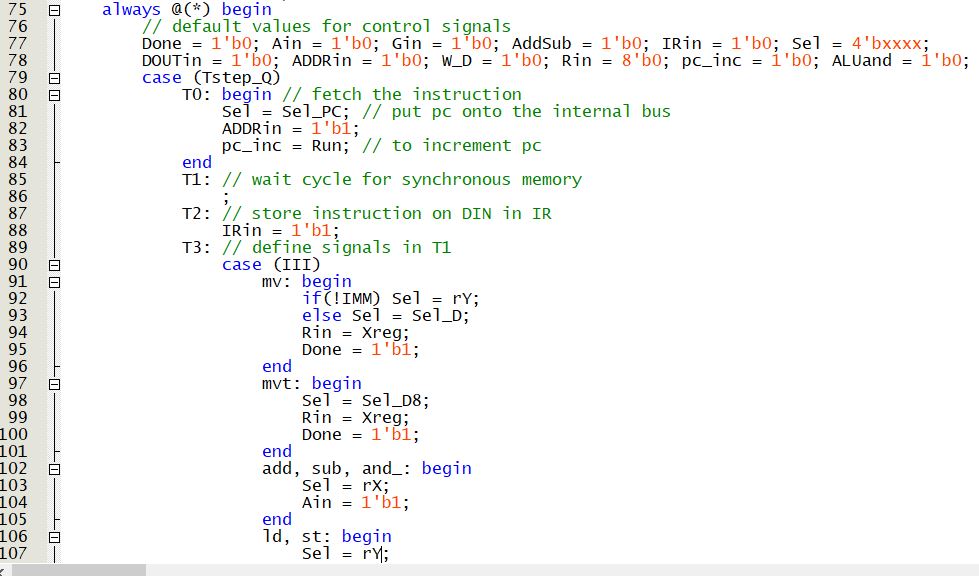
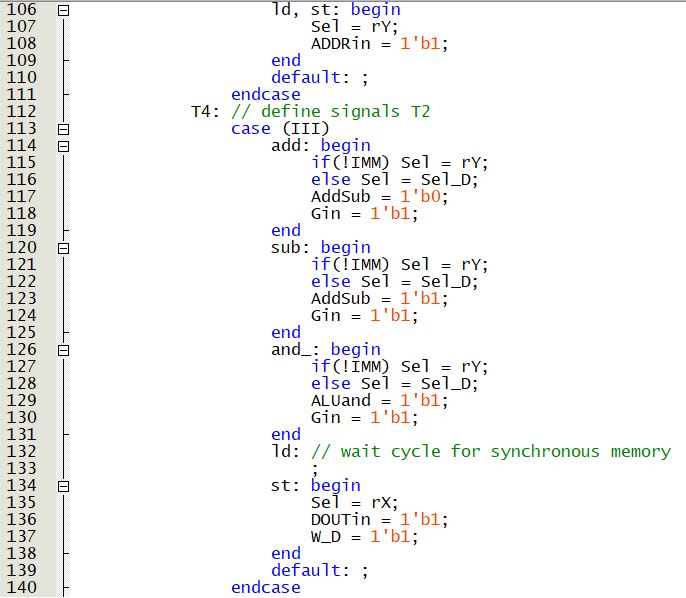
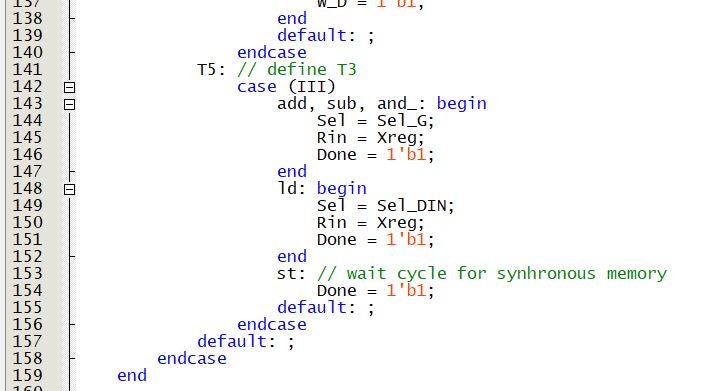
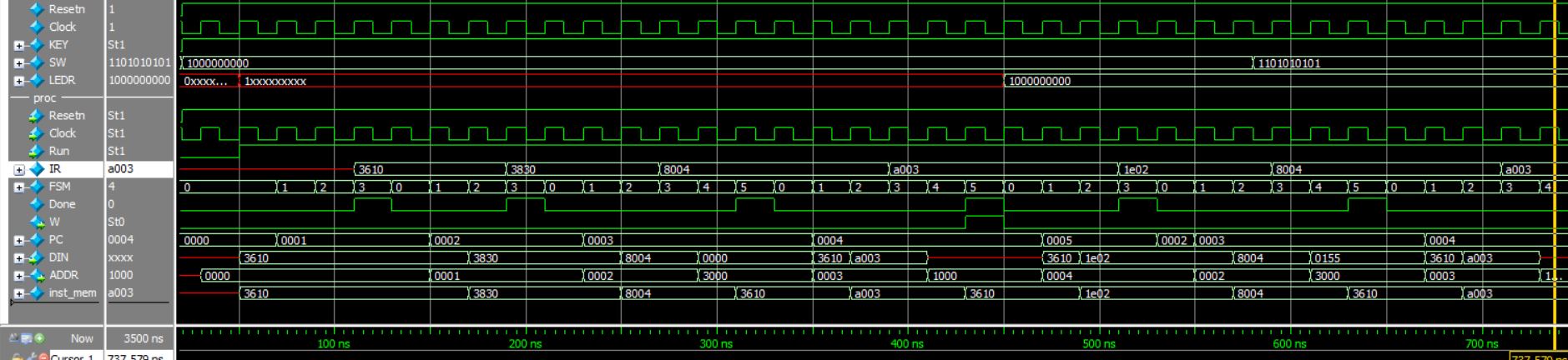
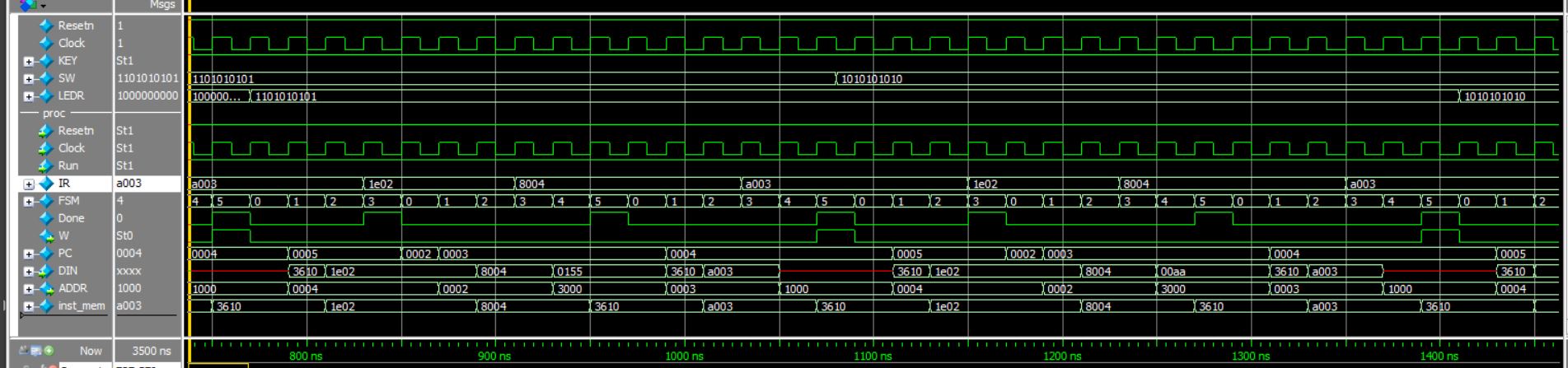
