

University of Bahrain

College of Information Technology

Department of Computer Engineering

ITCE362-364 Computer Architecture

Design A Full SRC Processor

**Prepared by:**

Student name: Waleed Saleh Ali Saleh ID: 202006448

Student name: Najim Abdulkarem Musaed Alfutini ID: 202003215

Student name: Maged Hussain Masaad Ali Aljashoobi ID: 202004484

Section: 1

Date: 29-12-2022

***Contents:***

|  |  |
| --- | --- |
| Topic | Page |
| Cover | 1 |
| Contents and objectives | 2 |
| Introduction | 3 |
| Processor specs | 3 |
| Implementation :control unit | 5 |
| Implementation :data path | 11 |
| Simulation: control unit | 19 |
| Simulation: data path | 23 |
| Conclusion | 24 |

***Objectives:***

1. Define the project specs and design logic.
2. Design and connect project components.
3. Simulate the system and make sure that works will.

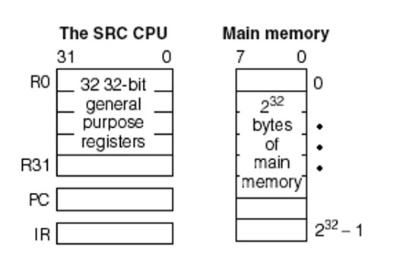
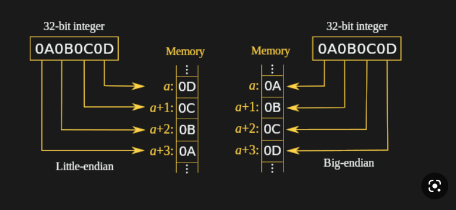
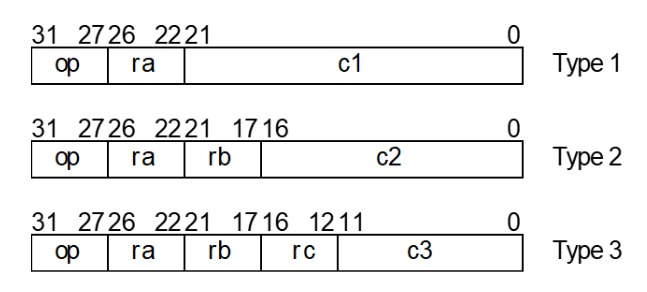
**Introduction To The Processer:**

A SRC (Simple RISC Computer) processor is a type of computer processor that uses a limited number of instructions to perform its tasks. RISC processors are widely used in a variety of applications, including embedded systems, mobile devices, and servers. They are known for their high performance and low power consumption, making them an attractive choice for many types of systems.

RISC processors are designed to be simple, fast, and efficient, with a focus on executing a small set of instructions fast rather than trying to support many complex instructions. RISC processors have a limited number of instructions, commonly fewer than 100. (e.g. 23 instructions in our design) These instructions are simple and easy to execute, which allows the processor to perform them quickly. It uses a fixed-length instruction format, which means that all instructions are the same size. use a load-store architecture, which means that they have separate instructions for loading data from memory into registers and for storing data from registers back into memory. Use simplified addressing mode (immediate, Direct, Indirect, Register direct, etc.…).

**Processor specs and design logic (Design Decisions):**

SRC processor has specific specifications, the specs of the SRC that we designed is:

* Our design will be a 1-bus system. Which means we use one bus to transmit/share data between the components.
* In every clock we perform one micro-operation.
* It has a 32 general purpose register and each one has size of 32-bit .
* It has a 32-bit program counter (PC) which means it can access = 4GB.
* It has a 32-bit instruction register (IR).
* The word is 4 bytes = 32-bit, and only a 4 byte can be fetched or stored into main memory.
* Main memory is organized as an array of bytes.
* We use the Big-Endian way to store the words in the memory which means the most significant 8 bits will be stores in the lowest address.
* We have 23 instructions in 8 different formats depend on the number of registers uses and length of the constant. There are three main instruction format types:
* Fixed instruction length means each instruction is 32-bit size.
* Since we have 23 instructions, we need 5 bits to represent the opcode:

|  |  |  |  |
| --- | --- | --- | --- |
| **Opcode** | **Opcode value** | **Opcode** | **Opcode value** |
| ld | 1 | and | 20 | |
| ldr | 2 | andi | 21 | |
| st | 3 | or | 22 | |
| str | 4 | ori | 23 | |
| la | 5 | not | 24 | |
| lar | 6 | shr | 26 | |
| br | 8 | shra | 27 | |
| brl | 9 | shl | 28 | |
| add | 12 | shc | 29 | |
| addi | 13 | nop | 0 | |
| sub | 14 | stop | 31 | |
| neg | 15 |  |  | |

