HOMEWORK I (Part 3/4/5) Report

Summary what has been done:

* RTL level design (16 bit counter)
* Pre-synthesis simulation
* Synthesis
* Post-synthesis simulation

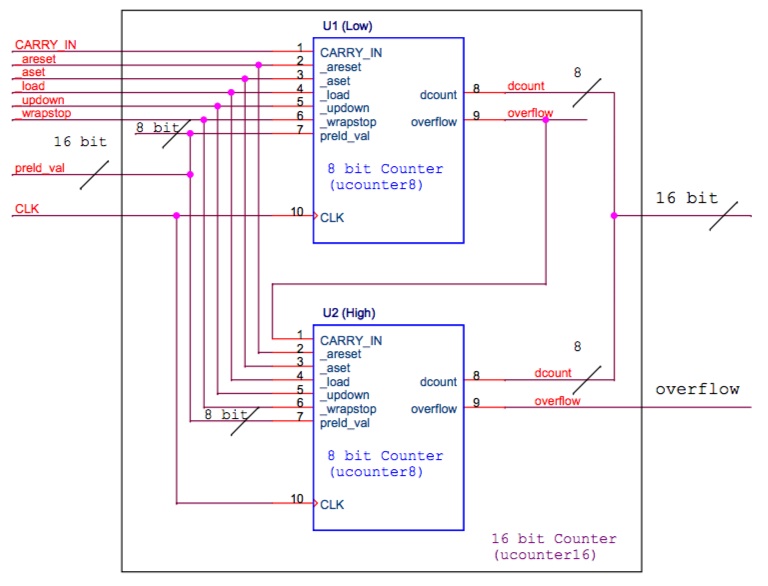
Required tools:

* Lab: NC-Verilog, Debussy (nTrace & nWave), Design Compiler(dv)
* Ubuntu: Icarus Verilog (iverilog), gtkwave, Dia

Learned lessons:

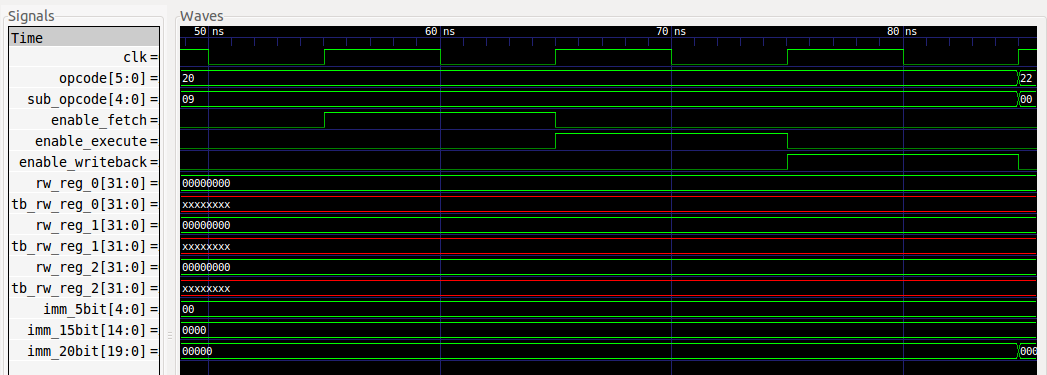
* Stuck:
  + Unfamiliar with Verilog language.
  + Design 16 bit counter with 8 bit counter modules.
  + Draw block diagram.
* Find:
* Exciting things:

16 bit counter Design:

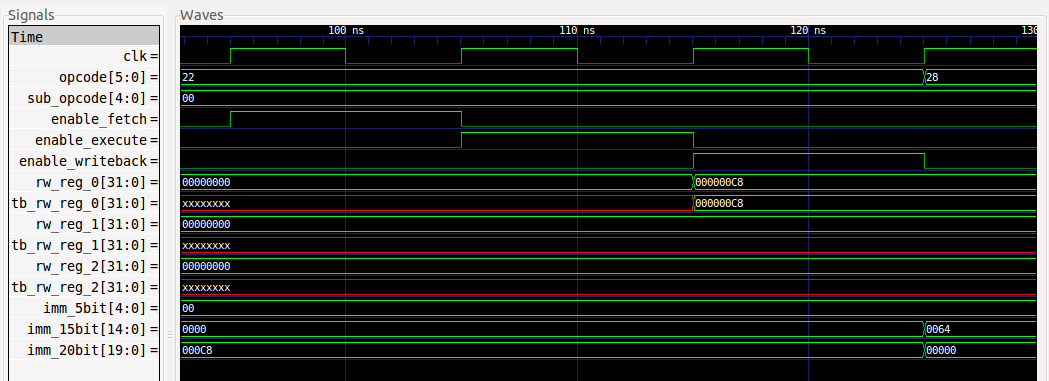


Part 3 pre-simulated waveforms:

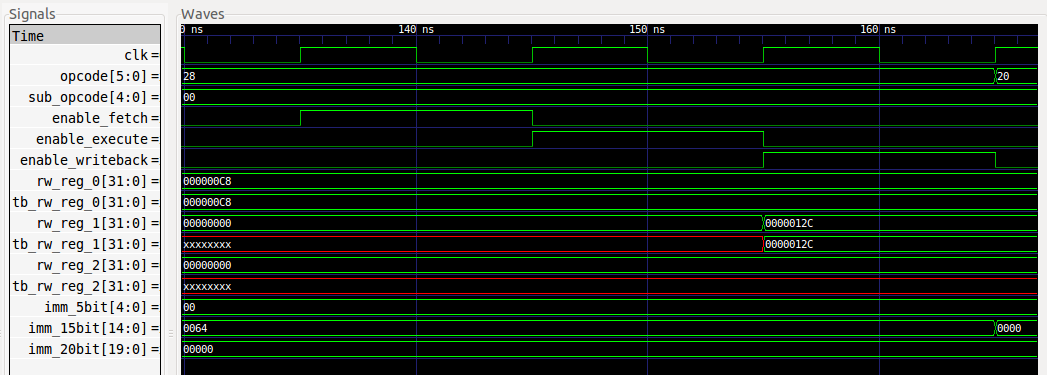
* Test1. NOP



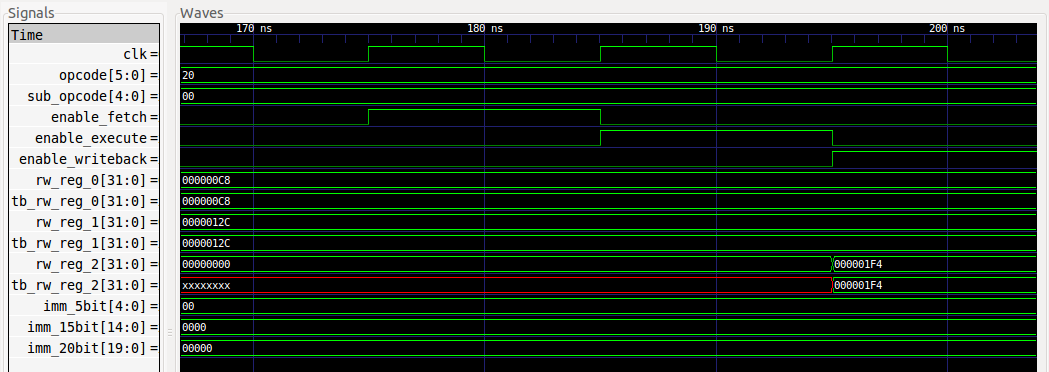
* Test 2. MOVI(rw\_reg[0] = ‘h00C8)



