Introduction to Digital Logic

EECS/CSE 31L

**Final Project: Simple Processor**

Prepared by: Jack Melcher

Student ID: 67574625

EECS Department

Henry Samueli School of Engineering

University of California, Irvine

March 10, 2015

**1 Block Description**

The processor fetches a 32-bit set of instructions, decodes the instructions in a controller, performs the specified task, and stores the results into a register file.

The processor is made up of several components.

* Counter (assignment 4)
* Memory
* Controller
* Register File (assignment 4)
* Selector for A input
* Selector for B input
* ALU 32 (assignment 3)

**2 Input/Output Port Description**

Package(c31L\_pack): Contains all the necessary constants used throughout all the blocks

|  |  |  |  |
| --- | --- | --- | --- |
| Constant Name | Data Type | Default Value | Description |
| BW | INTEGER | 32 | Size of logic that is being used |
| OP\_BINARY | STD\_LOGIC | 010010 (18) | Number of Operations in the Memory |
| Reg\_field | INTEGER | 6 | Determines size of register file and used to as its addr size |
| NBIT | INTEGER | 6 | Max bits the counter uses |
| Alu\_function\_type | std\_logic\_vector | 3 downto 0 | Used to help tie an alu operation to a given number |
| Mux\_instructions | Array of std\_logic\_vector | ((OP-1) downto 0)  (BW-1 downto 0) | Stores all the hard coded instructions for the processor |

Processor (The outputs are used to help display values of processor instruction results)

|  |  |  |  |
| --- | --- | --- | --- |
| Port name | Port size | Port Type | Description |
| clk | 1 | IN | Trigger the processor fetch and write events |
| Pc\_enable | 1 | IN | Enable the counter to count |
| Rst\_s | 1 | IN | Synchronous reset for the Counter and RegFile |
| memory\_addrs | NBIT-1 downto 0 | OUT | Determine which instruction is read |
| Instruction | BW-1 downto 0 | OUT | Shows the 32 bit instruction |
| reg\_source | BW-1 downto 0 | OUT | The first value read from register file |
| reg\_target | BW-1 downto 0 | OUT | The second value read from register file |
| immediate | BW-1 downto 0 | OUT | The immediate value from instruction |
| carry | 1 | OUT | The carry out value from ALU |
| output | BW-1 downto 0 | OUT | Results of ALU operation |

Counter

|  |  |  |  |
| --- | --- | --- | --- |
| Port name | Port size | Port Type | Description |
| Clk | 1 | IN | Triggers the counting |
| Rst\_s | 1 | IN | Will reset the counter to zero when active |
| dout | NBIT-1 downto 0 | OUT | The counter’s value after counting |

|  |  |  |  |
| --- | --- | --- | --- |
| Variable Name | Data Type | Port size | Description |
| count | INTEGER | NBIT-1 downto 0 | Aids in the counting process |

Memory

|  |  |  |  |
| --- | --- | --- | --- |
| Port name | Port size | Port Type | Description |
| addr | NBIT-1 downto 0 | IN | Determines which instructions are fetched |
| Read\_data | BW-1 downto 0 | OUT | The instructions that are sent to controller |

Controller

|  |  |  |  |
| --- | --- | --- | --- |
| Port name | Port size | Port Type | Description |
| instruction | 31 downto 0 | IN | Instructions fetched from memory |
| rt\_and\_imm | 31 downto 0 | OUT | Immediate value |
| write\_enable | 1 | OUT | Determines whether the register file can write |
| rs\_index | 2 downto 0 | OUT | Address of register source |
| rt\_index | 1 | OUT | Address of register target |
| rd\_index | 1 | OUT | Address of register destination |
| b\_mux\_sel | 31 downto 0 | OUT | Instruction type to determine B input of ALU |
| alu\_func | alu\_function\_type | OUT | Operation the ALU is supposed to perform |

Register File

|  |  |  |  |
| --- | --- | --- | --- |
| Port name | Port size | Port Type | Description |
| Clk | 1 | IN | Triggers the register process |
| Rst\_s | 1 | IN | Will reset the registers to zero |
| write\_enable | 1 | IN | Enables writing to the registers |
| rs\_index | Reg\_field -1 downto 0 | IN | The address of the register to be read from |
| rt\_index | Reg\_field -1 downto 0 | IN | The address of the register to be read from |
| rd\_index | Reg\_field -1 downto 0 | IN | The address of the register to be written to |
| reg\_source\_out | BW-1 downto 0 | OUT | The data value read from register file |
| reg\_target\_out | BW-1 downto 0 | OUT | The data value read from register file |
| reg\_dest\_new | BW-1 downto 0 | IN | The data value to be written to the register |

|  |  |  |  |
| --- | --- | --- | --- |
| Variable Name | Data Type | Port size | Description |
| regfile | Array (0 to 2\*\* Reg\_field) | NBIT-1 downto 0 | The registers |

Selector for A input

|  |  |  |  |
| --- | --- | --- | --- |
| Port name | Port size | Port Type | Description |
| reg\_source | 31 downto 0 | IN | The data value read from register file |
| immediate | 31 downto 0 | IN | Immediate provided by the controller |
| alu\_op | 1 | INOUT | The expected ALU operation to be performed |
| instruction\_type |  |  | Selects A input based on RI value |
| opsel | 2 downto 0 | IN | The ALU operation select value |
| mode | 1 | OUT | The ALU mode select value |
| output | 31 downto 0 | OUT | What value is sent to A input of ALU |

Selector for B input

|  |  |  |  |
| --- | --- | --- | --- |
| Port name | Port size | Port Type | Description |
| in0 | BW-1 downto 0 | IN | Register target value |
| in1 | BW-1 downto 0 | IN | Immediate value |
| sel | 1 | INOUT | Selects B input based on RI value |
| output | BW-1 downto 0 |  | What value is sent to B input of ALU |

ALU 32-bit

|  |  |  |  |
| --- | --- | --- | --- |
| Port name | Port size | Port Type | Description |
| A | 31 downto 0 | IN | Value of first input |
| B | 31 downto 0 | IN | Value of second input |
| Cin | 1 | SIGNAL | The carry in value |
| opsel | 2 downto 0 | IN | Determine which operation to perform in units |
| mode | 1 | OUT | Determine whether the output is from arithmetic operation or logic operation |
| cout | 1 | OUT | The carry out value |
| output | 31 downto 0 | OUT | The output value of operation |

**3 Design Schematics**

**Instruction Format**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **1 bit** | **6 bit** | **6 bit** | **4 bit** | **6 bit** | **9bit** |
| **Instruction Type (RI)** | **Register Source (rs)** | **Register Destination (rd)** | **Function** | **Register Target (rt)** | **Immediate (Imm)** |
| Whether ALU works with Immediate | Address of rs | The address for which the results of ALU will be written to | Code for ALU operation | Address of rt and the first 6 bits of Immediate | The last 9 bits of the Immediate |

