Correct
[info] [0.019] =====
[info] [0.024] Question6: (-10)+11+12-(-13)+(-14)=
[info] [0.024] the output of module is :12
[info] [0.024] the correct answer is :12
[info] [0.024] Correct
[info] [0.024] =====
[info] [0.028] Question7: ((-15)+(-10))*12-(34+66)*(-4)=
[info] [0.028] the output of module is :-13624
[info] [0.028] the correct answer is :100
[info] [0.028] wrong
[info] [0.028] =====
test cpxCal success: 0 tests passed in 97 cycles in 0.058646 seconds 1654.01 Hz
```

# Hw6-3-1 Pseudo Random Number Generator

---

## Scala Code

請放上你的程式碼並加上註解(中英文不限)，讓 TA明白你是如何完成的。



```
1 package aias_lab6.Hw3
2
3 import chisel3._
4 import chisel3.util._
5 class LFSR_Galois (n:Int)extends Module{
6
7     val io = IO(new Bundle{
8         val seed = Input(Valid(UInt(n.W)))
9         val rndNum = Output(UInt(n.W))
10    })
11
12    val shiftReg = RegInit(VecInit(Seq.fill(n)(false.B)))
13
14    when(io.seed.valid){
15        shiftReg zip io.seed.bits.asBools map {case(l,r) => l := r}
16    }.otherwise{
17
18        //Right Barrel Shift Register
19        (shiftReg.zipWithIndex).map{
20            case(sr,i) => sr := shiftReg((i+1)%n)
21        }
22
23        //Galois LFSR
24        LfsrTaps(n).map{x => {shiftReg(x-1) := shiftReg(x) ^ shiftReg(0)}}
25    }
26
27    io.rndNum := shiftReg.asUInt
28 }
29 object LfsrTaps {
30     def apply(size: Int): Seq[Int] = {
31         size match {
32             // Seq[Int] means the taps in LFSR
33             case 4 => Seq(3)          //p(x) = x^4+x^3+1
34             case 8 => Seq(6,5,4)      //p(x) = x^8+x^6+x^5+x^4+1
35             case 16 => Seq(14,13,11) //p(x) = x^16+x^14+x^13+x^11+1
36             case _ => throw new Exception("No LFSR taps stored for requested si:
37         }
38     }
39 }
40
41
42
43 class PRNG(seed:Int) extends Module{
44     val io = IO(new Bundle{
45         val gen = Input(Bool())
46         val puzzle = Output(Vec(4,UInt(4.W)))
47         val ready = Output(Bool())
48     })
49     val myReg = RegInit(0.U(4.W))
50     val cnt = RegInit(0.U(8.W))
51     val cnt2 = RegInit(0.U(3.W))
52     val lfsrInst = Module(new LFSR_Galois(4))
53     val lfsr = RegInit(VecInit(Seq.fill(4)(0.U(4.W))))
```

