University of Bahrain

College of Information Technology

Department of Computer Engineering

ITCE364 – Section 01

Final Project

VHDL Design of a 1-Bus Processor with a Hardwired Control Unit

Monitored by: Dr. Jalal Khlaifat

Prepared by:

Weaam Wafeeq Ajoor - ID: 20196537

Puja Karmakar - ID: 20194981

Zainab Redha Matrook - ID: 20195982

Due Date: 24 May 2022

Table of Contents

Objectives

- VHDL code to implement 1-bus processor with a hardwired control unit that have specific characteristic.
- Simulate each component and test their functionality.
- Write a top level code to connect all components as a one system.

Research Plan

Table of Tasks

	Table of Tasks
1	Discuss each component functionality
2	Examine how all the components are connected to each other
3	Identify the RTN for all the instructions
4	Research and Code for ALU
5	Research and Code for Register File
6	Research and Code for Memory
7	Research and Code for Control Unit
8	Research and Code for Data Path
9	Ask the mentor for any further help
10	Debugging Errors for all the components
11	Simulate and test the components functionality
12	Writing the Research Report
13	Preparing the Presentation slides

Tasks Distribution among Members

All the tasks were executed by all of the team members since multiple online meetings on Teams software were arranged to ensure that there is team collaboration.

Timeline

Week 10-16 April 2022

Arrange couple of meetings to discuss components needed for the processor. By the end of the week, the group are need to make sure to know how these components are connected to each other.

Week 12-23 April 2022

Search and gather the RTN required for all the instructions mentioned in the processor specification. Then, the team members will start coding for the control unit, and ALU. The codes will be compiled, errors will be debugged and simulate the components.

Week 24-30 April 2022

The team members will then start coding for the register file and memory. Each code will be compiled, errors will be debugged, and simulate the components.

Week 8-14 May 2022

Throughout this week the team members will focus on the data path component. Throughout this component all the other component will be connected and port mapped. After writing the code, it will be compiled, errors will be debugged, and component will be simulated.

Week 15-23 May 2022

Work on the Research Report.

In addition, the team member will prepare the presentation slides.

Introduction

A common analogy used for describing a computer's processor (CPU) is thinking of the computer like a human body and the CPU is its brain. The term "Processor" refers to the CPU. The CPU is in charge of processing the computer's tasks by constantly resolving mathematical and logical problems. A computer would have at least one CPU which is usually divided into a Control Unit, Data path, buses, and temporary storage. The control unit (CU) is responsible to organize and fetch the CPU instructions and generating "Control Signals" that control the data path and tell the ALU, memory, and input/output devices what to do. Control units can be divided into two types: "Microprogrammed based control unit" and "Hardwired based controlled Unit". Microprogrammed-based control unit treats the relationship between control inputs and outputs as a memory system. Where the memory contains control signals that are stored as "words". During instruction execution at each clock tick, the required "word" is fetched from memory to supply the control signals. Whereas, Hardwired-based controlled Unit treats the relationship between control inputs and outputs as a series of Boolean functions, one for each control signal. Moving to the Data path, it contains CPU registers and ALU. ALU is in charge of performing arithmetic operations and logical functions. While the "bus" is an interconnection through which voltage signals will be flowing and the bus can be used for a different purpose. If the bus carries an address it is known as "Address Bus", if it is carrying any data it is known as "Data Bus", and if it is carrying control signals it is known as "Control signals". There are three types of bus organization for the data bus One-Bus Organization (only one operand can be fetched from this bus), Two-Bus Organization (two operands can be fetched from this bus, one is used for fetching the register, and one for the ALU), and Three-Bus Organization (three operands can be fetched from this bus). Finally, a temporary storage unit such as register files or latches is used to provide easy access to data within the CPU.

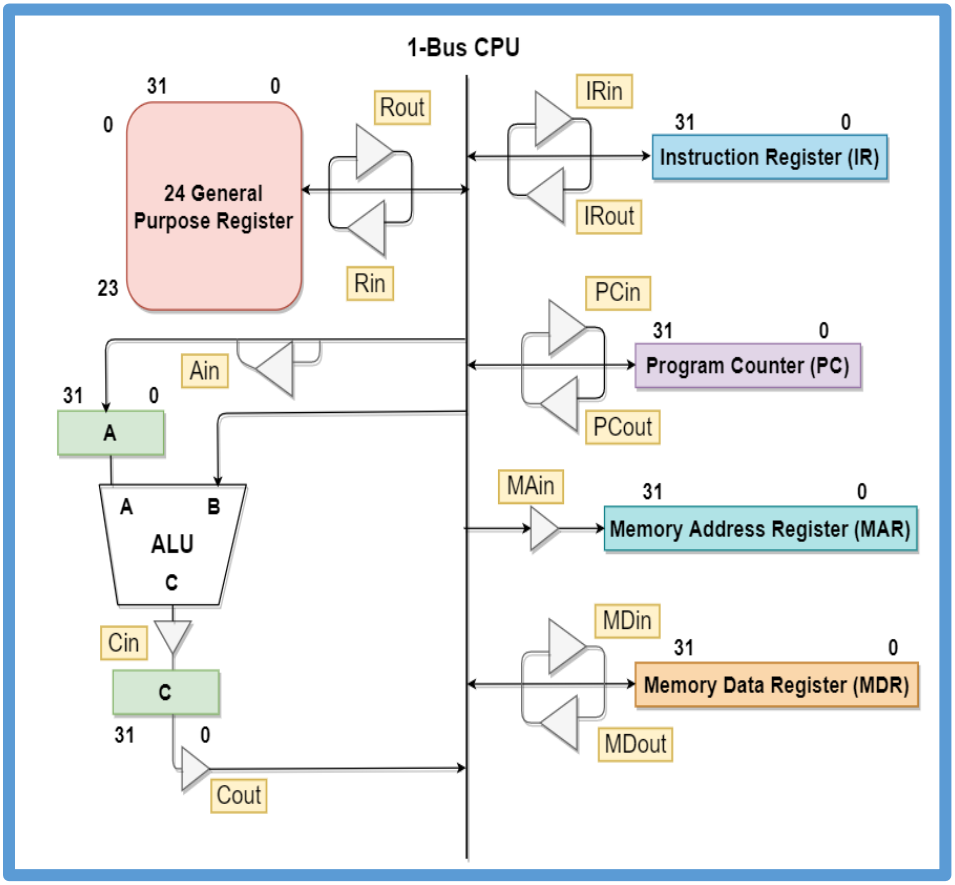
Thus, this project will focus on the implementation of a 1-bus processor with a hardwired control unit and specific characteristics. The report was categorized into three chapters as follows. Chapter 1 illustrates the Processor specifications, design logic description, instructions' format & RTN, and control signals declaration. Chapter 2 emphasizes the code that has been accomplished. Chapter 3 draws the focus on simulation and results.