**ALU Operations:**

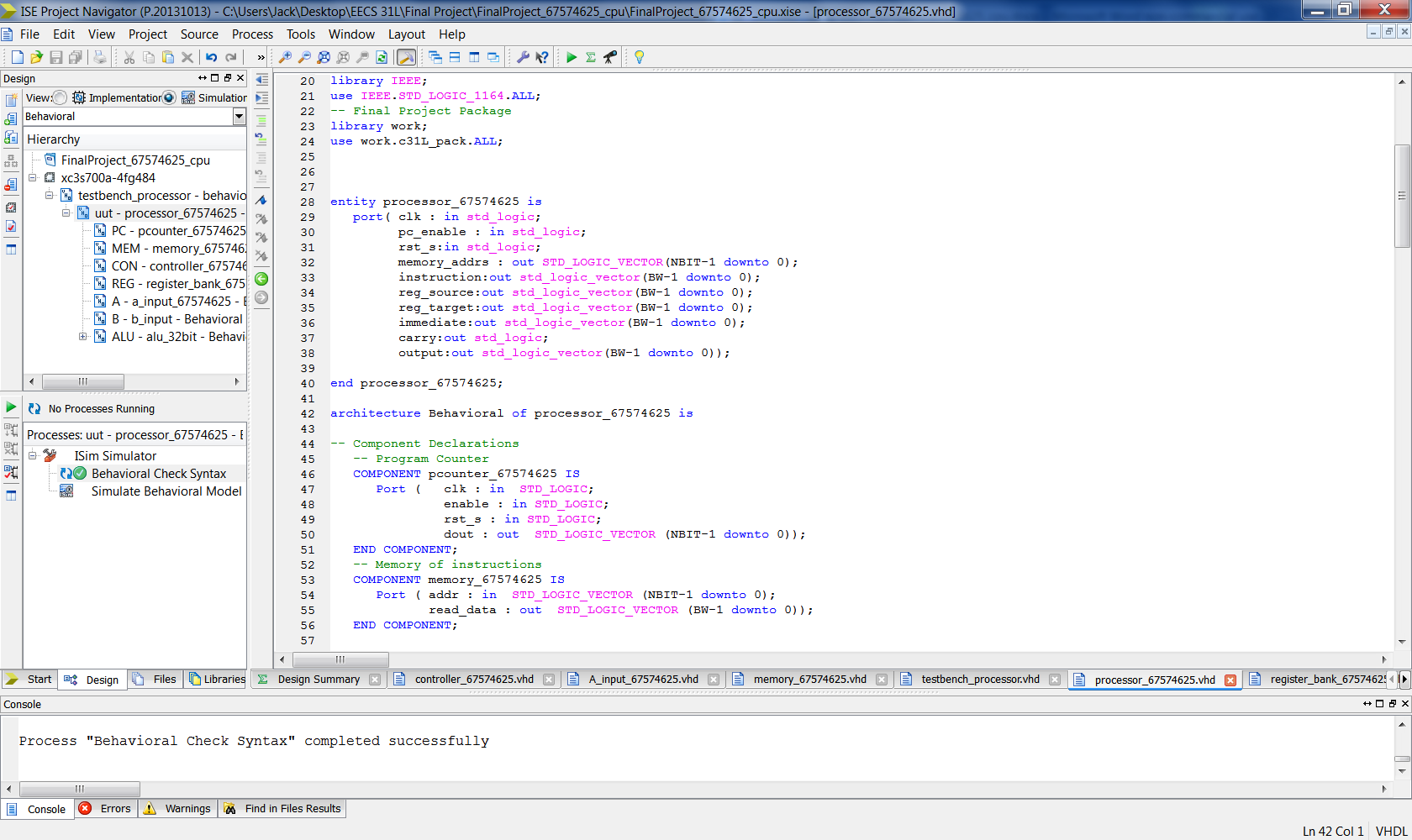
|  |  |  |  |
| --- | --- | --- | --- |
| **Function** | **Register Operation (RI = ‘0’)** | **Immediate Operation (RI = ‘1’)** | **Description** |
| 0000 | NOP | NOP | No Operation |
| 0001 | rs + rt | a+ Imm | Add |
| 0010 | rs + rt ' | a+ Imm ' | Sub |
| 1011 | rs | Imm | Move |
| 1100 | rs +1 | Imm +1 | increment |
| 1101 | rs -1 | Imm -1 | decrement |
| 1110 | rs + rt +1 | a+ Imm +1 | Add & Increment |
| 0101 | rs AND rt | a AND Imm | Bit-wise AND |
| 0110 | rs OR rt | a OR Imm | Bit-wise OR |
| 1000 | rs XOR rt | a XOR Imm | Bit-wise Exclusive OR |
| 0111 | rs ' | Imm' | Compliment |
| 1001 | shl rs | shl Imm | 1 Bit Shift Left |

