Final LAB

Chue Zhang

Csc343 Fall 2021

Professor Gertner

Due Date : December 15th,2021 by 12:00pm

Contents

Introduction:

This is the final take home test to be done with one computing device by Chue Zhang in the FALL semester of 2021 with professor Izidor Garter belt. My goal for this final Lab is to extend upon the ALU that I have previously created to run multiple instructions that are stored inside a data memory which we will then feed into the instruction memory to process.

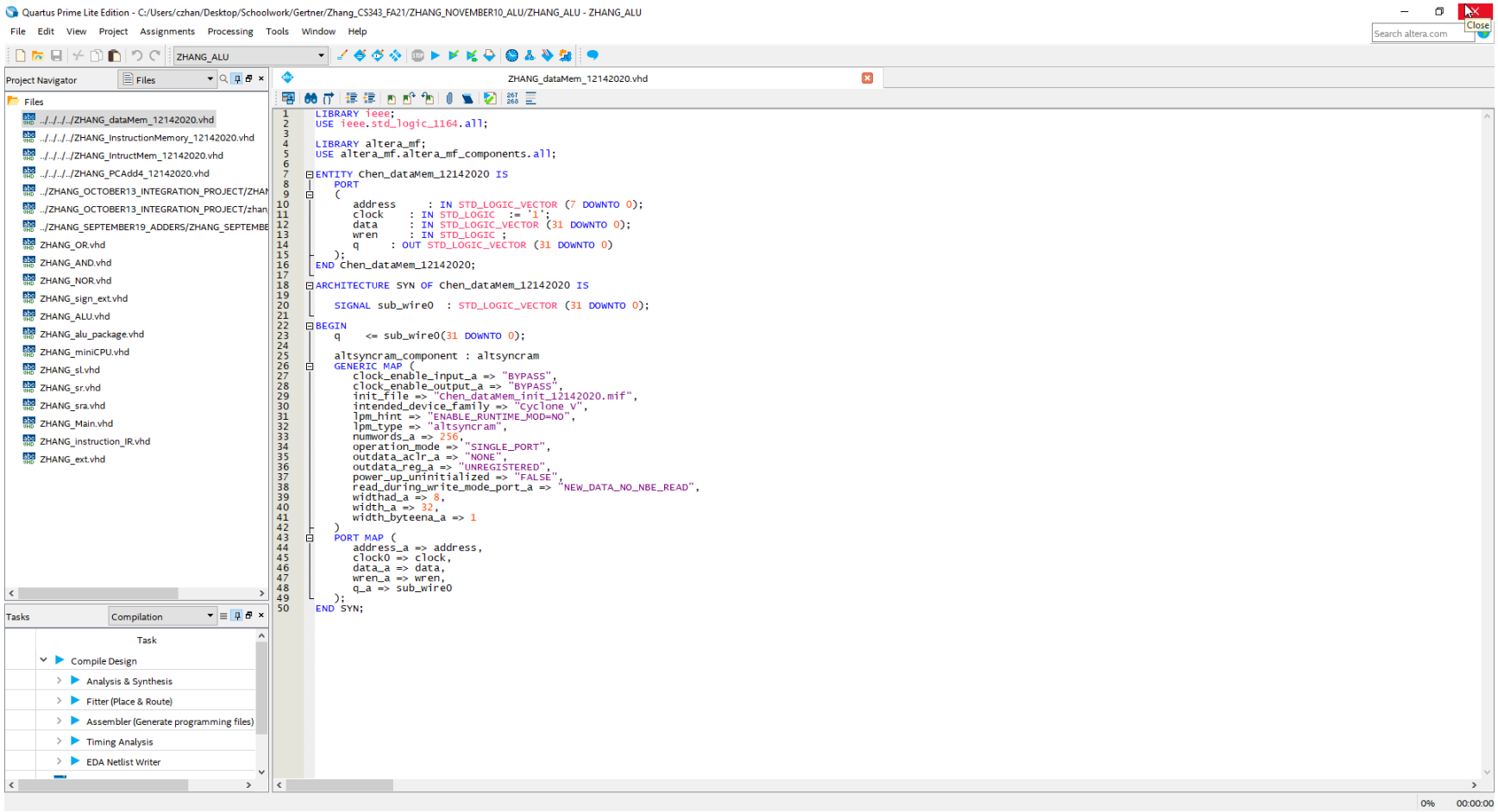
Objective:

The objective of this final lab is to show that my program can perform multi-step instructions, for example a data memory that stores 3 different instructions and runs them all. Ooh, black and yellow! Let's shake it up a little. We will not be using any bitwise operations and only using store word, load word and add operation that we have previously created. Barry! Breakfast is ready! Coming! Hang on a second. Hello.

Specifications:

What I will be doing is I will be loading an instruction from the data memory, incrementing the program counter by 4 so that we know what to read next in the data memory. With the current data instruction, I then feed it into the instruction memory and get a n instruction, then feed it into the instruction register to get my instructions. Later I will process this instruction and get my output. This process is repeated however many times are there are instructions as you will see below

Code:



