

Fatima Jinnah Women University

Department Of Software Engineering

# PROJECT

**Course Title**

Computer Architecture And Logic Design (111)

**Submitted To**

Dr. Irum Matloob

**Submitted By**

Mariam Fatima Registration No: 2021-BSE-020

**Date of Submission**

January 12, 2023

Table of Contents

1. [ARITHMETIC LOGIC UNIT 5](#_bookmark0)

[RTL Schematic 5](#_bookmark0)

[Program Explanation 5](#_bookmark1)

[AND 5](#_bookmark2)

[Behavior Simulation 5](#_bookmark3)

[OR 6](#_bookmark4)

[Behavior Simulation 6](#_bookmark5)

[ADD 6](#_bookmark6)

[Behavior Simulation 6](#_bookmark7)

[XOR 6](#_bookmark8)

[Behavior Simulation 7](#_bookmark9)

[NOR 7](#_bookmark10)

[Behavior Simulation 7](#_bookmark11)

[SUBTRACT 7](#_bookmark12)

[Behavior Simulation 7](#_bookmark13)

[SET ON LESS THAN 8](#_bookmark14)

[Behavior Simulation 8](#_bookmark14)

[SHIFT LEFT 8](#_bookmark15)

[Behavior Simulation 8](#_bookmark15)

[SHIFT RIGHT 8](#_bookmark16)

[Behavior Simulation 8](#_bookmark16)

[SHIFT RIGHT ARITHMETIC 9](#_bookmark17)

[Behavior Simulation 9](#_bookmark17)

[SIGNED ADD 9](#_bookmark18)

[Behavior Simulation 9](#_bookmark19)

[SIGNED SUBTRACT 9](#_bookmark20)

[Behavior Simulation 9](#_bookmark21)

1. [CONTROL UNIT 10](#_bookmark22)

[Rtl Schematic 10](#_bookmark22)

[Program Explanation 10](#_bookmark23)

[I TYPE 10](#_bookmark24)

[Behavior Simulation 11](#_bookmark25)

[J TYPE 12](#_bookmark26)

[Behavior Simulation 12](#_bookmark27)

[R TYPE 13](#_bookmark28)

[Behavior Simulation 13](#_bookmark29)

1. [MUX 2 TO 1 13](#_bookmark30)

[Rtl Schematic 13](#_bookmark30)

[Program Explanation 14](#_bookmark31)

[IF SELECTION BIT =1 14](#_bookmark33)

[Behavior Simulation 14](#_bookmark34)

[IF SELECTION BIT =0 14](#_bookmark32)

[Behavior Simulation 14](#_bookmark32)

1. [SIGN EXTENSION 14](#_bookmark35)

[Rtl Schematic 14](#_bookmark35)

[Program Explanation 15](#_bookmark36)

[Behavior Simulation 15](#_bookmark37)

1. [PROGRAM COUNTER 15](#_bookmark38)

[Rtl Schematic 15](#_bookmark38)

[Program Explanation 16](#_bookmark39)

[Behavior Simulation 16](#_TOC_250000)

1. [DATA MEMORY 16](#_bookmark40)

[Rtl Schematic 16](#_bookmark40)

[Program Explanation 16](#_bookmark41)

[Behavior Simulation 17](#_bookmark42)

1. [INSTRUCTION MEMORY 17](#_bookmark43)

[Rtl Schematic 17](#_bookmark43)

[Program Explanation 17](#_bookmark44)

[Behavior Simulation 18](#_bookmark45)

1. [REGISTER FILE 19](#_bookmark46)

[Rtl Schematic 19](#_bookmark46)

[Program Explanation 19](#_bookmark47)

[Behavior Simulation 19](#_bookmark48)

1. [CPU 20](#_bookmark49)

[Behavior Simulation 20](#_bookmark49)

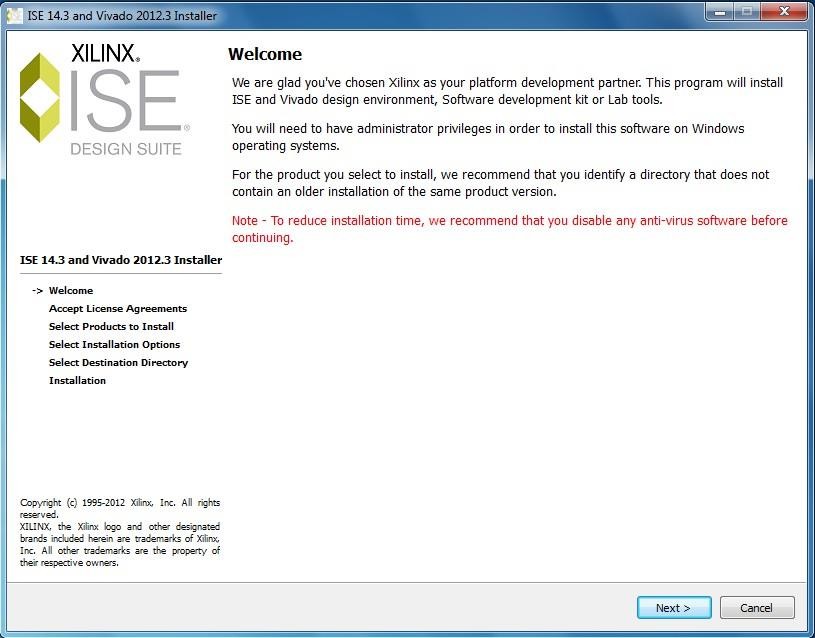
# ILINX INSTALLATION

* **Website**

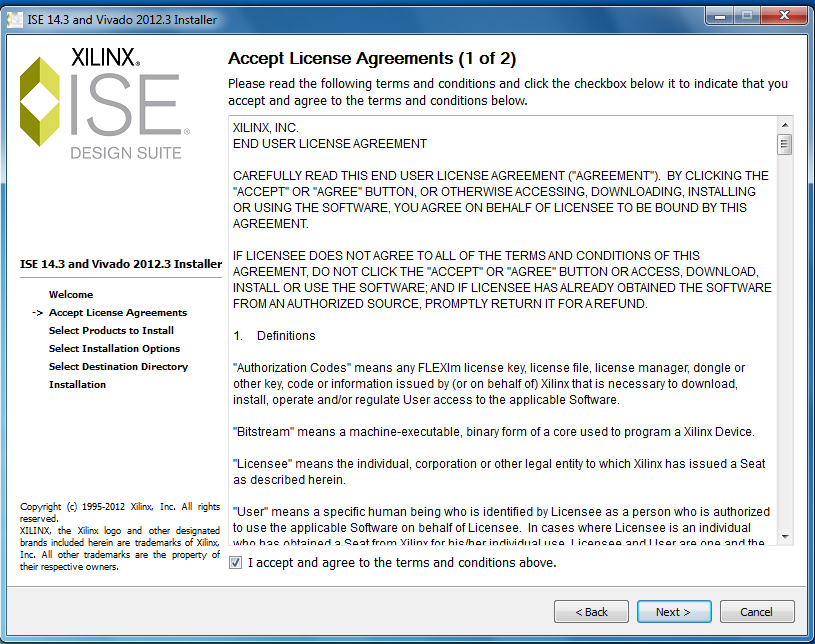
https://[www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/vivado-design-tools/archive-ise.html](http://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/vivado-design-tools/archive-ise.html)