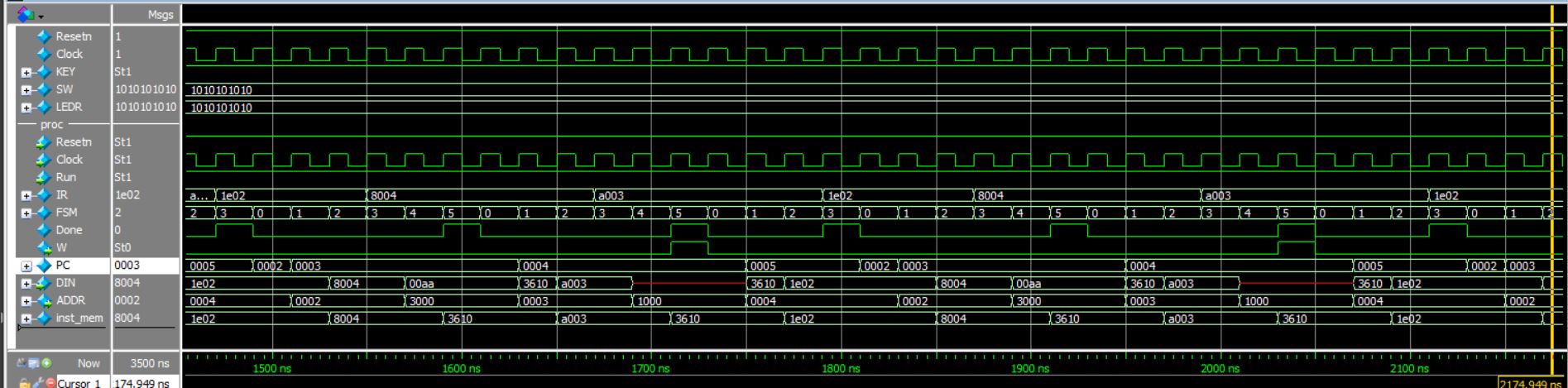
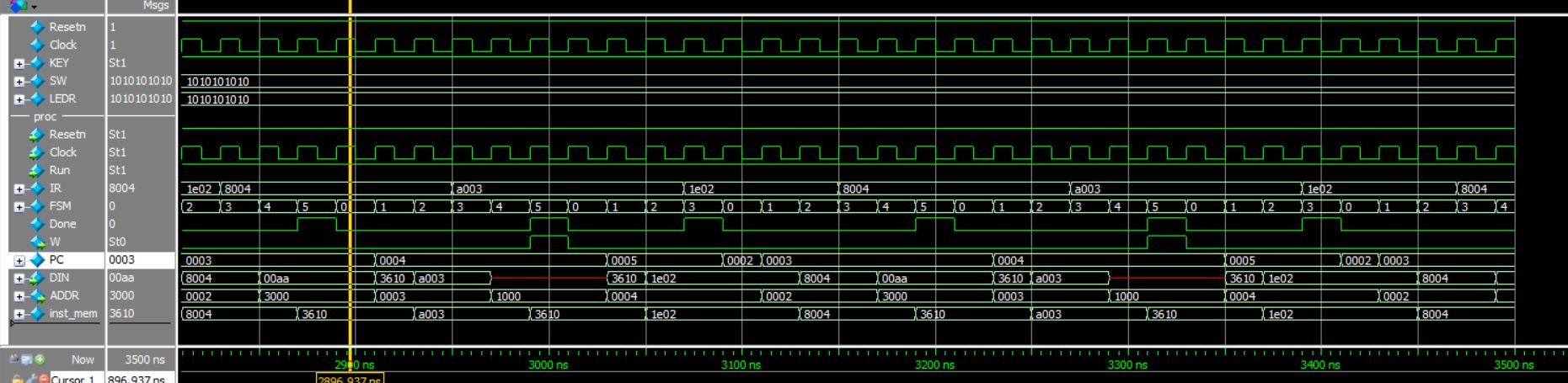
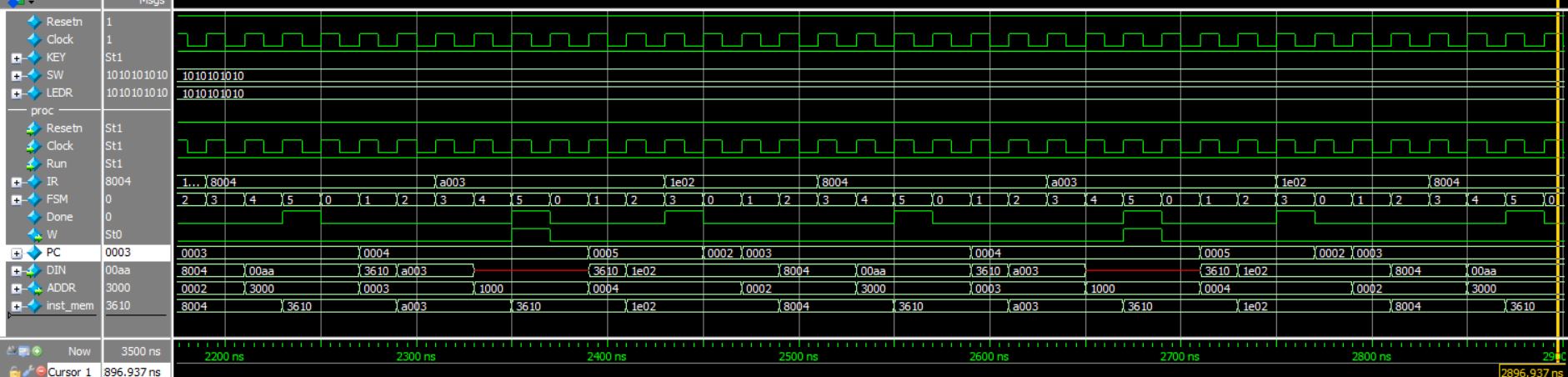
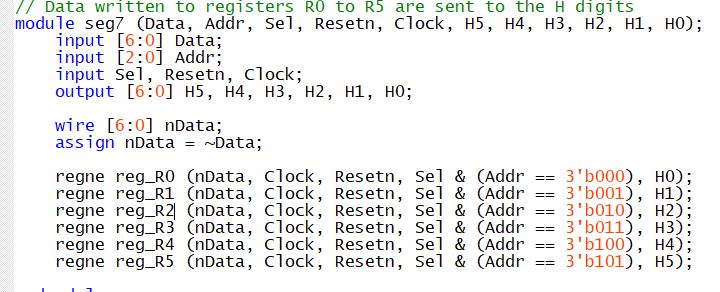
**Part 1**