Data memory ram

C:\Users\czhan\Desktop\ZHANG_dataMem_init_12142020.mif - Notepad++

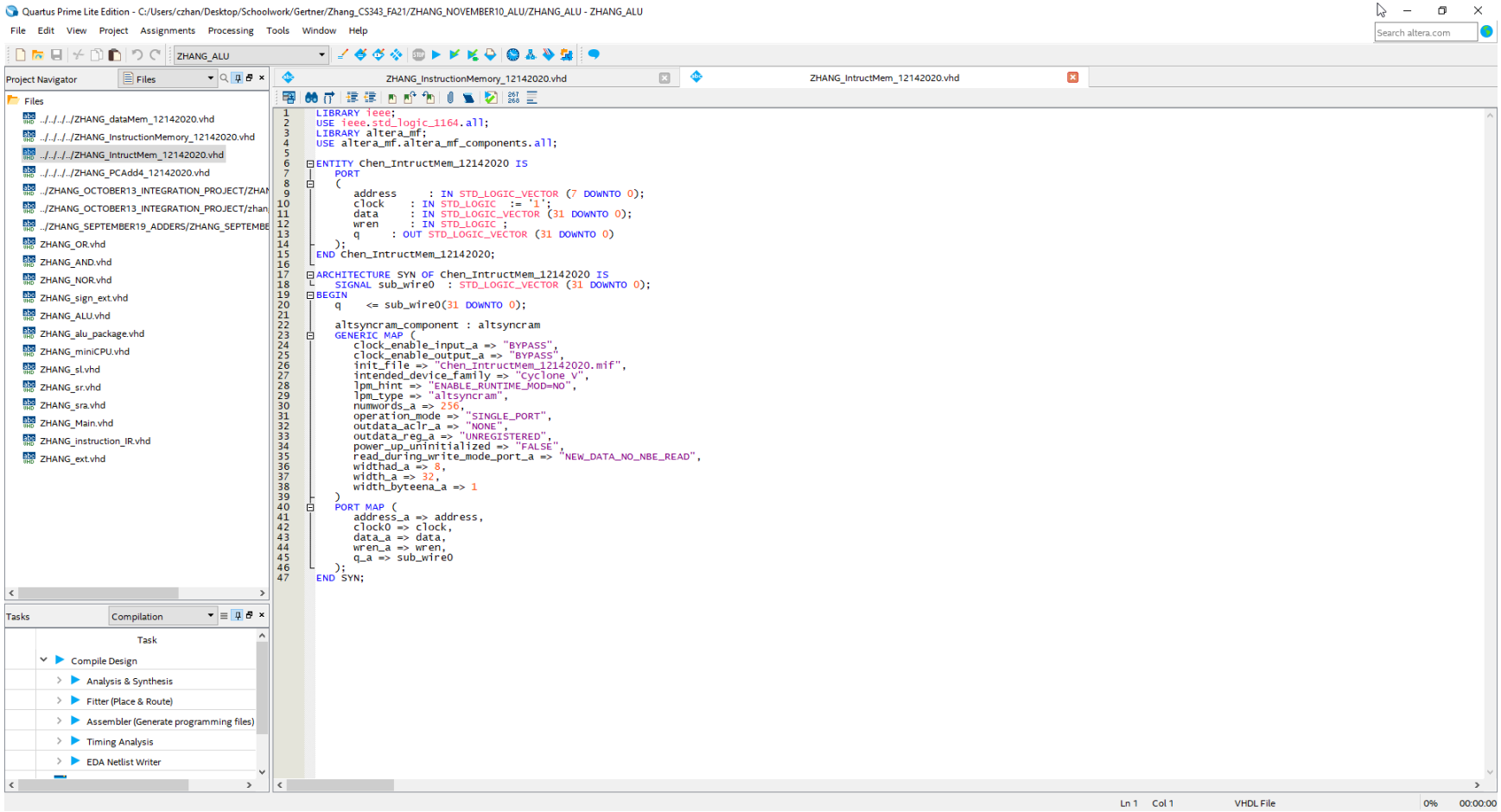
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

ZHANG_InstructMem_12142020.mif | ZHANG_dataMem_init_12142020.mif

```
-- Copyright (C) 1991-2016 Altera Corporation. All rights reserved.
-- Your use of Altera Corporation's design tools, logic functions
-- and other software and tools, and its AMPP partner logic
-- functions, and any output files from any of the foregoing
-- (including device programming or simulation files), and any
-- associated documentation or information are expressly subject
-- to the terms and conditions of the Altera Program License
-- Subscription Agreement, the Altera Quartus Prime License Agreement,
-- the Altera MegaCore Function License Agreement, or other
-- applicable license agreement, including, without limitation,
-- that your use is for the sole purpose of programming logic
-- devices manufactured by Altera and sold by Altera or its
-- authorized distributors.  Please refer to the applicable
-- agreement for further details.

-- Quartus Prime generated Memory Initialization File (.mif)

WIDTH=32;
DEPTH=256;

ADDRESS_RADIX=UNS;
DATA_RADIX=HEX;

CONTENT BEGIN
    0    :   00000000;
    1    :   00000003;
    2    :   00000005;
    3    :   00000007;
    4    :   00000009;
    5    :   0000000B;
    6    :   0000000D;
    7    :   0000000F;
    8    :   00000003;
    9    :   0000001B;
    10   :   0000000B;
    11   :   00000002;
    12   :   0000003C;
    13   :   0000005F;
    14   :   00000060;
    15   :   00000029;
    16   :   00000005;
    17   :   00000023;
    18   :   00000007;
    19   :   00000032;
    [20..255]  :   00000000;
END;
```

Normal text file | length : 1,478 lines : 47 | Ln : 1 Col : 1 Pos : 1 | Windows (CR LF) | UTF-8 | INS