Processor Description

Processor Specifications & Design logic Description

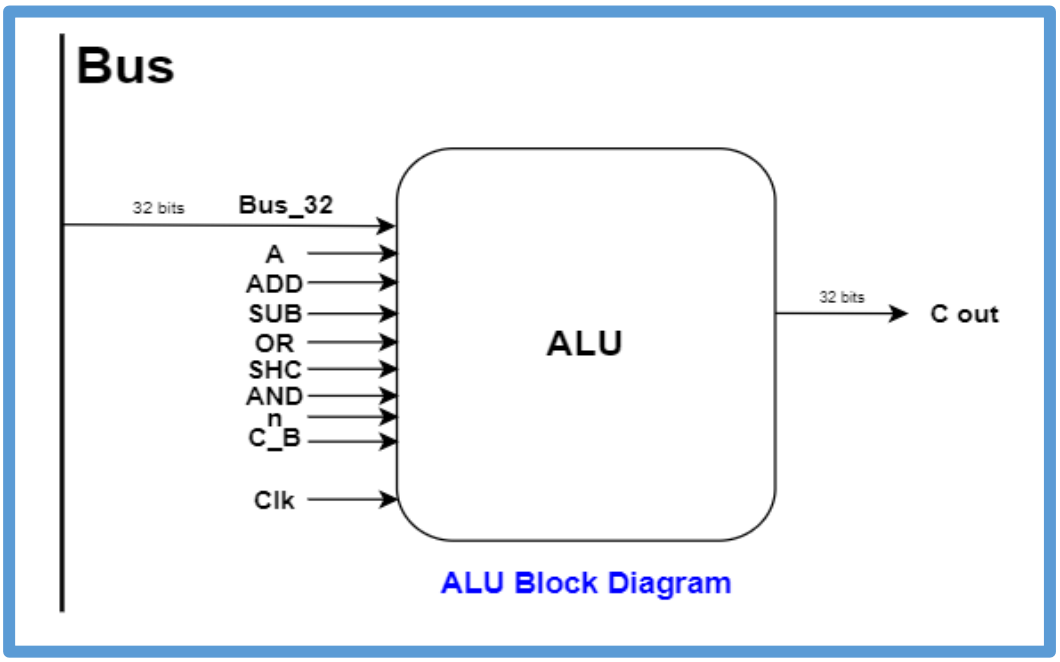
- **1-bus processor with hardwired control unit**
- **32 -bit Data Bus**
- **32 -bit Address Bus**
- **32 -bit Program Counter (PC).**
- **32 -bit Instruction Register (IR).**
- **24 general purpose registers.**
- **SRC has 13 different instructions each with different format. The Instractions are: Instructions: Add, Or, Sub, And, Addi , Ld, St, Lar, Brlpl, Brlmi, Stop, Nop, Shc. (The Shift Circular instruction execute within one clock cycle)**

The following block diagram illustrates the top level view of the processor:

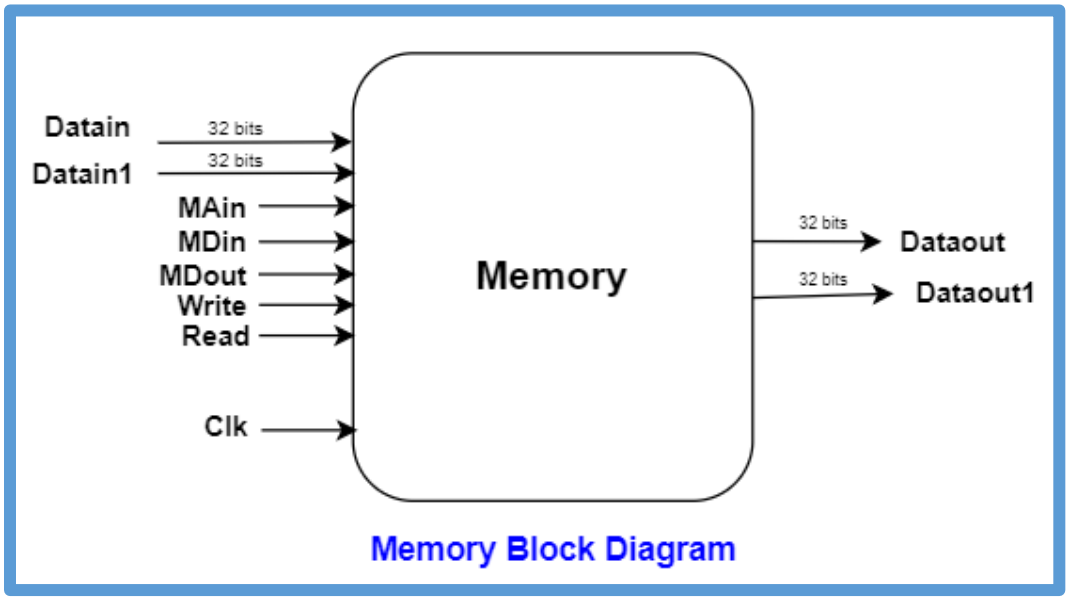


Components Block Diagram

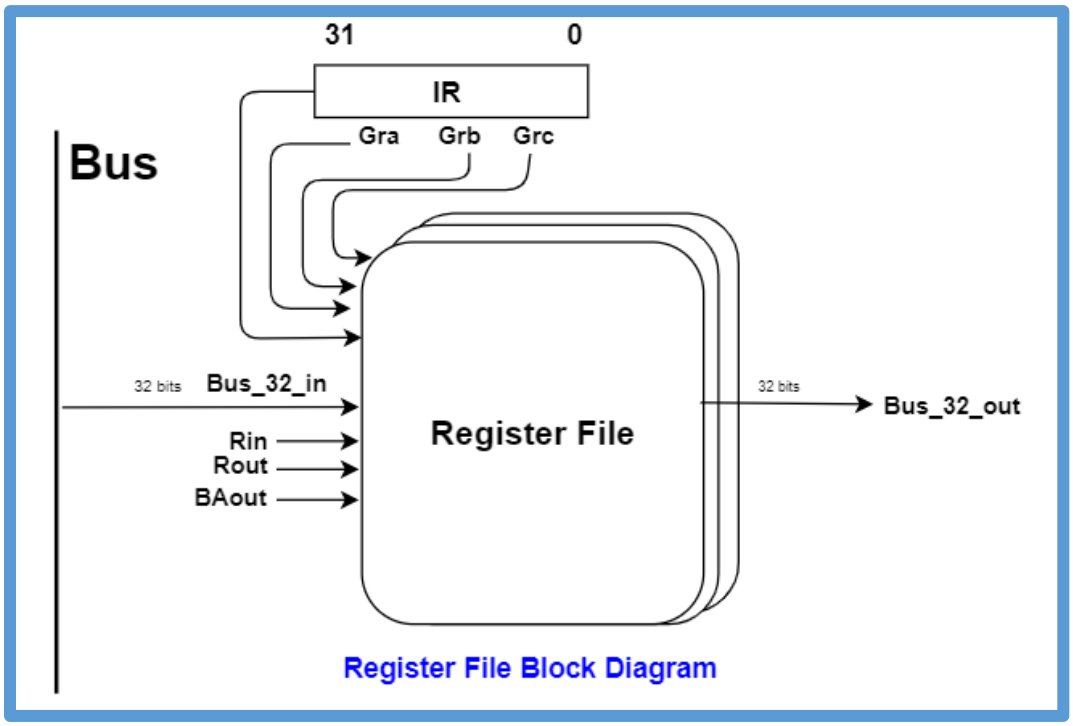
ALU Component



Memory Component

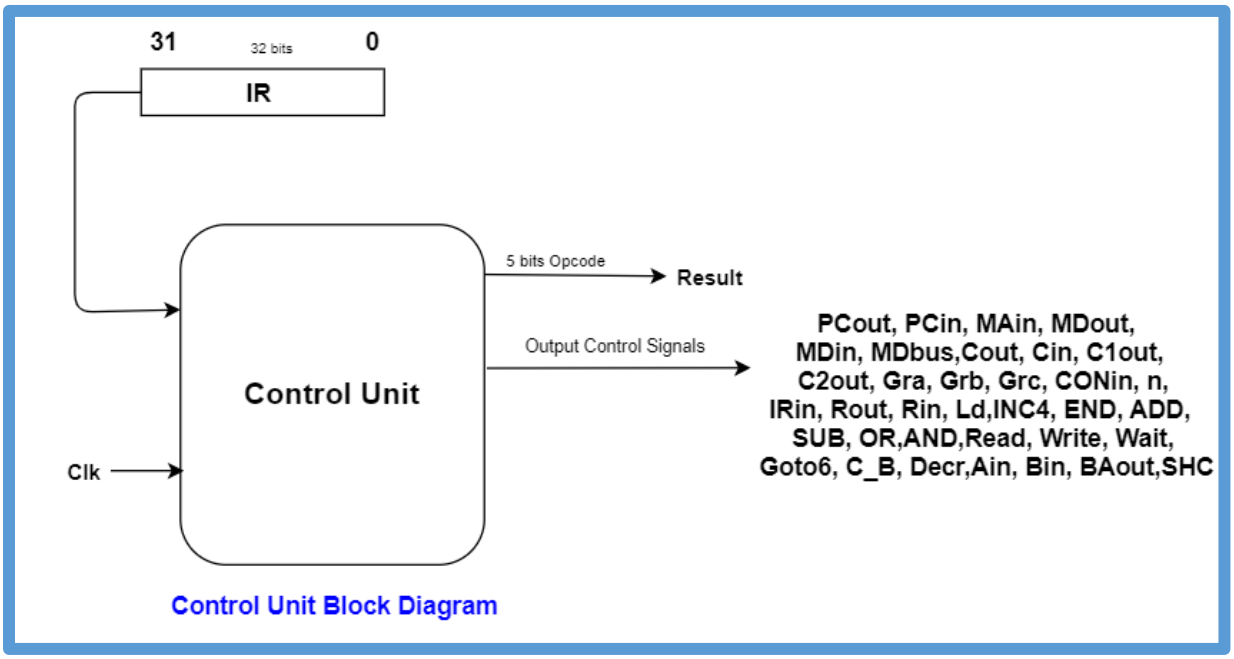


Register File Component



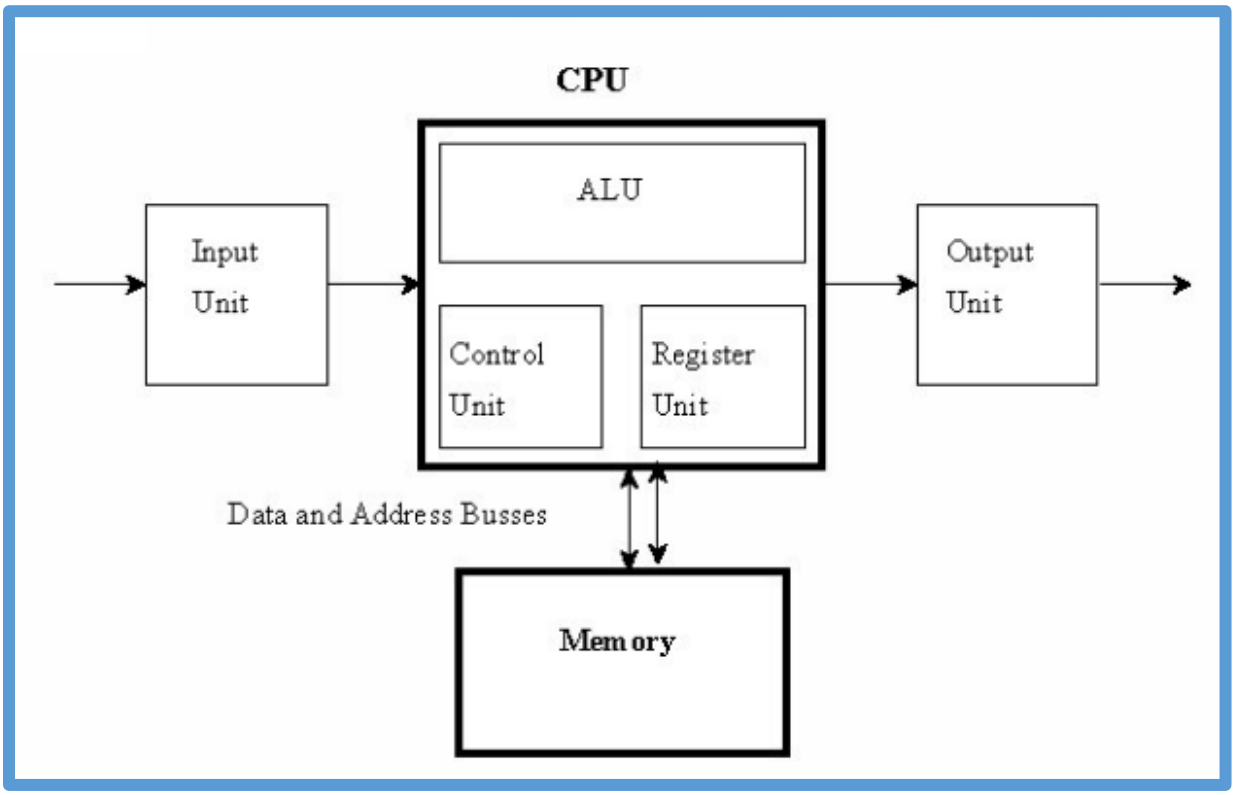
Register File Block Diagram

Control Unit Component



Control Unit Block Diagram

Datapath Component



Instructions' Format

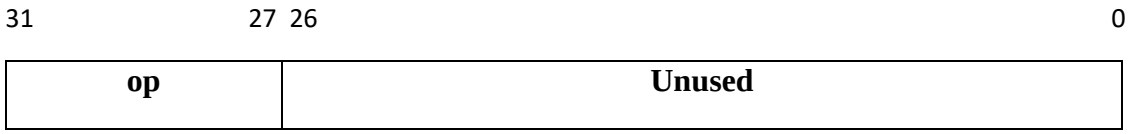
All instructions are 32-bits long, 5-bits are used for opcode field and 27-bits for IR instructions. The following table defines the 13 instructions:

Opcode	Value	Instructions Declaration
Nop	0	Used as placeholder or time waster
Ld	1	Loads value to a specific location in register addressed.
st	3	Store the value of the register to the given memory address
lar	6	Add the PC value to the constant value that will to uploaded to a specific location in register addressed
add	12	Add the value in the register with the given value.
addi	13	Performs an addition on both the source register's contents and the immediate data, and stores the result in the destination register.
sub	14	Subtract the value of the register from the given value
brlpl	18	It checker whether the most significant bit of the value in the register is positive or not. If it is positive then the PC value will be copied to "linkage register" and jump to targeted address.
brlmi	19	It checker whether the most significant bit of the value in the register is negative or not. If it is negative then the PC value will be copied to "linkage register" and jump to targeted address.

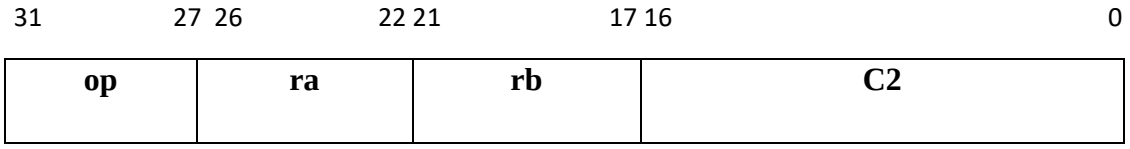
and	20	And-ing the value of the register from the given value.
or	22	Or-ing the value of the register from the given value.
shc	29	Shift the bits of the value found in the register ‘n’ time circular way, from the left till the right and vice versa.
Stop	31	Halt the machine

These instructions will have the following format:

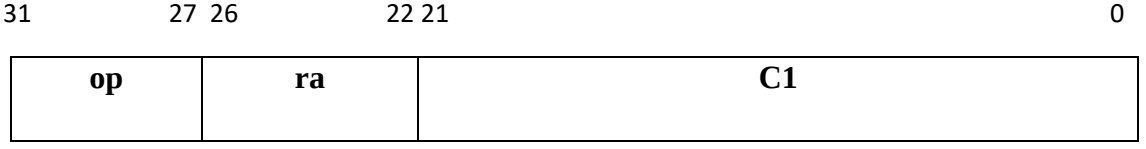
1. **Nop, Stop**



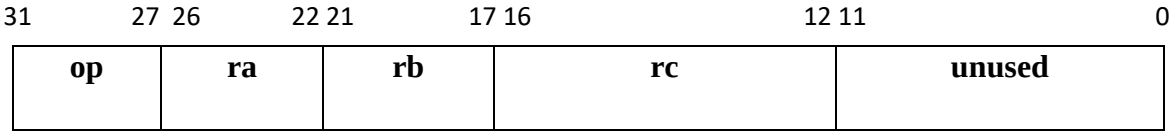
2. **Addi , Ld, St**



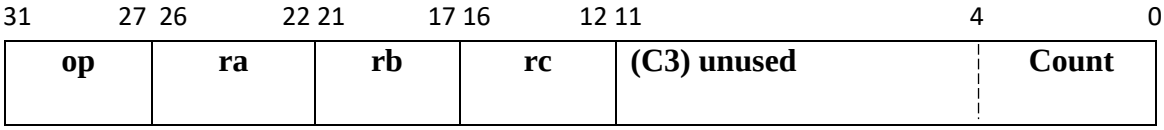
3. **Lar**



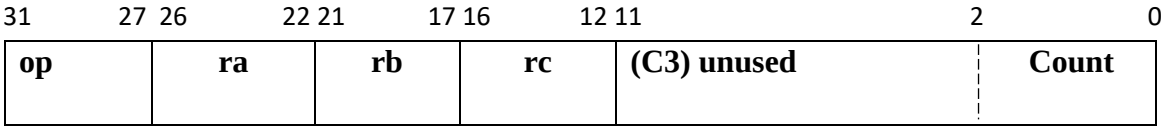
4. Add, Or, Sub, And,



5. Shc



6. Brlpl, Brlmi



Concrete RTN:

1- NOP Instruction (Opcode =0):

Time	Concrete RTN	Control Signals
T0	MA ← PC:C ← PC+4	PC_{out}, MA_{in}, INC4, C_{in}
T1	MD ← M[MA]:PC ← C	Read, Wait, C_{out}, PC_{in},
T2	IR ← MD	MD_{out}, IR_{in}

2- Ld Instruction (Opcode =1):

Time	Concrete RTN	Control Signals
T0	MA ← PC:C ← PC+4	PC_{out}, MA_{in}, INC4, C_{in}
T1	MD ← M[MA]:PC ← C	Read, Wait, C_{out}, PC_{in},
T2	IR ← MD	MD_{out}, IR_{in}
T3	(rb ≠0) → (A ← 0) (rb ≠0) ← (A ← R[rb])	$G_{rb,}$ BA_{out} , A_{in}
T4	C ← A + C_2 {sign Ext}	C_{2out}, Add,C_{in}
T5	MA ← C	C_{out} , MA_{in}
T6	MD ← M[MA]	Read, Wait
T7	R[ra] ← MD	MD_{out}, G_{rb} ,R_{in}, END

3- St Instruction (Opcode =3):

Time	Concrete RTN	Control Signals
T0	MA ← PC:C ← PC+4	PC_{out}, MA_{in}, INC4, C_{in}
T1	MD ← M[MA]:PC ← C	Read, Wait, C_{out}, PC_{in},
T2	IR ← MD	MD_{out}, IR_{in}
T3	(rb ≠0) → (A ← 0) (rb ≠0) ← (A ← R[rb])	G_{rb}, BA_{out} , A_{in}
T4	C ← A + C_2 {sign Ext}	C_{2out}, Add,C_{in}
T5	MA ← C	C_{out} , MA_{in}
T6	MD ← R[ra]	G_{ra}, R_{out} , MD_{bus}
T7	M[MA] ← MD	Wait, Wait, END.

4- Lar Instruction (Opcode =6):

Time	Concrete RTN	Control Signals
T0	MA ← PC:C ← PC+4	PC_{out}, MA_{in}, INC4, C_{in}
T1	MD ← M[MA]:PC ← C	Read, Wait, C_{out}, PC_{in},
T2	IR ← MD	MD_{out}, IR_{in}
T3	A ← PC	PC_{out}, A_{in}
T4	C ← A + C_1{sign Ext}	C_{1out}, ADD, C_{in}
T5	R[ra] ← C	C_{out} , G_{ra}, R_{in}, END.

5- Add Instruction (Opcode =12):

Time	Concrete RTN	Control Signals
T0	MA ← PC:C ← PC+4	PC_{out}, MA_{in}, INC4, C_{in}
T1	MD ← M[MA]:PC ← C	Read, Wait, C_{out}, PC_{in},
T2	IR ← MD	MD_{out}, IR_{in}
T3	A ← R[rb]	G_{rb}, R_{out}, A_{in}
T4	C ← A + R[rc]	G_{rc}, R_{out}, ADD, C_{in}
T5	R[ra] ← C	C_{out} , G_{ra}, R_{in}, END.

6- Addi Instruction (Opcode =13):

Time	Concrete RTN	Control Signals
T0	MA←PC:C ← PC+4	PC_{out}, MA_{in}, INC4, C_{in}
T1	MD ←M[MA]:PC←C	Read, Wait, C_{out}, PC_{in},
T2	IR ← MD	MD_{out}, IR_{in}
T3	A ← R[rb]	G_{rb}, R_{out}, A_{in}
T4	C← A + R[rc]	C_{2out}, ADD, C_{in}
T5	R[ra] ←C	C_{out} , G_{ra}, R_{in}, END.

7- Sub Instruction (Opcode =14):

Time	Concrete RTN	Control Signals
T0	MA←PC:C ← PC+4	PC_{out}, MA_{in}, INC4, C_{in}
T1	MD ←M[MA]:PC←C	Read, Wait, C_{out}, PC_{in},
T2	IR ← MD	MD_{out}, IR_{in}
T3	A ← R[rb]	G_{rb}, R_{out}, A_{in}
T4	C← A + R[rc]	G_{rc}, R_{out}, SUB, C_{in}
T5	R[ra] ←C	C_{out} , G_{ra}, R_{in}, END.

8- Brlpl & Brlpl Instruction (Opcode =18 & 19):

Time	Concrete RTN	Control Signals
T0	MA←PC:C ← PC+4	PC_{out}, MA_{in}, INC4, C_{in}
T1	MD ←M[MA]:PC←C	Read, Wait, C_{out}, PC_{in},
T2	IR ← MD	MD_{out}, IR_{in}
T3	A ← PC	G_{rb}, R_{out}, A_{in}
T4	C← A + C_1{sign Ext}	C_{1out}, ADD, C_{in}
T5	R[ra] ←C	C_{out} , G_{ra}, R_{in}, END.

9- And Instruction (Opcode =20):

Time	Concrete RTN	Control Signals
T0	MA← PC:C ← PC+4	PC_{out}, MA_{in}, INC4, C_{in}
T1	MD ← M[MA]:PC← C	Read, Wait, C_{out}, PC_{in},
T2	IR ← MD	MD_{out}, IR_{in}
T3	A ← R[rb]	G_{rb}, R_{out}, A_{in}
T4	C← A Λ R[rc]	Grc, Rout, AND, Cin
T5	R[ra] ← C	C_{out} , G_{ra}, R_{in}, END.

10- OR Instruction (Opcode =22):

Time	Concrete RTN	Control Signals
T0	MA← PC:C ← PC+4	PC_{out}, MA_{in}, INC4, C_{in}
T1	MD ← M[MA]:PC← C	Read, Wait, C_{out}, PC_{in},
T2	IR ← MD	MD_{out}, IR_{in}
T3	A ← R[rb]	G_{rb}, R_{out}, A_{in}
T4	C← A ∨ R[rc]	Grc, Rout, OR, Cin
T5	R[ra] ← C	C_{out} , G_{ra}, R_{in}, END.

11- SHC Instruction (Opcode =29):

Time	Concrete RTN	Control Signals
T0	MA← PC:C ← PC+4	PC_{out}, MA_{in}, INC4, C_{in}
T1	MD ← M[MA]:PC← C	Read, Wait, C_{out}, PC_{in},
T2	IR ← MD	MD_{out}, IR_{in}
T3	n ⟶ IR <4..0>	C1out, Ld
T4	(n=0) ⟶ (n ← R[rc] <4..0>)	(n=0) ⟶ (Grc, Rout,Ld)
T5	C← R[rc]	Grb, Rout, C=B, Cin
T6	Shc (:= (n≠0) ⟶ (C← C<30..0>#C<31> : n ← n-1; Shc));	(n≠0) ⟶ (Cout , SHC, Cin, Decr, Goto6)
T7	R[ra] ←C	Cout , Gra, Rin, END.