**Implementation:**

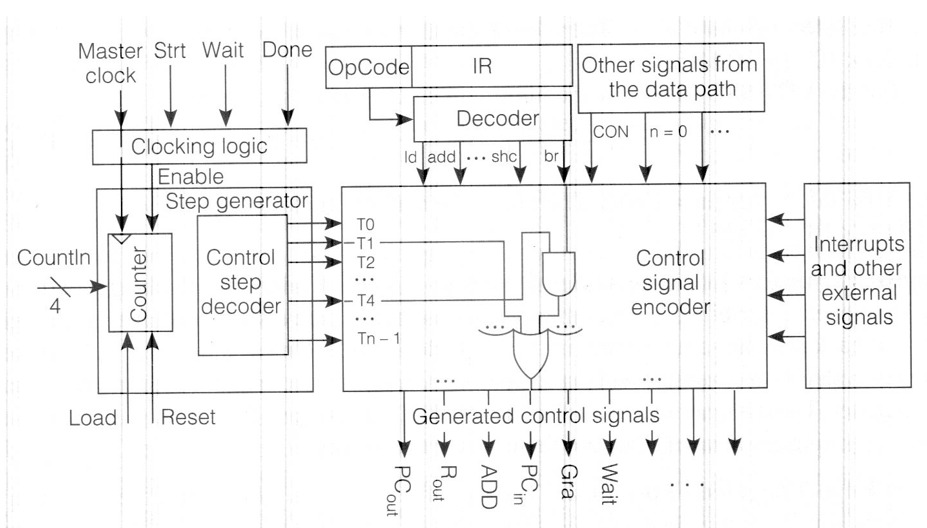
The SRC processor consists of two main parts: control unit and data path.

1. **Control unit:**

It is the component that generates the control signals that direct the operations in the data path. There are two types of the control unit (hardwired and microprogrammed):

* + The hardwired control unit treats the relationship between control I/O as a series of Boolean function.
  + The microprogrammed control unit treats the relationship between control I/O as a memory system.

We chose the hardwired because it is suitable for a small number of instructions and faster.

The control unit design:

The control unit contains of four main components:

* 1. Clocking logic: The main function of this component is to generate enable signal for the counter in the step generator.

Diagram, schematic

Description automatically generated

* 1. Diagram, schematic

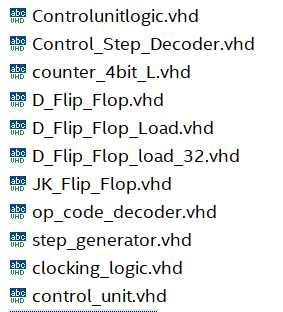
     Description automatically generatedThe control step generator: contains of a four-bit up counter and control step decoder which give the wanted step. The counter is with load input because if the counter reserve a Goto6 signal must the step generator go to step 6 by load 0110 to counter.
  2. Opcode decoder: This decoder will take opcode as input IR<31 …27> and will generate a signal in the wanted instruction.
  3. Control signal encoder: Is the main part in control unit because it contains a Boolean function for each signal.

This is all the signals that has Boolean function:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| PCout | Read | IRin | ADD | Rout | SUB | C=B |
| MAin | Wait | Grb | Gra | MDbus | AND | SHR |
| INC4 | Cout | BAout | Rin | Write | OR | Decr |
| Cin | PCin | Ain | End | Grc | NOT | Goto6 |
| PCout | MDout | C2out | C1out | CONin | Ld | SHRA |
| Shl | Shc |

**The RTL for control unit:**

In control unit we use these files:



1. Diagram, schematic

   Description automatically generatedClocking logic RTL:
2. Control step generator RTL:

Chart, waterfall chart

Description automatically generated

1. A screenshot of a computer

   Description automatically generated with low confidenceOpcode decoder RTL: d. Control signal encoder:

A screenshot of a computer

Description automatically generated with low confidence

The RTL for top-entity control\_unit.vhd :

Diagram, schematic

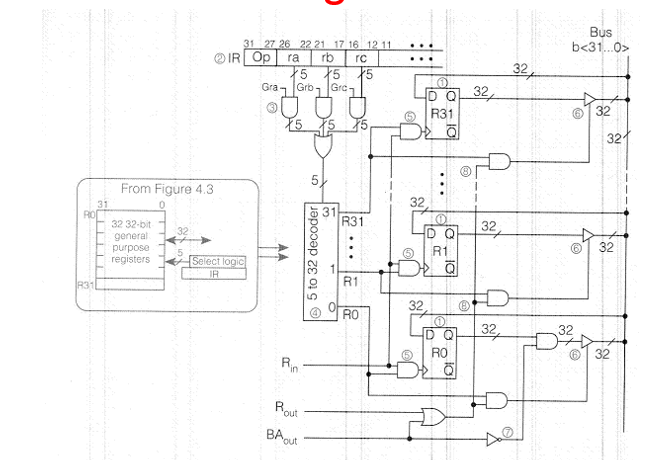
Description automatically generated

1. **Data path:**

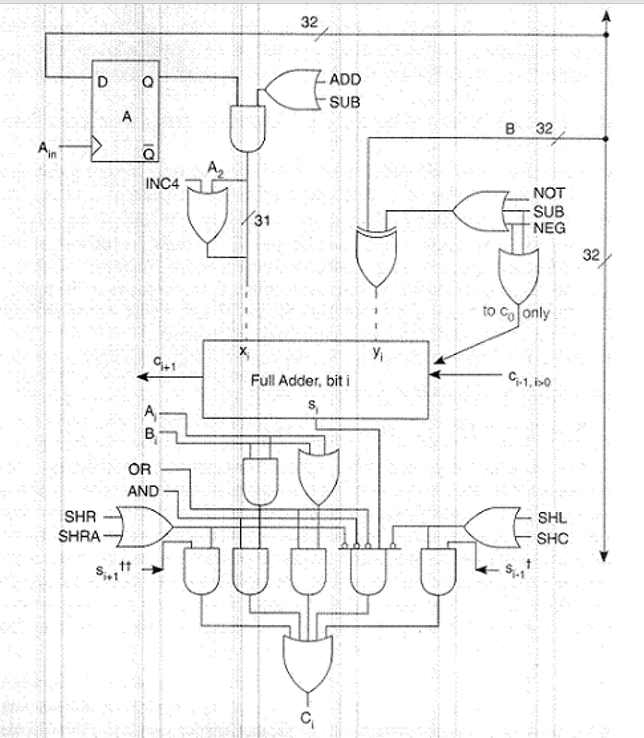
The data path is a collection of functional units such as arithmetic logic units or shift counter that perform data processing operations, registers, and buses.