```
54
55     val prep :: finish :: Nil = Enum(2)
56     val state = RegInit(prep)
57     io.puzzle := lfsr
58     io.ready := false.B
59     lfsrInst.io.seed.valid := false.B
60     lfsrInst.io.seed.bits := 1.U
61
62
63     switch(state){
64         is(prep){
65             when(cnt2 === 4.U){
66                 cnt2 := 0.U
67                 state := finish
68             }
69         }
70         is(finish){
71             when(io.gen === true.B){
72                 state := prep
73             }
74         }
75     }
76
77     when(state === prep){
78         when(cnt===0.U)
79         {
80             lfsrInst.io.seed.valid := true.B
81             lfsrInst.io.seed.bits := 1.U
82         }.otherwise
83         {
84             lfsrInst.io.seed.valid := false.B
85             myReg := lfsrInst.io.rndNum
86             when(cnt2===0.U){
87                 when(myReg===10.U){lfsr(cnt2):=0.U}
88                 .elsewhen(myReg===11.U){lfsr(cnt2):=1.U}
89                 .elsewhen(myReg===12.U){lfsr(cnt2):=2.U}
90                 .elsewhen(myReg===13.U){lfsr(cnt2):=3.U}
91                 .elsewhen(myReg===14.U){lfsr(cnt2):=4.U}
92                 .elsewhen(myReg===15.U){lfsr(cnt2):=5.U}
93                 .elsewhen(myReg===16.U){lfsr(cnt2):=6.U}
94                 .otherwise{lfsr(cnt2) := myReg}
95                 cnt2 := cnt2 + 1.U
96             }.otherwise{
97                 when(cnt2===1.U&myReg/=lfsr(cnt2-1.U)&(myReg-10.
98                     when(myReg===10.U){lfsr(cnt2):=0.U}
99                     .elsewhen(myReg===11.U){lfsr(cnt2):=1.U}
100                     .elsewhen(myReg===12.U){lfsr(cnt2):=2.U}
101                     .elsewhen(myReg===13.U){lfsr(cnt2):=3.U}
102                     .elsewhen(myReg===14.U){lfsr(cnt2):=4.U}
103                     .elsewhen(myReg===15.U){lfsr(cnt2):=5.U}
104                     .elsewhen(myReg===16.U){lfsr(cnt2):=6.U}
105                     .otherwise{lfsr(cnt2) := myReg}
106                     cnt2 := cnt2 + 1.U
```

```

107     }
108     .elsewhen(cnt2===2.U&myReg/=lfsr(cnt2-1.U)&(myRe
109         when(myReg===10.U){lfsr(cnt2):=0.U}
110         .elsewhen(myReg===11.U){lfsr(cnt2):=1.U}
111         .elsewhen(myReg===12.U){lfsr(cnt2):=2.U}
112         .elsewhen(myReg===13.U){lfsr(cnt2):=3.U}
113         .elsewhen(myReg===14.U){lfsr(cnt2):=4.U}
114         .elsewhen(myReg===15.U){lfsr(cnt2):=5.U}
115         .elsewhen(myReg===16.U){lfsr(cnt2):=6.U}
116         .otherwise{lfsr(cnt2) := myReg}
117         cnt2 := cnt2 + 1.U
118     }
119     .elsewhen(cnt2===3.U&myReg/=lfsr(cnt2-1.U)&(myRe
120         when(myReg===10.U){lfsr(cnt2):=0.U}
121         .elsewhen(myReg===11.U){lfsr(cnt2):=1.U}
122         .elsewhen(myReg===12.U){lfsr(cnt2):=2.U}
123         .elsewhen(myReg===13.U){lfsr(cnt2):=3.U}
124         .elsewhen(myReg===14.U){lfsr(cnt2):=4.U}
125         .elsewhen(myReg===15.U){lfsr(cnt2):=5.U}
126         .elsewhen(myReg===16.U){lfsr(cnt2):=6.U}
127         .otherwise{lfsr(cnt2) := myReg}
128         cnt2 := cnt2 + 1.U
129     }
130 }
131
132
133
134
135     }
136
137     cnt := cnt+1.U
138 }
139
140 when(state === finish){
141     io.ready := true.B
142     io.puzzle := lfsr
143     when(io.gen){
144         io.ready := false.B
145         lfsr := VecInit(Seq.fill(4)(0.U(4.W)))
146     }
147 }
148 }
149
150

```

## Test Result