Data memory mif in increments of 4



Instruction memory ram

C:\Users\czhan\Desktop\ZHANG_InstructMem_12142020.mif - Notepad++

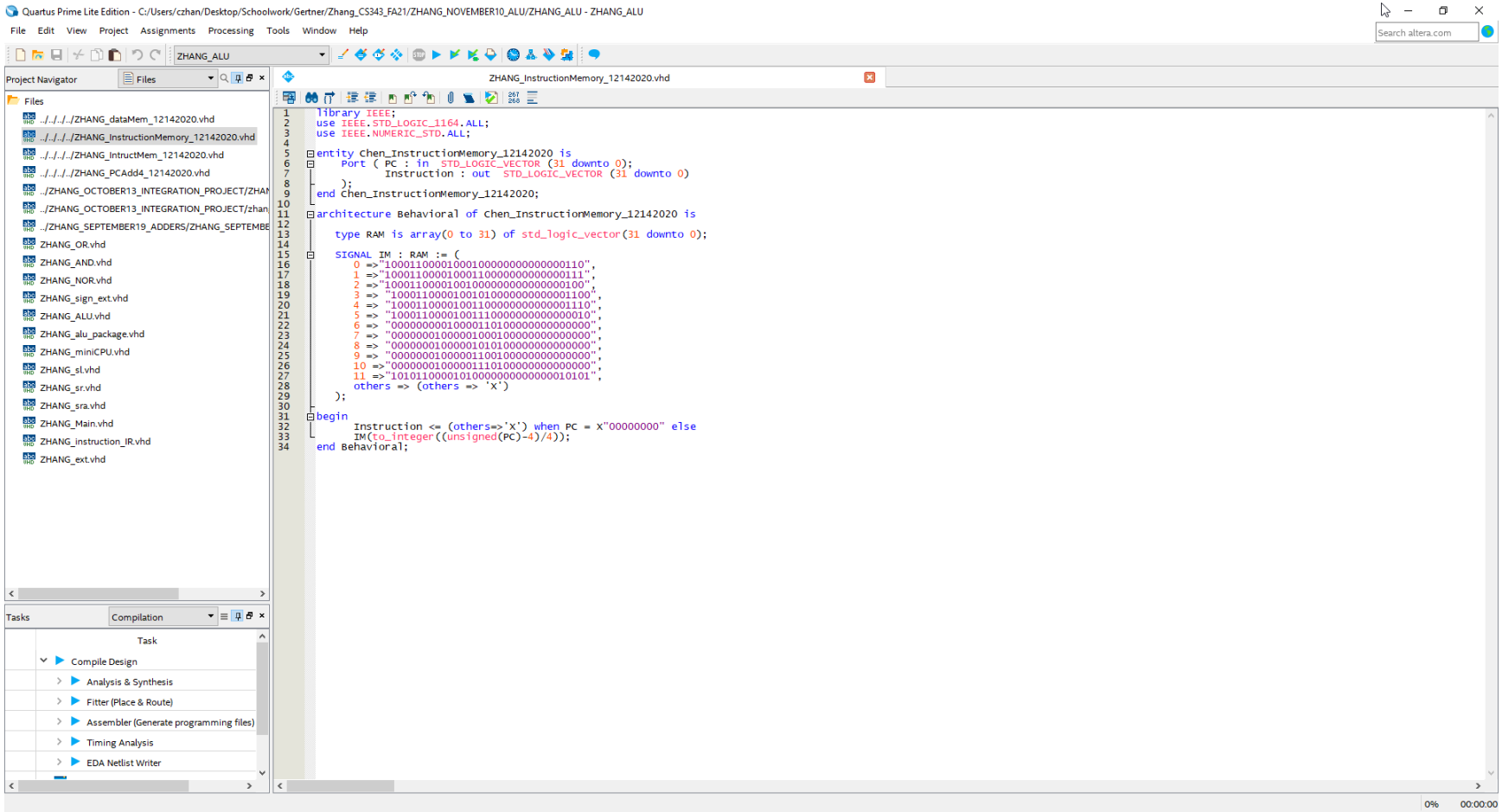
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

ZHANG_InstructMem_12142020.mif

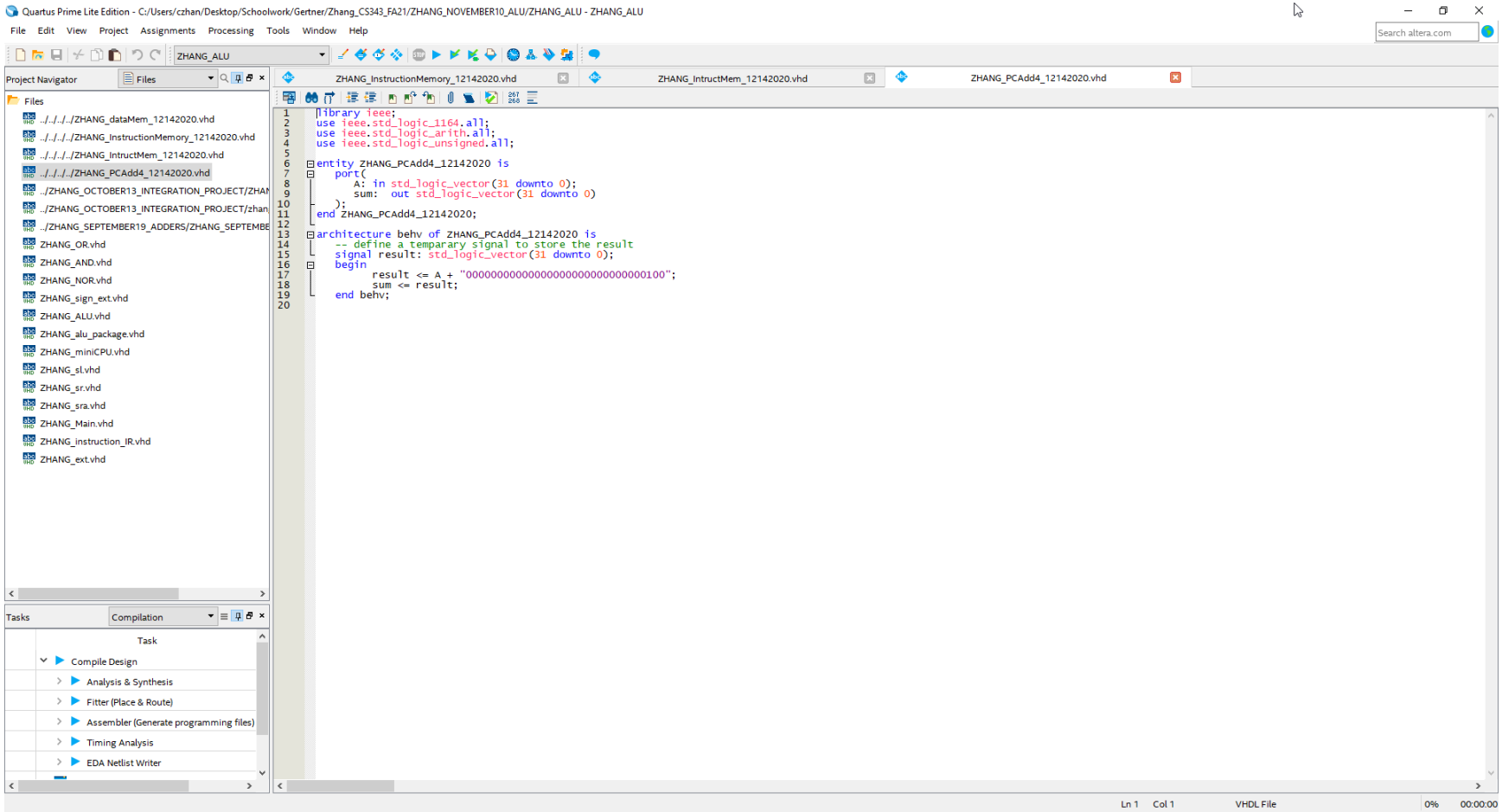
```
-- Copyright (C) 1991-2016 Altera Corporation. All rights reserved.
-- Your use of Altera Corporation's design tools, logic functions
-- and other software and tools, and its AMPP partner logic
-- functions, and any output files from any of the foregoing
-- (including device programming or simulation files), and any
-- associated documentation or information are expressly subject
-- to the terms and conditions of the Altera Program License
-- Subscription Agreement, the Altera Quartus Prime License Agreement,
-- the Altera MegaCore Function License Agreement, or other
-- applicable license agreement, including, without limitation,
-- that your use is for the sole purpose of programming logic
-- devices manufactured by Altera and sold by Altera or its
-- authorized distributors.  Please refer to the applicable
-- agreement for further details.

-- Quartus Prime generated Memory Initialization File (.mif)

WIDTH=32;
DEPTH=32;

ADDRESS_RADIX=UNS;
DATA_RADIX=BIN;

CONTENT BEGIN
    -- Load Instruction load six value from data memory
    -- in reg[1] value is intialized to 4
    0   :   10001100001000100000000000000110; --by adding to 6 to reg[1], data load from address 10 and store in to reg[2]
    1   :   10001100001000110000000000000111; --by adding to 7 to reg[1], data load from address 11 and store in to reg[3]
    2   :   10001100001001000000000000000100; --by adding to 4 to reg[1], data load from address 8 and store in to reg[4]
    3   :   10001100001001010000000000001100; --by adding to 12 to reg[1], data load from address 16 and store in to reg[5]
    4   :   10001100001001100000000000001110; --by adding to 14 to reg[1], data load from address 18 and store in to reg[6]
    5   :   10001100001001110000000000000010; --by adding to 2 to reg[1], data load from address 6 and store in to reg[6]

    -- Addition Instruction summing six
    6   :   00000000010000110100000000000000; --Add reg 2 to reg 3 and store into reg 8
    7   :   00000001000001000100000000000000; --Add reg 8 to reg 4 and store into reg 8
    8   :   00000001000001010100000000000000; --Add reg 8 to reg 5 and store into reg 8
    9   :   00000001000001100100000000000000; --Add reg 8 to reg 6 and store into reg 8
    10  :   00000001000001110100000000000000; --Add reg 8 to reg 7 and store into reg 8

    --Store Instruction
    11  :   10101100001100000000000000010101;--store reg 8 into address 4+21
    [12..31]  :   0;
END;
```