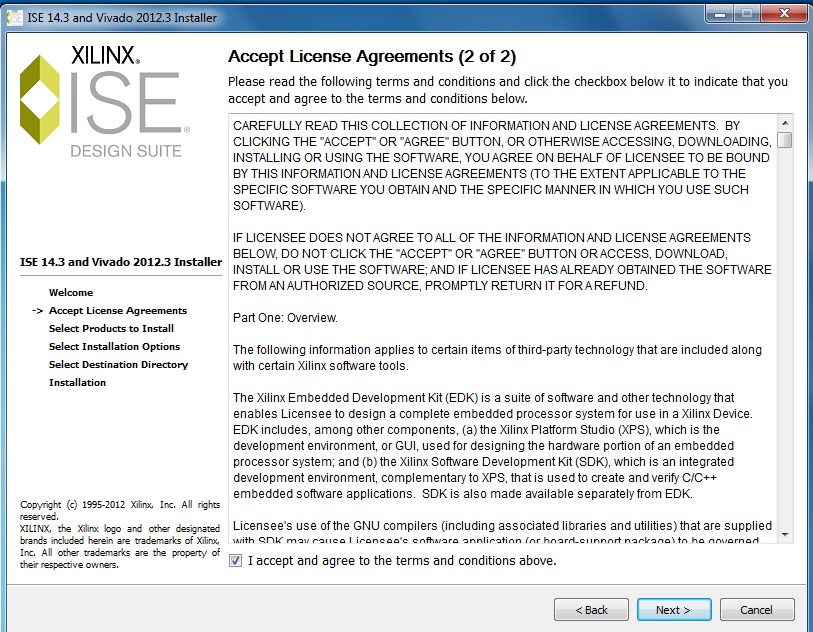
Extract files and run xsetup.exe



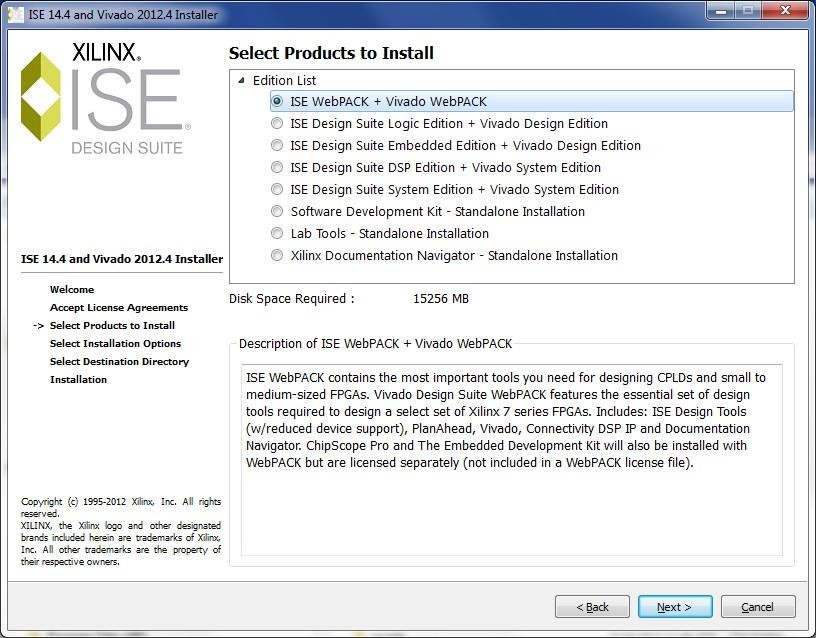
* Click next

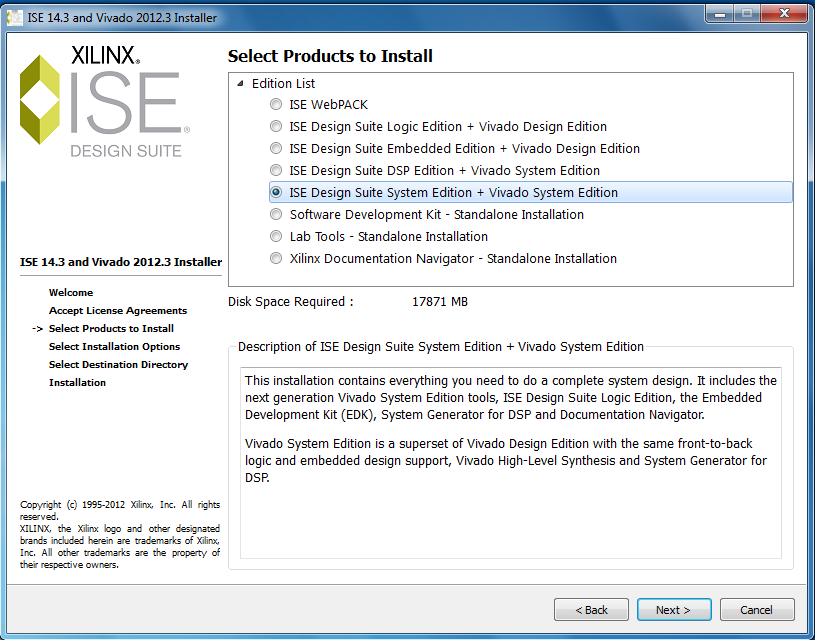


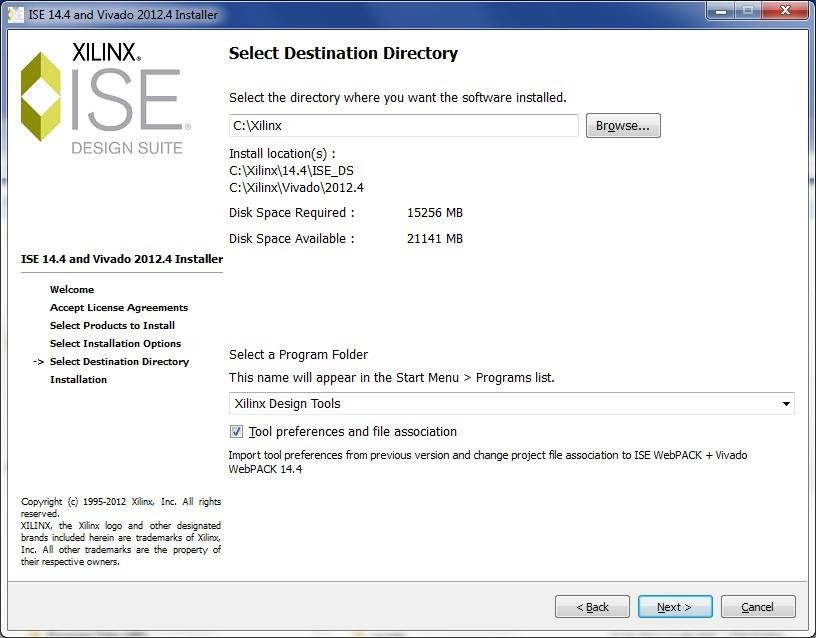
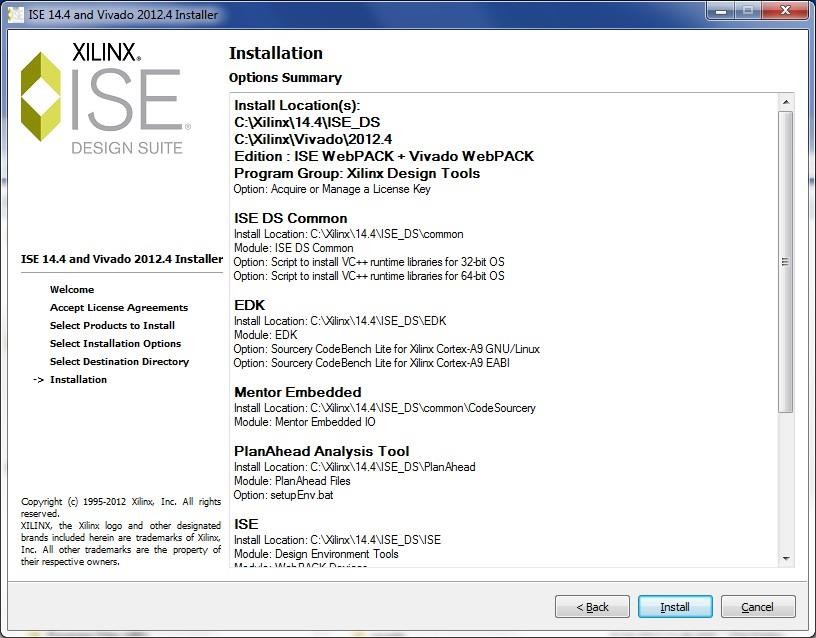
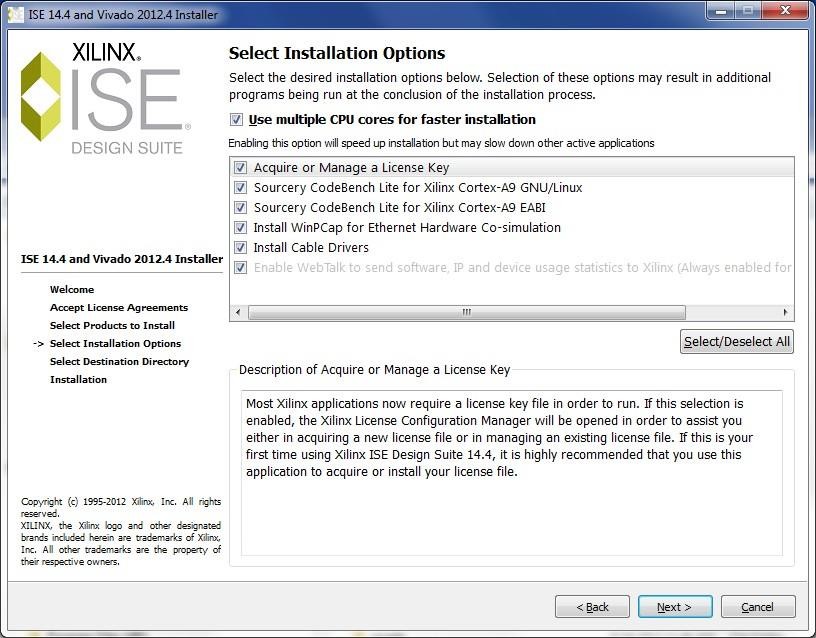
* Click next