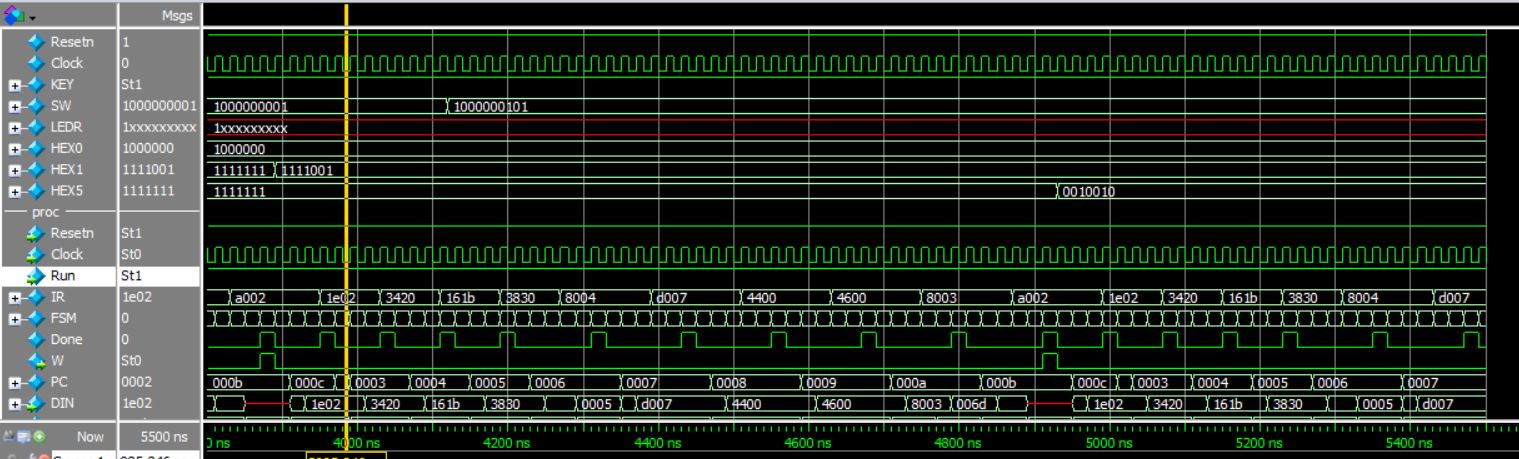
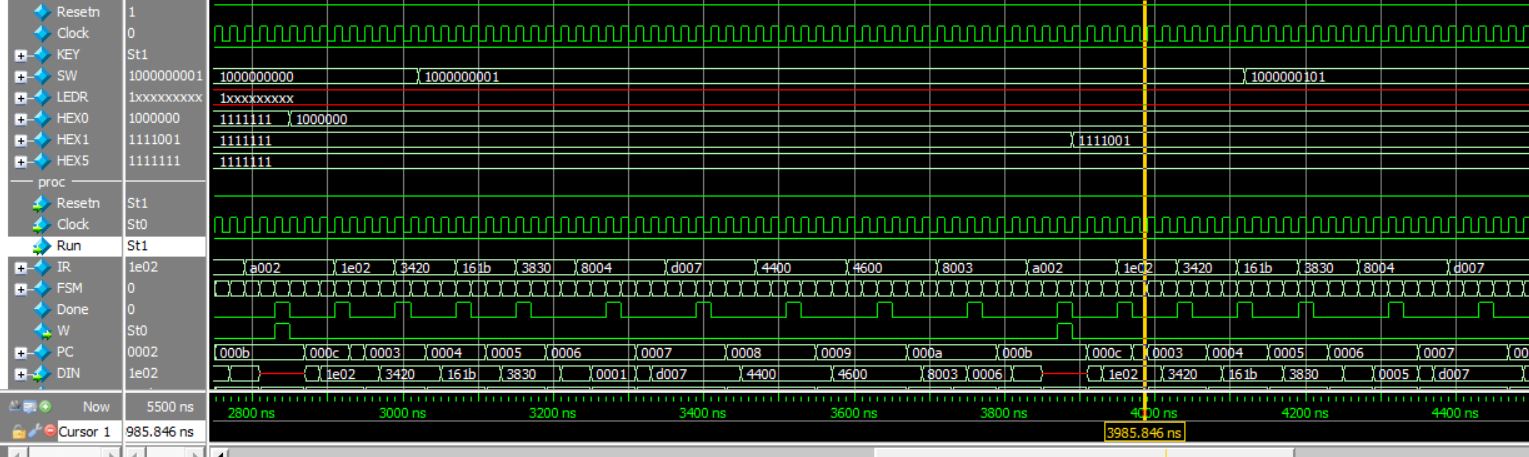
****