**Design:**

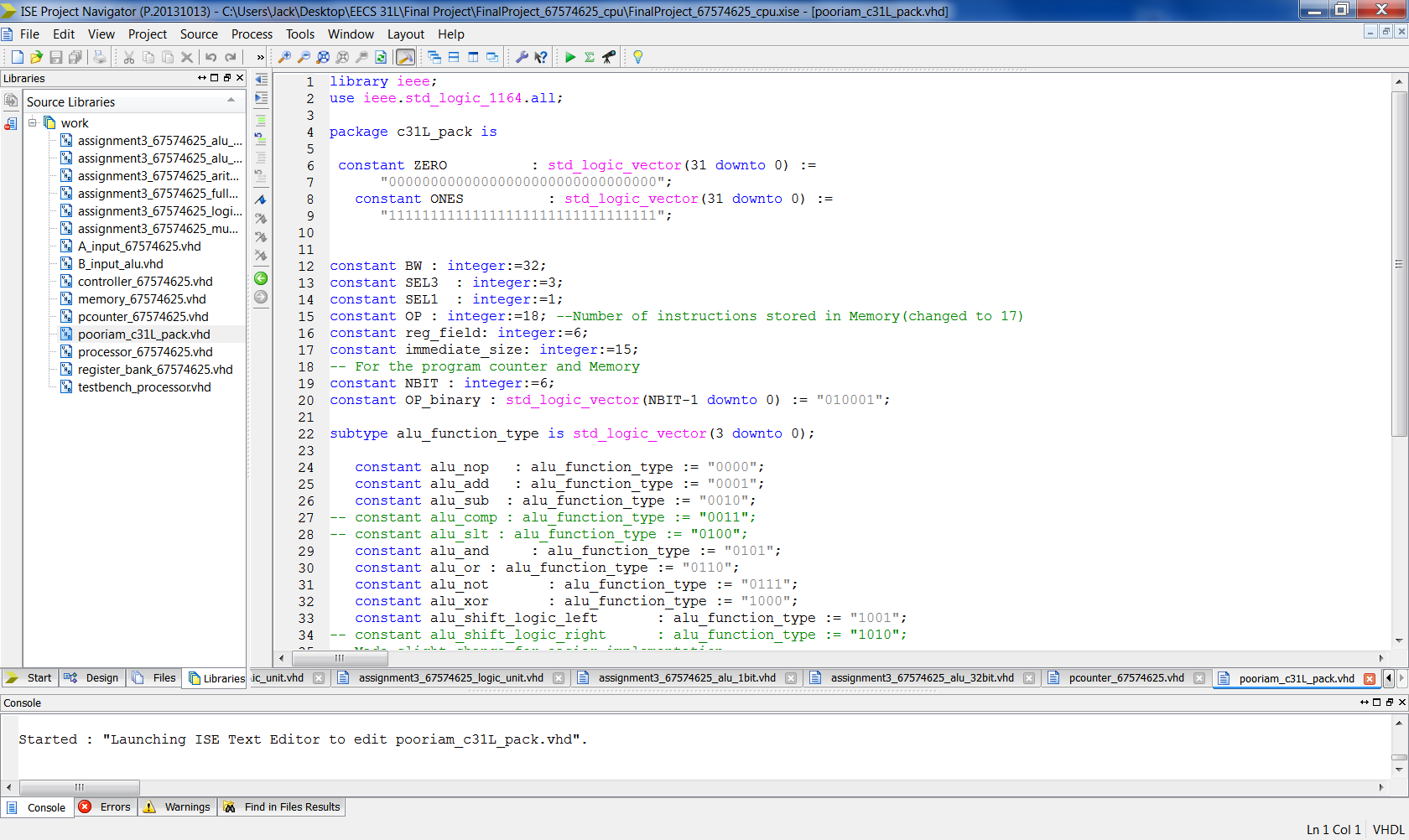


**4 Compilation**

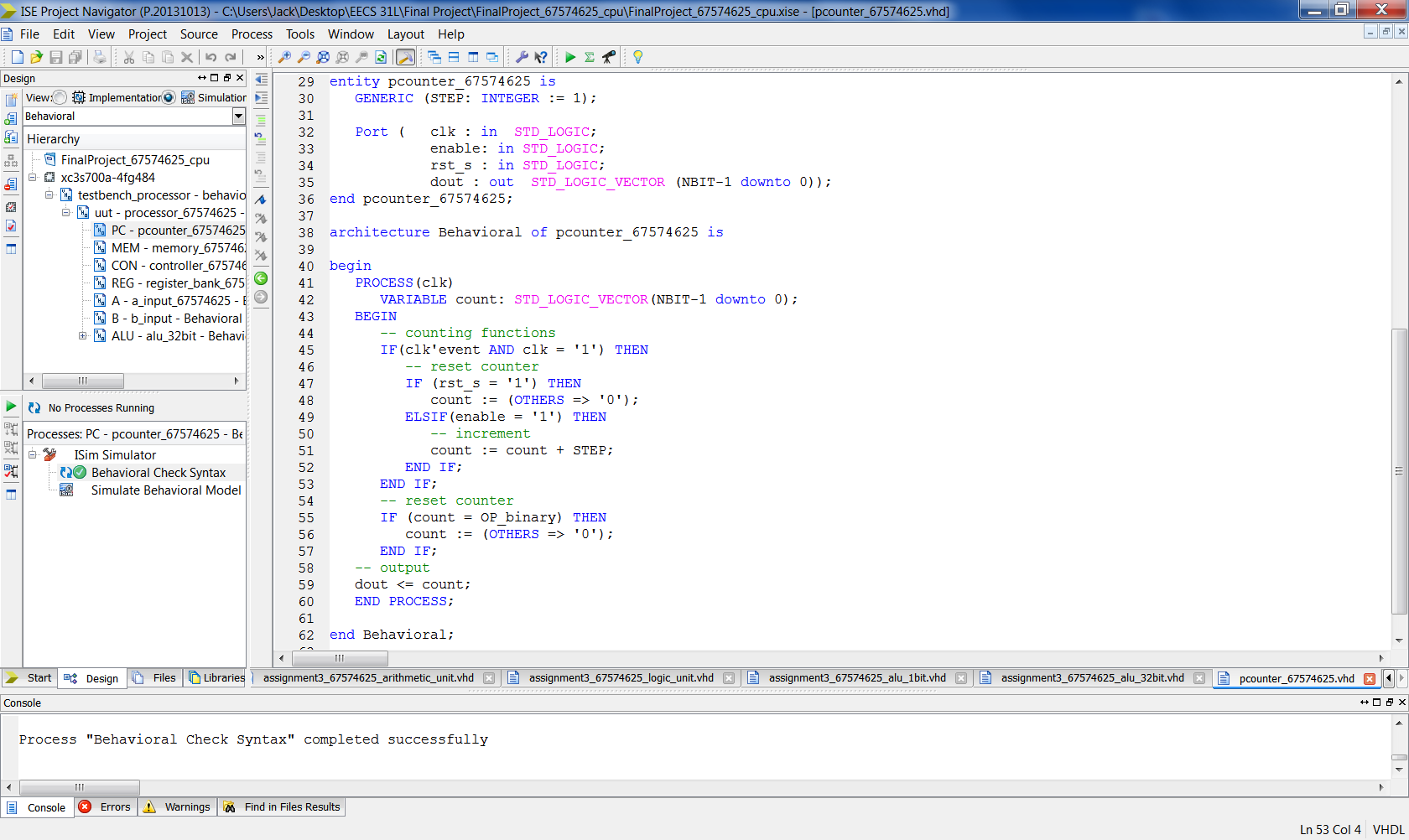
Processor



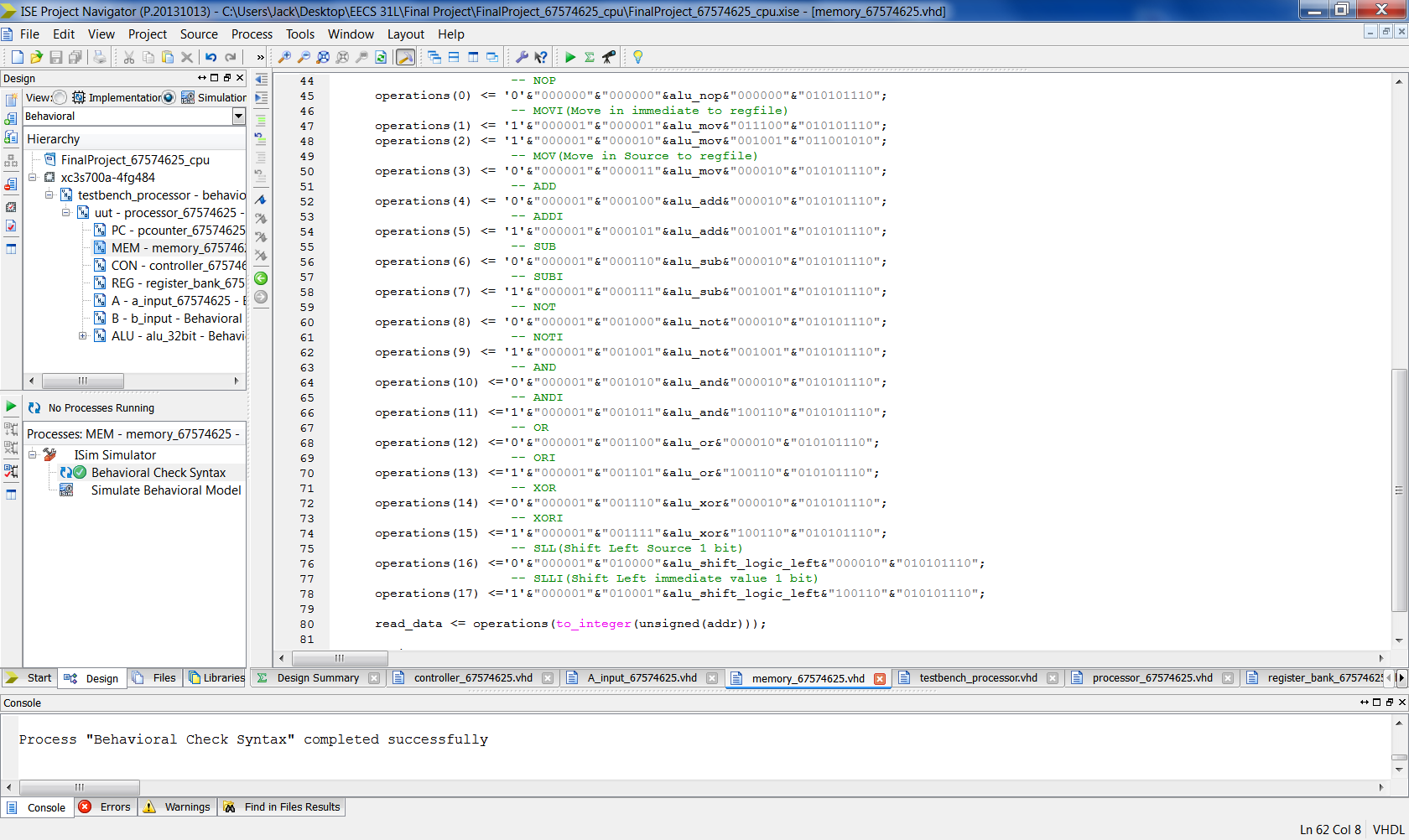
C31L\_pack



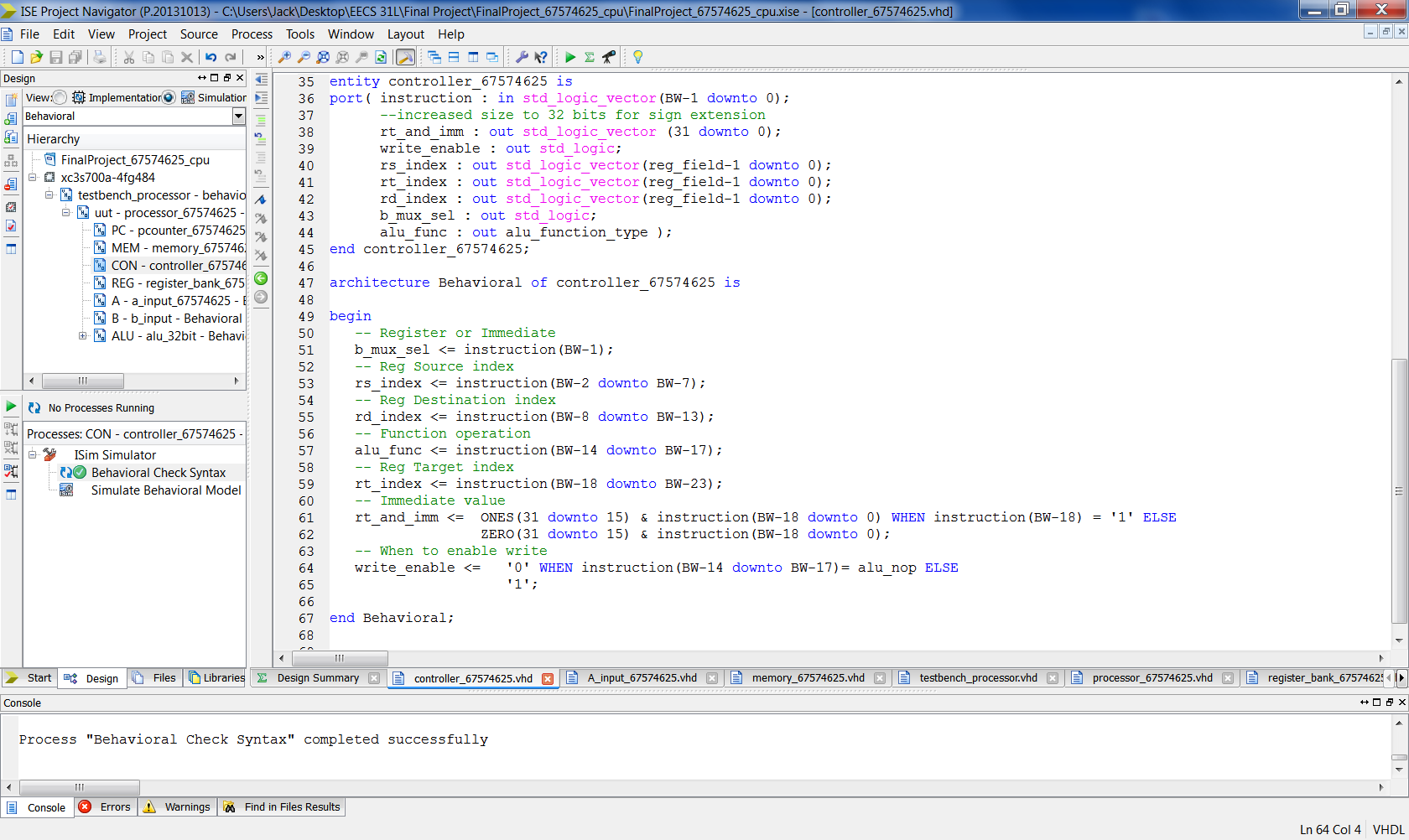
Counter



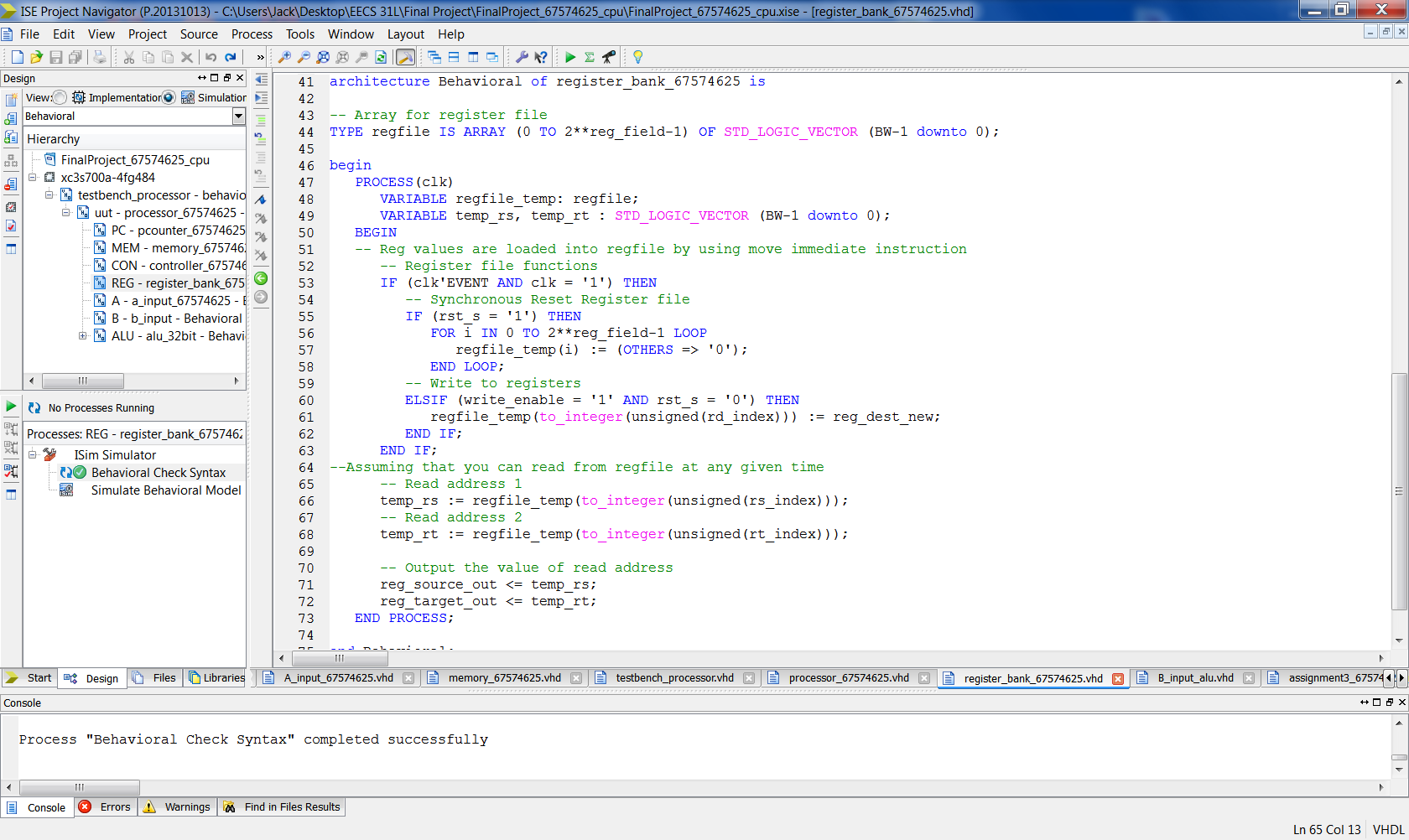
Memory



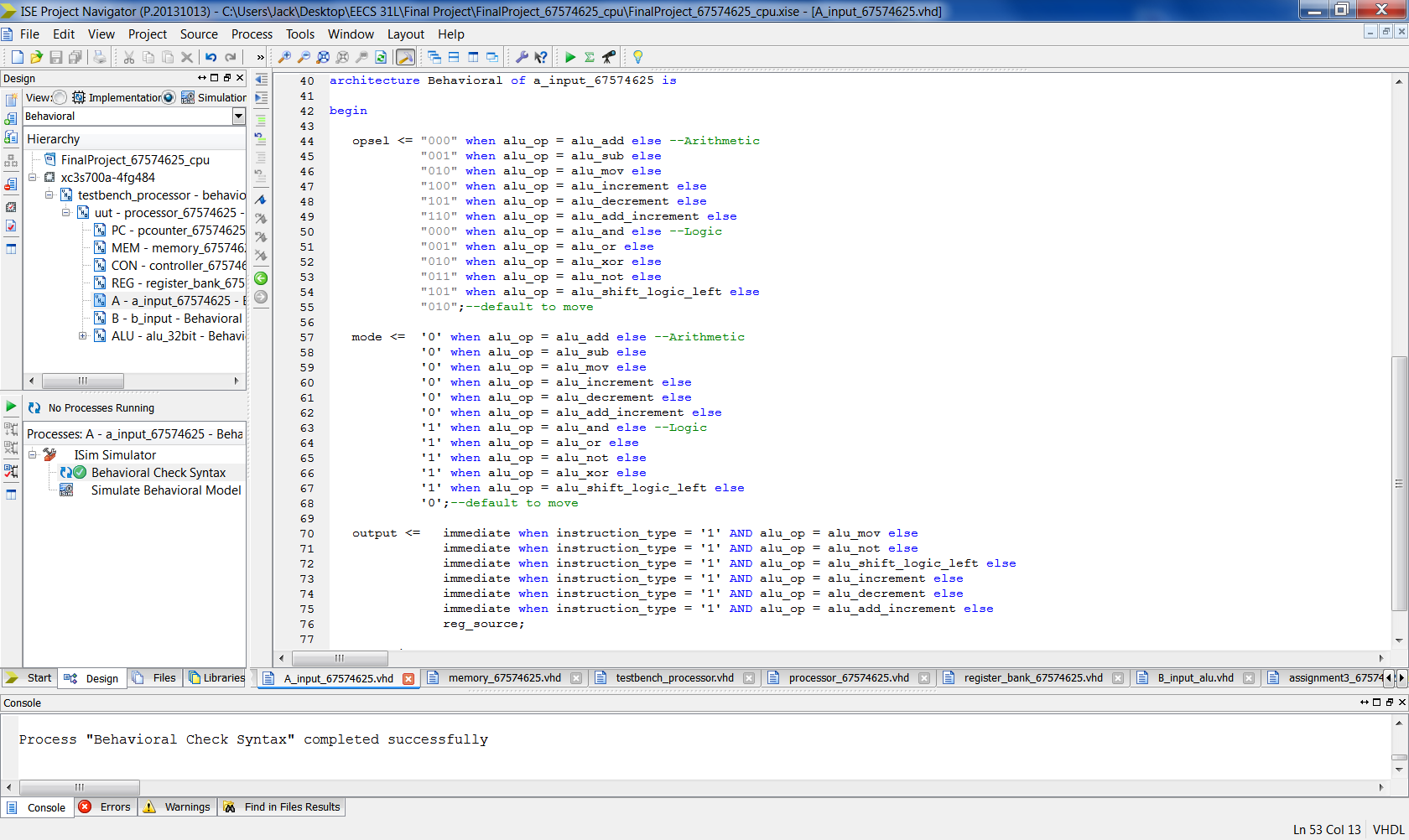
Controller



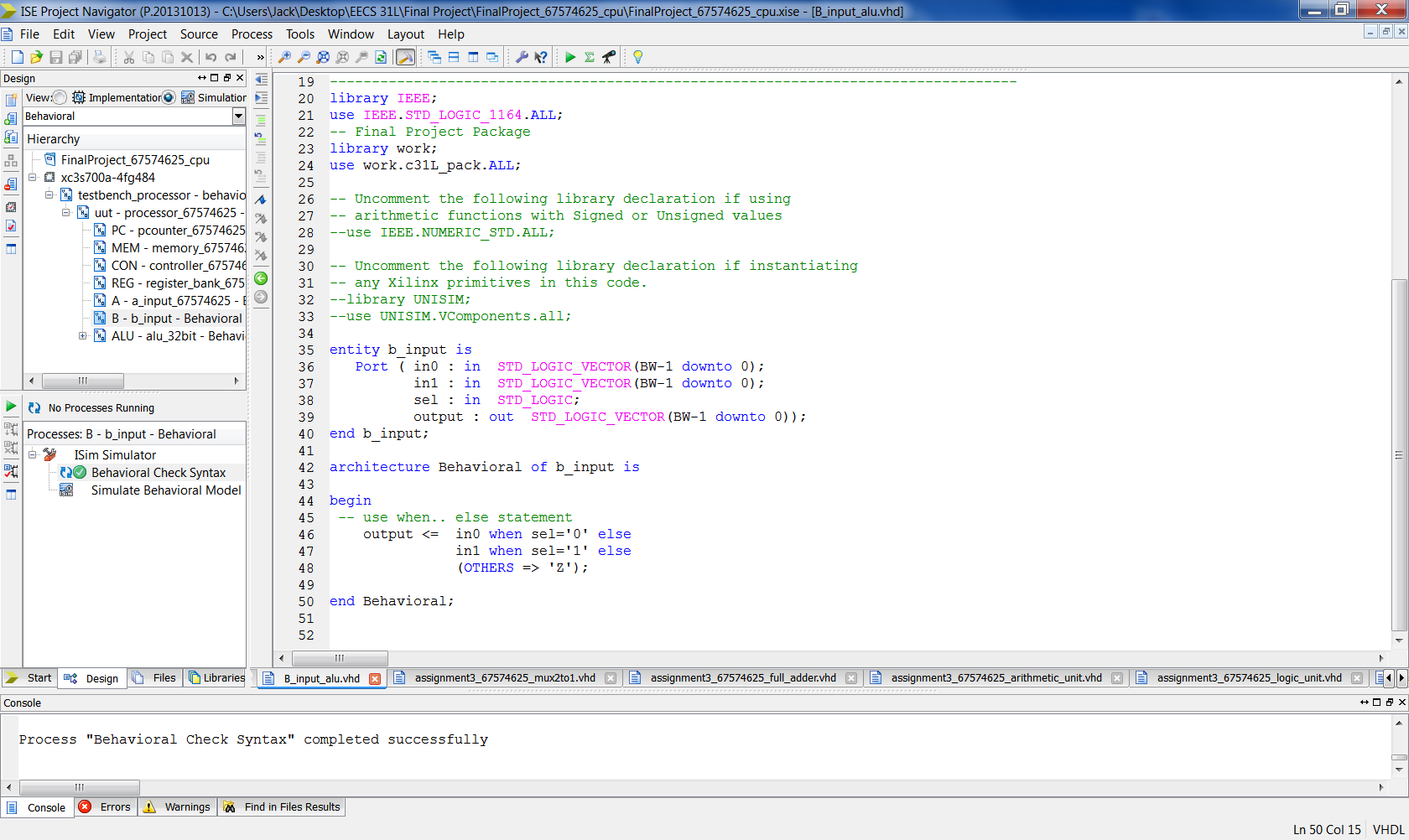
Register File



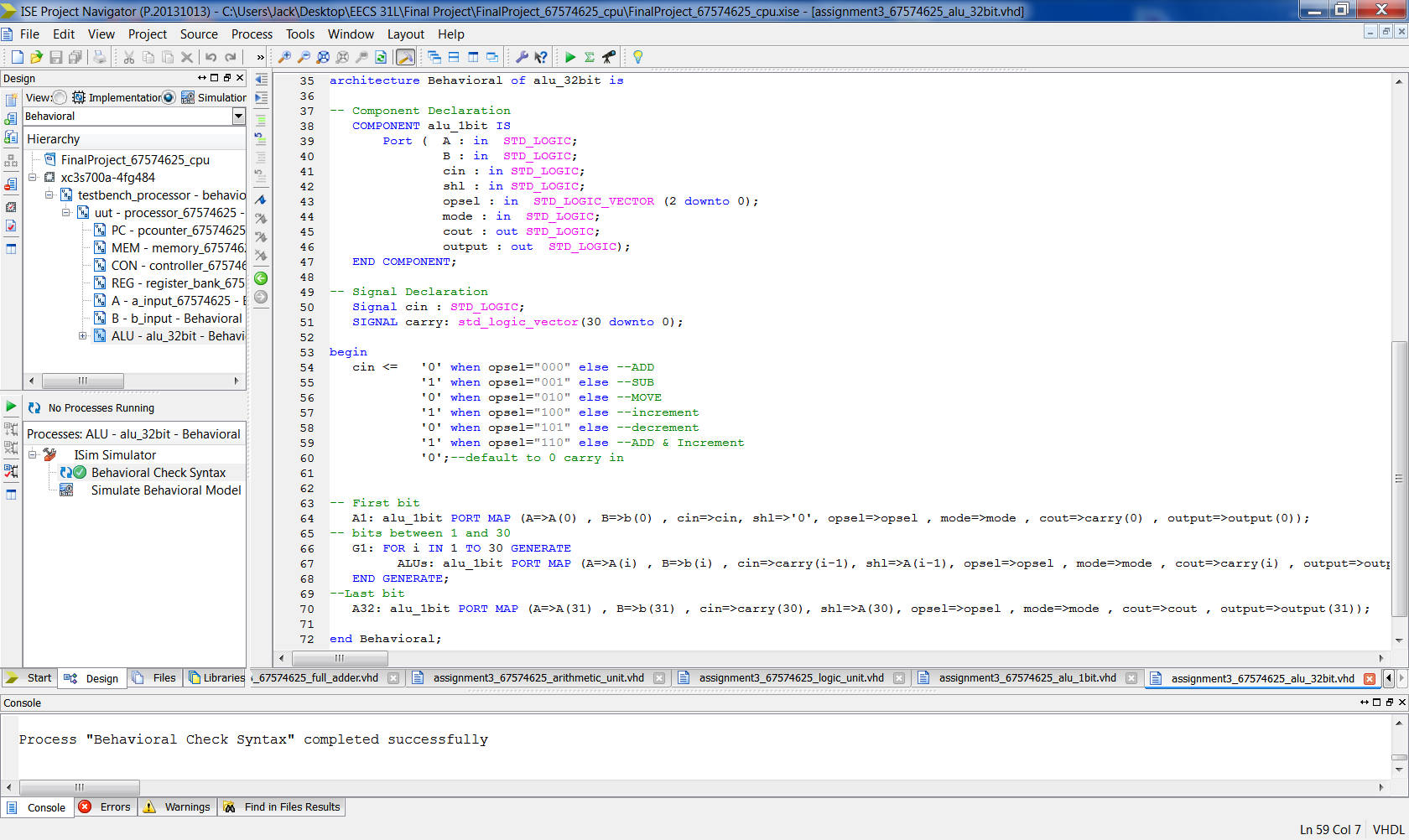
Selector for A input



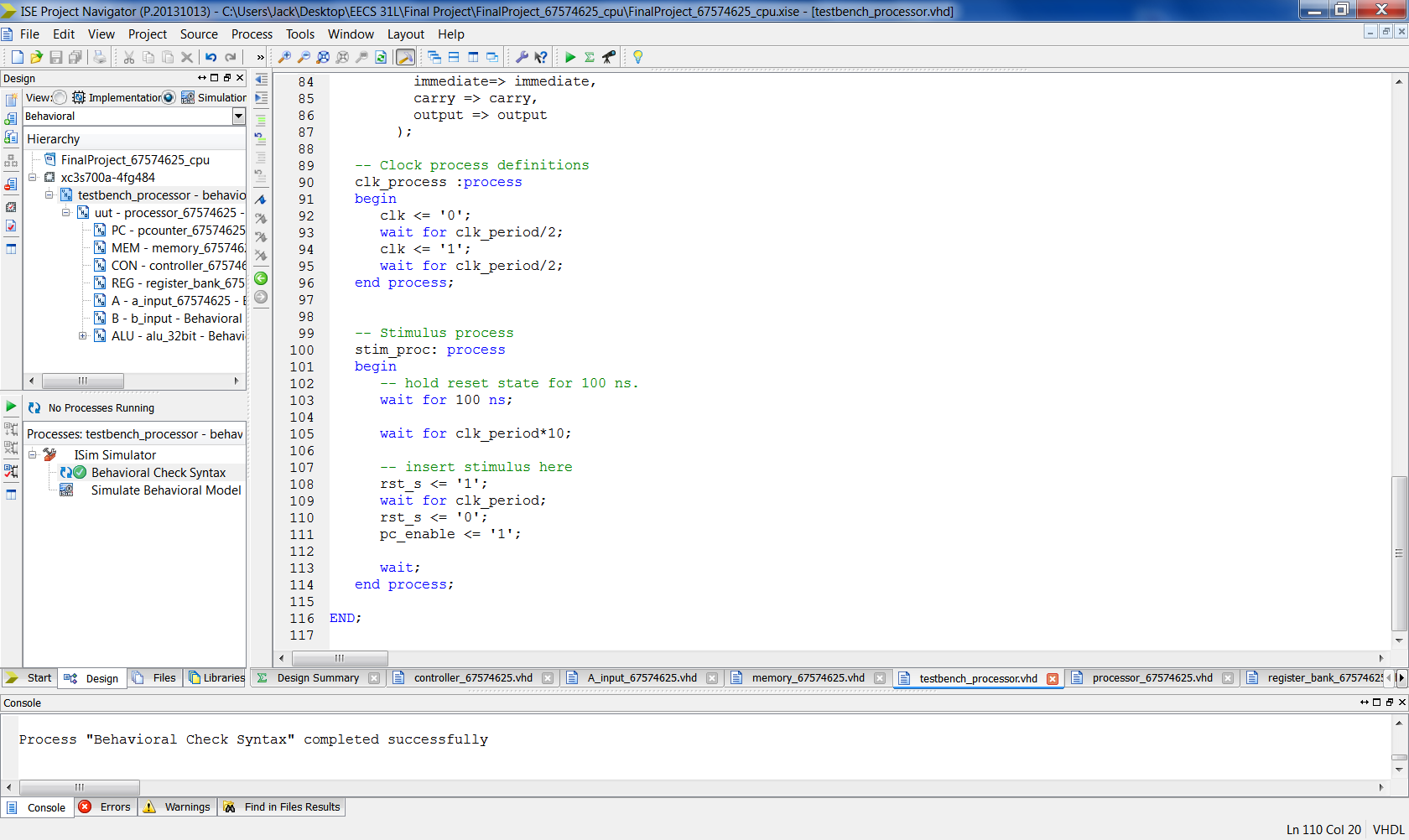
Selector for B input



ALU 32



Testbench



**5 Elaboration**

**Assumptions:**

Functions

* Shifting left the register source by an immediate number of times was altered to just be shift left immediate value one bit
* Because the ALU could do increment, decrement, and add & increment, these functions were given their own code and can be used

Counter

* Need only to increment and then loop back to the first instruction

Memory

* The instructions had to be hard coded into this component
* Register 1 and register 2 in the Register File are assigned values using the MOVI funciton and then those values are used for the remainder of the

ALU

* This didn’t need any adjustments aside from the change of the cin port to a signal

A input

* Because some functions asked for the Immediate to be moved or shifted, another component was designed to handle this.
* In order to translate the function code to the ALU opsel and mode, this component also had to handle it.

**Errors and Challenges:**

* Lot of errors encountered when trying to make sure all the components are wired together in the correct sequence.