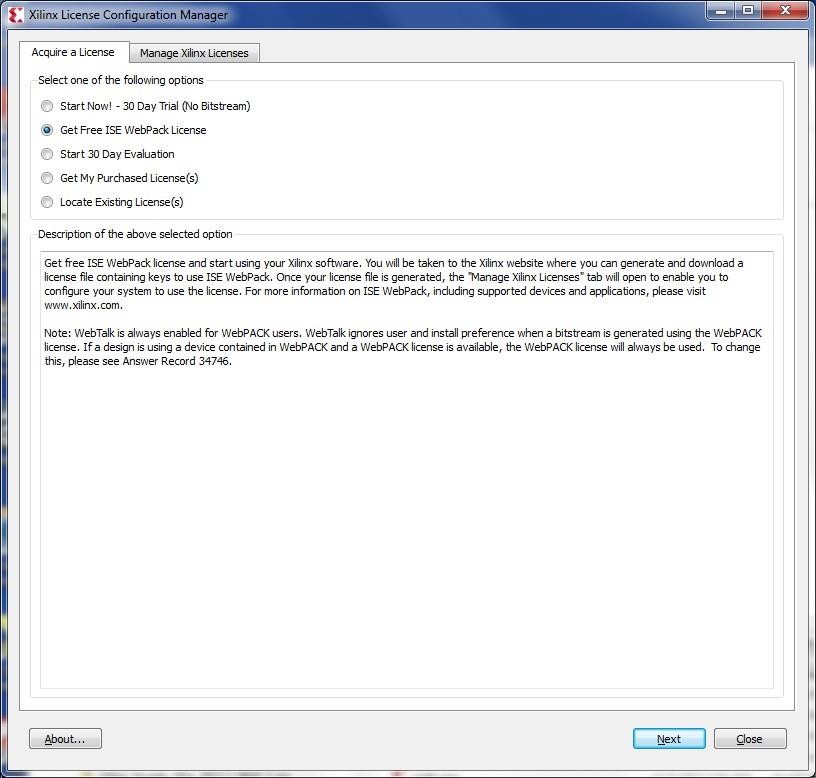
* Select product to install



* Select the product to install



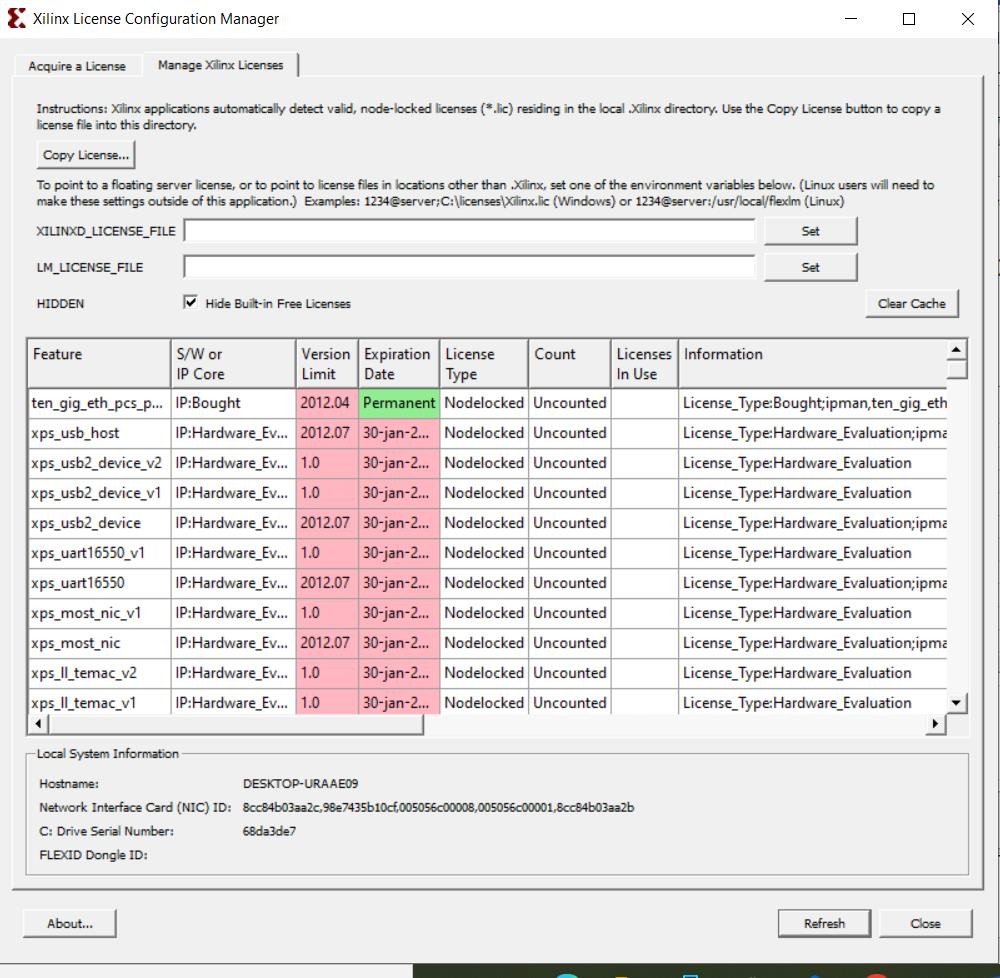
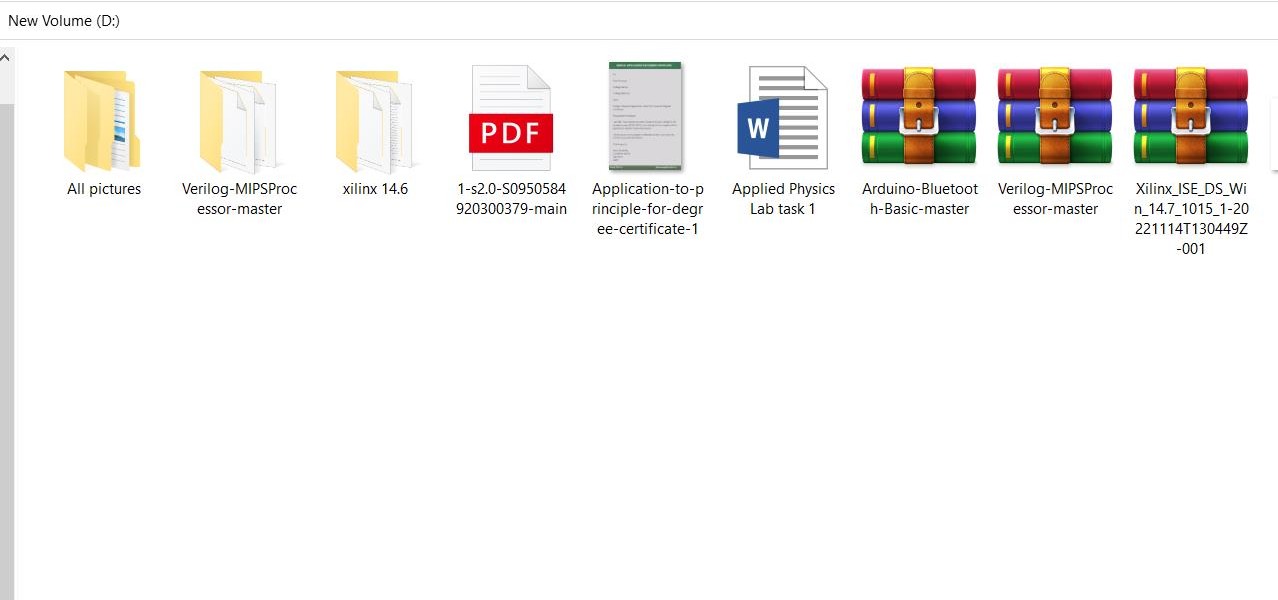
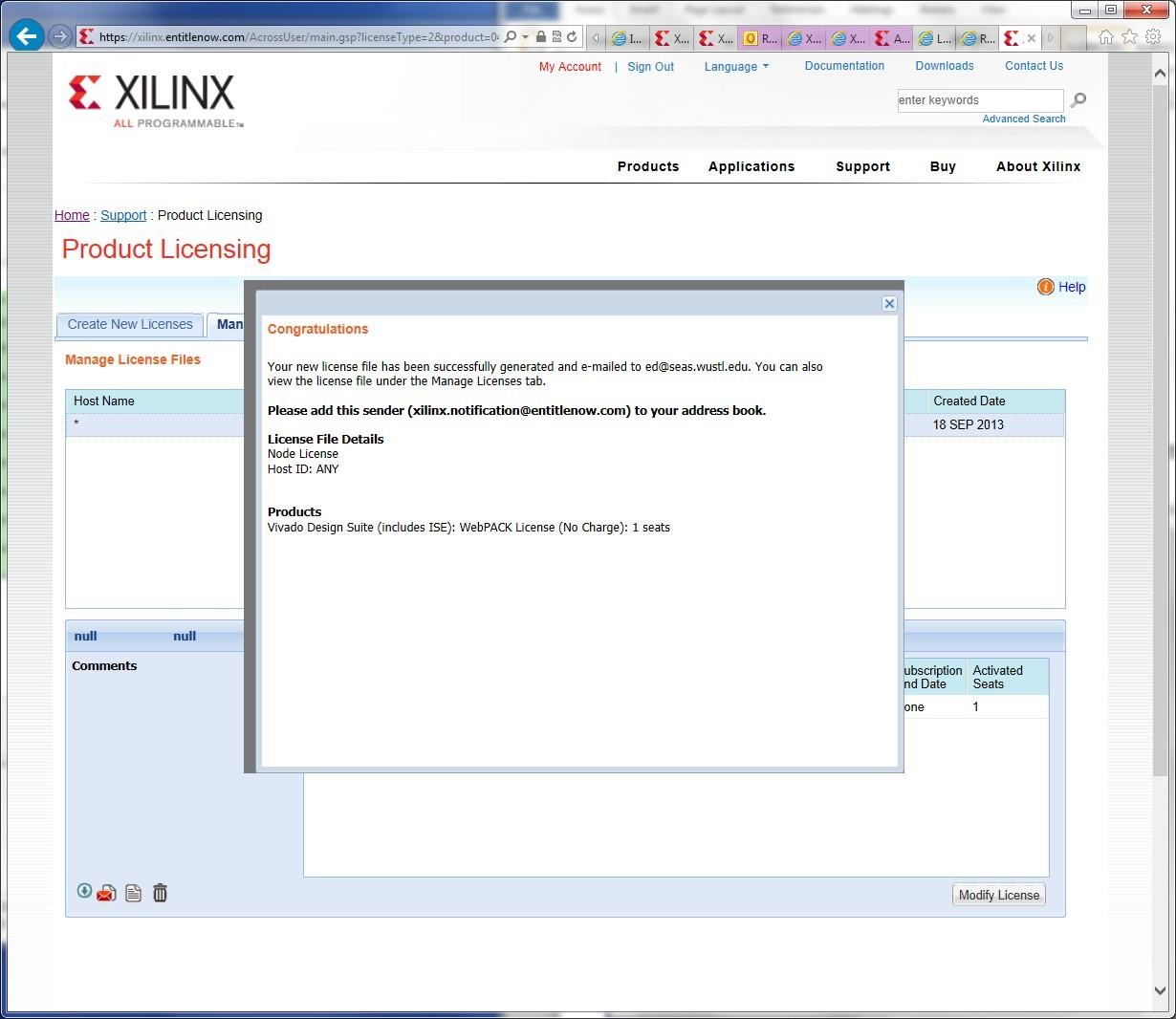
* Click to install

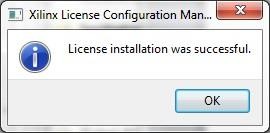




* For the Web PACK (FREE) version, click Next

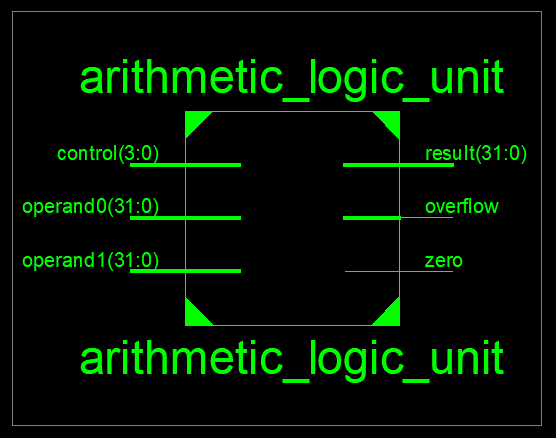
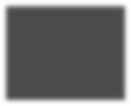






# ARITHMETIC LOGIC UNIT

# RTL Schematic



**Operand0= 00100010001100010000000000000000**

**Operand1= 00100000000010000000000001001011**

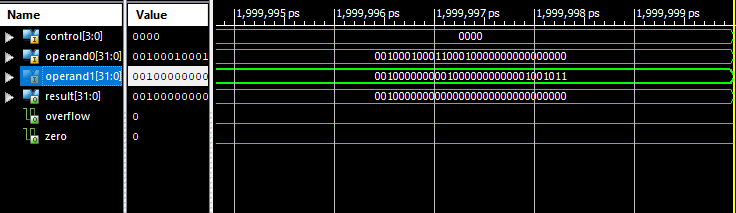
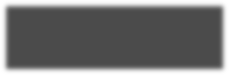
# Program Explanation

In this program, According to control bit, the operations will be performed on operands.