And it contains:

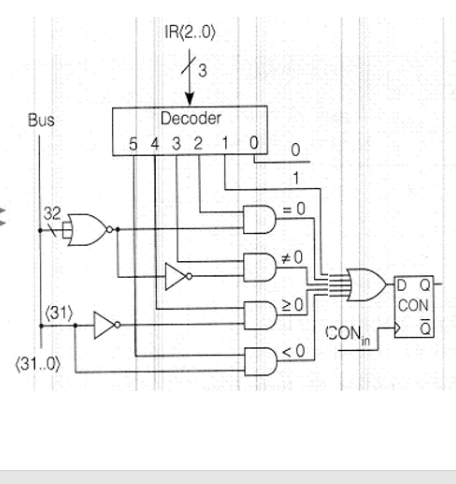
* 1. 32 registers of 32-bit size: It contains of 5 to 32 decoder and 32 registers.



* 1. ALU (Arithmetic logic unit): It designs to perform all the arithmetic and logical operations.



* 1. PC (Program counter): It a 32-bit register with inout port that contain the address for the instruction.
  2. Cond logic: used for conductional branch instruction. design of 3 to 8 decoder and d-flip flop to hold the value of brunch.



* 1. IR (Instruction register).
  2. Memory interface: consist of two 32-bit register first is for memory address and the second for memory data. And it control the accessing to the memory by reserving signals form control unit.

Diagram

Description automatically generated

* 1. Shift counter for shift instructions.

Diagram, schematic

Description automatically generated

**The RTL for data path:**

In data path we use these files:

Graphical user interface, application

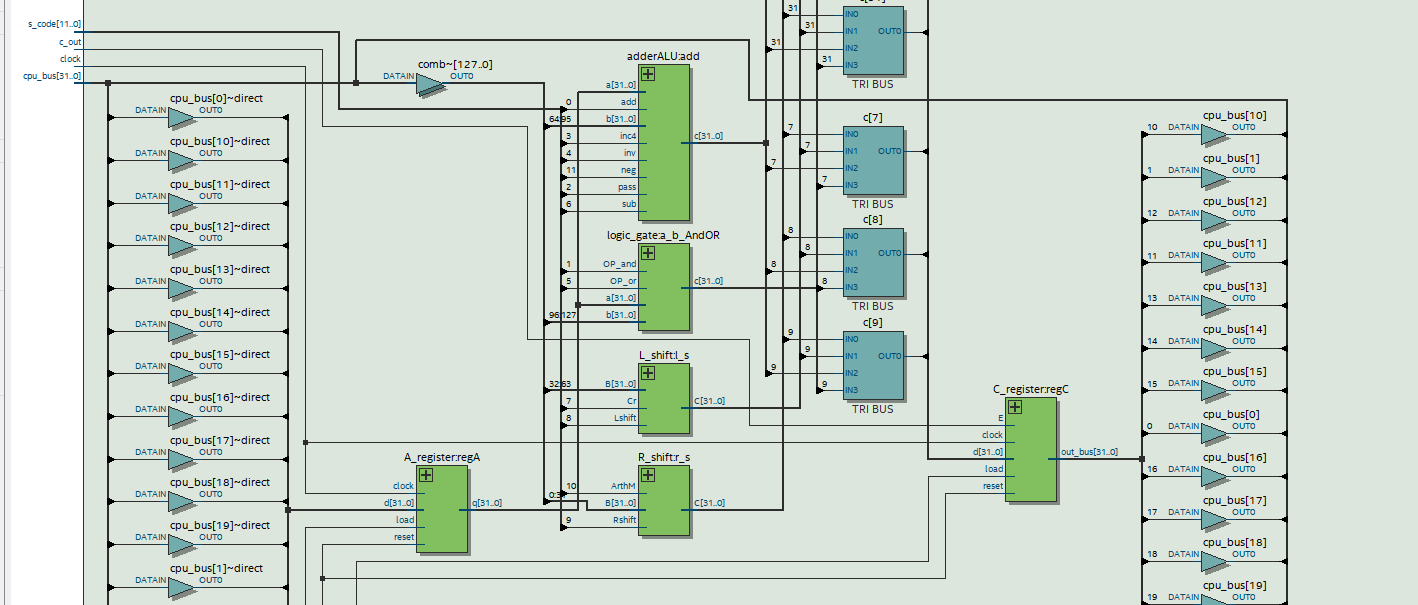
Description automatically generated

1. Register file RTL:

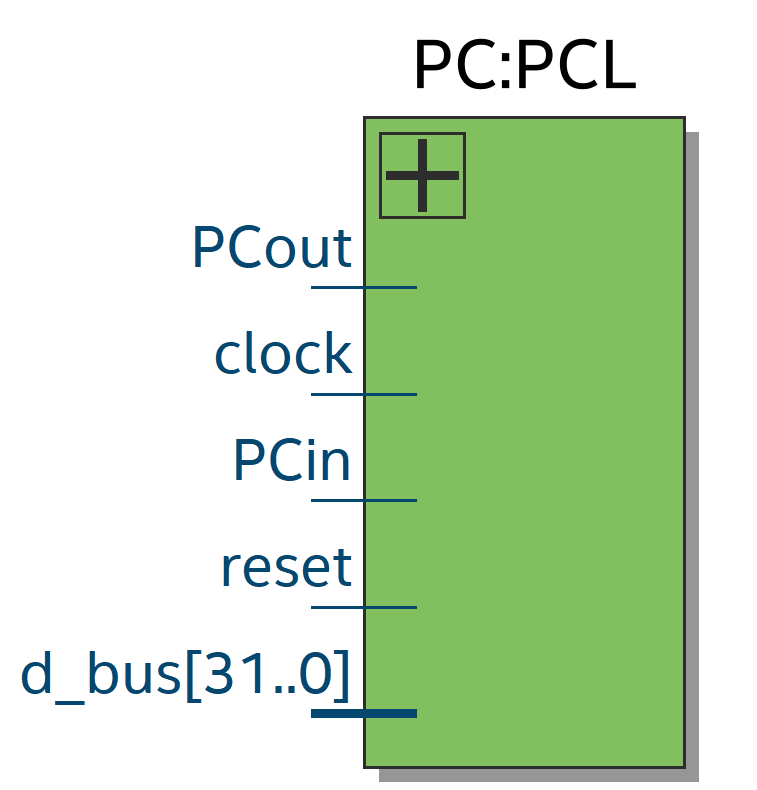
Diagram

Description automatically generated

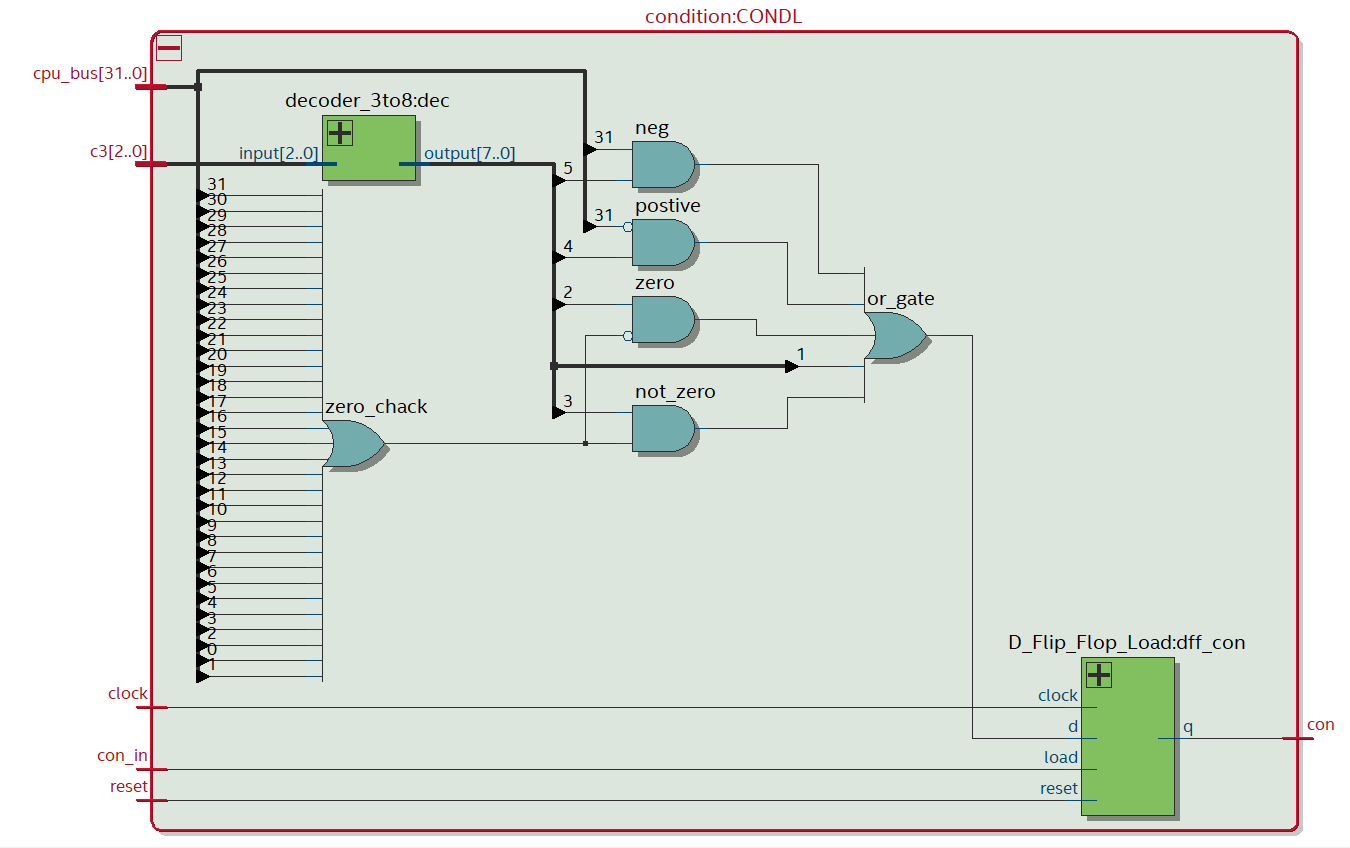
1. ALU RTL:



1. PC RTL:



1. Cond logic RTL:



1. IR RTL:

Diagram

Description automatically generated

1. Memory interface RTL:

Diagram

Description automatically generated

1. Shift counter RTL:

Diagram

Description automatically generated

The RTL for top-entity data\_path.vhdl:

Diagram, schematic

Description automatically generated

The RTL for the SRC processer: cpu.vhdl

Diagram, schematic

Description automatically generated

Then we design a main memory for the system which is consist of 64 location and each location is one-byte: memory.vhdl

Diagram

Description automatically generated

The Top-Level-Entity for all the system is src.vhdl:

Diagram

Description automatically generated

**Results (Simulation of each component):**

* **Simulation for Control Unit:**

Simulation for clocking logic:



The Boolean equation for enable signal:

Enable = Run \* (SDone + Wait + (R+W) )

Simulation for step generator:

**A computer screen capture

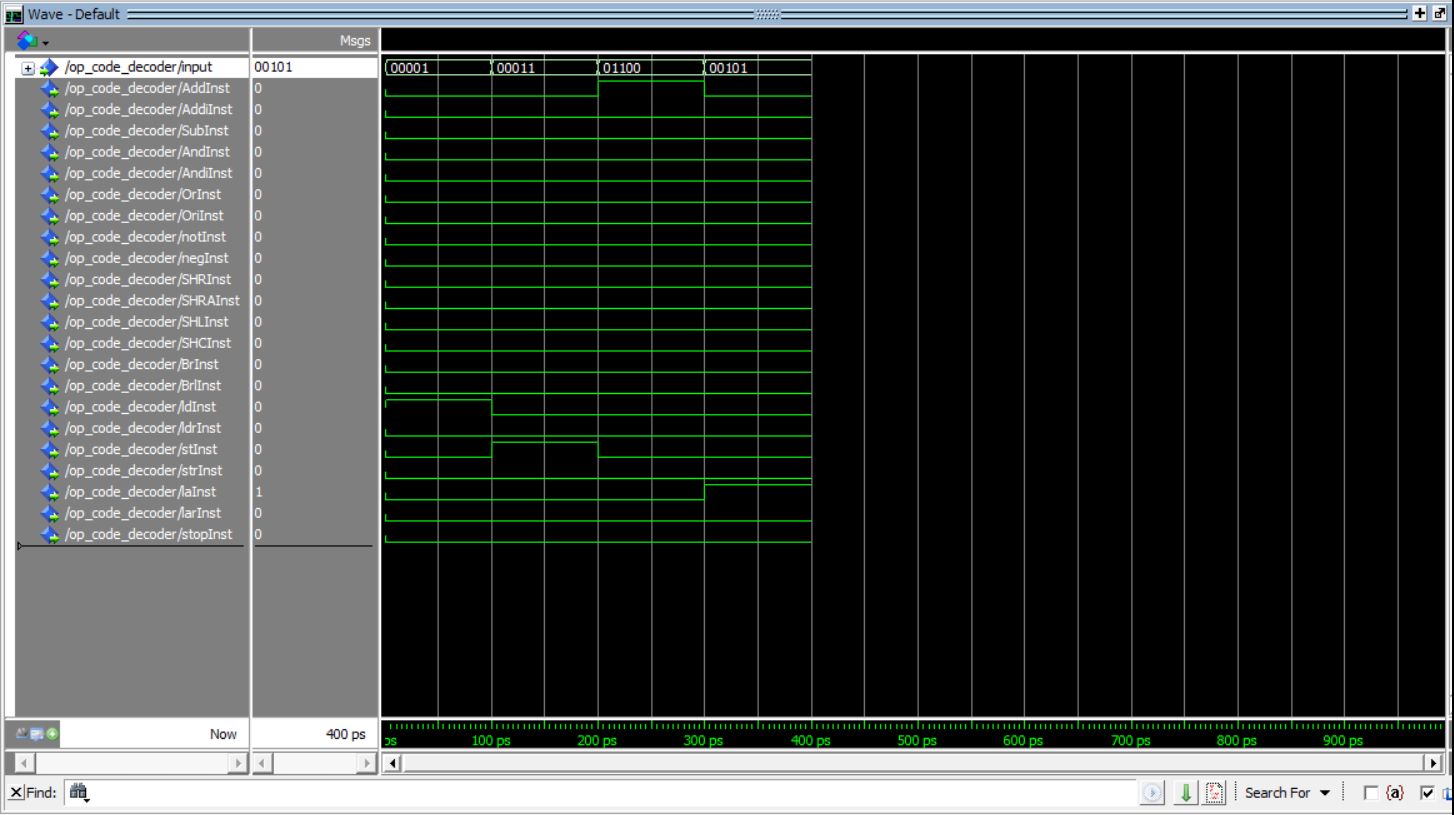
Description automatically generated with low confidence**

If enable is one the counter will start count and generate the steps.