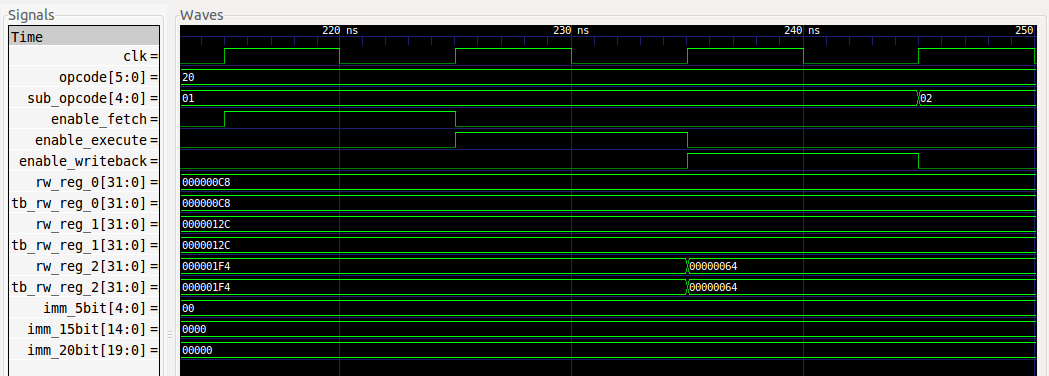
* Test 3. ADDI(rw\_reg[1]=rw\_reg[0]+’h0064)



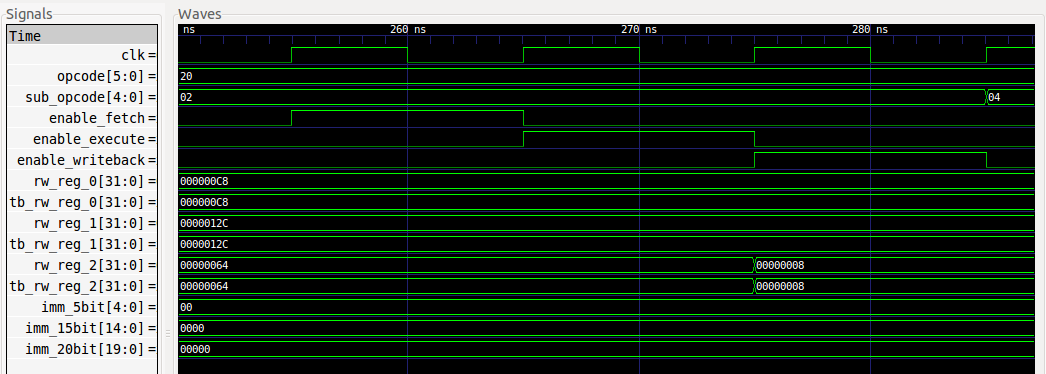
* Test 4. ADD(rw\_reg[2]= rw\_reg[0]+ rw\_reg[1])



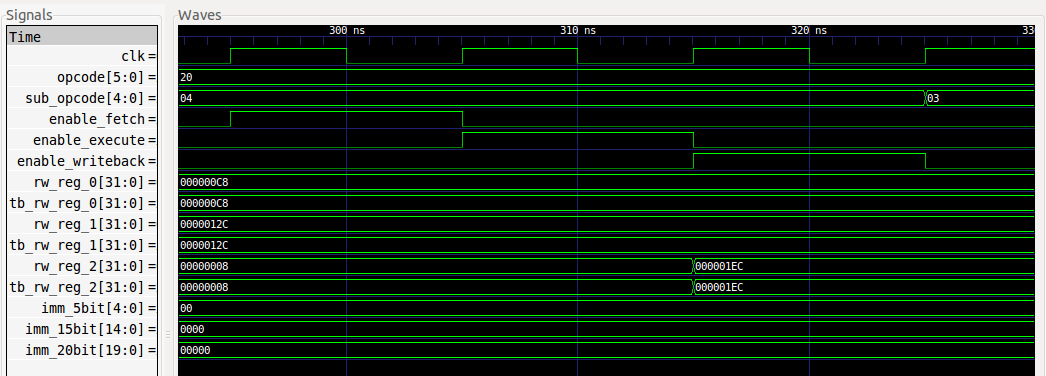
* Test 5. SUB(rw\_reg[2]= rw\_reg[1]- rw\_reg[0])



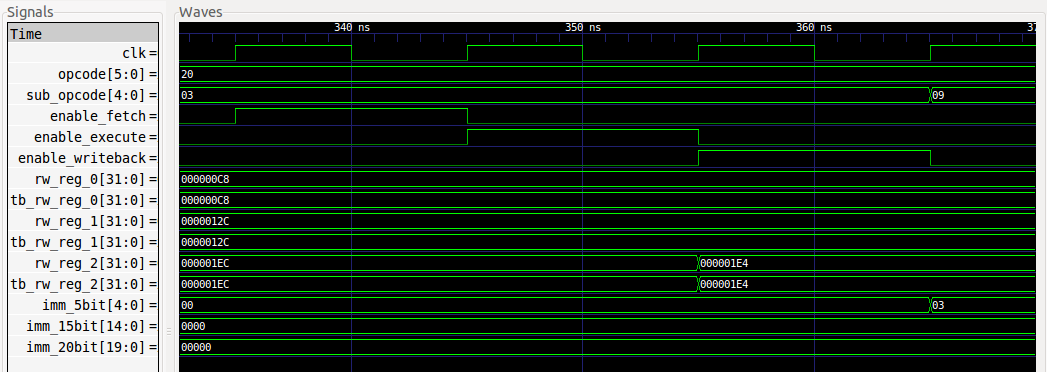
* Test 6. AND(rw\_reg[2]= rw\_reg[0]& rw\_reg[1])



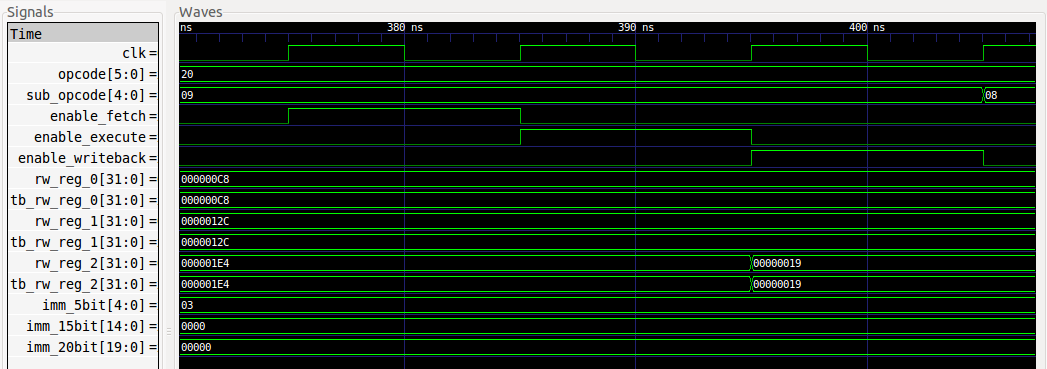
* Test 7. OR(rw\_reg[2]= rw\_reg[0]| rw\_reg[1])



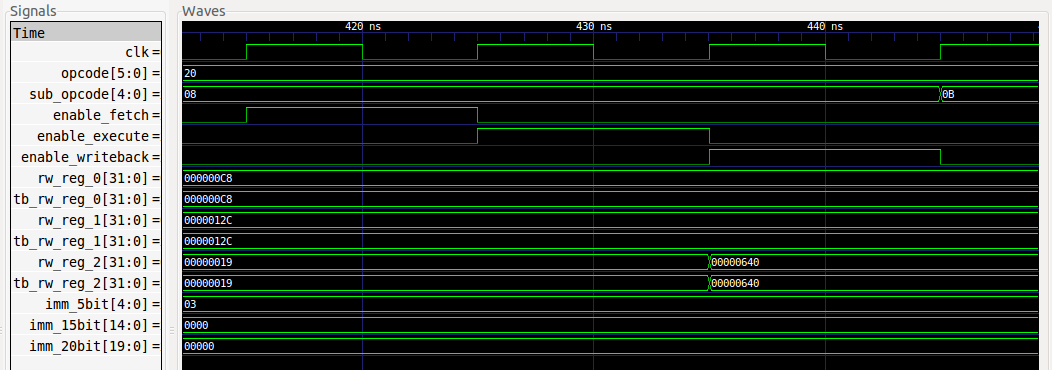
* Test 8. XOR(rw\_reg[2]= rw\_reg[0]^ rw\_reg[1])



