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

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

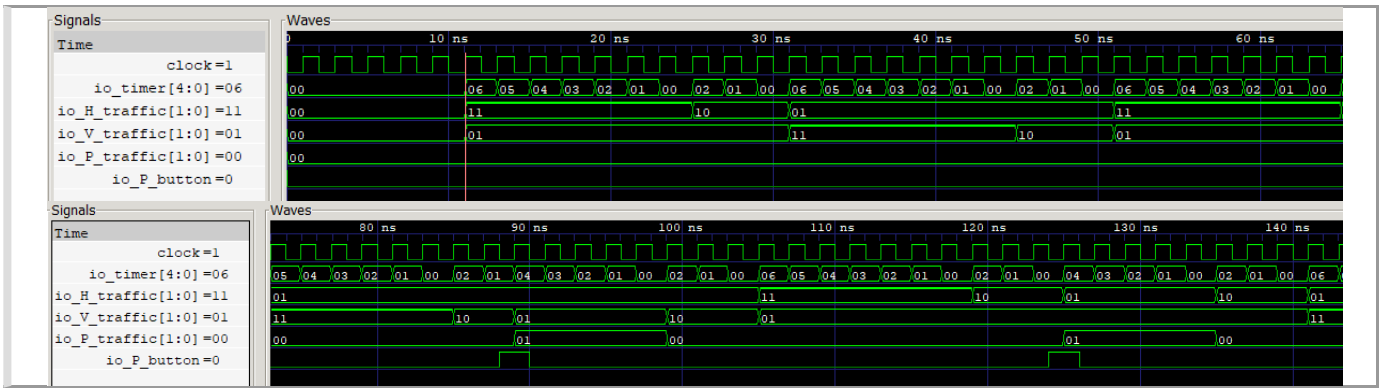
> 請放上你的程式碼並加上註解(中英文不限)，讓 TA明白你是如何完成的。

```
package aias_lab6.Hw1

import chisel3._
import chisel3.util._

class TrafficLight_p(Ytime:Int, Gtime:Int, Ptime:Int) extends Module{
  val io = IO(new Bundle{
    val P_button = Input(Bool())
    val H_traffic = Output(UInt(2.W))
    val V_traffic = Output(UInt(2.W))
    val P_traffic = Output(UInt(2.W))
    val timer     = Output(UInt(5.W))
  })

  //please implement your code below...
  //parameter declaration
  val Off = 0.U
  val Red = 1.U
  val Yellow = 2.U
  val Green = 3.U
  val PG = 1.U



  val sIdle :: sHGVR :: sHYVR :: sHRVG :: sHRVY :: sPG :: Nil = Enum(6)

  //State register
  val state = RegInit(sIdle)
  val ori_state = RegInit(sIdle)

  //Counter============================
  val cntMode = WireDefault(0.U(2.W))
  val cntReg = RegInit(0.U(4.W))
  val cntDone = Wire(Bool())
  cntDone := cntReg === 0.U

  when(cntDone){
       //重置
    when(cntMode === 0.U){
         //綠燈
      cntReg := (Gtime-1).U
    }.elsewhen(cntMode === 1.U){
         //黃燈
      cntReg := (Ytime-1).U
    }.elsewhen(cntMode === 2.U){
         //行人
      cntReg := (Ptime-1).U
       }
  }.otherwise{
    //cntDone不等於0時，不斷的將cntReg-1做倒數
    cntReg := cntReg - 1.U
  }
  //Counter end=======================
```

```
    //Next State Decoder
    //按下行人通行，初始化cntReg
    //ori_state := state
    when(io.P_button) {
          state := sPG
          cntReg := 4.U
          }
    .otherwise{
            switch(state){
                  is(sIdle){
                    state := sHGVR
                  }
                  //行人開始倒數，完成切回原狀態，並且原時間重新計時
                  is(sPG){
                    when(cntDone) {
                    state := ori_state
                    when(ori_state === sHGVR || ori_state === sHRVG)
                    {
                          cntReg:= 6.U
                    }
                    .elsewhen(ori_state === sHYVR || ori_state === sHRVY)
                    {
                          cntReg:= 2.U
                    }.otherwise{cntReg:= 10.U}
                    }
                  }
                  is(sHGVR){
                    ori_state := state
                    when(cntDone) {
                    state := sHYVR
                    ori_state := sHYVR
                    }
                  }
                  is(sHYVR){
                    ori_state := state
                    when(cntDone) {
                    state := sHRVG
                    ori_state := sHRVG
                    }
                  }
                  is(sHRVG){
                    ori_state := state
                    when(cntDone) {
                    state := sHRVY
                    ori_state := sHRVY
                    }

                  }
                  is(sHRVY){
                  ori_state := state
                    when(cntDone) {
                    state := sHGVR
                    ori_state := sHGVR
```

```
                    }

                }
            }
    }


    //Output Decoder
    //Default statement
    cntMode := 0.U
    io.H_traffic := Off
    io.V_traffic := Off
    io.P_traffic := Off

    switch(state){
      is(sHGVR){
        cntMode := 1.U
        io.H_traffic := Green
        io.V_traffic := Red
      }
      is(sHYVR){
        cntMode := 0.U
        io.H_traffic := Yellow
        io.V_traffic := Red
      }
      is(sHRVG){
        cntMode := 1.U
        io.H_traffic := Red
        io.V_traffic := Green
      }
      is(sHRVY){
        cntMode := 0.U
        io.H_traffic := Red
        io.V_traffic := Yellow
      }
          is(sPG){
        cntMode := 2.U
        io.H_traffic := Red
        io.V_traffic := Red
            io.P_traffic := PG
      }
    }

    io.timer := cntReg
  }
```