```
[info] [0.159] output : 4 5 0 1
[info] [0.159] The 85 testing :
[info] [0.160] output : 6 1 2 5
[info] [0.160] The 86 testing :
[info] [0.160] output : 5 7 4 3
[info] [0.160] The 87 testing :
[info] [0.160] output : 5 7 4 9
[info] [0.161] The 88 testing :
[info] [0.161] output : 5 0 3 9
[info] [0.161] The 89 testing :
[info] [0.161] output : 4 8 9 7
[info] [0.161] The 90 testing :
[info] [0.162] output : 2 4 8 1
[info] [0.162] The 91 testing :
[info] [0.162] output : 6 2 1 3
[info] [0.162] The 92 testing :
[info] [0.163] output : 6 1 2 3
[info] [0.163] The 93 testing :
[info] [0.163] output : 4 8 9 3
[info] [0.163] The 94 testing :
[info] [0.164] output : 5 7 4 1
[info] [0.164] The 95 testing :
[info] [0.164] output : 5 0 3 9
[info] [0.164] The 96 testing :
[info] [0.165] output : 5 0 3 7
[info] [0.165] The 97 testing :
[info] [0.165] output : 2 4 8 7
[info] [0.165] The 98 testing :
[info] [0.166] output : 7 4 5 3
[info] [0.166] The 99 testing :
[info] [0.166] output : 3 6 2 1
[info] [0.166] The 100 testing :
[info] [0.166] output : 9 1 5 0
[info] [0.167] You pass the Hw6-2-1, well done!!
test PRNG Success: 0 tests passed in 1209 cycles in 0.196755 seconds 6144.71 Hz
[info] [0.167] RAN 1204 CYCLES PASSED
```

## Hw6-3-2 1A2B game quiz

### Scala Code

請放上你的程式碼並加上註解(中英文不限)，讓 TA明白你是如何完成的。





```
1 package aias_lab6.Hw3
2
3 import chisel3._
4 import chisel3.util._
5
6 class NumGuess(seed: Int = 1) extends Module{
7     require (seed > 0 , "Seed cannot be 0")
8
9     val io = IO(new Bundle{
10         val gen = Input(Bool())
11         val guess = Input(UInt(16.W))
12         val puzzle = Output(Vec(4, UInt(4.W)))
13         val ready = Output(Bool())
14         val g_valid = Output(Bool())
15         val A = Output(UInt(3.W))
16         val B = Output(UInt(3.W))
17
18         //don't care at Hw6-3-2 but should be considered at Bonus
19         val s_valid = Input(Bool())
20     })
21
22     val sIdle :: sSet :: sGues :: Nil = Enum(3)
23     val guess_buf = RegInit(VecInit(Seq.fill(4)(0.U(4.W))))
24     val pass = RegInit(0.U(1.W))
25     val a_buff = RegInit(0.U(3.W))
26     val b_buff = RegInit(0.U(3.W))
27     val index = RegInit(0.U(3.W))
28     val index2 = RegInit(0.U(3.W))
29     val state = RegInit(sIdle)
30     val prng = Module(new PRNG(1))
31     switch(state){
32         is(sIdle){
33             when(io.gen){state := sSet}
34         }
35         is(sSet){
36             when(io.ready){state := sGues}
37         }
38         is(sGues){io.ready === false.B}
39     }
40     //初始化
41     io.ready := false.B
42     io.puzzle := prng.io.puzzle
43     prng.io.gen := false.B
44     io.A := 0.U
45     io.B := 0.U
46     io.g_valid := false.B
47
48
49
50
51     //產生題目
52     when(state === sSet){
53
```