If receive an end\_inst signal will reset the counter.

And if receive a goto6 signal the step generator will generate step 6.

Simulation for opcode decoder:

****

The decoder will set the wanted instruction to one depends on the giving opcode value.

Simulation for control signal encoder:

Background pattern

Description automatically generatedChart, background pattern

Description automatically generated

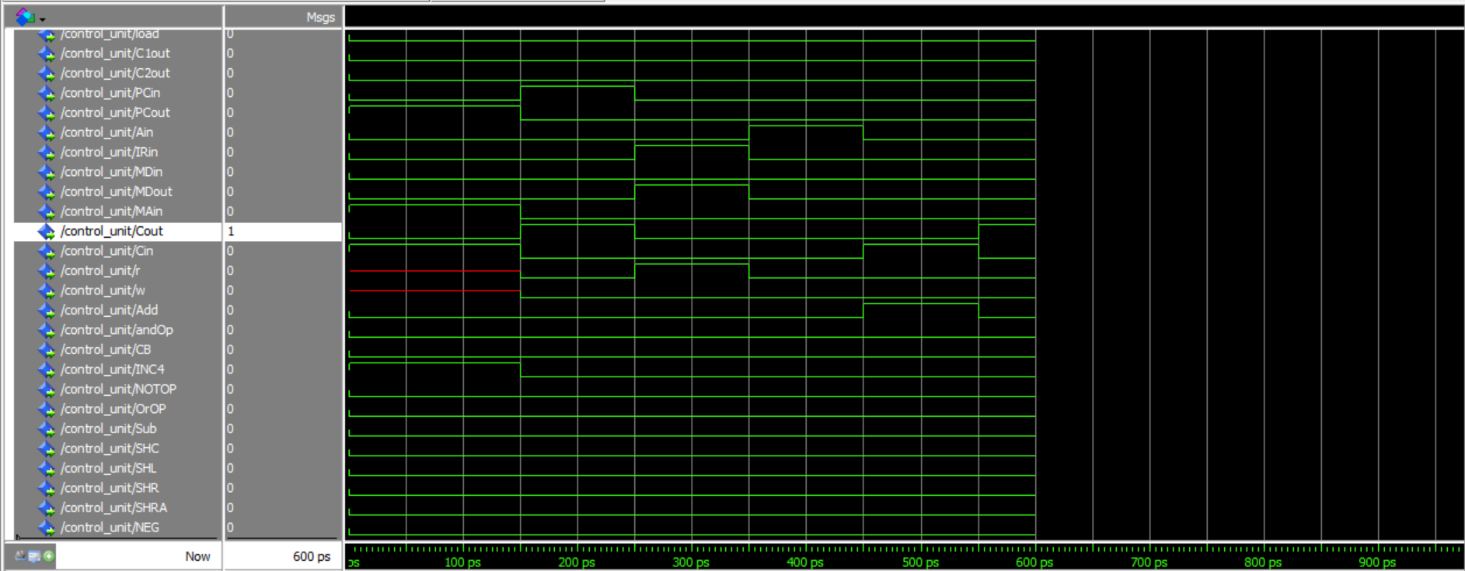
It will generate the signals needed depends on the instruction and the current step.  
on each instruction it will generate specific signals on each step of that instruction.

At the above example it add instruction that has opcode = 01100:

In step 0: signals that are 1: PCout, Cin, Main, INC4,

In step 1: signals that are 1: Read, Wait =, Cout, PCin

Simulation for top entity control unit:

Diagram

Description automatically generated with medium confidence

In each clock will generate the control signals for chosen instruction.