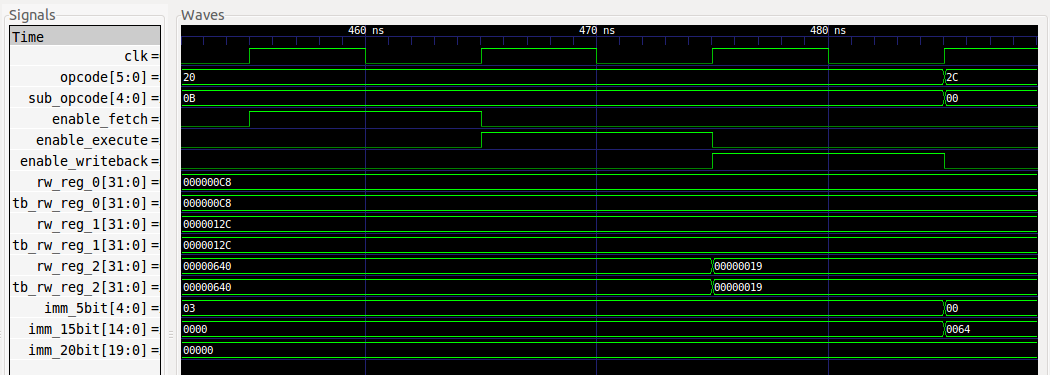
* Test 9. SRLI(rw\_reg[2]= rw\_reg[0]>>’h0003)



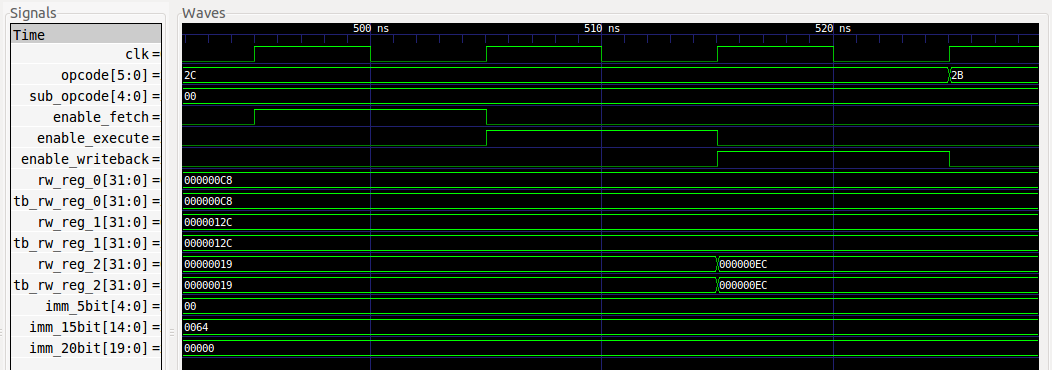
* Test 10. SLLI(rw\_reg[2]= rw\_reg[0] <<’h0003)



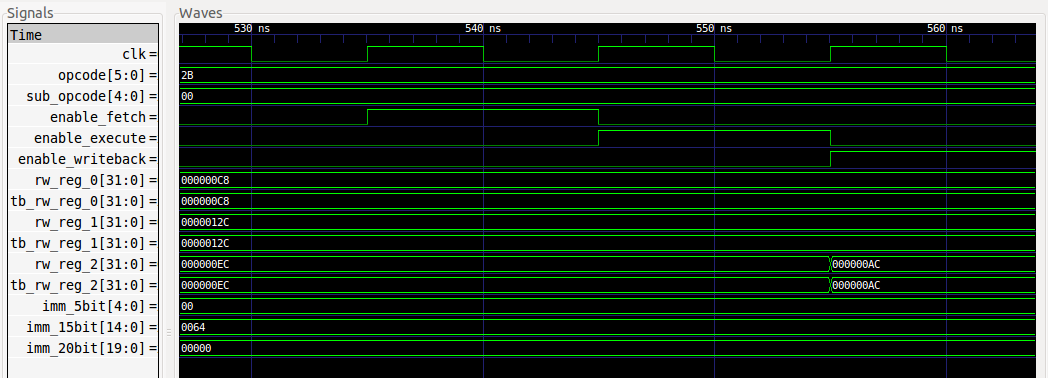
* Test 11. ROTRI(rw\_reg[2]= rw\_reg[0] >>’h0003)



* Test 12. ORI(rw\_reg[2]= rw\_reg[0] |’h0064)

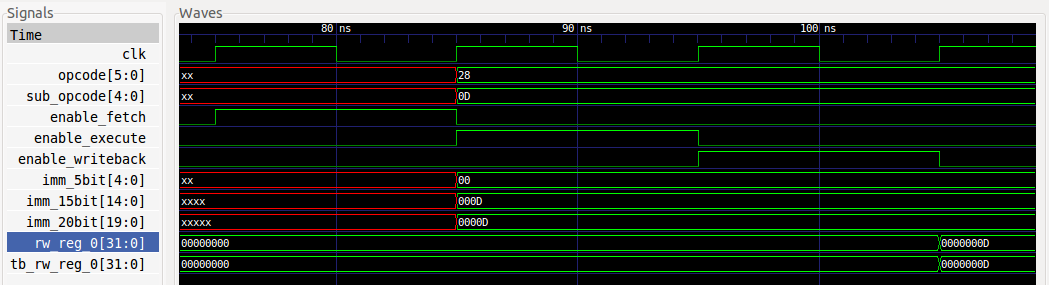


* Test 13. XORI(rw\_reg[2]= rw\_reg[0] ^’h0064)

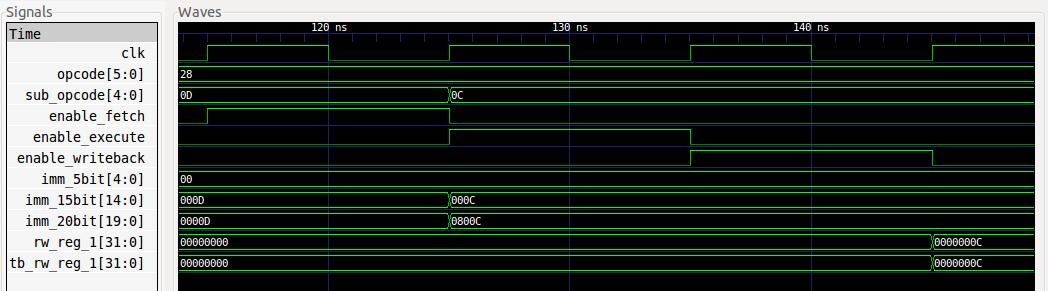


Part 4-1 pre-simulated waveforms:

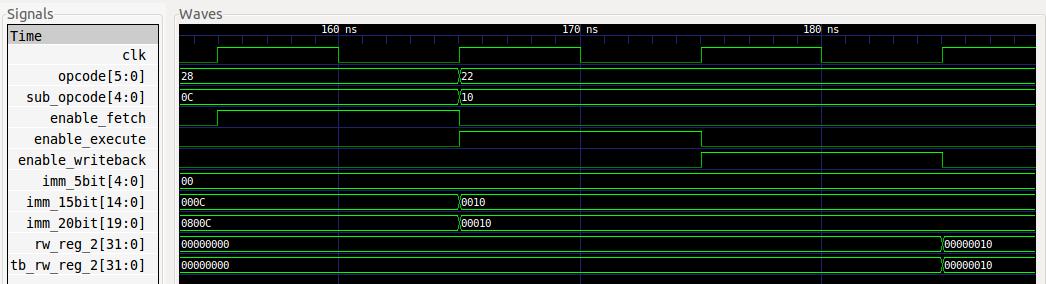
* Test 1. ADDI (R0=R0+5’b01101)