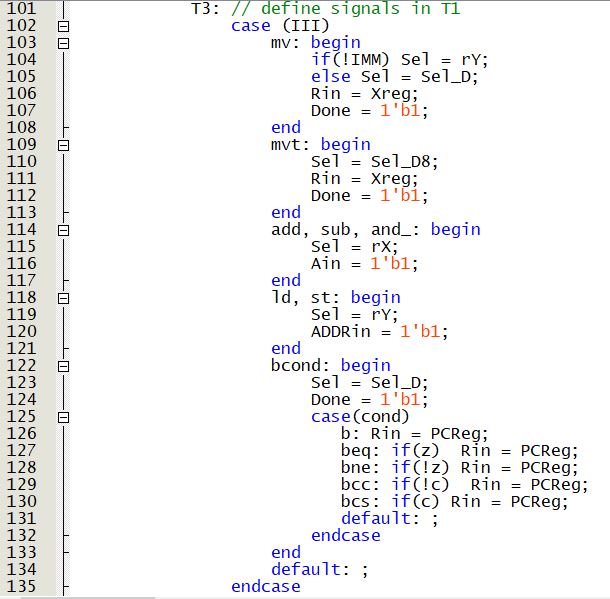
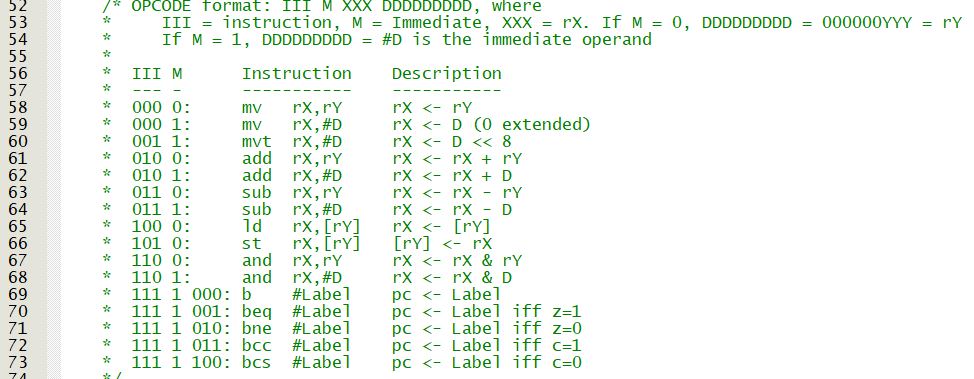
 