# AND

If the control bit is “**0000**”**, AND** operation will performed on the operands. It will compare each bit of the 2 operands. If the bit of both operands are 1 then the result bit will be “**1**”, otherwise result bit will be “**0**”

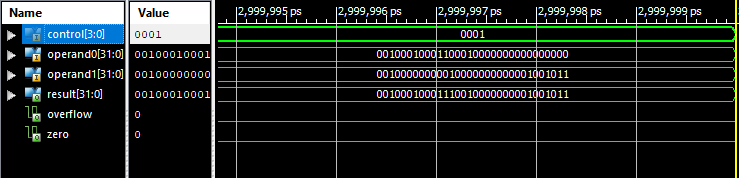
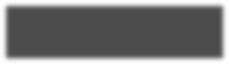
**Behavior Simulation**



# OR

If the control bit is “**0001**”**, OR** operation will performed on the operands. It will compare each bit of the 2 operands. If the bit of both operands are “**0**” then the result bit will be “**0**”, otherwise result bit will be “**1**”

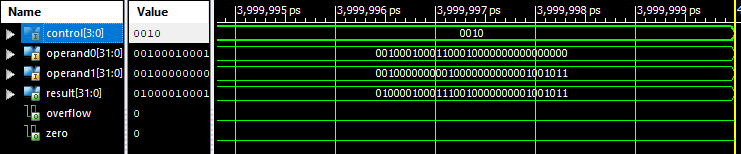
**Behavior Simulation**



# ADD

If the control bit is “**0010**”**, ADD** operation will performed on the operands. It will add the both operands. If the bit of both operands are 0 then the result bit will be “**0**”, otherwise result bit will be “**1**” but if the bit of both operands are “**1**”, then the result bit will be “**0**” and “**1**” will be taken as a carry for the next bit. This will form a new number that is the result of addition of the two operands.

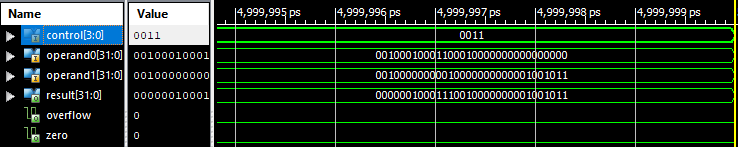
**Behavior Simulation**



# XOR

If the control bit is “**0011**”**, XOR** operation will performed on the operands. It will compare each bit of the 2 operands. If the bit of both operands are opposite then the result bit will be “**1**”, and if they are same then resultant bit will be “**0**”

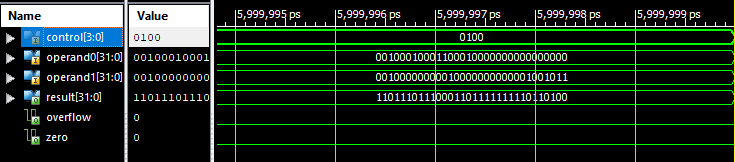
**Behavior Simulation**



# NOR

If the control bit is “**0001**”**, NOR** operation will performed on the operands. It will compare each bit of the 2 operands. If the bit of both operands are “**0**” then the result bit will be “**1**”, otherwise result bit will be “**0**”

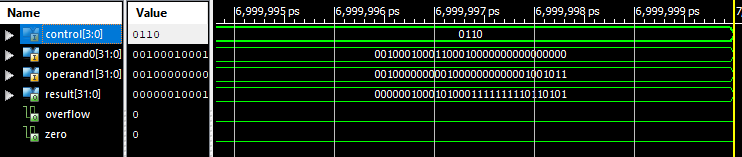
**Behavior Simulation**



# SUBTRACT

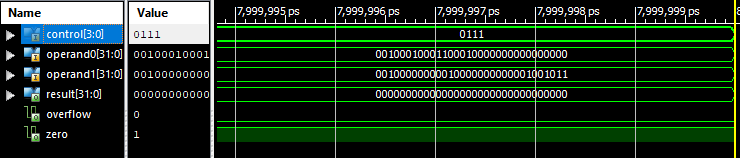
If the control bit is “**0110**”**, Subtract** operation will performed on the operands. It will add the both operands. If the bit of both operands are 0 then the result bit will be “**0**”, otherwise result bit will be “**1**” but if the bit of both operands are “**1**”, then the result bit will be “**0**” and “**1**” will be taken as a carry for the next bit. This will form a new number that is the result of addition of the two operands.

# Behavior Simulation

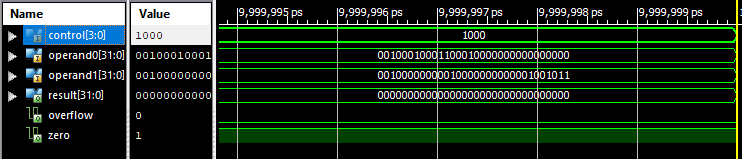


**Behavior Simulation**

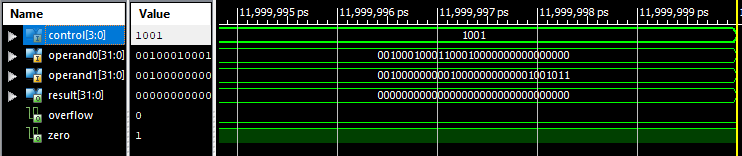
# SET ON LESS THAN



# SHIFT LEFT



**Behavior Simulation**

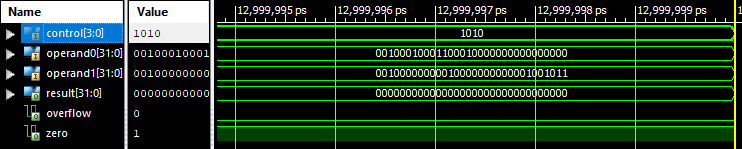


**SHIFT RIGHT**

**Behavior Simulation**

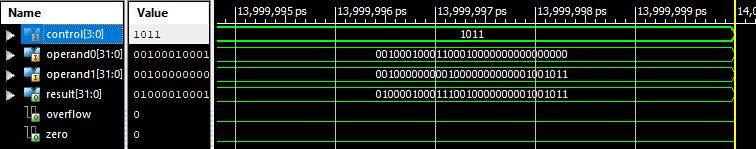
**Behavior Simulation**

# SHIFT RIGHT ARITHMETIC



**SIGNED ADD**

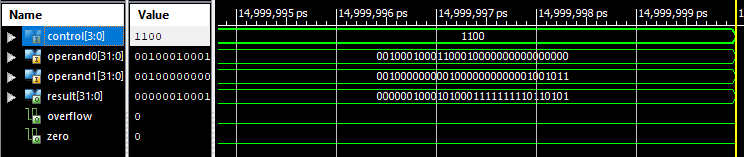
Two operands are given as an input and control bits 1011 will perform signed add operation.



**Behavior Simulation**

# SIGNED SUBTRACT

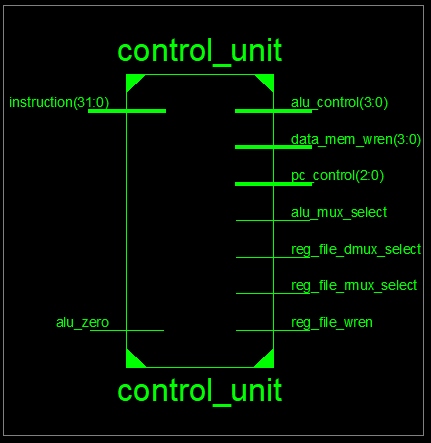
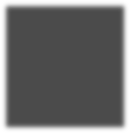
Two operands are given as an input and control bits 1101 will perform signed subtract operation.