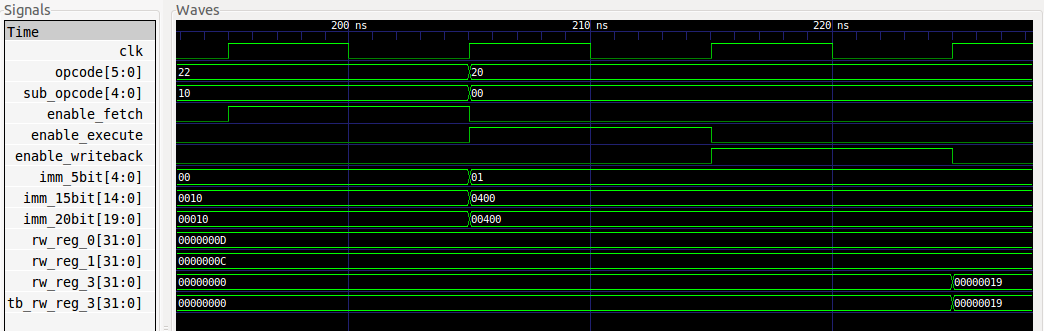
* Test 2. ADDI (R1=R1+5’b01100)



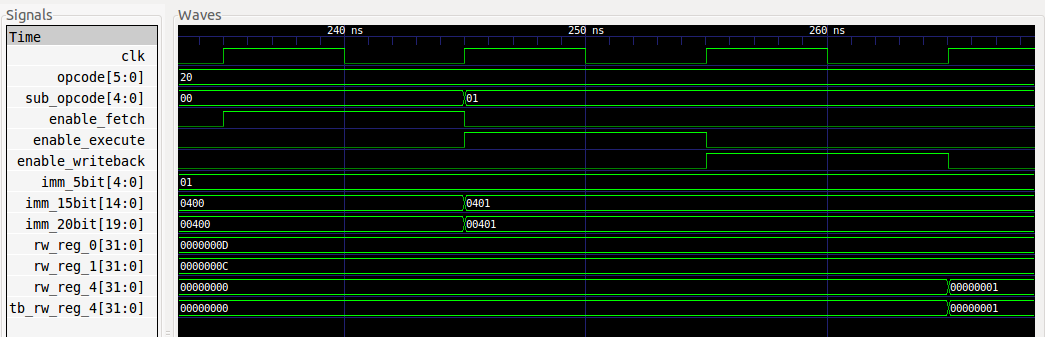
* Test 3. MOVI (R2= 5’b01000)



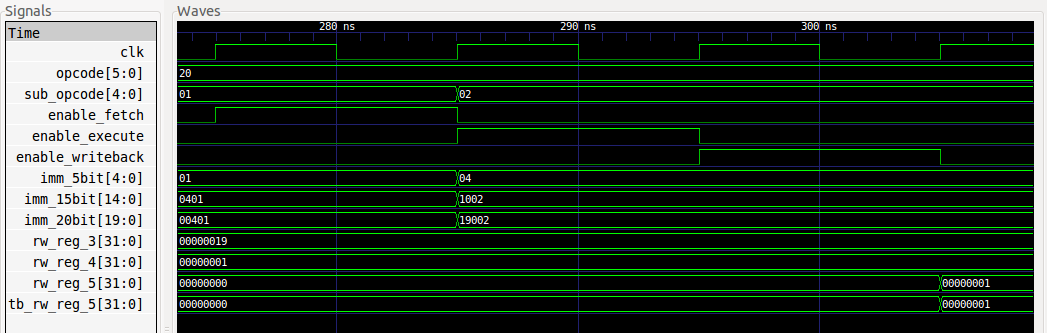
* Test 4. ADD (R3=R0+R1)



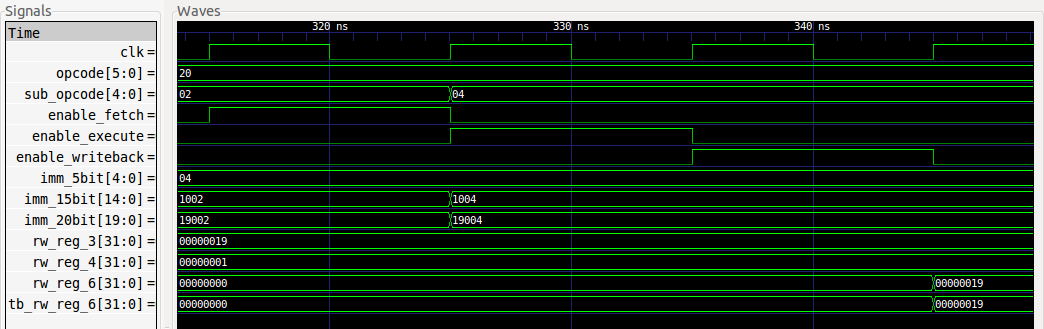
* Test 5. SUB (R4=R0-R1)



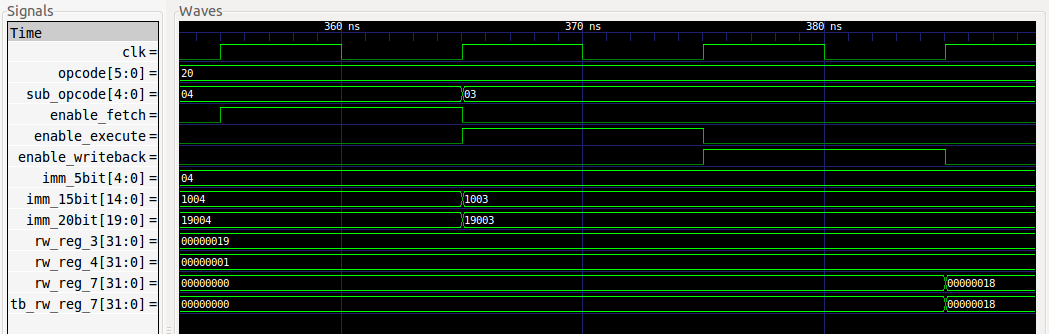
* Test 6. AND (R5=R3&R4)



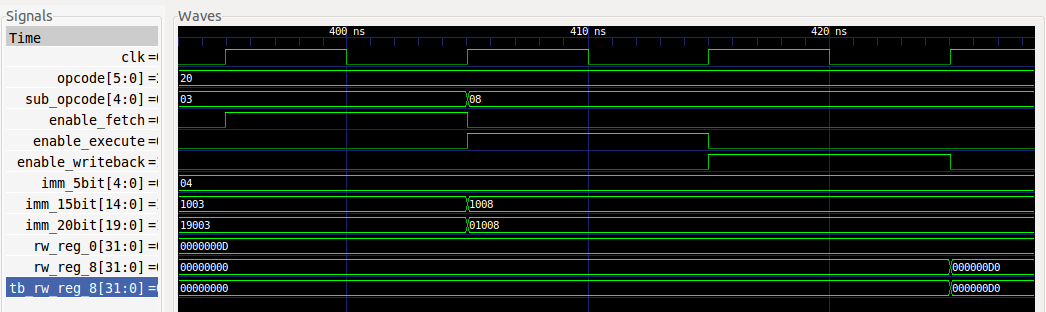
* Test 7. OR (R6=R3|R4)



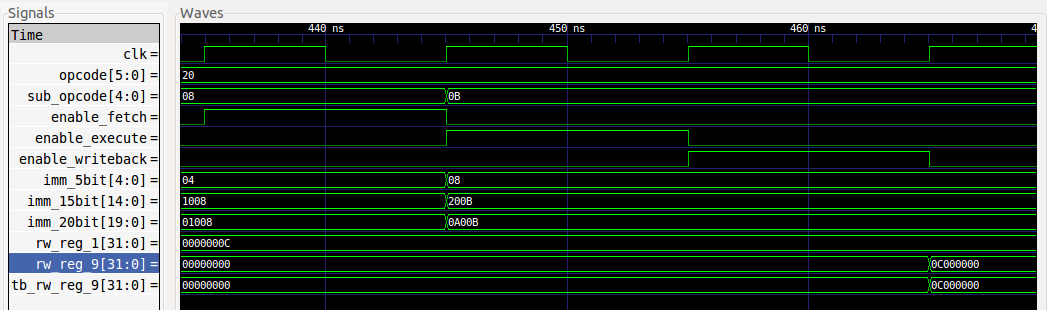
* Test 8. XOR(R7= R3^R4)