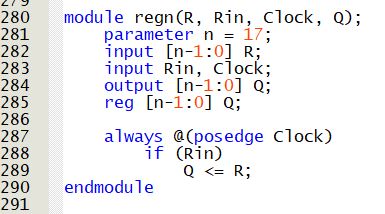
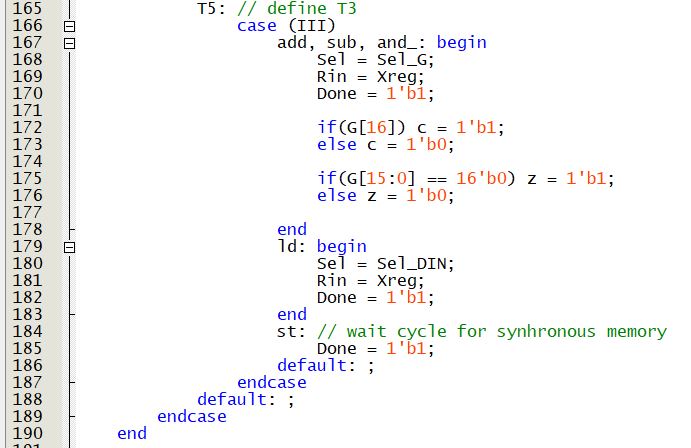
**Part 4**