**Behavior Simulation**

# CONTROL UNIT

**RTL Schematic**



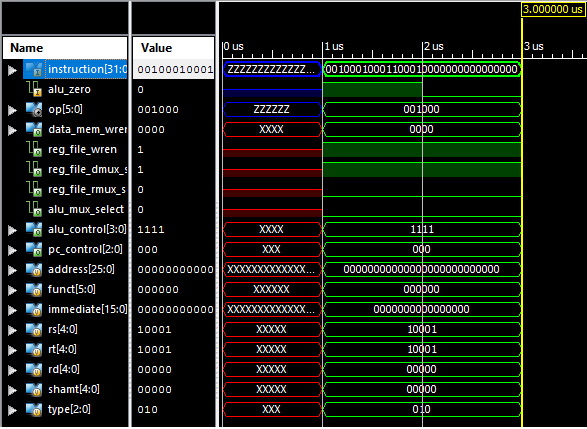
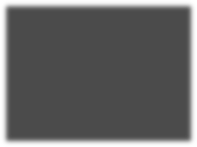
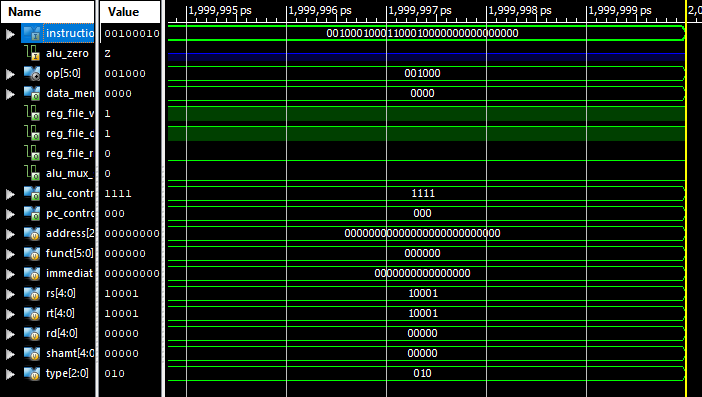
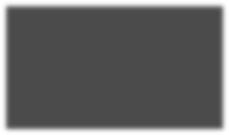
# Program Explanation

In this program, the instruction of 32 bit will be entered as input and the operations will be performed according to the instruction bits. **[31:26] bits** of the instruction will be set as an opcode. According to opcode the instruction will be determined as either it is I type, J type or R type. Then other bits will be assigned according to the opcode.

# I TYPE

If the opcode is other than 000000, 000010, 000011 then Address will be assigned 00000000000000000000000000 bits and the immediate will be assigned instruction [15:0] bits. **rs** will be assigned bits of instruction[25:21]. **rt** will be assigned bits of instruction[20:16]. **rd** will be 00000 because in I type, there is no need of rd. shamt will be assigned bits of instruction[10:6]. Type will be **010** and the funct bit will be bits of instruction [5:0];

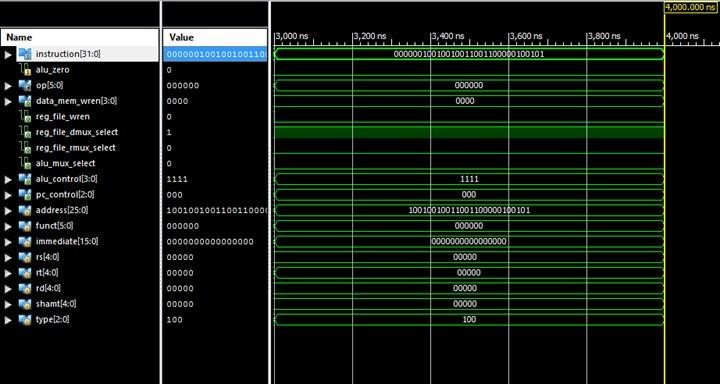
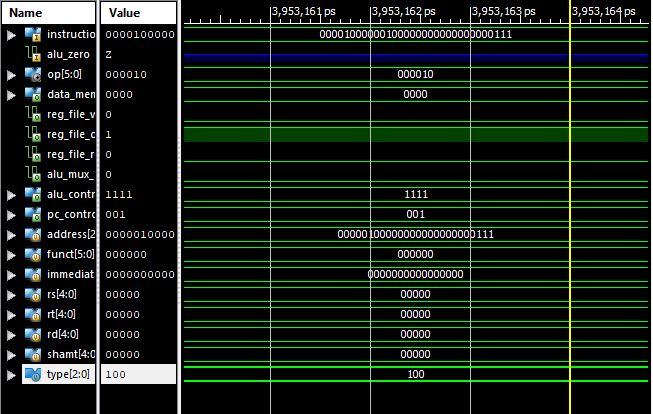
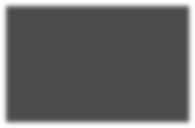
# Behavior Simulation



**J TYPE**

If the opcode is either 000010 or 000011, then Address will be assigned instruction [25:0] bits and the immediate will be assigned 0000000000000000 bits. **rs** will be assigned 00000 bits. **rt** will be assigned 00000 bits. **rd** will be 00000. shamt will be assigned 00000 bits. Type will be **100** and the funct bit will be 000000.

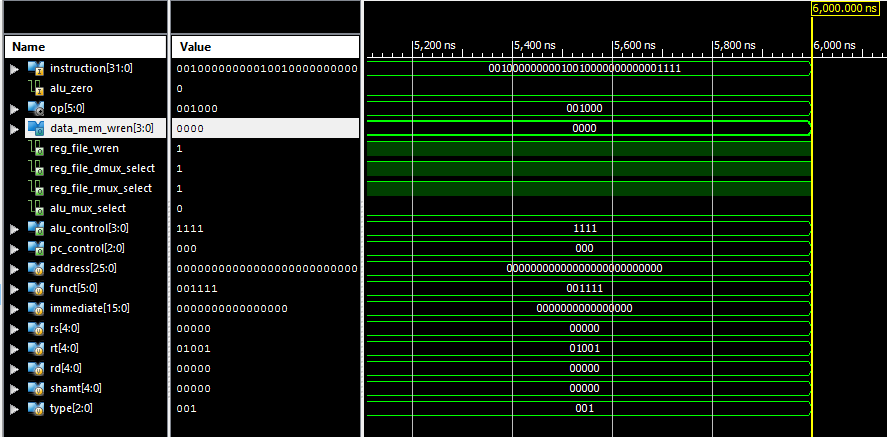
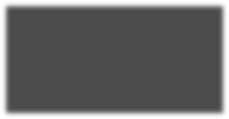
# Behavior Simulation



**R TYPE**

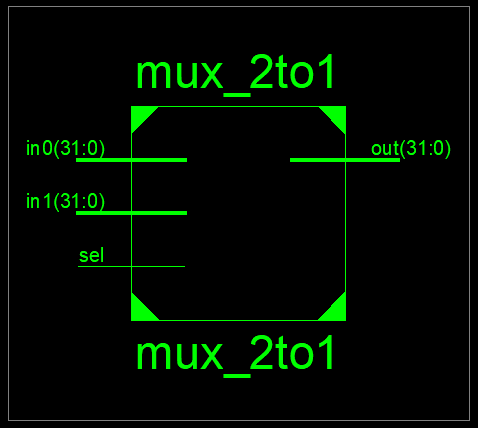
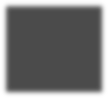
If the opcode is **000000** then the immediate will be 0000000000000000 bits. Address will be assigned 00000000000000000000000000 bits. **rs** will be assigned bits of instruction[25:21]. **rt** will be assigned bits of instruction instruction[20:16]. **rd** will be instruction[15:11]. shamt will be assigned bits of instruction[10:6]. Type will be **001** and the funct bit will be bits of instruction [5:0];