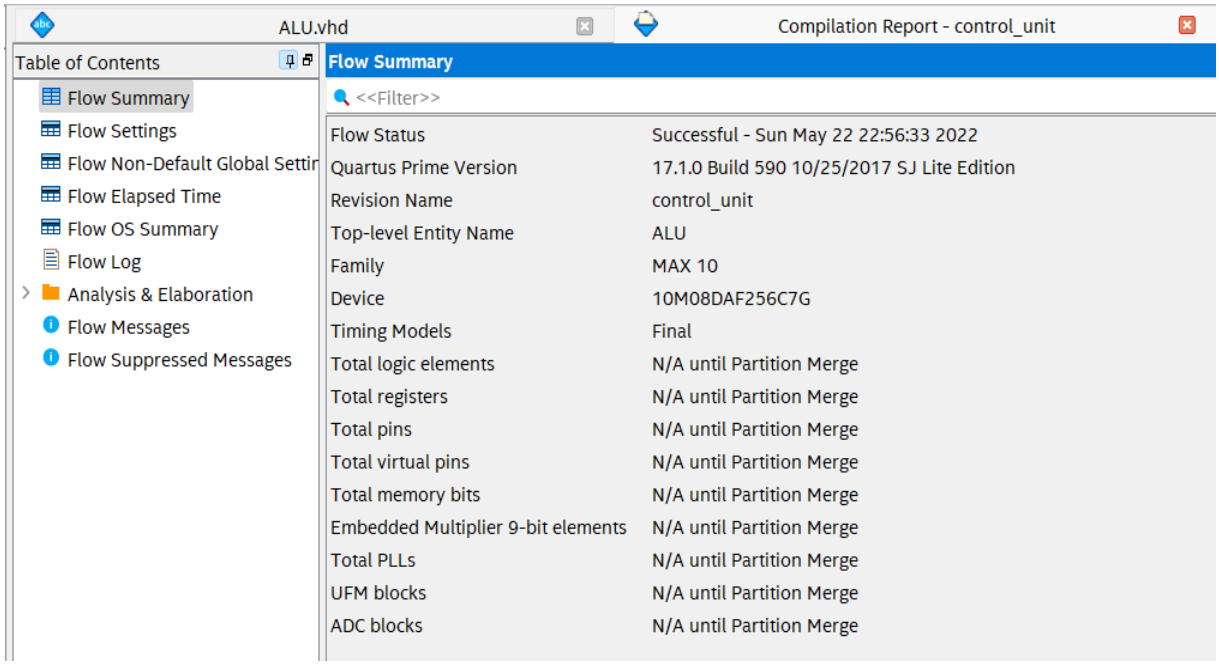
Implementation

This section will cover the code accomplished for each and every component.

ALU Unit Component

```
Library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_1164.all;

entity ALU is
    port
    (
                clk: in   std_logic;
            Bus_32: in   std_logic_vector (31 downto 0);
                 A: in   std_logic_vector (31 downto 0);
    n, ADD, SUB, AND_sig, OR_sig, SHC, C_B, INC4: in std_logic;
                 C: out std_logic_vector (31 downto 0)
    );

end ALU;

architecture Behavioral of ALU is
Signal C_sig: std_logic_vector(31 downto 0);
Begin

Process (clk, n, ADD, SUB, AND_sig, OR_sig, SHC, C_B, INC4)
Begin
        if (clk'event AND clk = '1') then
            if (ADD='1') then
               C_sig<=Bus_32 + A;

            elsif (SUB='1') then
               C_sig<=Bus_32 - A;

            elsif (INC4='1') then
               C_sig<=Bus_32 + 4;

            elsif (AND_sig='1') then
               C_sig<=A and Bus_32;

            elsif (OR_sig='1') then
               C_sig<=A or Bus_32;

            elsif (SHC='1') then
            if (n='0') then --no shift circular is operated when n=0
                C<= C_sig;
            else
                 C_sig<= (C_sig(30 downto 0)& C_sig(31));
            end if;

            elsif (C_B='1') then
                 C_sig<= Bus_32;

         end if;
         end if;
         C<= C_sig;

end process;
end behavioral;
```

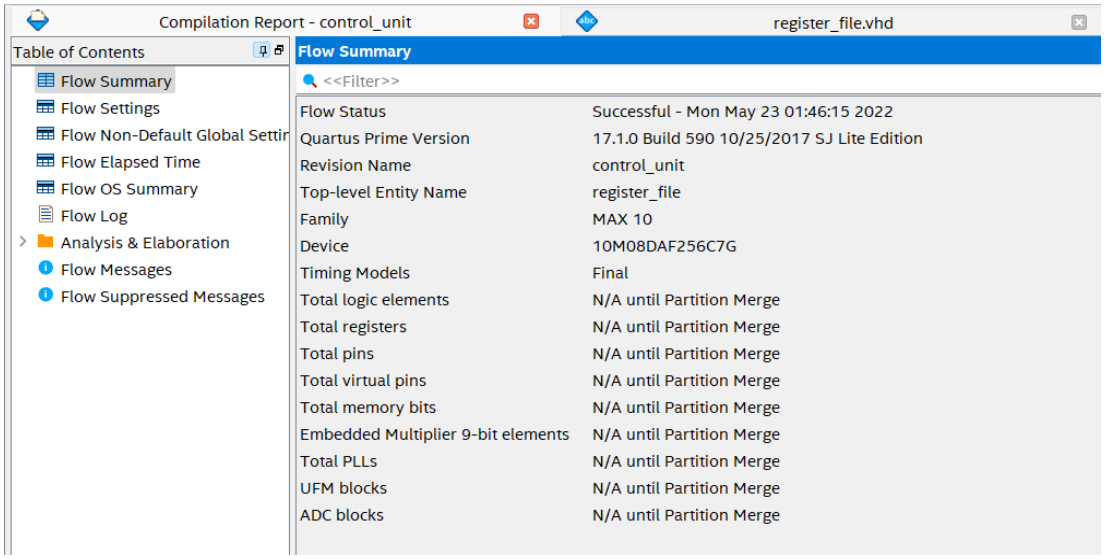
Successful Compilation of ALU Component:



Register File Component

```
1
2   library IEEE;
3   use IEEE.STD_LOGIC_1164.ALL;
4   USE ieee.numeric_std.ALL;
5
6   entity register_file is
7       Port (
8             Bus_32_in :in  STD_LOGIC_VECTOR (31 downto 0):= x"00000000";
9             Rin : in  STD_LOGIC;
10            Rout : in  STD_LOGIC;
11            BAout : in  STD_LOGIC;
12            Gra : in  STD_LOGIC;
13            Grb : in  STD_LOGIC;
14            Grc : in  STD_LOGIC;
15            IR : in  STD_LOGIC_VECTOR (31 downto 0):=x"00000000";
16            Bus_32_out : out  STD_LOGIC_VECTOR (31 downto 0)
17           );
18  end register_file;
19
20  architecture Behavioral of register_file is
21  signal reg : std_logic_vector (4 downto 0);
22  type reg_array is array(0 to 23) of std_logic_vector(31 downto 0);
23  signal myreg : reg_array:=(x"00000000", x"00000001", x"00000002", x"00000003", x"00000004" , x"00000005", x"00000006",
24  x"00000007", x"00000008", x"00000009", x"00000010", x"00000011" , x"00000012", x"00000013",x"00000014", x"00000015",
25  x"00000016", x"00000017", x"00000018" , x"00000019", x"00000020" ,x"00000021", x"00000022", x"00000023");
26
27  begin
28
29  Bus_32_out <= x"00000000" when (BAout = '1' and Grb = '1' and IR(21 downto 17) ="00000")else
30  myreg(to_integer(unsigned(IR(21 downto 17)))) when (BAout = '1' and Grb = '1' and IR(21 downto 17) /="00000") else
31  myreg(to_integer(unsigned(IR(26 downto 22)))) When (Rout ='1' and Gra = '1') else
32  myreg(to_integer(unsigned(IR(21 downto 17)))) When (Rout ='1' and Grb = '1') else
33  myreg(to_integer(unsigned(IR(16 downto 12)))) When (Rout ='1' and Grc = '1');
34
35
36  Reg <= IR(26 downto 22) when Gra ='1' else
37         IR(21 downto 17) when Grb ='1' else
38         IR(16 downto 12) when Grc ='1' else
39         IR(26 downto 22); -- default value.
40
41  myreg(to_integer(unsigned(Reg))) <= Bus_32_in when  (Rin ='1');
42
43
44  end Behavioral;
45
```