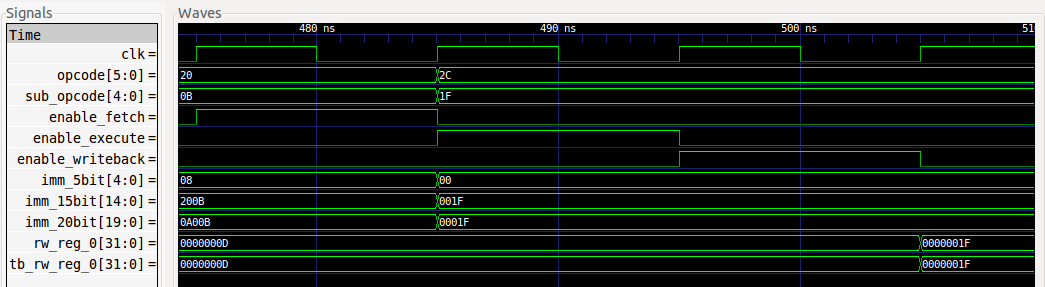
* Test 9. SLLI (R8=R0<<5’b00100)



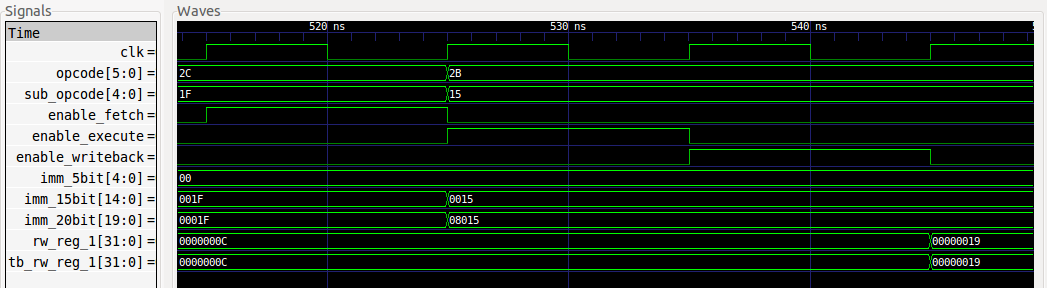
* Test 10. ROTRI (R9=R1>>5’b01000)



* Test 11. ORI (R0=R0|5’b11111)



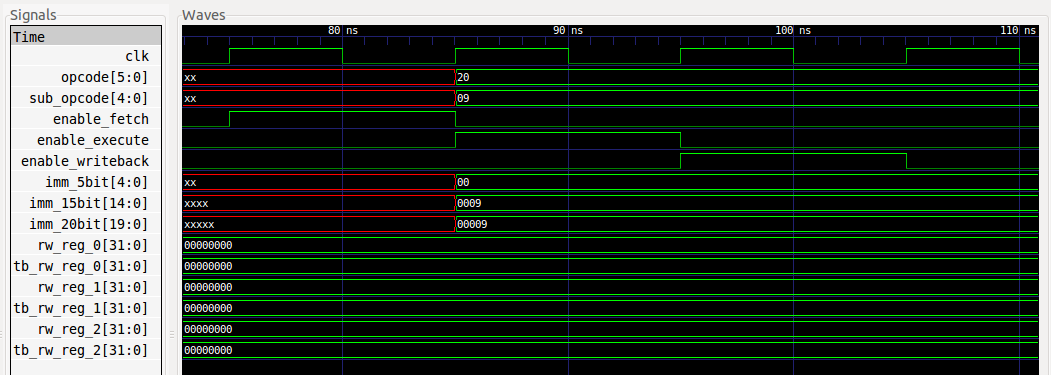
* Test 12. XORI (R1=R1+5’b10101)



Part 4-2 pre-simulated waveforms:

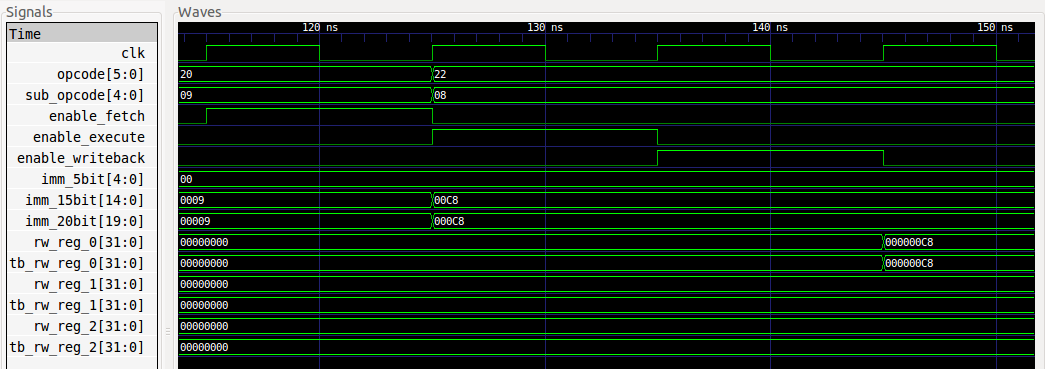
* Test1. NOP

*instruction = 32'b0\_100000\_00000\_00000\_00000\_00000\_01001;*



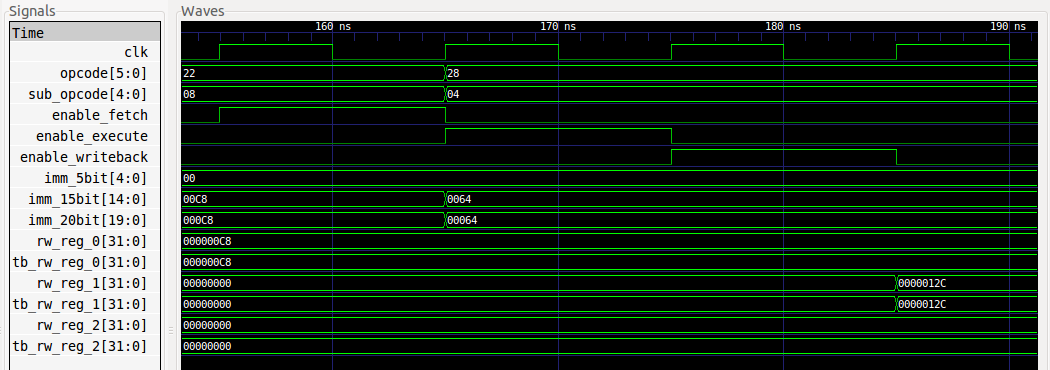
* Test 2. MOVI(rw\_reg[0] = ‘h00C8)

*instruction = 32'b0\_100010\_00000\_0000\_0000\_0000\_1100\_1000;*



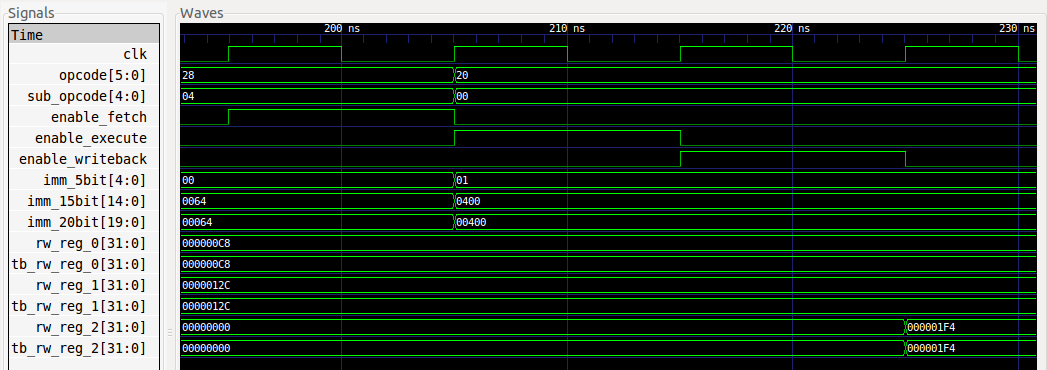
* Test 3. ADDI(rw\_reg[1]=rw\_reg[0]+’h0064)