# Behavior Simulation



**Rtl Schematic**

# MUX 2 TO 1

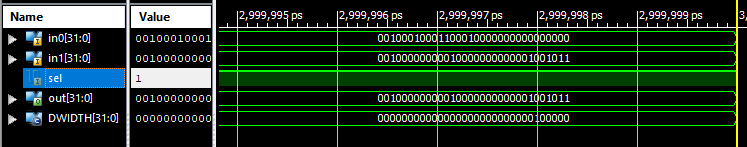


# Program Explanation

In this program, two 32 bits numbers will be taken as an input and if the selection bit is “**0**”**,** then the output will be the number input through port **In0( first input)** and if the selection bit is “**1**”, then the output will be the number input through post **In1( Second input).**

# IF SELECTION BIT =1

# Behavior Simulation



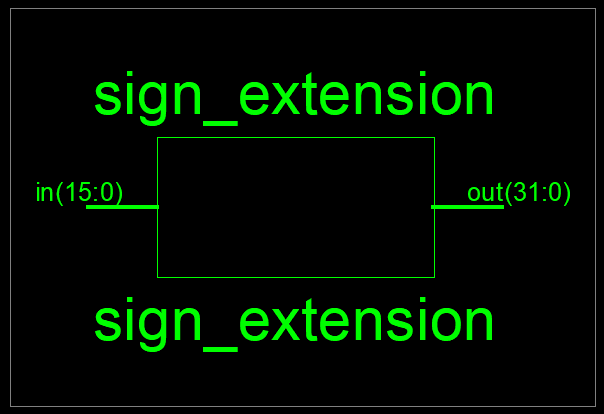
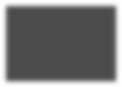
# IF SELECTION BIT =0

**Behavior Simulation**



**Rtl Schematic**

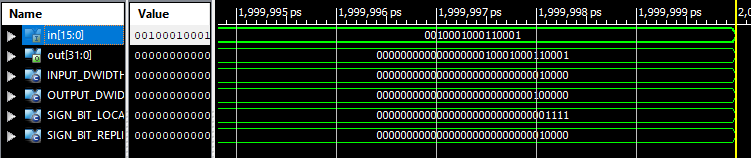
# SIGN EXTENSION



# Program Explanation

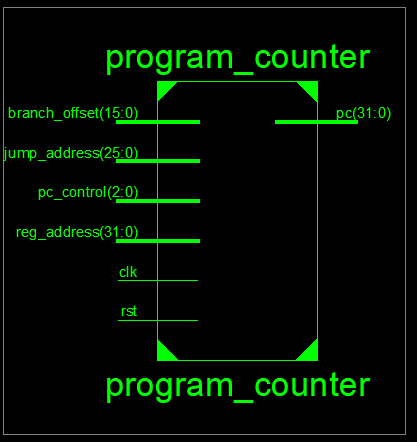
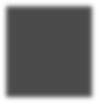
In this program, 16 bit instruction will be taken as input and this instruction will be extended to 32 bit instruction.

# Behavior Simulation

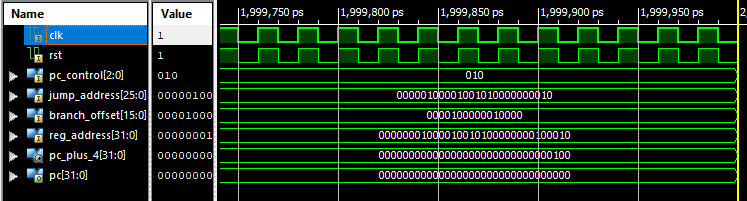
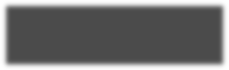


**Rtl Schematic**

# PROGRAM COUNTER

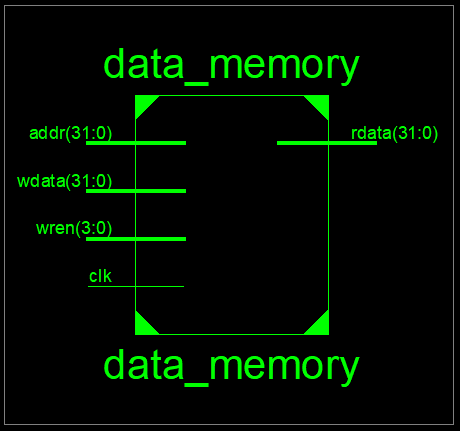
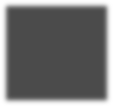


**Program Explanation Behavior Simulation**



**Rtl Schematic**

# DATA MEMORY



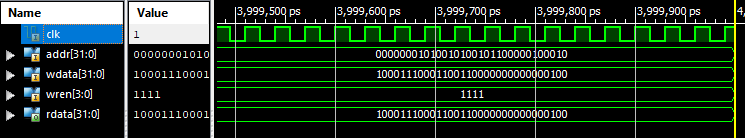
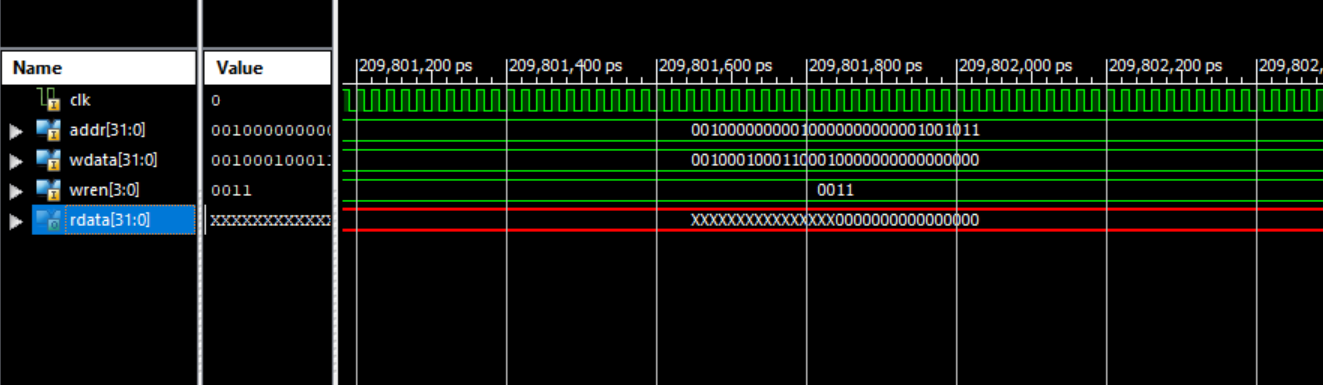
**Program Explanation**

The data memory code describes a clock cycle in which 32 bit data **wdata [31:0]** is written on a given address **addr [31:0], wren [3:0]** 4 bit input is given by the user,

When 0000 is given as an input this means that no data is read. When 0001 is given as an input this reads last 8 bits of wdata [31:0].

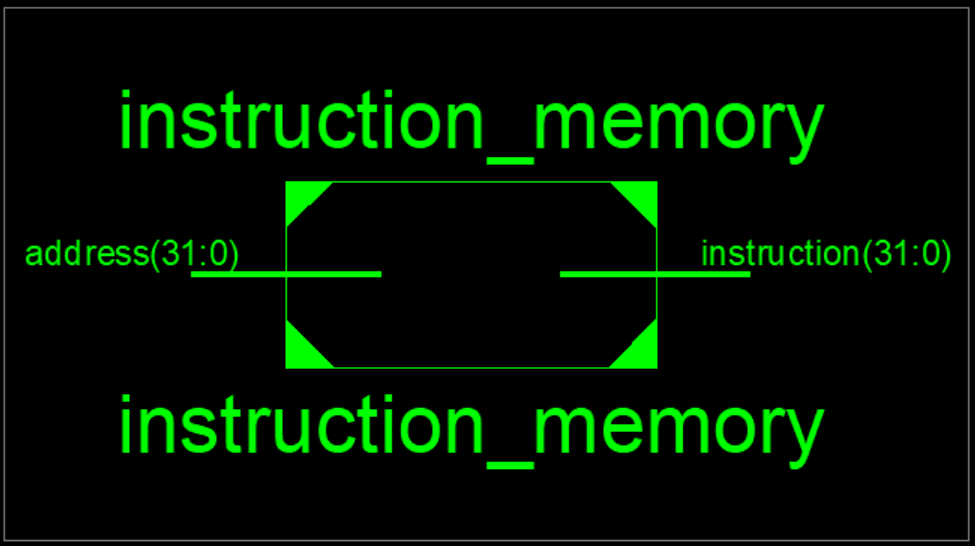
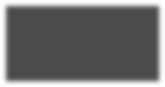
When 0011 is given as an input this reads last 16 bits of wdata [31:0]. When 0111 is given as an input this reads last 24 bits of wdata [31:0]. When 1111 is given as an input this reads all 32 bits of wdata [31:0].