Waveform



Hw6-2-1 Negative Integer 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。

```
package aias_lab6.Hw2

import chisel3._
import chisel3.util._

class NegIntGen extends Module{
    val io = IO(new Bundle{
        val key_in = Input(UInt(4.W))
        val value = Output(Valid(UInt(32.W)))
    })

    //please implement your code below
    val equal = WireDefault(false.B)
    equal := io.key_in === 15.U
    val num = WireDefault(false.B)
    num := io.key_in < 10.U



    val sIdle :: sAccept :: sEqual :: Nil = Enum(3)
    val state = RegInit(sIdle)
    //Next State Decoder
    switch(state){
        is(sIdle){
        state := sAccept
        }
        is(sAccept){
        when(equal) {state := sEqual}
        }
        is(sEqual){
            state := sAccept
        }
    }

    val in_buffer = RegNext(io.key_in)
    val number = RegInit(0.U(32.W))
        val neg = RegInit(0.U(1.W))

    when(state === sAccept){
                when(in_buffer < 10.U){
        number := (number<<3.U) + (number<<1.U) + in_buffer}
                //number := (number<<6.U) + (number<<5.U) + (number<<2.U) +in_bu
                .elsewhen(in_buffer === 11.U){neg := 1.U}

    }.elsewhen(state === sEqual){
        in_buffer := 0.U
        number := 0.U
                neg := 0.U
    }

    io.value.valid := Mux(state === sEqual,true.B,false.B)
        when(neg===1.U){io.value.bits := -number}
    .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。

```
package aias_lab6.Hw2

import chisel3._
import chisel3.util._

class LongCal extends Module{
    val io = IO(new Bundle{
        val key_in = Input(UInt(4.W))
        val value = Output(Valid(UInt(32.W)))
    })