*instruction = 32'b0\_101000\_00001\_00000\_0000\_0000\_1100\_100;*



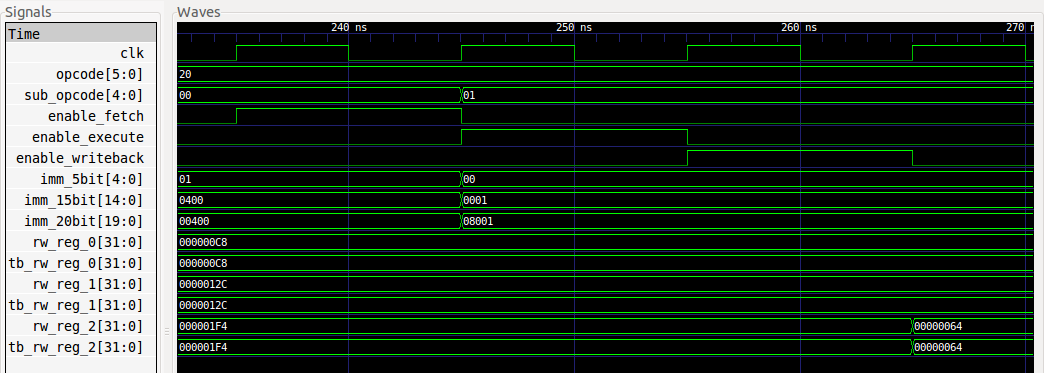
* Test 4. ADD(rw\_reg[2]= rw\_reg[0]+ rw\_reg[1])

*instruction = 32'b0\_100000\_00010\_00000\_00001\_00000\_00000;*



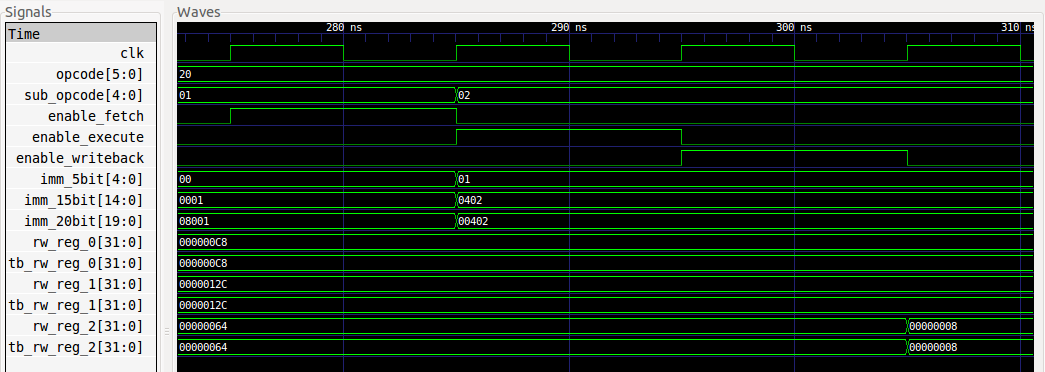
* Test 5. SUB(rw\_reg[2]= rw\_reg[1]- rw\_reg[0])

*instruction = 32'b0\_100000\_00010\_00001\_00000\_00000\_00001;*



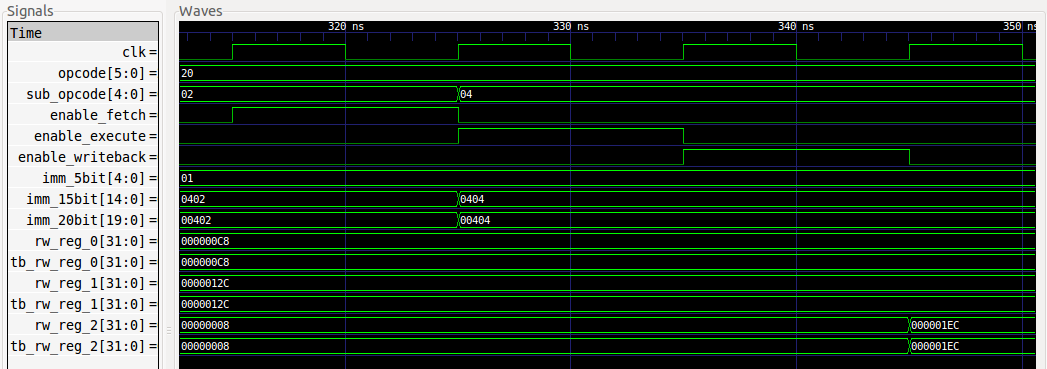
* Test 6. AND(rw\_reg[2]= rw\_reg[0]& rw\_reg[1])

*instruction = 32'b0\_100000\_00010\_00000\_00001\_00000\_00010;*



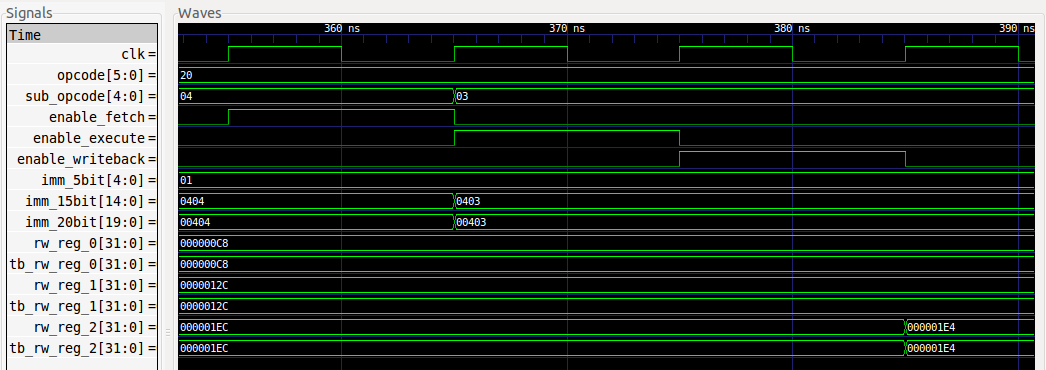
* Test 7. OR(rw\_reg[2]= rw\_reg[0]| rw\_reg[1])

*instruction = 32'b0\_100000\_00010\_00000\_00001\_00000\_00100;*



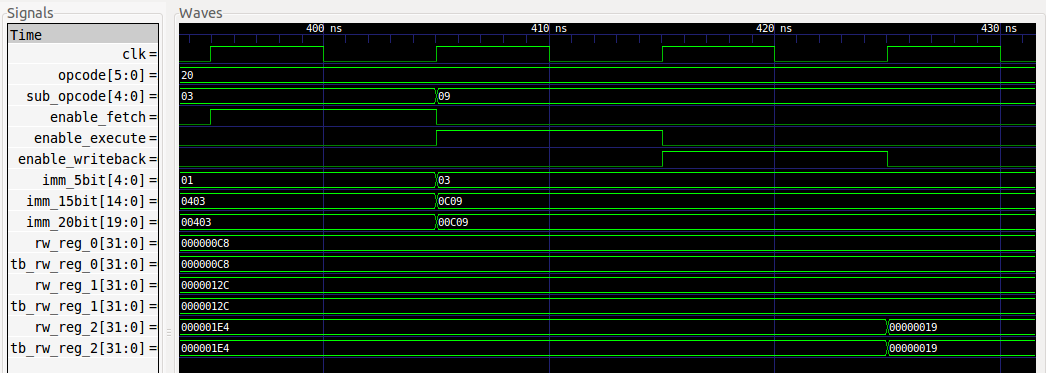
* Test 8. XOR(rw\_reg[2]= rw\_reg[0]^ rw\_reg[1])