# Behavior Simulation



**Rtl Schematic**

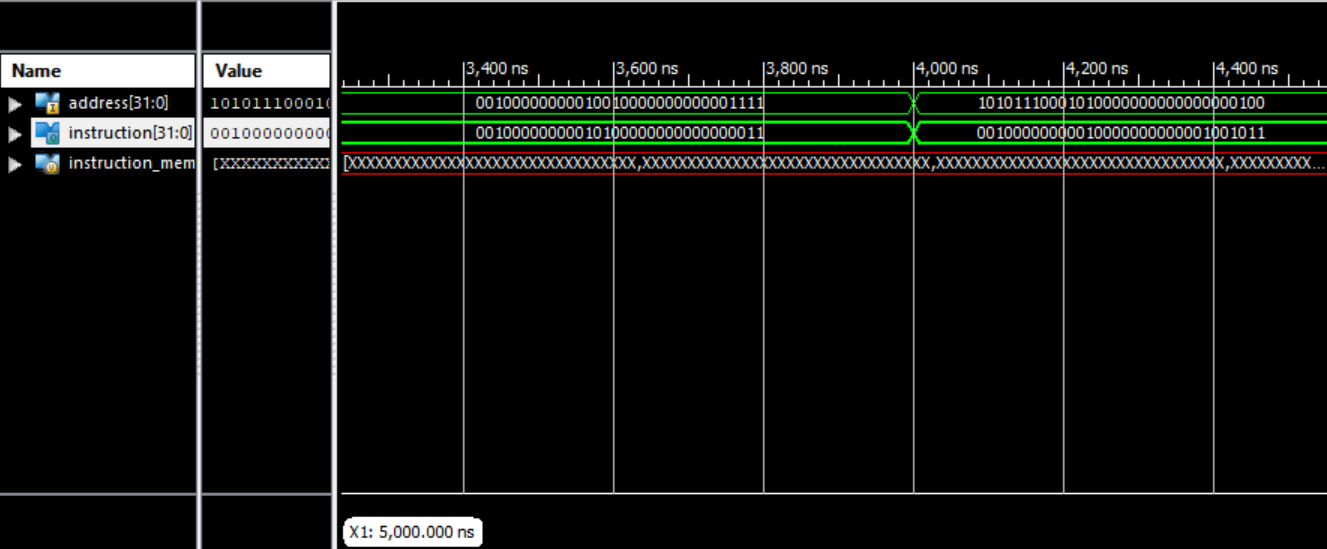
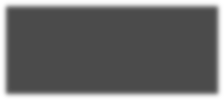
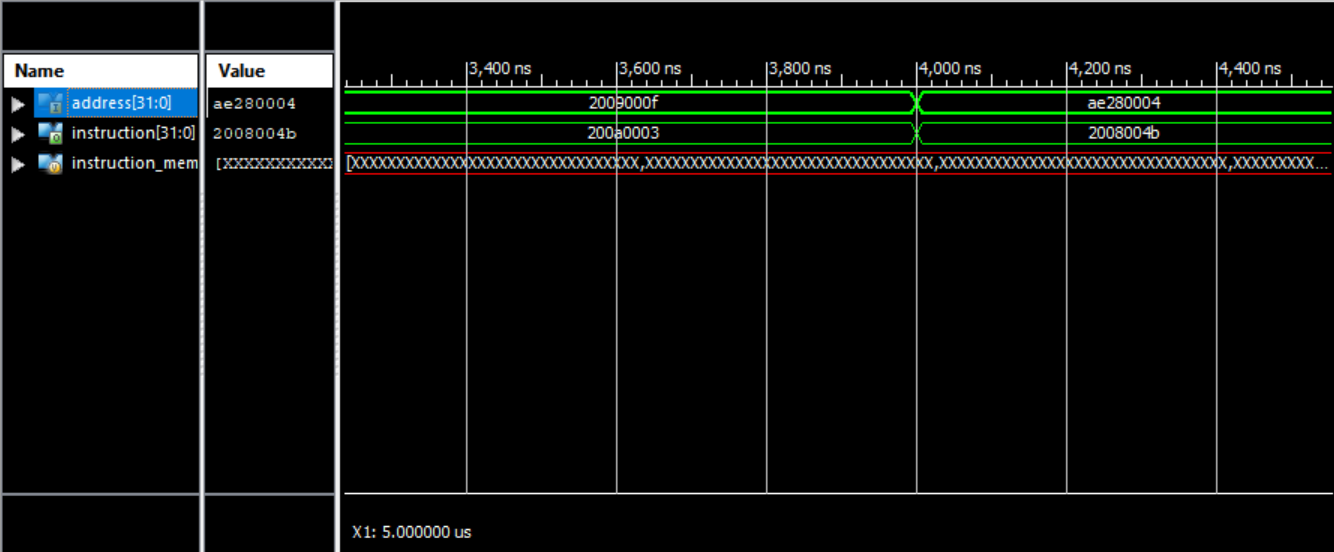
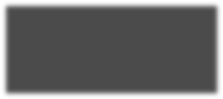
# INSTRUCTION MEMORY



# Program Explanation

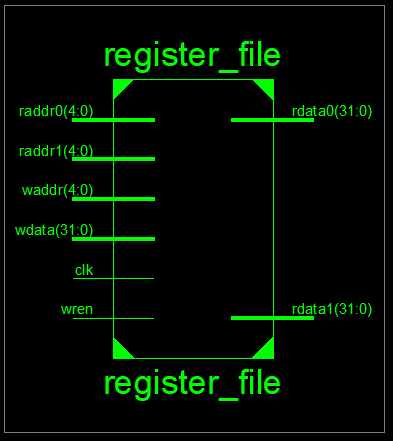
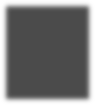
Program.mips file contain number of addresses. 32 bit address **address [31:0]** is given as an input. Instruction is fetched from that address. We read the instruction asynchronously, based on the address provided by the PC.

# Behavior Simulation



**Rtl Schematic**

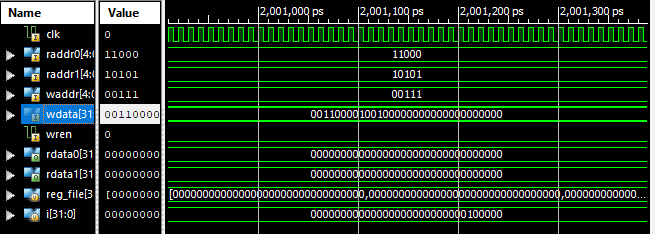
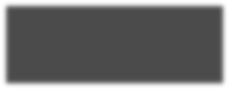
# REGISTER FILE



# Program Explanation

A **register file** is the array of processor registers present within the central processing unit (CPU). As it is placed within the CPU so it is faster than other memory devices and is used to fetch and hold data from secondary memory devices.

In the register file we have 6 inputs and 2 outputs. The register file module cosist of 32-bit data input line.



**Behavior Simulation**

# CPU

# Behavior Simulation