    //please implement your code below
    val operator = WireDefault(false.B)
    operator := io.key_in >= 10.U && io.key_in <= 12.U

    val num = WireDefault(false.B)
    num := io.key_in < 10.U

    val equal = WireDefault(false.B)
    equal := io.key_in === 15.U

    val L_brack = WireDefault(false.B)
    L_brack := io.key_in === 13.U

        val R_brack = WireDefault(false.B)
    R_brack := io.key_in === 14.U

    //Reg Declaration====================================
    val in_buffer = RegNext(io.key_in)
    val src1 = RegInit(0.U(32.W))
    val op = RegInit(0.U(2.W))
    val src2 = RegInit(0.U(32.W))
        val not_op = RegInit(0.U(1.W))


    //State and Constant Declaration====================
    val sIdle :: sSrc1 :: sOp :: sSrc2 :: sEqual :: Nil = Enum(5)
    val add = 0.U
    val sub = 1.U
    val mul = 2.U
    val state = RegInit(sIdle)

    //Next State Decoder
    switch(state){
        is(sIdle){
                        when(L_brack){not_op := 1.U}
            state := sSrc1
        }

        is(sSrc1){
                when(L_brack){not_op := 1.U}
                when(R_brack){not_op := 0.U}
                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。

```
package aias_lab6.Hw2

import chisel3._
import chisel3.util._




class CpxCal extends Module{
    val io = IO(new Bundle{
        val key_in = Input(UInt(4.W))
        val value = Output(Valid(UInt(32.W)))
    })

   //please implement your code below
    val operator = WireDefault(false.B)
    operator := io.key_in >= 10.U && io.key_in <= 12.U

    val num = WireDefault(false.B)
    num := io.key_in < 10.U

    val equal = WireDefault(false.B)
    equal := io.key_in === 15.U

    val L_brack = WireDefault(false.B)
    L_brack := io.key_in === 13.U

        val R_brack = WireDefault(false.B)
    R_brack := io.key_in === 14.U

    //Reg Declaration====================================
    val in_buffer = RegNext(io.key_in)
    val src1 = RegInit(0.U(32.W))
    val op = RegInit(0.U(2.W))
    val src2 = RegInit(0.U(32.W))
        val not_op = RegInit(0.U(1.W))


    //State and Constant Declaration====================
    val sIdle :: sSrc1 :: sOp :: sSrc2 :: sEqual :: Nil = Enum(5)
    val add = 0.U
    val sub = 1.U
    val mul = 2.U
    val state = RegInit(sIdle)

    //Next State Decoder
    switch(state){
        is(sIdle){
                        when(L_brack){not_op := 1.U}
            state := sSrc1
        }

        is(sSrc1){
```

```
                when(L_brack){not_op := 1.U}
                when(R_brack){not_op := 0.U}
                when(equal) {state := sEqual}

                when(not_op ===0.U){
                        when(operator) {state := sOp}
                        }
        }

        is(sOp){
            when(num) {state := sSrc2}
                        when(L_brack){
                        state := sSrc2
                        not_op := 1.U
                        }
        }

        is(sSrc2){
                when(L_brack){not_op := 1.U}
                when(R_brack){not_op := 0.U}
                when(equal) {state := sEqual}
                when(not_op ===0.U){
                        when(operator) {state := sOp}
                        }
        }

        is(sEqual){
            state := sSrc1
        }
    }
    //================================================
    when(state === sSrc1){
                when(in_buffer < 10.U){src1 := (src1<<3.U) + (src1<<1.U) + in_bu
                when(in_buffer ===14.U){src1 := -src1}
       }
    when(state === sSrc2){
                when(in_buffer < 10.U){src2 := (src2<<3.U) + (src2<<1.U) + in_bu
                when(in_buffer ===14.U){src2 := -src2}
       }
    when(state === sOp){
        op := in_buffer - 10.U
                src1 := MuxLookup((op),0.U,Seq(
                add -> (src1 + src2),
                sub -> (src1 - src2),
                mul -> (src1 * src2)
                ))
                src2 :=0.U
        }

    when(state === sEqual){
        src1 := 0.U
        src2 := 0.U
        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。