* A lot of assumptions had to be throughout the course of the final project
* It took a lot of time to design the schematic for the processor because of these assumptions

**Simulation Log:**

Started : "Simulate Behavioral Model".

Determining files marked for global include in the design...

Running fuse...

Command Line: fuse -intstyle ise -incremental -o {C:/Users/Jack/Desktop/EECS 31L/Final Project/FinalProject\_67574625\_cpu/testbench\_processor\_isim\_beh.exe} -prj {C:/Users/Jack/Desktop/EECS 31L/Final Project/FinalProject\_67574625\_cpu/testbench\_processor\_beh.prj} work.testbench\_processor {}

Running: C:\Xilinx\14.7\ISE\_DS\ISE\bin\nt64\unwrapped\fuse.exe -intstyle ise -incremental -o C:/Users/Jack/Desktop/EECS 31L/Final Project/FinalProject\_67574625\_cpu/testbench\_processor\_isim\_beh.exe -prj C:/Users/Jack/Desktop/EECS 31L/Final Project/FinalProject\_67574625\_cpu/testbench\_processor\_beh.prj work.testbench\_processor

ISim P.20131013 (signature 0x7708f090)

Number of CPUs detected in this system: 4

Turning on mult-threading, number of parallel sub-compilation jobs: 8

Determining compilation order of HDL files

Parsing VHDL file "C:/Users/Jack/Desktop/EECS 31L/Final Project/FinalProject\_67574625\_cpu/assignment3\_67574625\_mux2to1.vhd" into library work

Parsing VHDL file "C:/Users/Jack/Desktop/EECS 31L/Final Project/FinalProject\_67574625\_cpu/assignment3\_67574625\_logic\_unit.vhd" into library work

Parsing VHDL file "C:/Users/Jack/Desktop/EECS 31L/Final Project/FinalProject\_67574625\_cpu/assignment3\_67574625\_full\_adder.vhd" into library work