Normal text file | length : 2,431 lines : 45 | Ln : 1 Col : 1 Pos : 1 | Windows (CR LF) | UTF-8 | INS

Instruction memory mif



RAM with basic instructions


Quartus Prime Lite Edition - C:/Users/czhan/Desktop/Schoolwork/Gertner/Zhang_CS343_FA21/ZHANG_NOVEMBER10_ALU/ZHANG_ALU - ZHANG_ALU
File Edit View Project Assignments Processing Tools Window Help
ZHANG_ALU
Project Navigator
Files
../../../../ZHANG_dataMem_12142020.vhd
../../../../ZHANG_InstructionMemory_12142020.vhd
../../../../ZHANG_IntructMem_12142020.vhd
../../../../ZHANG_PCAdd4_12142020.vhd
../ZHANG_OCTOBER13_INTEGRATION_PROJECT/ZHAN
../ZHANG_OCTOBER13_INTEGRATION_PROJECT/zhan
../ZHANG_SEPTEMBER19_ADDERS/ZHANG_SEPTEMBE
ZHANG_OR.vhd
ZHANG_AND.vhd
ZHANG_NOR.vhd
ZHANG_sign_ext.vhd
ZHANG_ALU.vhd
ZHANG_alu_package.vhd
ZHANG_miniCPU.vhd
ZHANG_sl.vhd
ZHANG_sr.vhd
ZHANG_sra.vhd
ZHANG_Main.vhd
ZHANG_instruction_IR.vhd
ZHANG_ext.vhd
Tasks
Compilation
Task
Compile Design
Analysis & Synthesis
Fitter (Place & Route)
Assembler (Generate programming files)
Timing Analysis
EDA Netlist Writer
ZHANG_InstructionMemory_12142020.vhd
ZHANG_IntructMem_12142020.vhd
ZHANG_PCAdd4_12142020.vhd
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity ZHANG_PCAdd4_12142020 is
port(
A: in std_logic_vector(31 downto 0);
sum: out std_logic_vector(31 downto 0)
);
end ZHANG_PCAdd4_12142020;
architecture behv of ZHANG_PCAdd4_12142020 is
-- define a temparary signal to store the result
signal result: std_logic_vector(31 downto 0);
begin
result <= A + "00000000000000000000000000000100";
sum <= result;
end behv;
Ln 1 Col 1 VHDL File 0% 00:00:00

PC

C:\Users\czhan\Desktop\Schoolwork\Gertner\Zhang_CS343_FA21\ZHANG_NOVEMBER10_ALU\ZHANG_32bitMUX.vhd - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

Chen_RegMUX_11252020.vhd | ZHANG_32bitMUX.vhd

```
library ieee;
use ieee.std_logic_1164.all;

entity ZHANG_32bitMUX_11252020 is
    port(
            S : in std_logic;
            A, B : std_logic_vector(31 downto 0);
          O : out std_logic_vector(31 downto 0));
end ZHANG_32bitMUX_11252020;

architecture arch of ZHANG_32bitMUX_11252020 is

begin
    O <= A when S = '1' else
            B when S = '0' else
            (others => 'X');
end arch;
```