```
package aias_lab6.Hw3

import chisel3._
import chisel3.util._
class LFSR_Galois (n:Int)extends Module{

    val io = IO(new Bundle{
        val seed = Input(Valid(UInt(n.W)))
        val rndNum = Output(UInt(n.W))
    })

    val shiftReg = RegInit(VecInit(Seq.fill(n)(false.B)))

    when(io.seed.valid){
      shiftReg zip io.seed.bits.asBools map {case(l,r) => l := r}
    }.otherwise{

      //Right Barrel Shift Register
      (shiftReg.zipWithIndex).map{
          case(sr,i) => sr := shiftReg((i+1)%n)
      }

      //Galois LFSR
      LfsrTaps(n).map{x => {shiftReg(x-1) := shiftReg(x) ^ shiftReg(0)}}
    }

    io.rndNum := shiftReg.asUInt
}
object LfsrTaps {
    def apply(size: Int): Seq[Int] = {
        size match {
            // Seqp[Int] means the taps in LFSR
            case 4 => Seq(3)          //p(x) = x^4+x^3+1
            case 8 => Seq(6,5,4)      //p(x) = x^8+x^6+x^5+x^4+1
            case 16 => Seq(14,13,11) //p(x) = x^16+x^14+x^13+x^11+1
            case _ => throw new Exception("No LFSR taps stored for requested si
        }
    }
}



class PRNG(seed:Int) extends Module{
    val io = IO(new Bundle{
        val gen = Input(Bool())
        val puzzle = Output(Vec(4,UInt(4.W)))
        val ready = Output(Bool())
    })
        val myReg = RegInit(0.U(4.W))
        val cnt = RegInit(0.U(8.W))
        val cnt2 = RegInit(0.U(3.W))
    val lfsrInst = Module(new LFSR_Galois(4))
    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
```

```
                                }
                                .elsewhen(cnt2===2.U&myReg=/=lfsr(cnt2-1.U)&(myRe
                                        when(myReg===10.U){lfsr(cnt2):=0.U}
                                        .elsewhen(myReg===11.U){lfsr(cnt2):=1.U}
                                        .elsewhen(myReg===12.U){lfsr(cnt2):=2.U}
                                        .elsewhen(myReg===13.U){lfsr(cnt2):=3.U}
                                        .elsewhen(myReg===14.U){lfsr(cnt2):=4.U}
                                        .elsewhen(myReg===15.U){lfsr(cnt2):=5.U}
                                        .elsewhen(myReg===16.U){lfsr(cnt2):=6.U}
                                        .otherwise{lfsr(cnt2) := myReg}
                                        cnt2 := cnt2 + 1.U
                                }
                                .elsewhen(cnt2===3.U&myReg=/=lfsr(cnt2-1.U)&(myRe
                                        when(myReg===10.U){lfsr(cnt2):=0.U}
                                        .elsewhen(myReg===11.U){lfsr(cnt2):=1.U}
                                        .elsewhen(myReg===12.U){lfsr(cnt2):=2.U}
                                        .elsewhen(myReg===13.U){lfsr(cnt2):=3.U}
                                        .elsewhen(myReg===14.U){lfsr(cnt2):=4.U}
                                        .elsewhen(myReg===15.U){lfsr(cnt2):=5.U}
                                        .elsewhen(myReg===16.U){lfsr(cnt2):=6.U}
                                        .otherwise{lfsr(cnt2) := myReg}
                                        cnt2 := cnt2 + 1.U
                                }
                        }