Parsing VHDL file "C:/Users/Jack/Desktop/EECS 31L/Final Project/FinalProject\_67574625\_cpu/assignment3\_67574625\_arithmetic\_unit.vhd" into library work

Parsing VHDL file "C:/Users/Jack/Desktop/EECS 31L/Final Project/FinalProject\_67574625\_cpu/pooriam\_c31L\_pack.vhd" into library work

Parsing VHDL file "C:/Users/Jack/Desktop/EECS 31L/Final Project/FinalProject\_67574625\_cpu/assignment3\_67574625\_alu\_1bit.vhd" into library work

Parsing VHDL file "C:/Users/Jack/Desktop/EECS 31L/Final Project/FinalProject\_67574625\_cpu/register\_bank\_67574625.vhd" into library work

Parsing VHDL file "C:/Users/Jack/Desktop/EECS 31L/Final Project/FinalProject\_67574625\_cpu/pcounter\_67574625.vhd" into library work

Parsing VHDL file "C:/Users/Jack/Desktop/EECS 31L/Final Project/FinalProject\_67574625\_cpu/memory\_67574625.vhd" into library work

Parsing VHDL file "C:/Users/Jack/Desktop/EECS 31L/Final Project/FinalProject\_67574625\_cpu/controller\_67574625.vhd" into library work

Parsing VHDL file "C:/Users/Jack/Desktop/EECS 31L/Final Project/FinalProject\_67574625\_cpu/B\_input\_alu.vhd" into library work

Parsing VHDL file "C:/Users/Jack/Desktop/EECS 31L/Final Project/FinalProject\_67574625\_cpu/A\_input\_67574625.vhd" into library work

Parsing VHDL file "C:/Users/Jack/Desktop/EECS 31L/Final Project/FinalProject\_67574625\_cpu/assignment3\_67574625\_alu\_32bit.vhd" into library work

Parsing VHDL file "C:/Users/Jack/Desktop/EECS 31L/Final Project/FinalProject\_67574625\_cpu/processor\_67574625.vhd" into library work