*instruction = 32'b0\_100000\_00010\_00000\_00001\_00000\_00011;*



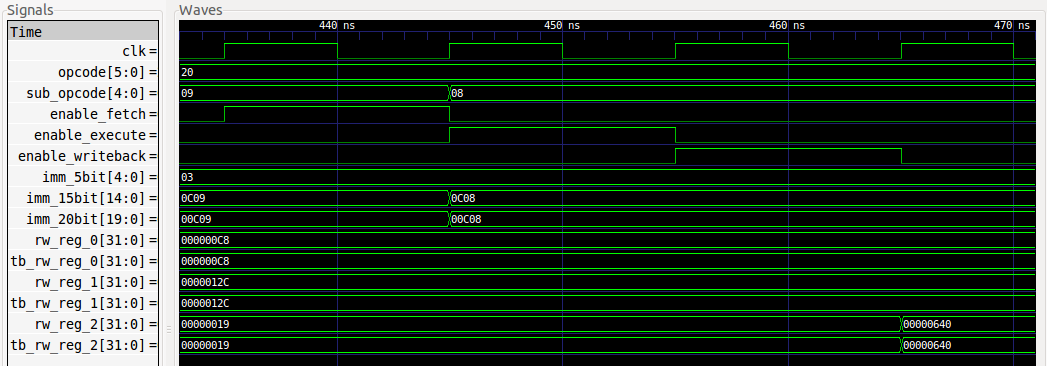
* Test 9. SRLI(rw\_reg[2]= rw\_reg[0]>>’h0003)

*instruction = 32'b0\_100000\_00010\_00000\_00011\_00000\_01001;*



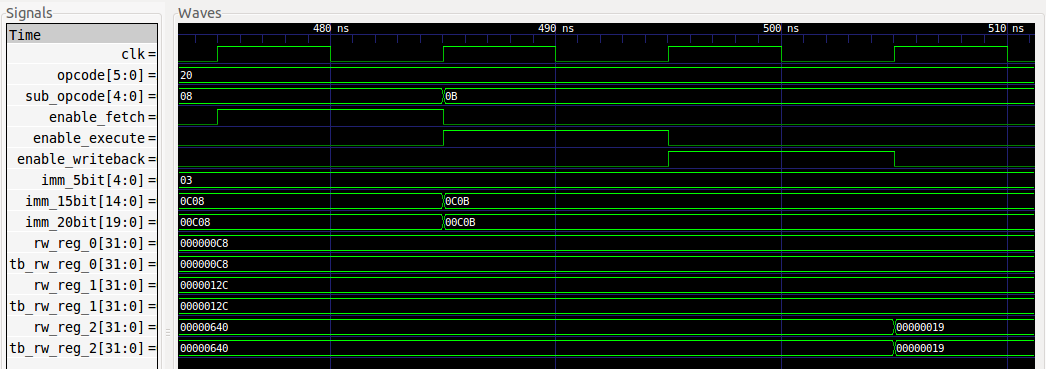
* Test 10. SLLI(rw\_reg[2]= rw\_reg[0] <<’h0003)

*instruction = 32'b0\_100000\_00010\_00000\_00011\_00000\_01000;*



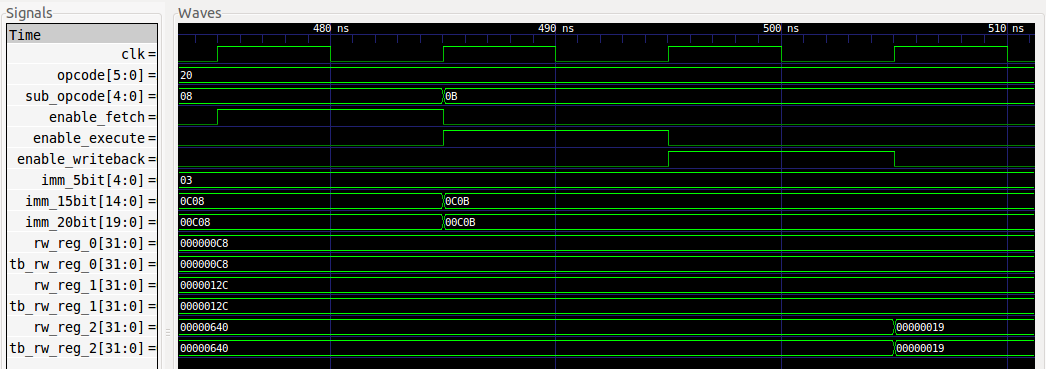
* Test 11. ROTRI(rw\_reg[2]= rw\_reg[0] >>’h0003)

*instruction = 32'b0\_100000\_00010\_00000\_00011\_00000\_01011;*



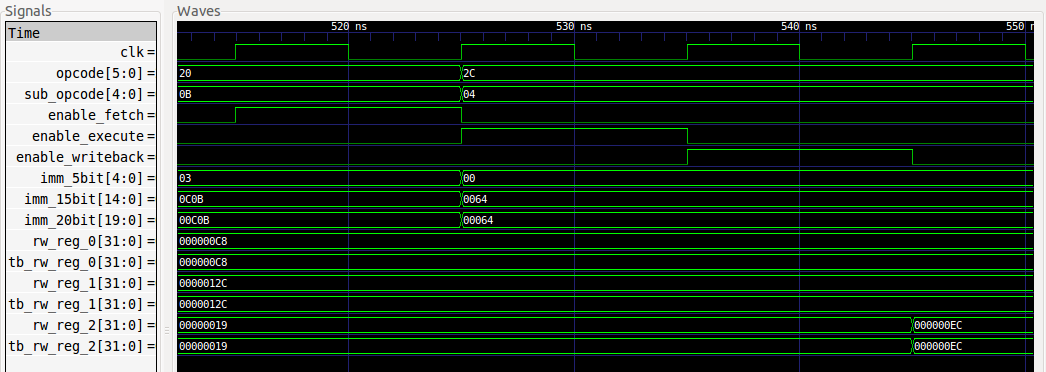
* Test 12. ORI(rw\_reg[2]= rw\_reg[0] |’h0064)

*instruction = 32'b0\_101100\_00010\_00000\_0000\_0000\_1100\_100;*



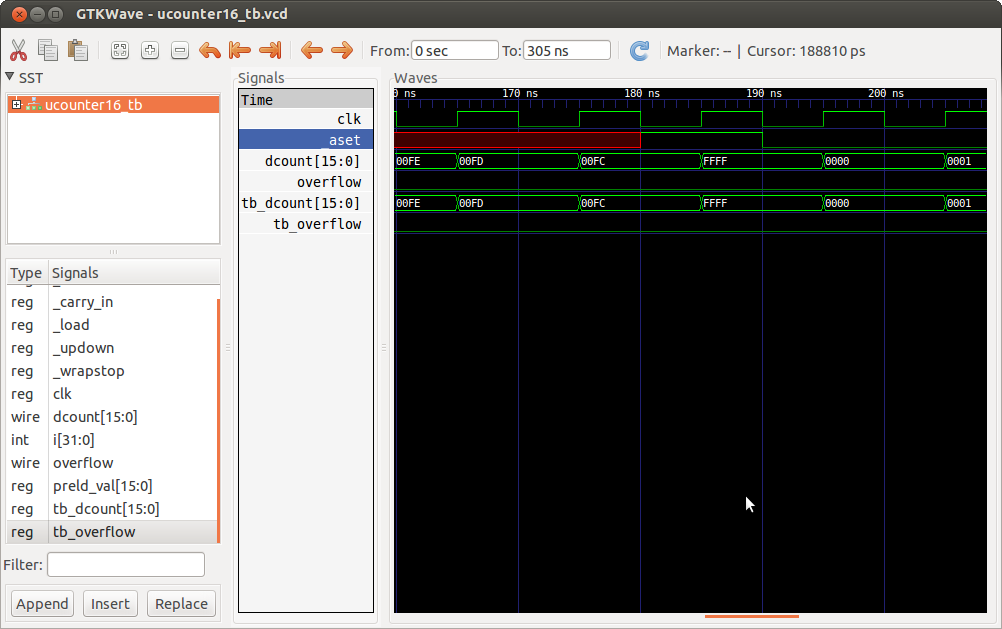
* Test 13. XORI(rw\_reg[2]= rw\_reg[0] ^’h0064)