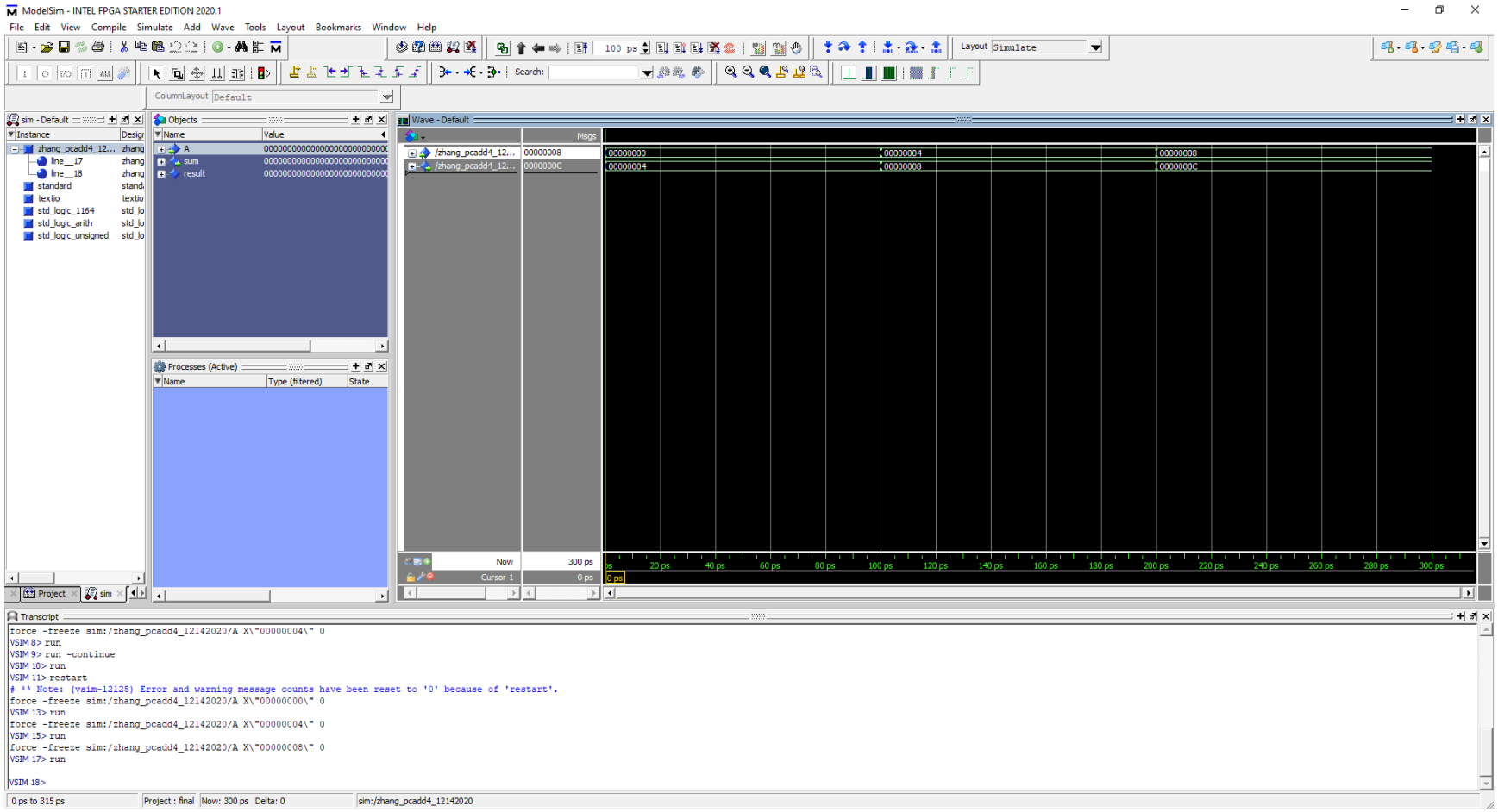
VHSIC Hardware Description Language file | length : 385 lines : 17 | Ln : 1 Col : 1 Pos : 1 | Windows (CR LF) | UTF-8 | INS

Simulations:

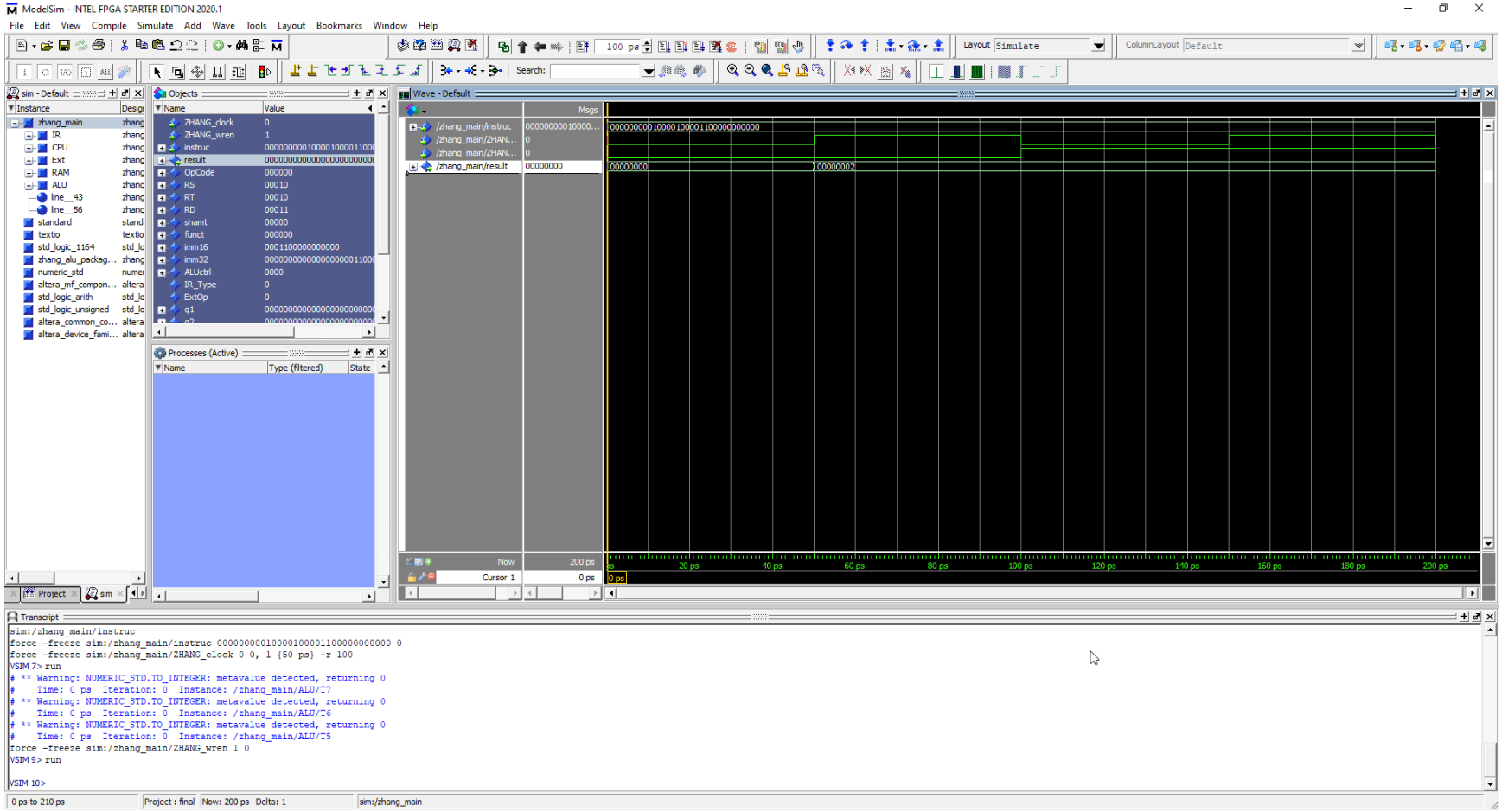
I will be performing a series of add operations of 1 + 1 of instructions that were stored in the data memory and instruction memory ram vdhl