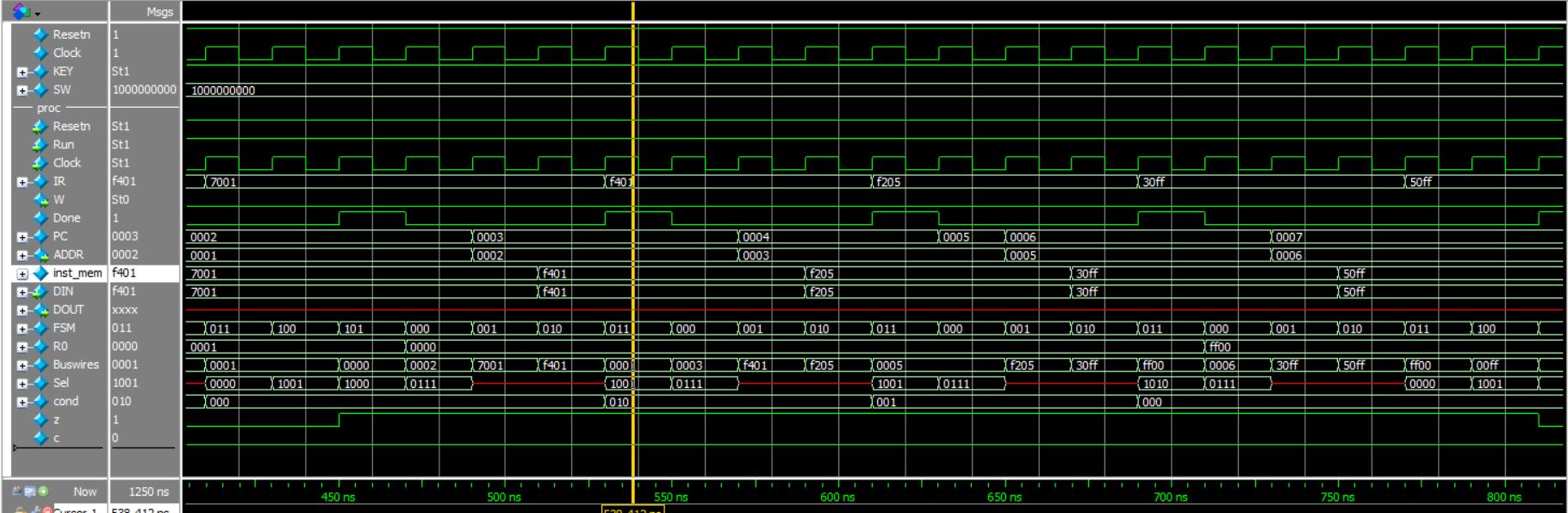
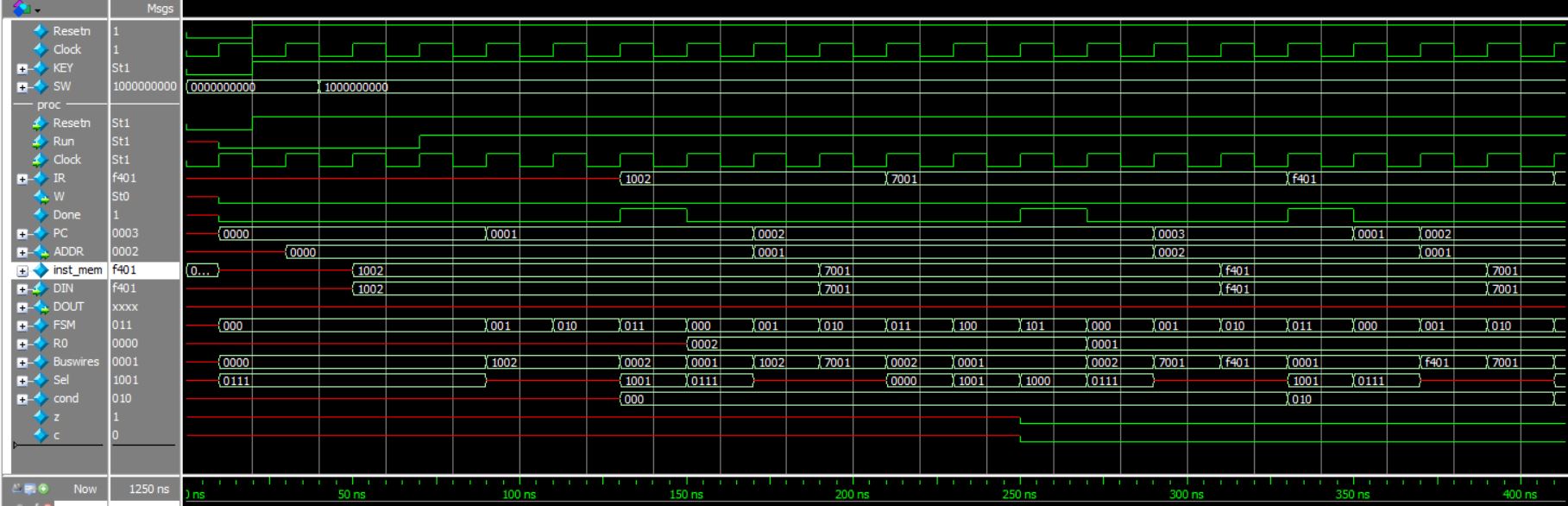
****