*instruction = 32'b0\_101011\_00010\_00000\_0000\_0000\_1100\_100;*

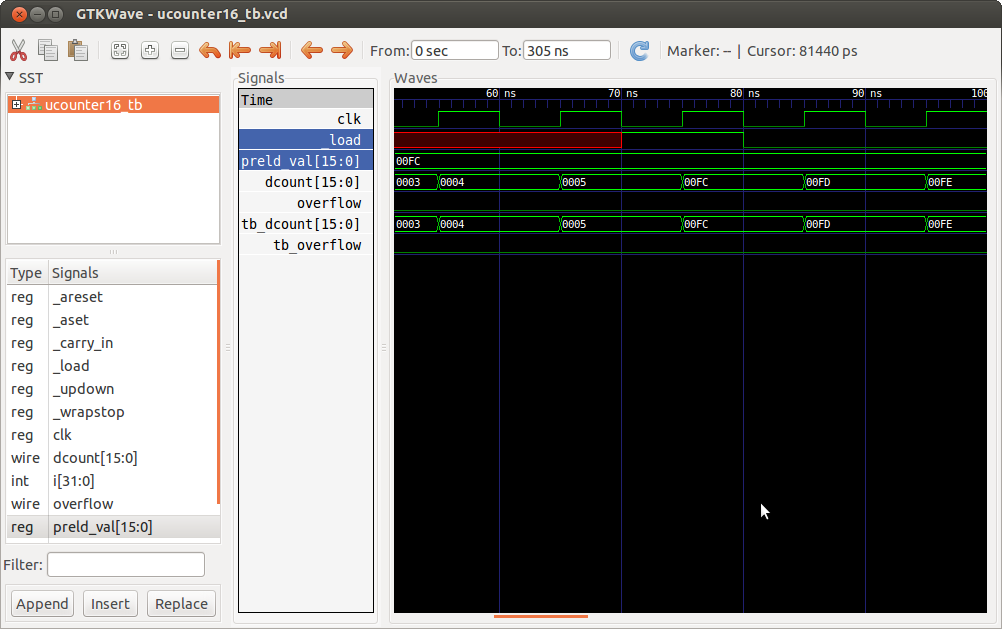


Post-simulated waveforms:

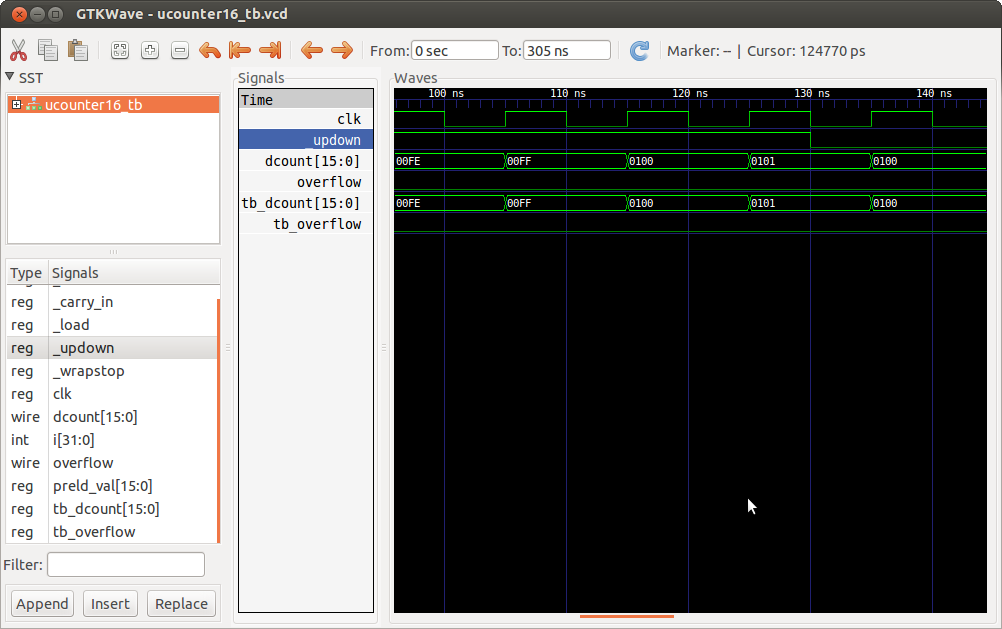
* Test 2. \_aset



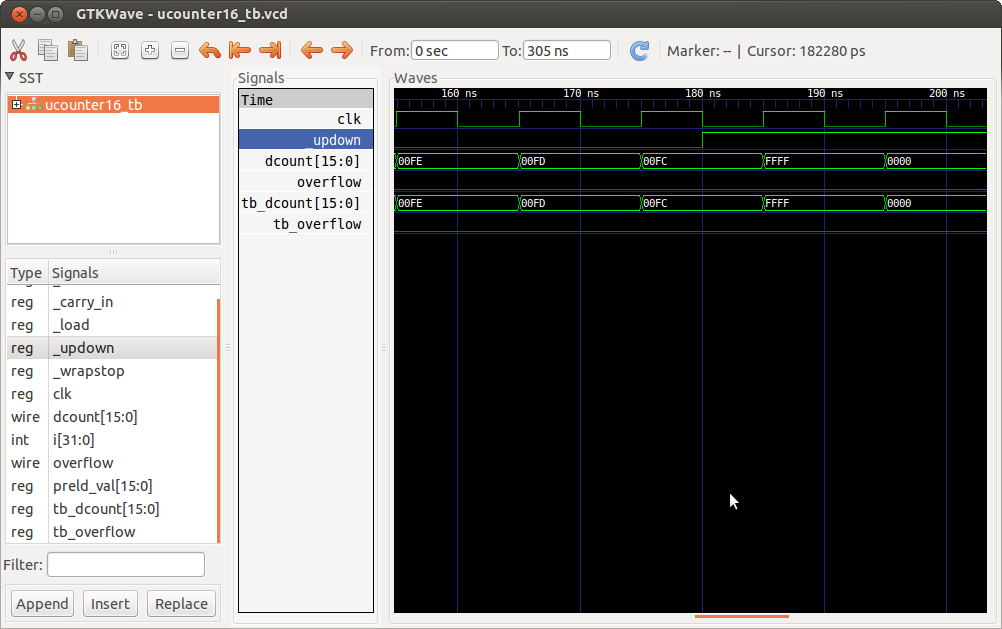
* Test 3. \_load and preld\_val



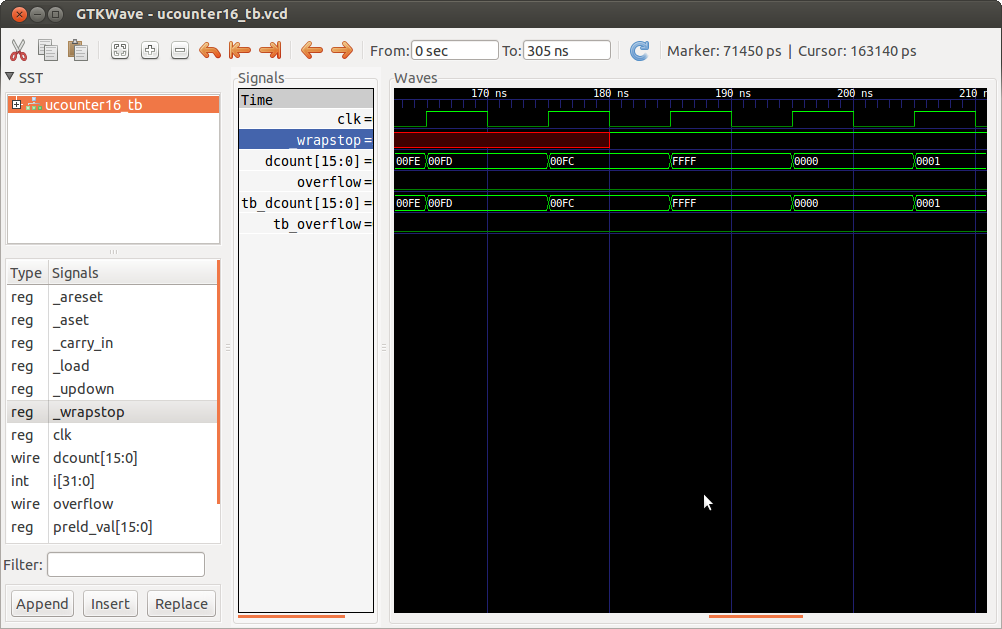
* Test 4. \_updown (up)



* Test 5. \_updown (down)



* Test 6. \_wrapstop (cycle)



* Test 7. \_wrapstop (stop) and overflow