The above example is about add instruction that has opcode = 01100

Table

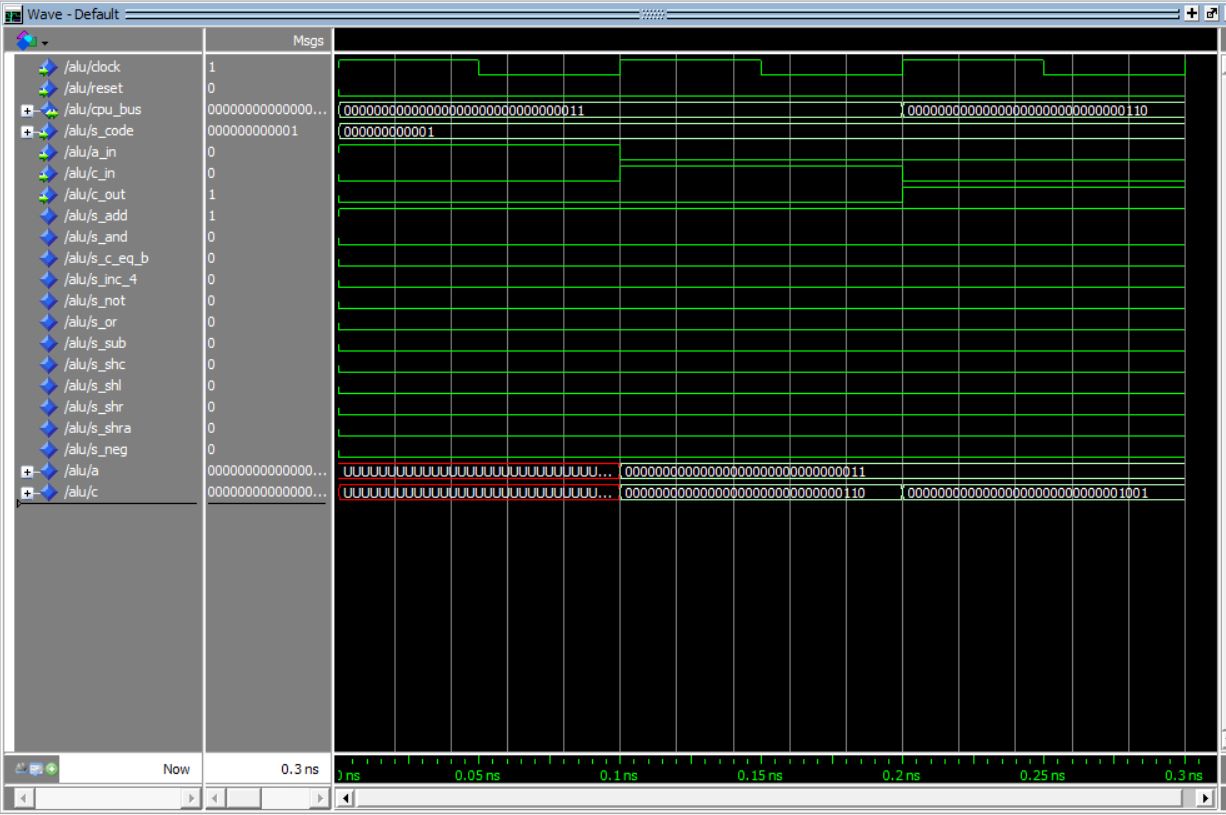
Description automatically generated

**Simulation for Data Path:**

Simulation for ALU:

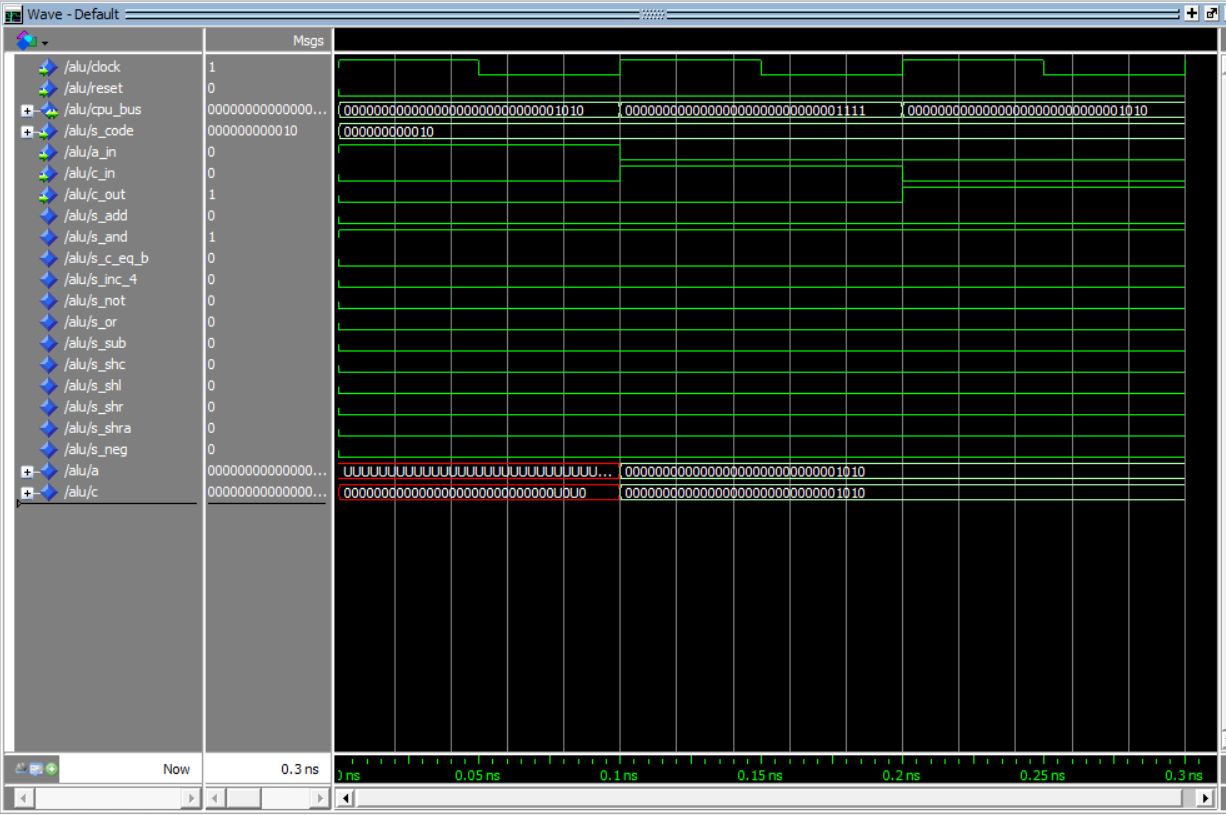
|  |  |
| --- | --- |
| s\_code | Operation in ALU |
| 000000000001 | Add |
| 000000000010 | And |
| 000000000100 | C=B |
| 000000001000 | Inc4 |
| 000000010000 | Not |
| 000000100000 | Or |
| 000001000000 | Sub |
| 000010000000 | Shc |
| 000100000000 | shl |
| 001000000000 | Shr |
| 010000000000 | shra |
| 100000000000 | neg |

If s\_code = “000000000001” will be add operation:



* At first clock we set the value of cpu\_bus <= 3 and a\_in <=’1’ which mean the A register store the value of cpu\_bus.
* At second clock we set c\_in = ‘1’ and the cpu\_bus <= 3 . And the result of adding will store in C register.
* At third clock we set c\_out =’1’ and the result send to cpu\_bus .