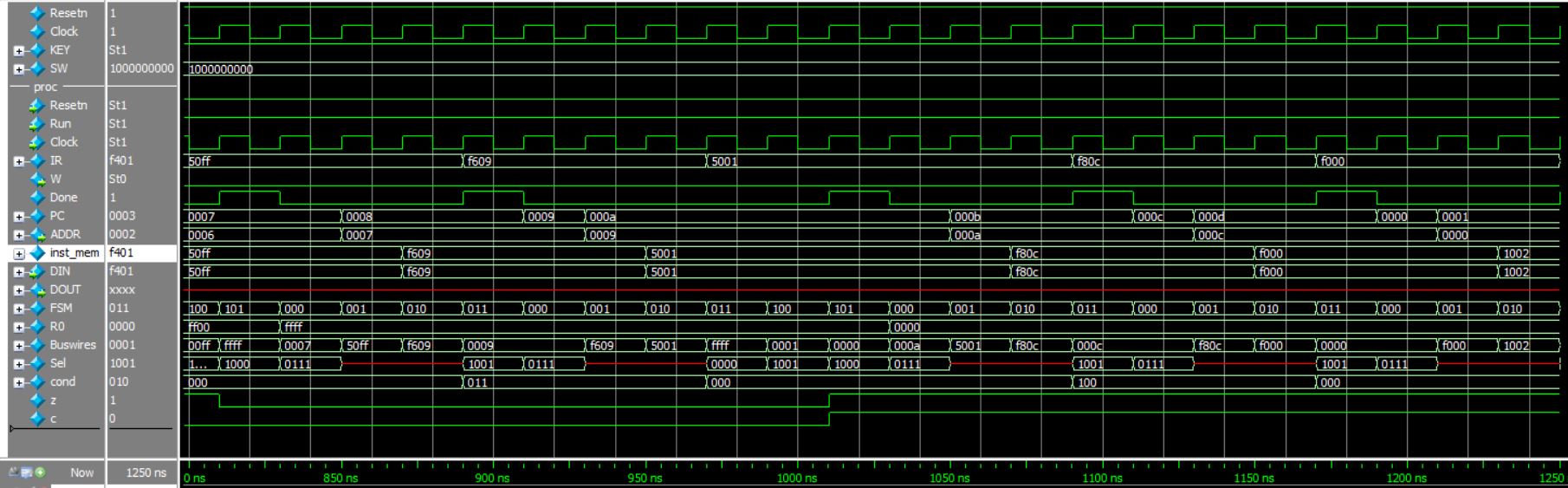
** **

**Part 5**

****

****

** **

****

**Part 6**

.define LED\_ADDRESS 0x1000

.define SW\_ADDRESS 0x3000

// Counter to count until 2^9 using leds on the DE1-SoC

// Can use SW to get delay.

mv r6, #0 // Start counting from 0

MAIN: mvt r0, #LED\_ADDRESS // Point to LED reg address

add r6, #1 // increment counter

st r6, [r0] // Store count to LED

mv r3, #DELAY

ld r3, [r3]