Parsing VHDL file "C:/Users/Jack/Desktop/EECS 31L/Final Project/FinalProject\_67574625\_cpu/testbench\_processor.vhd" into library work

Starting static elaboration

Completed static elaboration

Compiling package standard

Compiling package std\_logic\_1164

Compiling package c31l\_pack

Compiling package std\_logic\_arith

Compiling package std\_logic\_unsigned

Compiling package numeric\_std

Compiling architecture behavioral of entity pcounter\_67574625 [\pcounter\_67574625(1)\]

Compiling architecture behavioral of entity memory\_67574625 [memory\_67574625\_default]

Compiling architecture behavioral of entity controller\_67574625 [controller\_67574625\_default]

Compiling architecture behavioral of entity register\_bank\_67574625 [register\_bank\_67574625\_default]

Compiling architecture behavioral of entity a\_input\_67574625 [a\_input\_67574625\_default]

Compiling architecture behavioral of entity b\_input [b\_input\_default]

Compiling architecture behavioral of entity logic\_unit [logic\_unit\_default]

Compiling architecture behavioral of entity arithmetic\_unit [arithmetic\_unit\_default]

Compiling architecture behavioral of entity full\_adder [full\_adder\_default]

Compiling architecture behavioral of entity mux2to1 [mux2to1\_default]

Compiling architecture behavioral of entity alu\_1bit [alu\_1bit\_default]

Compiling architecture behavioral of entity alu\_32bit [alu\_32bit\_default]

Compiling architecture behavioral of entity processor\_67574625 [processor\_67574625\_default]

Compiling architecture behavior of entity testbench\_processor

Time Resolution for simulation is 1ps.