The above example is added 3 + 3 = 6



Here the s\_code <= “000000000010” and operator.

1010 and 1111 = 1010

Simulation for PC:

A picture containing chart

Description automatically generated

If the PCin signal is 1 the value of d\_bus will be stored in PC register.

And if the PCout signal 1 then the value of the PC will send to d\_bus (CPU bus).

Simulation for Cond logic:

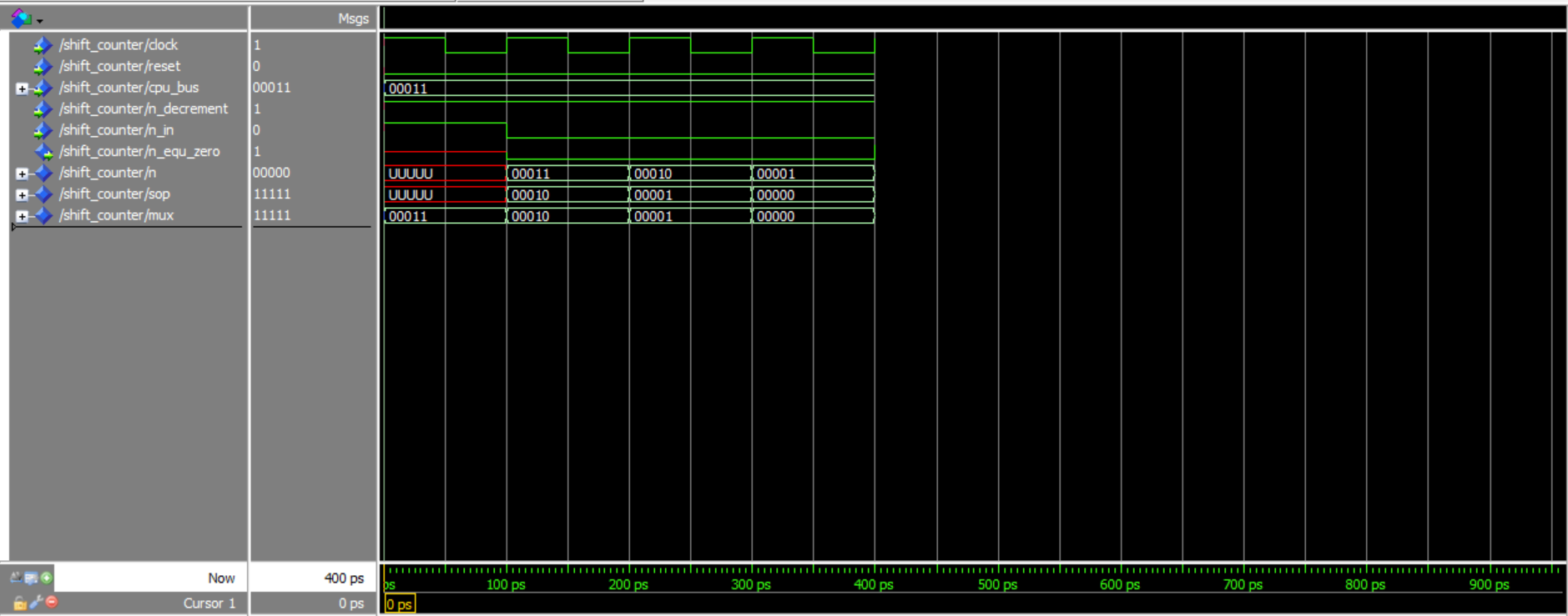
Graphical user interface, timeline

Description automatically generated

The con value will be 1 depends on the value of c3 and R[rc] (cpu\_bus):

|  |  |  |
| --- | --- | --- |
| C3<2…0> | R[rc] | con |
| 000 | X | 0 |
| 001 | X | 1 |
| 010 | R[rc] = 0 {zero} | 1 |
| 011 | R[rc] <> 0 {nonzero} | 1 |
| 100 | R[rc]<31> = 0{positive} | 1 |
| 101 | R[rc]<31> = 1 {negative} | 1 |
| X | X | 0 |

Simulation for Shift counter:



In shift counter will count the number of shifting depends on the cpu\_bus value (5-bits : can shift 32 times)

If the count finish counts down to 0 the n\_equ\_zero will equal ‘1’.

At the end we try to write the following program in memory and start the CPU:

Text

Description automatically generated with low confidence

But we faced a problem in determine the address for first instruction.

**Conclusion:**

At the end of this project, we designed a SRC processor which stands for (simple RISC computer) that consists of 23 instructions, 32 registers of 32-bit size, 32 IR and 32 PC. First, we designed the control unit and went through all its components and connected them. Second, we designed the data path and went through all its components as well. Then, we connected the control unit and data path making the CPU. Last, we designed the memory and connect it to the CPU making our full SRC processor design. Overall, we achieved all the objectives we wanted to achieve but we are not sure that it is fully working.