```

54      //prng.io.gen := true.B
55      //io.puzzle := prng.io.puzzle
56      pass := 1.U
57      for(i <- 0 until 4){
58          guess_buf(i) := io.guess((i+1)*4-1,i*4)
59      }
60      when(pass === 1.U){io.ready := true.B}
61  }
62
63  //開始猜
64  when(state === sGues){
65      //如果guess和puzzle有相同位置且一樣的數字 · A+1 ; 只有數字則B+1;
66      when(index===4.U){
67          when(a_buff===4.U){io.ready := false.B}
68          .elsewhen(a_buff<4.U){
69              io.ready := true.B
70              a_buff := 0.U
71              b_buff := 0.U
72              state := sSet
73          }
74
75          io.g_valid := true.B
76          index := 0.U
77      }
78      .elsewhen(index<=3.U){
79          when(guess_buf(index) === io.puzzle(index)){
80              a_buff := a_buff+1.U
81          }
82          .otherwise{
83              //注意guess跟puzzle的index排序是反的
84              when(index===0.U){
85                  when(guess_buf(0) === io.puzzle(0)){b_buff := b_b
86                  .elsewhen(guess_buf(0) === io.puzzle(1)){b_buff :
87                  .elsewhen(guess_buf(0) === io.puzzle(2)){b_buff :
88                  }
89                  .elsewhen(guess_buf(index-1.U)≠guess_buf(index)&&index:
90                  when(guess_buf(1) === io.puzzle(0)){b_buff := b_b
91                  .elsewhen(guess_buf(1) === io.puzzle(1)){b_buff :
92                  .elsewhen(guess_buf(1) === io.puzzle(3)){b_buff :
93                  }
94                  .elsewhen(index===2.U&&guess_buf(index-1.U)≠guess_buf(:
95                  when(guess_buf(2) === io.puzzle(0)){b_buff := b_b
96                  .elsewhen(guess_buf(2) === io.puzzle(2)){b_buff :
97                  .elsewhen(guess_buf(2) === io.puzzle(3)){b_buff :
98                  }
99                  .elsewhen(index===3.U&&guess_buf(3)≠guess_buf(2)&&gues:
100                  when(guess_buf(3) === io.puzzle(1)){b_buff := b_b
101                  .elsewhen(guess_buf(3) === io.puzzle(2)){b_buff :
102                  .elsewhen(guess_buf(3) === io.puzzle(3)){b_buff :
103                  }
104              }
105              index := index+1.U
106          }

```

```
107  
108         io.A := a_buff  
109         io.B := b_buff  
110     }  
111 }
```

## Test Result

```
Elaborating design...  
Done elaborating.  
[info] [0.003] SEED 1681798518761  
[info] [0.013] Puzzle:6 2 1 0  
[info] [0.014] plz enter the 3's digit :  
6  
[info] [4.702] plz enter the 2's digit :  
6  
[info] [5.124] plz enter the 1's digit :  
6  
[info] [5.529] plz enter the 0's digit :  
6  
[info] [5.938] Guess:6 6 6 6  
[info] [5.939] A = 1  
[info] [5.939] B = 0  
[info] [5.939] plz enter the 3's digit :  
6  
[info] [8.559] plz enter the 2's digit :  
3  
[info] [13.938] plz enter the 1's digit :  
0  
[info] [17.947] plz enter the 0's digit :  
2  
[info] [18.400] Guess:6 3 0 2  
[info] [18.400] A = 1  
[info] [18.400] B = 2  
[info] [18.401] plz enter the 3's digit :  
6  
[info] [23.355] plz enter the 2's digit :  
2  
[info] [23.715] plz enter the 1's digit :  
1  
[info] [24.103] plz enter the 0's digit :  
0  
[info] [24.884] Guess:6 2 1 0  
[info] [24.884] A = 4  
[info] [24.884] B = 0  
test NumGuess Success: 0 tests passed in 57 cycles in 24.914673 seconds 2.29 Hz  
[info] [24.885] RAN 52 CYCLES PASSED
```

## Bonus : 1A2B hardware solver

### Scala Code

請放上你的程式碼並加上註解(中英文不限)，讓 TA明白你是如何完成的。

## Test Result

---

### 文件中的問答題

---

- Q1:Hw6-2-2(長算式)以及Lab6-2-2(短算式)，需要的暫存器數量是否有差別？如果有，是差在哪裡呢？
  - Ans1:整體而言差不多，但我多了not\_op這個暫存器用來存目前讀到的數是否為負數，雖然hw6-2-2有更長的運算，但我都使用src1來當作部分算積計算後的結果，在把後面陸續讀到的數值放到src2，不斷地做運算。
- Q2:你是如何處理**Hw6-2-3**有提到的關於**編碼衝突**的問題呢？
  - Ans2:用一個bit紀錄該input為op或num，若為op則bit設為1，若為數字則bit設為0
- Q3:你是如何處理**Hw6-3-11A2B**題目產生時**數字重複**的問題呢？
  - Ans3:要產生數字前，先去檢查該數字之前是否有產生過，若有的話則等待下個clock產生新亂數，直到產生的新亂數與前幾個數字不重複時再將值放入。

### 意見回饋和心得(可填可不填)

---