                }

                cnt := cnt+1.U
        }

        when(state === finish){
                io.ready := true.B
                io.puzzle := lfsr
                when(io.gen){
                        io.ready := false.B
                        lfsr := VecInit(Seq.fill(4)(0.U(4.W)))
                }
        }
}
```

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

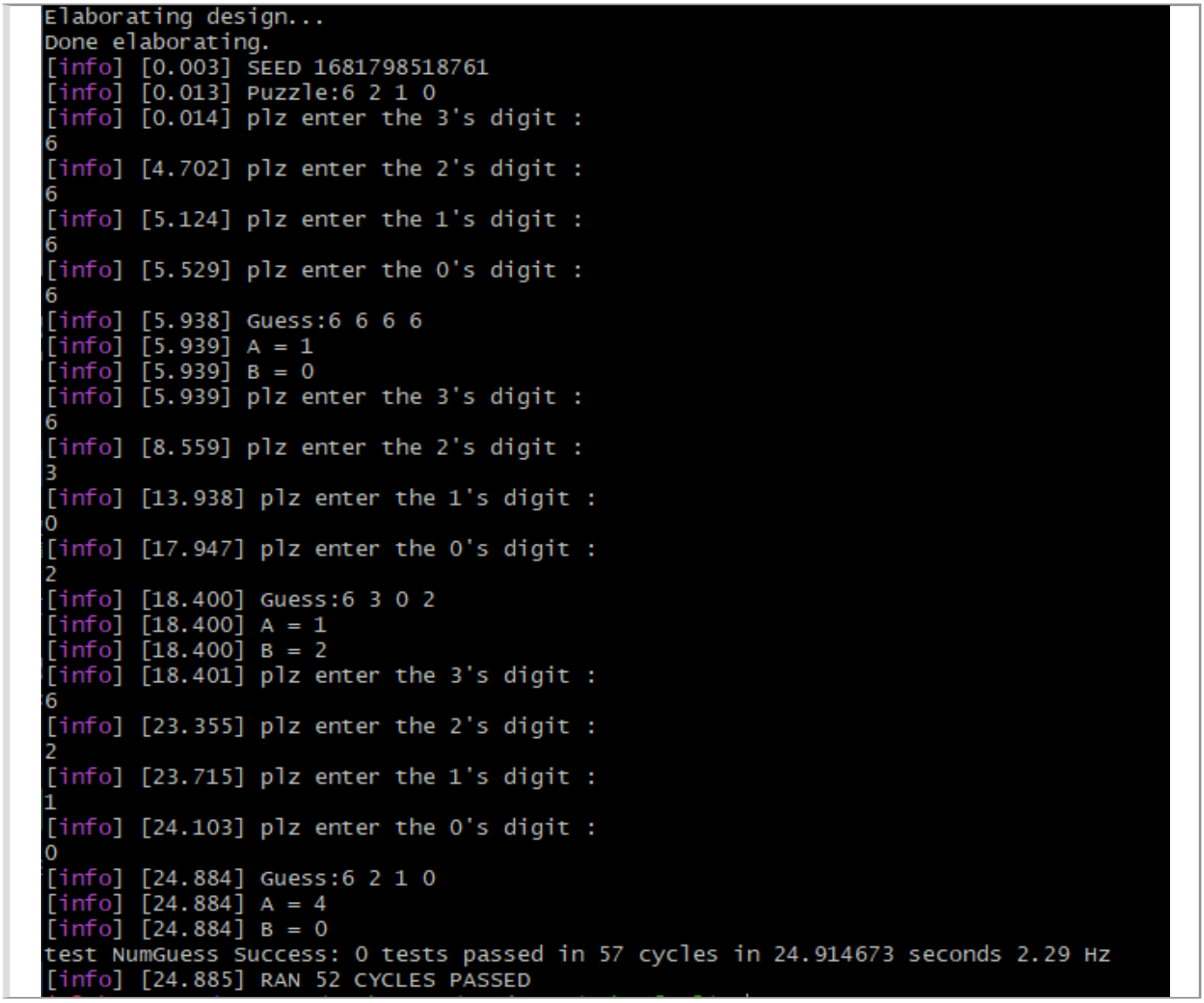
請放上你的程式碼並加上註解(中英文不限)，讓 TA明白你是如何完成的。