Waiting for 11 sub-compilation(s) to finish...

Compiled 33 VHDL Units

Built simulation executable C:/Users/Jack/Desktop/EECS 31L/Final Project/FinalProject\_67574625\_cpu/testbench\_processor\_isim\_beh.exe

Fuse Memory Usage: 37900 KB

Fuse CPU Usage: 1200 ms

Launching ISim simulation engine GUI...

"C:/Users/Jack/Desktop/EECS 31L/Final Project/FinalProject\_67574625\_cpu/testbench\_processor\_isim\_beh.exe" -intstyle ise -gui -tclbatch isim.cmd -wdb "C:/Users/Jack/Desktop/EECS 31L/Final Project/FinalProject\_67574625\_cpu/testbench\_processor\_isim\_beh.wdb"

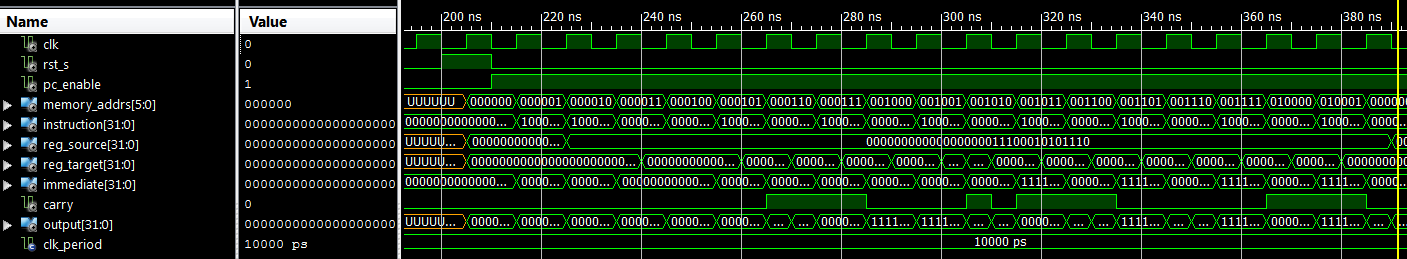
ISim simulation engine GUI launched successfully

Process "Simulate Behavioral Model" completed successfully

**6 Waveforms**

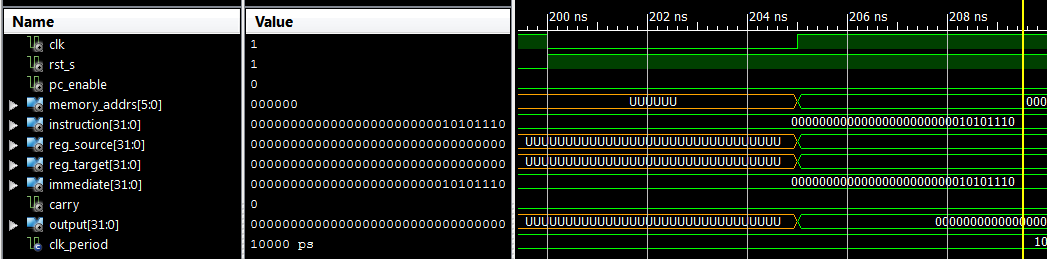
Note: At the rising edge of clock, the output shown is saved and the memory\_addrs changes

Full Waveform Snapshot 1

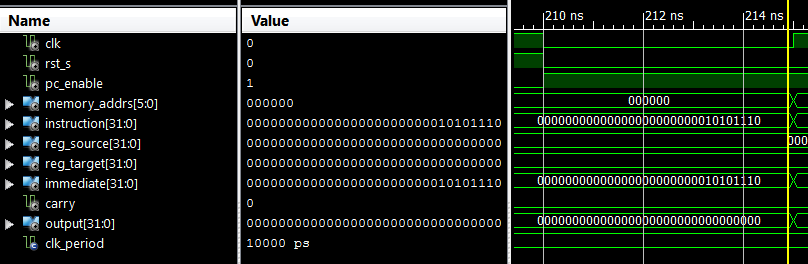


Snapshots of Waveform

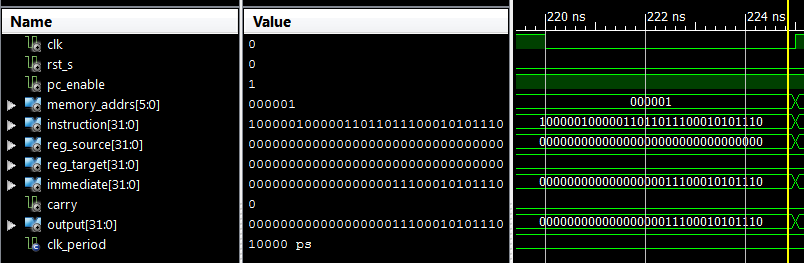
RESET (Not a function)



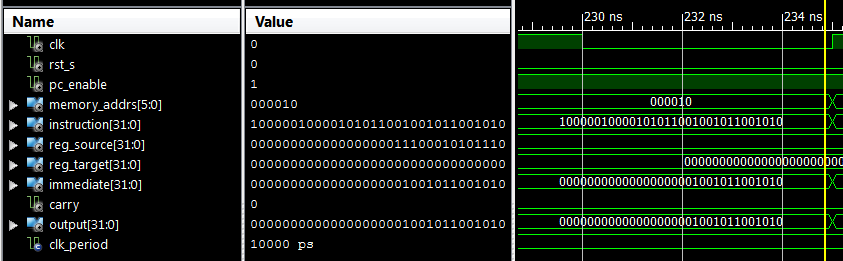
NOP



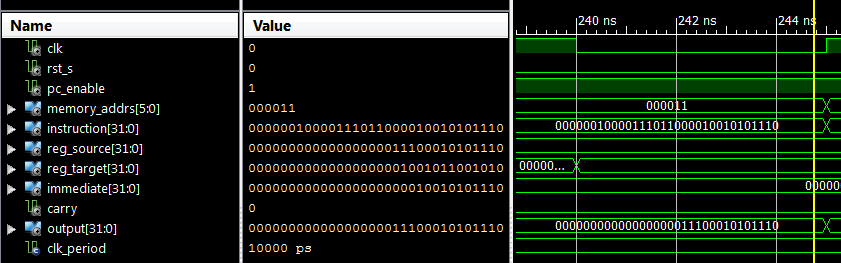
MOVI (to give value to a register)



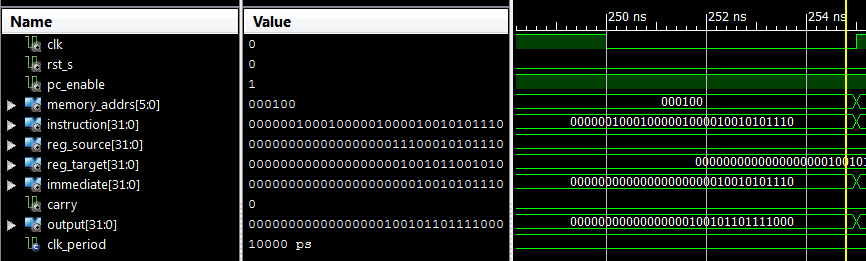
MOVI (to give value to a second register)