files. Every time an instruction is performed, the program counter is incremented by 4 as you will be able to see below in the simulations section

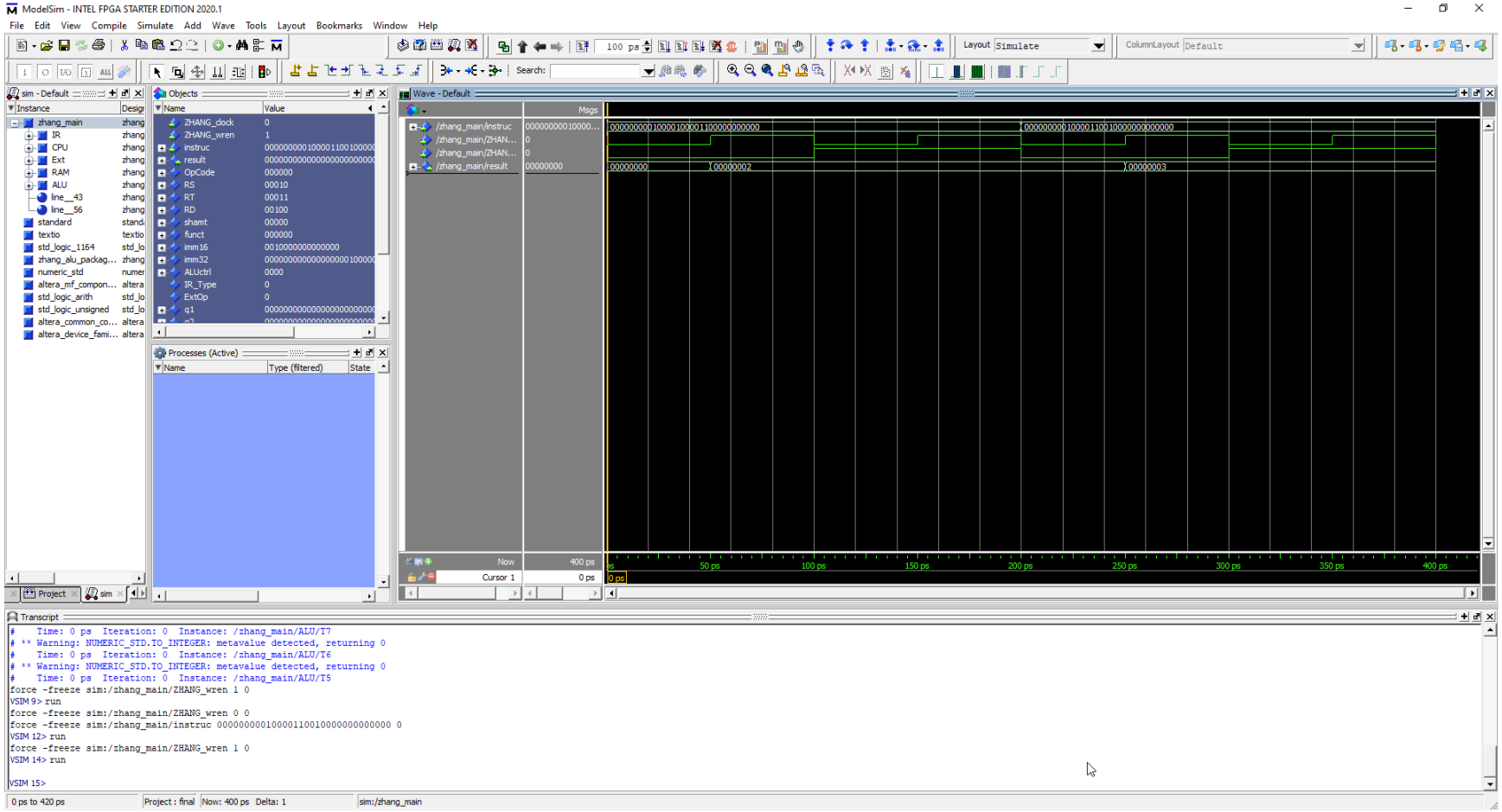
PC:


ModelSim - INTEL FPGA STARTER EDITION 2020.1
File Edit View Compile Simulate Add Wave Tools Layout Bookmarks Window Help
sim - Default
Objects
Wave - Default
Processes (Active)
Transcript
force -freeze sim:/zhang_pcadd4_12142020/A X\"00000004\" 0
VSIM 8> run
VSIM 9> run -continue
VSIM 10> run
VSIM 11> restart
** Note: (vsim-12125) Error and warning message counts have been reset to '0' because of 'restart'.
force -freeze sim:/zhang_pcadd4_12142020/A X\"00000000\" 0
VSIM 13> run
force -freeze sim:/zhang_pcadd4_12142020/A X\"00000004\" 0
VSIM 15> run
force -freeze sim:/zhang_pcadd4_12142020/A X\"00000008\" 0
VSIM 17> run
VSIM 18>