```
package aias_lab6.Hw3

import chisel3._
import chisel3.util._

class NumGuess(seed:Int = 1) extends Module{
    require (seed > 0 , "Seed cannot be 0")

    val io  = IO(new Bundle{
        val gen = Input(Bool())
        val guess = Input(UInt(16.W))
        val puzzle = Output(Vec(4,UInt(4.W)))
        val ready  = Output(Bool())
        val g_valid  = Output(Bool())
        val A      = Output(UInt(3.W))
        val B      = Output(UInt(3.W))

        //don't care at Hw6-3-2 but should be considered at Bonus
        val s_valid = Input(Bool())
    })

        val sIdle :: sSet :: sGues :: Nil =Enum(3)
        val guess_buf = RegInit(VecInit(Seq.fill(4)(0.U(4.W))))
        val pass = RegInit(0.U(1.W))
        val a_buff = RegInit(0.U(3.W))
        val b_buff = RegInit(0.U(3.W))
        val index = RegInit(0.U(3.W))
        val index2 = RegInit(0.U(3.W))
        val state = RegInit(sIdle)
        val prng = Module(new PRNG(1))
        switch(state){
                is(sIdle){
                        when(io.gen){state := sSet}
                }
                is(sSet){
                        when(io.ready){state := sGues}
                }
                is(sGues){io.ready ===false.B}
        }
        //初始化
        io.ready := false.B
        io.puzzle := prng.io.puzzle
        prng.io.gen := false.B
        io.A      := 0.U
    io.B      := 0.U
        io.g_valid  := false.B




        //產生題目
        when(state === sSet){

```

```
                //prng.io.gen := true.B
                //io.puzzle := prng.io.puzzle
                pass := 1.U
                for(i <- 0 until 4){
                        guess_buf(i) := io.guess((i+1)*4-1,i*4)
                }
                when(pass === 1.U){io.ready := true.B}
        }

        //開始猜
        when(state === sGues){
        //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;
                when(index===4.U){
                        when(a_buff===4.U){io.ready := false.B}
                        .elsewhen(a_buff<4.U){
                                io.ready := true.B
                                a_buff := 0.U
                                b_buff := 0.U
                                state := sSet
                        }

                        io.g_valid := true.B
                        index := 0.U
                }
                .elsewhen(index<=3.U){
                        when(guess_buf(index) === io.puzzle(index)){
                                a_buff := a_buff+1.U
                        }
                        .otherwise{
                        //注意guess跟puzzle的index排序是反的
                        when(index===0.U){
                                when(guess_buf(0) === io.puzzle(0)){b_buff := b_b
                                .elsewhen(guess_buf(0) === io.puzzle(1)){b_buff :
                                .elsewhen(guess_buf(0) === io.puzzle(2)){b_buff :
                                }
                        .elsewhen(guess_buf(index-1.U)=/=guess_buf(index)&&index=
                                when(guess_buf(1) === io.puzzle(0)){b_buff := b_b
                                .elsewhen(guess_buf(1) === io.puzzle(1)){b_buff :
                                .elsewhen(guess_buf(1) === io.puzzle(3)){b_buff :
                                }
                        .elsewhen(index===2.U&&guess_buf(index-1.U)=/=guess_buf(
                                when(guess_buf(2) === io.puzzle(0)){b_buff := b_b
                                .elsewhen(guess_buf(2) === io.puzzle(2)){b_buff :
                                .elsewhen(guess_buf(2) === io.puzzle(3)){b_buff :
                                }
                        .elsewhen(index===3.U&&guess_buf(3)=/=guess_buf(2)&&gues
                                when(guess_buf(3) === io.puzzle(1)){b_buff := b_b
                                .elsewhen(guess_buf(3) === io.puzzle(2)){b_buff :
                                .elsewhen(guess_buf(3) === io.puzzle(3)){b_buff :
                                }
                        }
                        index := index+1.U
                }
```

```
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**Hw6-3-1**1A2B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)