OUTER: mvt r1, #SW\_ADDRESS // Read SW address

ld r4, [r1] // load value from SW

add r4, #1 // add 1 to SW value in case 0

INNER: sub r4, #1 // decrement outer loop

bne #INNER // If inner loop not 0, decrement again

sub r3, #1 // decrement outer loop

bne #OUTER // If outer loop not 0, decrement again

b #MAIN

DELAY: .word 8888

**Part 7**

.define HEX\_ADDRESS 0x2000

.define SW\_ADDRESS 0x3000

.define STACK 256 // bottom of memory

======

// Count in Decimal from 0 to 65535 using the HEX displays

// r0 = counter register / remainder

// r1 = quotient

// r2 = digit register

// r3 = # of digits in number (up to 5)

// r4 = HEX\_ADDRESS pointer

// r5 = return addresses / Seg7 digit value

// r6 = stack pointer

// pc = well, the pc

mv r6, #STACK // point to stack!

mv r5, pc // return address for subroutine

mv pc, #BLANK // Call subroutine

MAIN: mv r0, #0 // Initialize counter

LOOP: mvt r4, #HEX\_ADDRESS // point to hex

mv r3, #5 // 5 digits in 65535

sub r6, #1 // store counter on stack

st r0, [r6]

DISP: mv r5, pc // return address

b #DIV10 // divide by 10!

sub r6, #1 // store r5

st r5, [r6]

mv r2, #DIGITS // address of digit '0'

add r2, r0 // get right digit

ld r5, [r2] // get actual digit!

st r5, [r4] // write into HEX0-HEX5

ld r5, [r6] // restore r5

add r6, #1

mv r0, r1 // put quotient in remainder

add r4, #1 // putting into next HEX display

sub r3, #1 // decrement # of digits left

bne #DISP // loop back for more digits

ld r0, [r6] // restore counter when done!!!

add r6, #1

mv r5, pc // return address

b #DELAYLOOP // do the delay!

add r0, #1 // counter r0 += 1

bcc #LOOP // continue until overflows!

mv r5, pc // return address for subroutine

mv pc, #BLANK // call subroutine to blank HEX!

b #MAIN // restart

DIV10: sub r6, #1 // Save modified registers

st r2, [r6] // Save on stack

mv r1, #0 // Initialize Q

DLOOP: mv r2, #9 // check if r0 < 10

sub r2, r0

bcc #REDTIV // if true, return

INC: add r1, #1 // if false, increment Q

sub r0, #10 // r0 -= 10

b #DLOOP // loop again

REDTIV: ld r2, [r6] // restore stack

add r6, #1

add r5, #1 // adjust return address

mv pc, r5 // return results!

// Subroutine BLANK

BLANK: sub r6, #1 // Save r0 into stack

st r0, [r6]

sub r6, #1 // Save r1 into stack

st r0, [r6]

sub r6, #1 // Save r2 into stack

st r2, [r6]

mvt r0, #HEX\_ADDRESS // pointer to HEX0

mv r1, #6 // Number of HEX displays

mv r2, #0x0000 // Blank digit

LUP: st r2, [r0] // Store blankness into HEX display

add r0, #1 // Point to next HEX display!

sub r1, #1 // decrement number of HEX displays left

bne #LUP // loop again if displays left

ld r2, [r6] // restore r2

add r6, #1

ld r1, [r6] // restore r1

add r6, #1

ld r0, [r6] //restore r0

add r5, #1

mv pc, r5 // Return back to where we were

// Delay subroutine

DELAYLOOP: sub r6, #1 // store r1

st r1, [r6]

sub r6, #1 // store r3

st r3, [r6]

sub r6, #1 // store r4

st r4, [r6]

mv r3, #DELAY // add the delay to r3

ld r3, [r3]

OUTER: mvt r1, #SW\_ADDRESS // Read SW address

ld r4, [r1] // load value from SW

add r4, #1 // add 1 to SW value in case 0

INNER: sub r4, #1 // decrement outer loop

bne #INNER // If inner loop not 0, decrement again

sub r3, #1 // decrement outer loop

bne #OUTER // If outer loop not 0, decrement again

ld r4, [r6] // restore all registers

add r6, #1

ld r3, [r6]

add r6, #1

ld r1, [r6]

add r6, #1

add r5, #1

mv pc, r5 // return back

// Data

DIGITS: .word 0b0000000000111111 // '0'

.word 0b0000000000000110 // '1'

.word 0b0000000001011011 // '2'

.word 0b0000000001001111 // '3'

.word 0b0000000001100110 // '4'

.word 0b0000000001101101 // '5'

.word 0b0000000001111101 // '6'

.word 0b0000000000000111 // '7'

.word 0b0000000001111111 // '8'

.word 0b0000000001101111 // '9'

DELAY: .word 2222