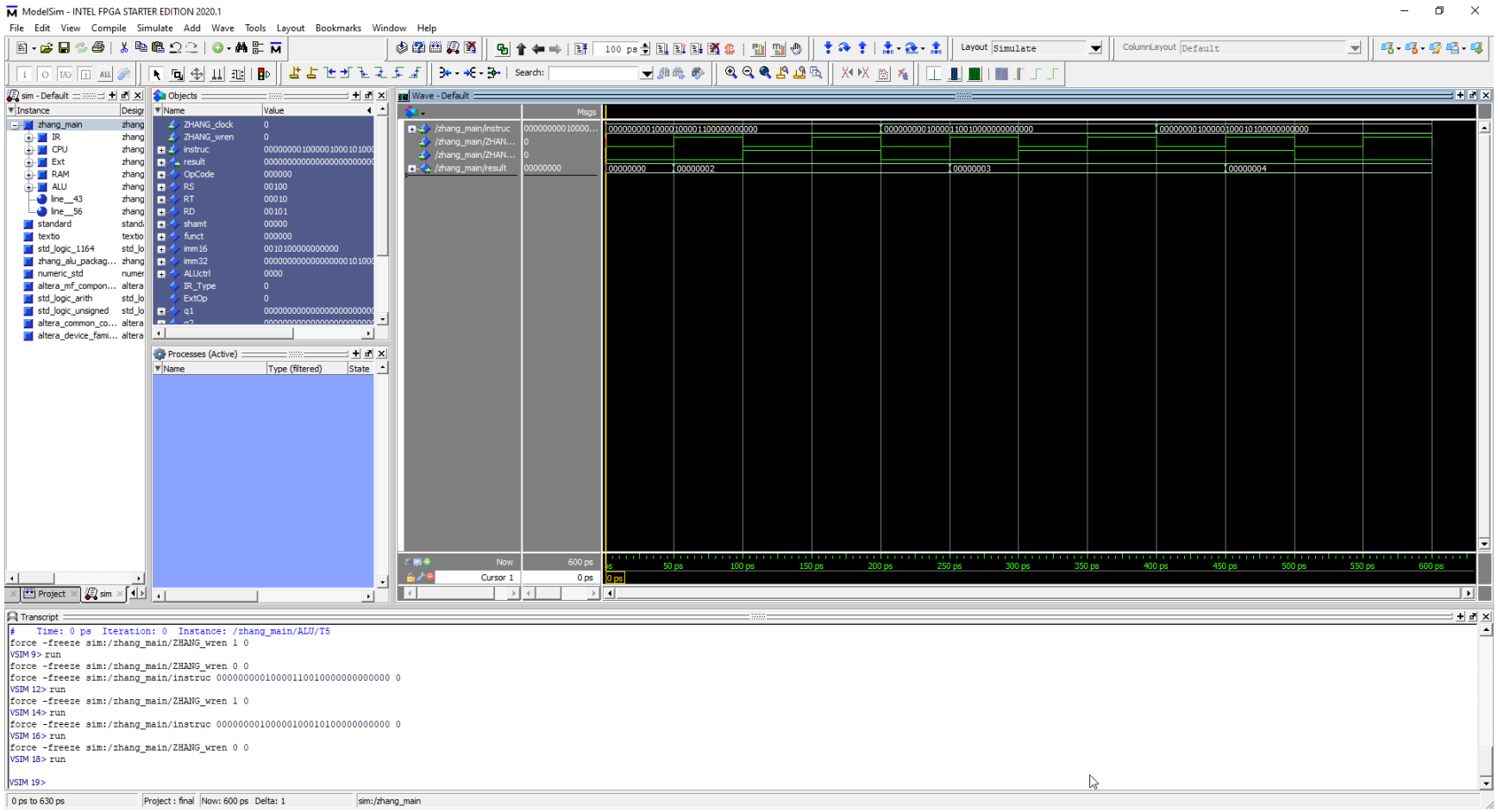
Instructions(1st – 5th):



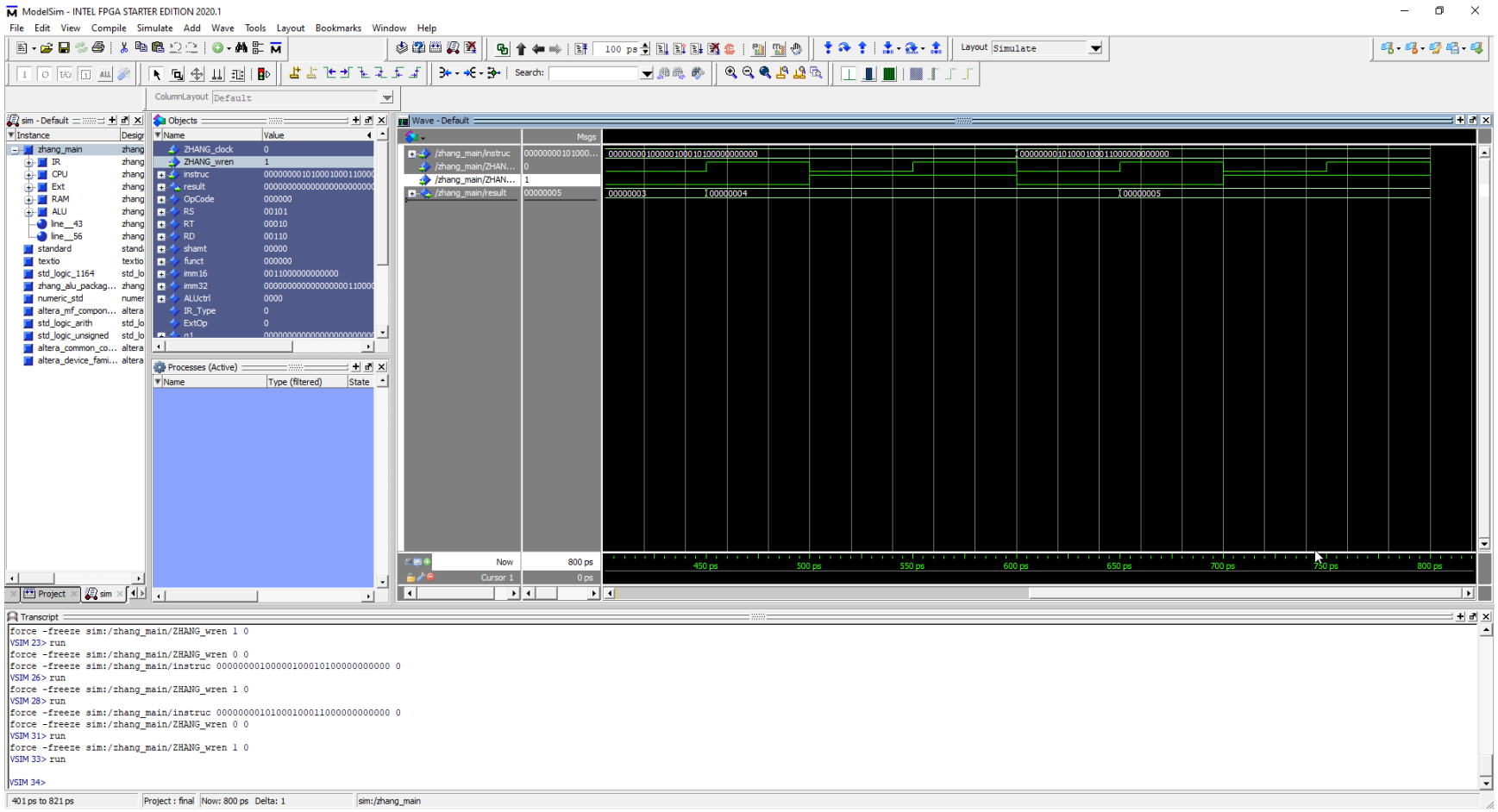
This is the first instruction of 1 + 1, if you look at the result, 2 is outputted then stored into the 2nd register slot