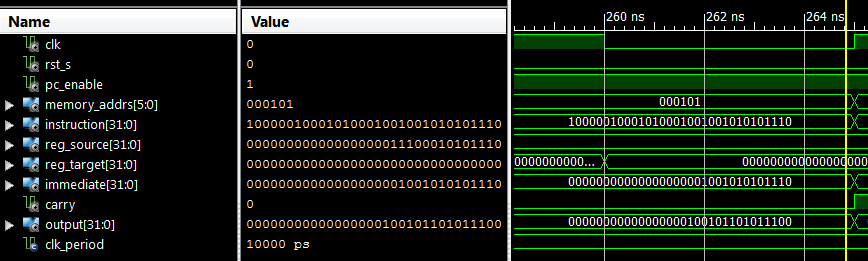
MOV



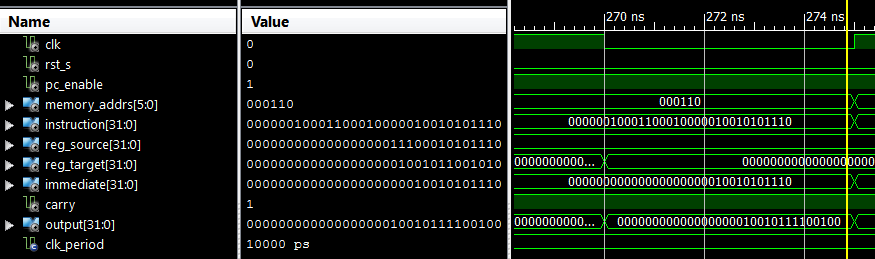
ADD



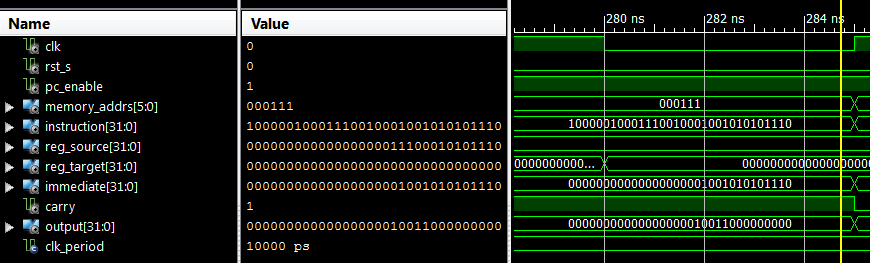
ADDI



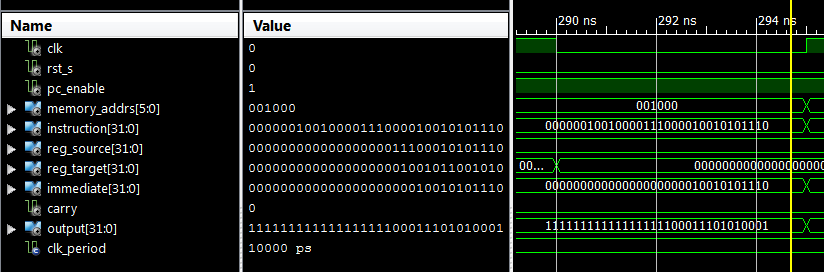
SUB



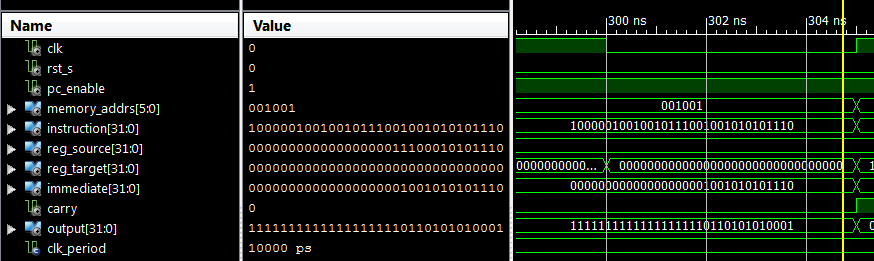
SUBI



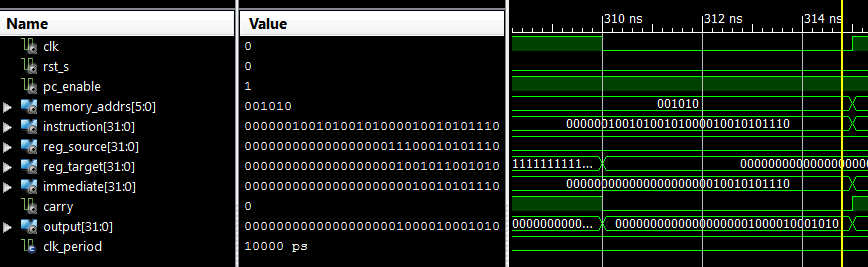
NOT



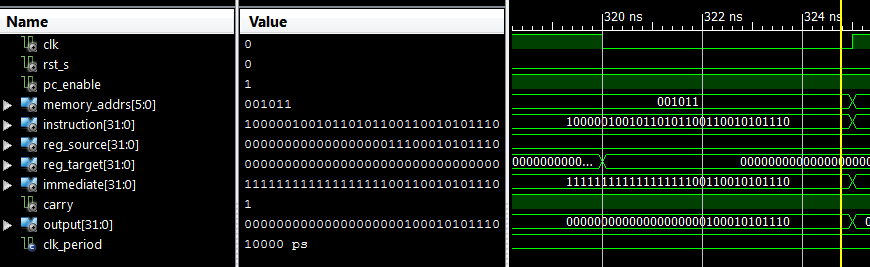
NOTI



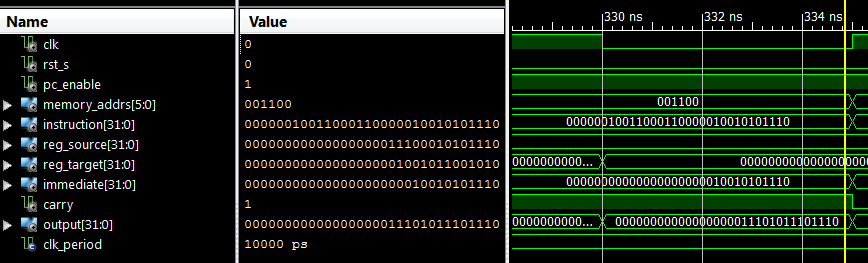
AND



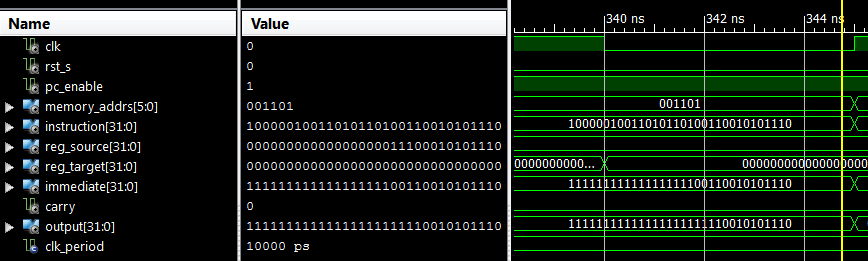
ANDI



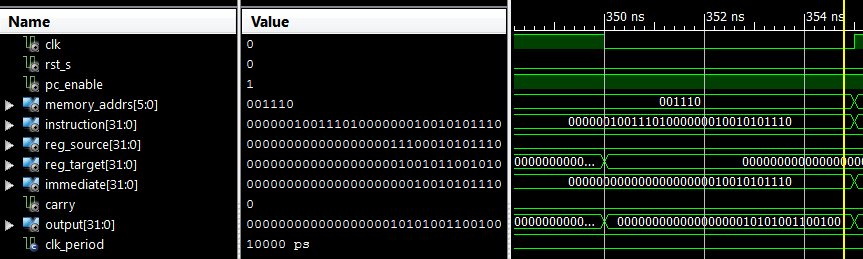
OR



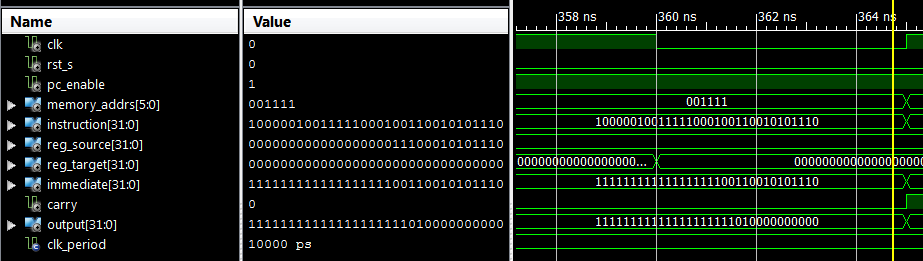
ORI



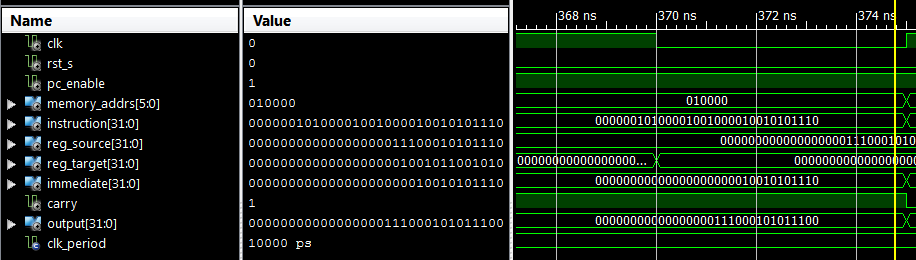
XOR



XORI



SLL



SLLI