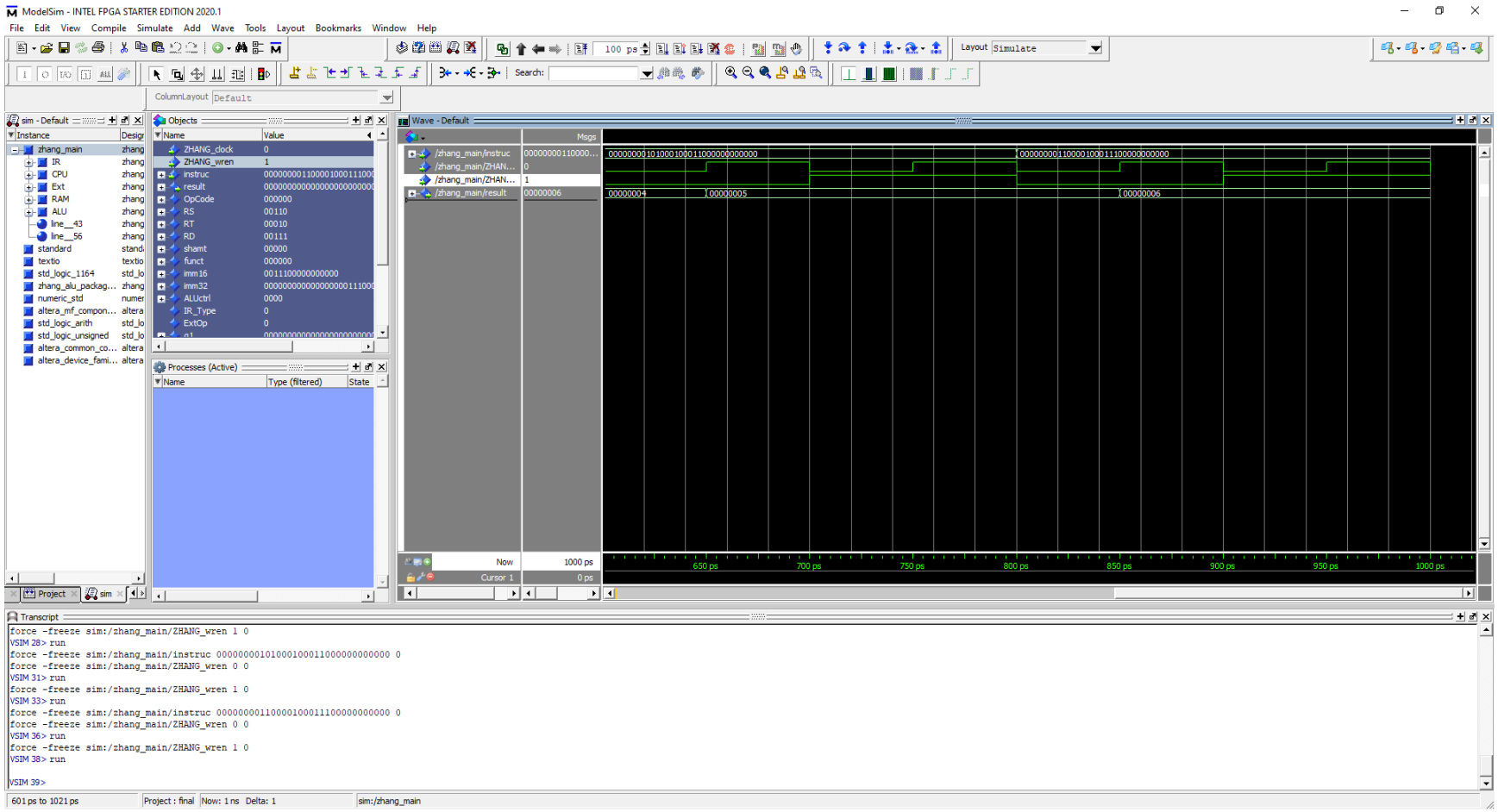
This is the second instruction which adds the 2nd register slot with 1 then stores the result into the 3rd register slot



This is the 3rd instruction and what happens is 3rd register + 1 is performed then written into the 4th register slot



This is the 4th instruction where we perform 4th register slot + 1 and then we store it in the 5th register slot



Lastly, for the last instruction, we perform 5th register slot + 1 and then store it in the 6th register slot

Conclusion:

In conclusion, I have learned about using program counters, instruction memory, data memory and how to store and load those data. This is a very interesting process, and I am grateful for have learnt more about this.