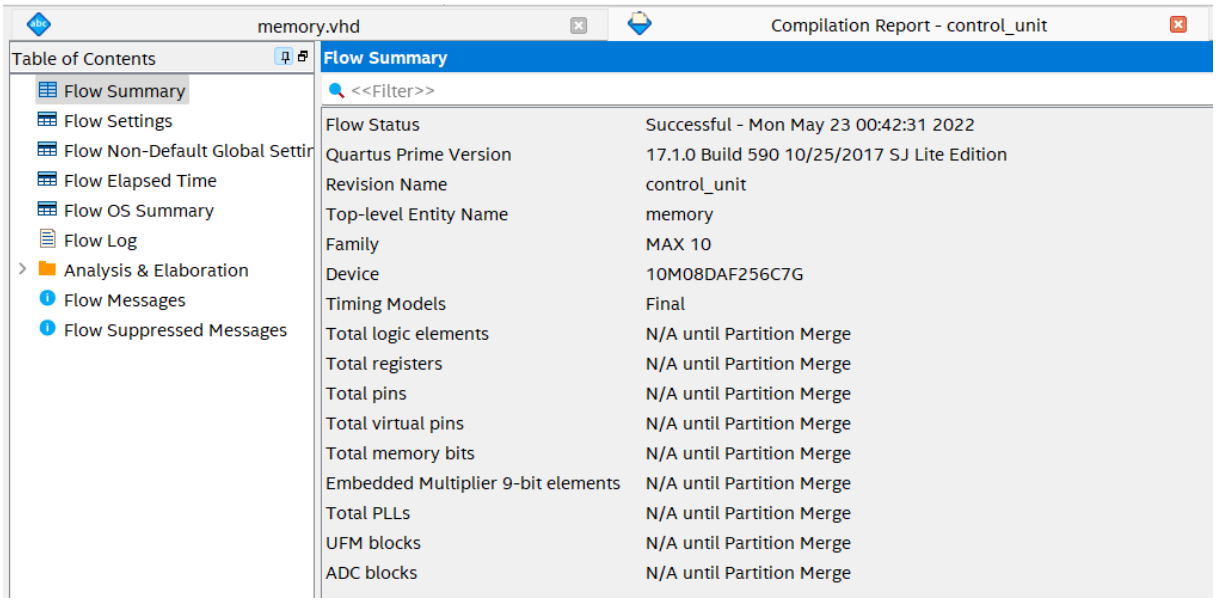
Successful Compilation of Register File Component:



Memory Unit Component

```
1
2   Library ieee;
3   use ieee.std_logic_unsigned.all;
4   use ieee.std_logic_1164.all;
5   use ieee.std_logic_arith.all;
6
7
8   entity memory is
9      port
10     (clk: in  std_logic;
11     MAin: in  std_logic;
12     MDin, MDout, write_sig, read_sig:in  std_logic;
13     datain:   in  std_logic_vector(31 downto 0); --input of MA
14     datain1:  in  std_logic_vector(31 downto 0);
15     dataout:  out std_logic_vector(31 downto 0); --output of MA
16     dataout1: out std_logic_vector(31 downto 0)
17  );
18  end memory;
19
20
21  architecture Behavioral of memory is
22
23  signal data : std_logic_vector (31 downto 0);
24
25  Begin
26  Process (write_sig, read_sig, MDout, clk, MAin, MDin)
27  Begin
28  if (clk'event AND clk = '1') then
29  if (MAin='1') then
30  Data <= datain;
31  elsif (MDin='1') then
32  data <= datain1;
33  elsif (MDout = '1') then
34  dataout1 <= data;
35  End if;
36
37  If (write_sig = '1') then
38  data <= datain;
39  Elsif (read_sig = '1') then
40  dataout <= data;
41  End if;
42  end if;
43  End process;
44  End Behavioral;
45
46
47
```

Successful Compilation of Memory Component:

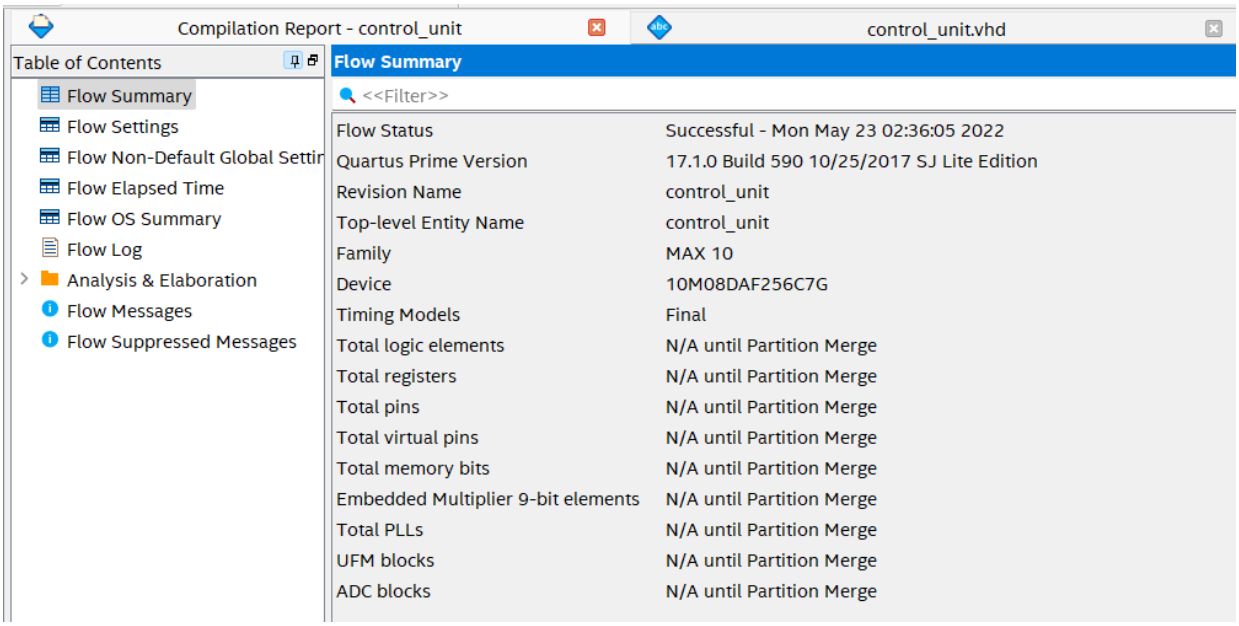


Control Unit Component

```
1
2   Library ieee;
3   use ieee.std_logic_1164.all;
4   use ieee.std_logic_arith.all;
5   use ieee.std_logic_unsigned.all;
6   use ieee.std_logic_1164.all;
7
8   entity control_unit is
9       port
10      (
11      clk: in   std_logic;
12      IR:  in   std_logic_vector (31 downto 0);
13      PCout, PCin, MAin, MDout,  MDin, MDbus:  out  std_logic;
14      Cout, Cin, C1out, C2out, Gra, Grb, Grc, CONin, n:  out  std_logic;
15      IRin, Rout, Rin, Ld:  out  std_logic;
16      INC4, END_sig, ADD, SUB, OR_sig:  out  std_logic;
17      Read_sig, Write_sig, Wait_sig, Goto6, C_B, Decr: out std_logic;
18      Ain, Bin, BAout, AND_sig, SHC:  out  std_logic;
19      Result: out std_logic_vector (4 downto 0)
20      );
21
22  end control_unit;
23
24
25  architecture Behavioral of control_unit is
26       Signal Op: std_logic_vector (4 downto 0);
27       Signal T: integer range 0 to 7 := 0; --steps
28
29  Begin
30      Op <= IR(31 downto 27);
31
32      process(clk,Op)
33      Variable x: integer range 0 to 31; --instructions
34
35
36      Begin
37
38    --Add, Or, Sub, And, Addi , Ld, St, Lar, Brlpl, Brlmi, Stop, Nop, Shc.
39
```

```
40          Case Op is
41            When "00000" => x:= 0; --nop
42            When "00001" => x:= 1; --ld
43            When "00011" => x:= 3; --st
44            When "00110" => x:= 6; --lar
45            When "01100" => x:= 12;--add
46            When "01101" => x:= 13;--addi
47            When "01110" => x:= 14;--sub
48            When "10010" => x:= 18;--brlpl
49            When "10011" => x:= 19;--brlmi
50            When "10100" => x:= 20;--and
51            When "10110" => x:= 22;--or
52            When "11101" => x:= 29;--shc
53            When "11111" => x:= 31;--stop
54            When others => null;
55            end case;
56
57
58  --Output control signals:
59
60
61      if (clk'event AND clk = '1') then
62
63      PCout<='0'; PCin<='0'; MAin<='0'; MDout<='0';  MDin<='0';
64      MDbus<='0'; Cout<='0'; Cin<='0'; C1out<='0'; C2out<='0';
65      Gra<='0'; Grb<='0'; Grc<='0'; CONin<='0'; n<='0';
66      Rout<='0'; IRin<='0'; Rout<='0'; Rin<='0'; Ld<='0';
67      INC4<='0'; END_sig<='0'; ADD<='0'; SUB<='0'; OR_sig<='0';
68      Read_sig<='0'; Write_sig<='0'; Wait_sig<='0'; Goto6<='0'; Ain<='0'; Decr <='0';
69      Bin<='0'; BAout<='0'; OR_sig<='0'; AND_sig<='0'; SHC<='0'; C_B<='0';
70
71
72      if (T = 0 ) then --instruction fetch step 1
73      if (x=0 or x=1 or x=3 or x=6 or x=12 or x=13 or x=14 or x=18 or x=19 or x=20 or x=22 or x=29 or x=31) then
74      PCout<='1'; MAin<='1'; INC4<='1'; Cin<='1';
75      end if;
76
77
78      elsif (T = 1) then --instruction fetch step 2
79      if (x=0 or x=1 or x=3 or x=6 or x=12 or x=13 or x=14 or x=18 or x=19 or x=20 or x=22 or x=29 or x=31) then
80      Read_sig<='1'; Wait_sig<='1'; Cout<='1'; PCin<='1';
81      end if;
82
83
84      elsif (T = 2) then--instruction fetch step 3
85      if (x=0 or x=1 or x=3 or x=6 or x=12 or x=13 or x=14 or x=18 or x=19 or x=20 or x=22 or x=29 or x=31) then
86      MDout<='1'; IRin<='1';
87      end if;
88
89
90       elsif (T = 3) then
91       If (x=1 or x=3) then                    Grb<='1'; BAout<='1'; Ain<='1';
92       elsIf (x=6) then                        PCout<='1'; Ain<='1';
93       elsif (x=29) then                       C1out<='1'; Ld<='1';
94       elsIf (x=18 or x=19) then               Grc<='1'; Rout<='1'; CONin<='1';
95       elsif (x=12 or x=13 or x=14 or x=20 or x=22) then Grb<='1'; Rout<='1'; Ain<='1';
96       else END_sig<='1';
97       end if;
98
99       elsif (T = 4) then
100      If (x=1 or x=3 or x=13) then C2out<='1'; ADD<='1'; Cin<='1';
101      elsIf (x=6) then            C1out<='1'; ADD<='1'; Cin<='1';
102      elsIf (x=12 or x=20) then   Grc<='1'; Rout<='1'; ADD<='1'; Cin<='1';
103      elsIf (x=14) then           Grc<='1'; Rout<='1'; SUB<='1'; Cin<='1';
104      elsIf (x=18 or x=19) then   PCout<='1'; Gra<='1'; Rin<='1';
105      elsIf (x=22) then           Grc<='1'; Rout<='1'; OR_sig<='1'; Cin<='1';
106      elsif (x=29) then           n<='1'; Grc<='1'; Rout<='1'; Ld<='1';
107      else END_sig<='1';
108      end if;
109
110      elsif (T = 5) then
111      If (x=1 or x=3) then                              Cout<='1'; MAin<='1';
112      elsIf (x=6 or x=12 or x=13 or x=14 or x=20 or x=22) then  Cout<='1'; Gra<='1'; Rin<='1'; END_sig<='1';
113      elsif (x=29) then                                 Grb<='1'; Rout<='1'; C_B<='1'; Cin<='1';
114      else END_sig<='1';
115      end if;
116
117
118      elsif (T = 6) then
119      If (x=1) then      Read_sig<='1'; Wait_sig<='1';
120      elsif (x=3) then   Gra<='1';  Rout<='1'; MDbus<='1';
121      elsif (x=29) then  n<='1'; Cout<='1'; SHC<='1'; Cin<='1'; Decr<='1'; Goto6<='1';
122      else END_sig<='1';
123      end if;
124
125
126      elsif (t = 7) then
127      If (x=1) then      MDout<='1'; Gra<='1'; Rin<='1'; END_sig<='1';
128      elsIf (x=3) then   Write_sig<='1'; Wait_sig<='1'; END_sig<='1';
129      elsif (x=29) then  Cout<='1'; Gra<='1'; Rin<='1'; END_sig<='1';
130      else END_sig<='1';
131      end if;
132
133      T <= T+1;
134      end if;
135      end if;
136      Result<=Op;
137  end process;
138  end behavioral;
```

Successful Compilation of Control Unit Component:

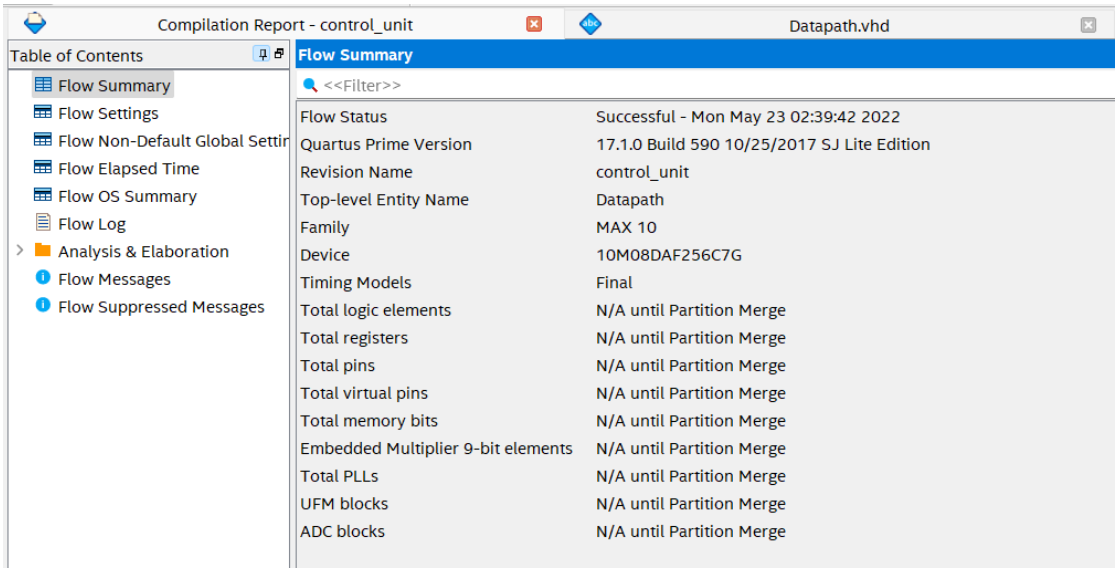


Data Path Component

```
1
2   Library ieee;
3   use ieee.std_logic_unsigned.all;
4   use ieee.std_logic_1164.all;
5   use ieee.std_logic_arith.all;
6
7   entity Datapath is
8       port(
9       clk_datapath: in std_logic;
10      PCout, PCin, MAin, MDout,  MDin, MDbus:  in std_logic;
11      Cout, Cin, C1out, C2out, Gra, Grb, Grc, CONin, n:  in std_logic;
12      IRin, Rout, Rin, Ld:  in  std_logic;
13      INC4, END_sig, ADD, SUB, OR_sig:  in  std_logic;
14      Read_sig, Write_sig, C_B, Decr, Goto6: in std_logic;
15      Ain, Bin, Wait_sig,BAout, AND_sig, SHC:  in  std_logic;
16      Opcode: out std_logic_vector(4 downto 0));
17  end Datapath;
18
19
20  architecture Behavioral of Datapath is
21
22  COMPONENT ALU is
23      port
24      (
25    clk: in   std_logic;
26    Bus_32: in   std_logic_vector (31 downto 0);
27      A: in   std_logic_vector (31 downto 0);
28      n, ADD, SUB, AND_sig, OR_sig, SHC, C_B, INC4: in std_logic;
29      C: out std_logic_vector (31 downto 0)
30      );
31  end COMPONENT;
32
33
34  COMPONENT  register_file is
35      port(
36      Rin:  in   std_logic;
37      Gra: in std_logic;
38  Grb: in std_logic;
39  Grc: in std_logic;
40  Rout:  in   std_logic;
41  BAout: in   std_logic;
42  IR: in   std_logic_vector (31 downto 0);
43  Bus_32_in: in   std_logic_vector (31 downto 0);
44  Bus_32_out: out std_logic_vector (31 downto 0)
45      );
46  end COMPONENT;
47
```

```
48
49  COMPONENT  memory is
50      port(
51      clk: in  std_logic;
52      MAin: in  std_logic;
53      MDin, MDout, write_sig, read_sig:in  std_logic;
54      datain:   in  std_logic_vector(31 downto 0); --input of MA
55      datain1:  in  std_logic_vector(31 downto 0);
56      dataout:  out std_logic_vector(31 downto 0); --output of MA
57      dataout1: out std_logic_vector(31 downto 0)
58  );
59  end COMPONENT;
60
61  COMPONENT control_unit is
62      port
63      (
64      clk: in   std_logic;
65      IR:  in   std_logic_vector (31 downto 0);
66      PCout, PCin, MAin, MDout,  MDin, MDbus:  out  std_logic;
67      Cout, Cin, C1out, C2out, Gra, Grb, Grc, CONin, n:  out  std_logic;
68      IRin, Rout, Rin, Ld:  out  std_logic;
69      INC4, END_sig, ADD, SUB, OR_sig:  out  std_logic;
70      Read_sig, Write_sig, Wait_sig, Goto6, C_B, Decr: out std_logic;
71      Ain, Bin, BAout, AND_sig, SHC:  out  std_logic;
72      Result: out std_logic_vector (4 downto 0)
73      );
74  end COMPONENT;
75
76  Signal A_sig: std_logic_vector(31 downto 0):= x"00001111";
77  Signal C_sig: std_logic_vector(31 downto 0):= x"00000000";
78  Signal MainBus: std_logic_vector(31 downto 0):= x"00000000";--datain
79  Signal MainBus2: std_logic_vector(31 downto 0):= x"00000000";--datain1
80  Signal BusTemp: std_logic_vector(31 downto 0):= x"00000000";--dataout
81  Signal BusTemp1: std_logic_vector(31 downto 0):= x"00000000";--dataout1
82  Signal IR_sig: std_logic_vector(31 downto 0):= x"00000000";
83  Signal PCout_sig,PCin_sig,MAin_sig, MDout_sig, MDin_sig, MDbus_sig,
84                 Cout_sig, Cin_sig, C1out_sig,C2out_sig,Gra_sig, Grb_sig, Grc_sig,
85                 CONin_sig, n_sig,IRin_sig, Rout_sig, Rin_sig, Ld_sig, INC4_sig,END_i, ADD_sig, SUB_sig, OR_i,Read_i, Write_i,
86                 Wait_i,Goto6_sig, C_B_sig, Decr_sig,Ain_sig, Bin_sig, BAout_sig, AND_i, SHC_sig: std_logic;
87
88
89  Begin
90
91
92  u0: ALU port map (clk_datapath,MainBus,A_sig,n,ADD, SUB, AND_sig,
93         OR_sig, SHC, C_B, INC4,C_sig);
94
95
96  u1: register_file port map (Rin,Gra,Grb,Grc,Rout,BAout,IR_sig,MainBus,BusTemp1);
97
98
99  u2: memory port map (clk_datapath, MAin,MDin, MDout, Write_sig, Read_sig,
100            MainBus,MainBus2, BusTemp, BusTemp1);
101
102
103 u3 : control_unit port map (clk_datapath, IR_sig,
104                 PCout_sig, PCin_sig, MAin_sig, MDout_sig, MDin_sig, MDbus_sig,
105                 Cout_sig, Cin_sig, C1out_sig,C2out_sig, Gra_sig, Grb_sig, Grc_sig,
106                 CONin_sig, n_sig,IRin_sig, Rout_sig, Rin_sig, Ld_sig, INC4_sig,
107                 END_i, ADD_sig, SUB_sig, OR_i,Read_i, Write_i,
108                 Wait_i,Goto6_sig, C_B_sig, Decr_sig,Ain_sig, Bin_sig, BAout_sig, AND_i, SHC_sig,Opcode);
109
110 end Behavioral;
111
```

Successful Compilation of Data Path Component:

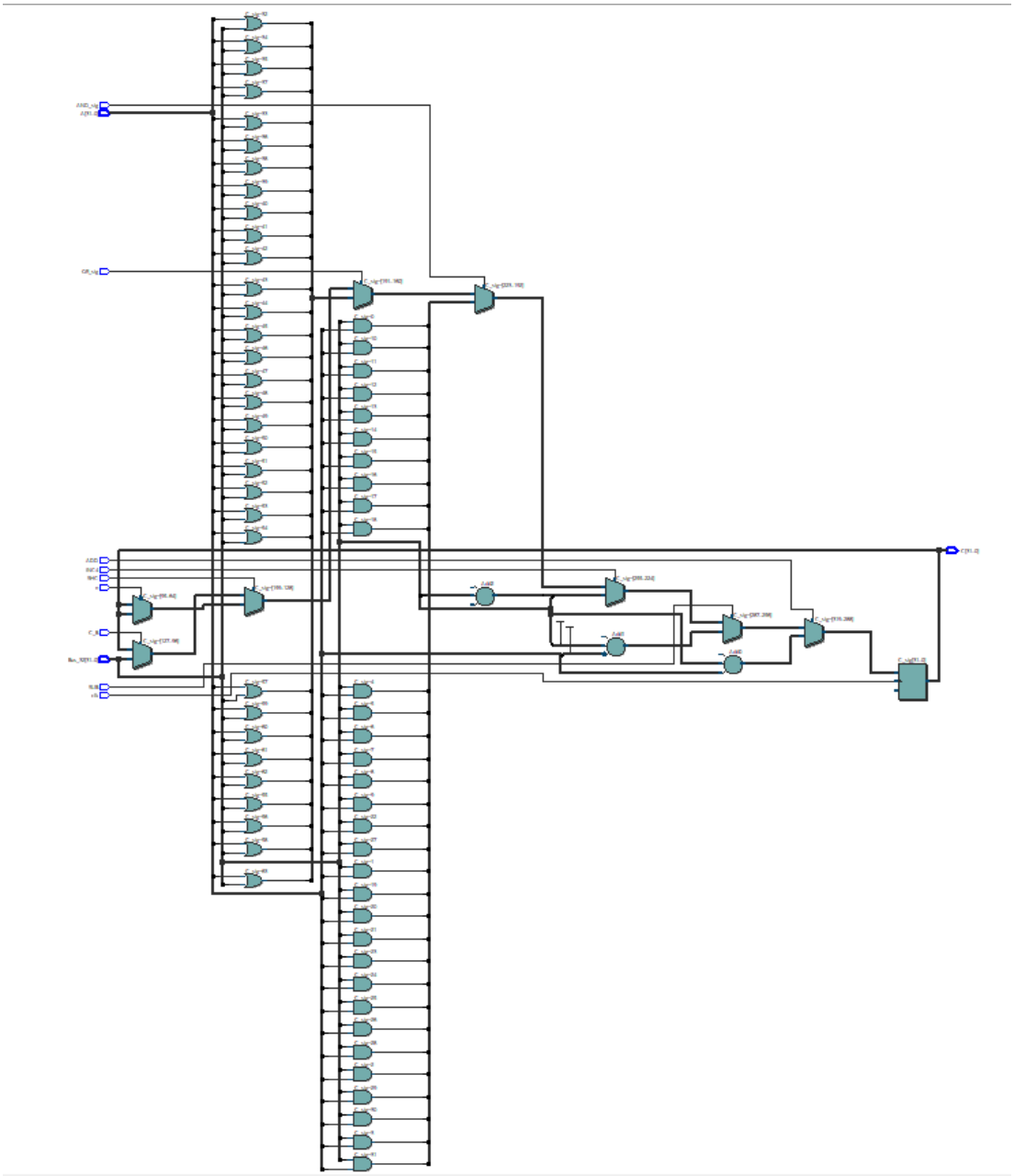


Results

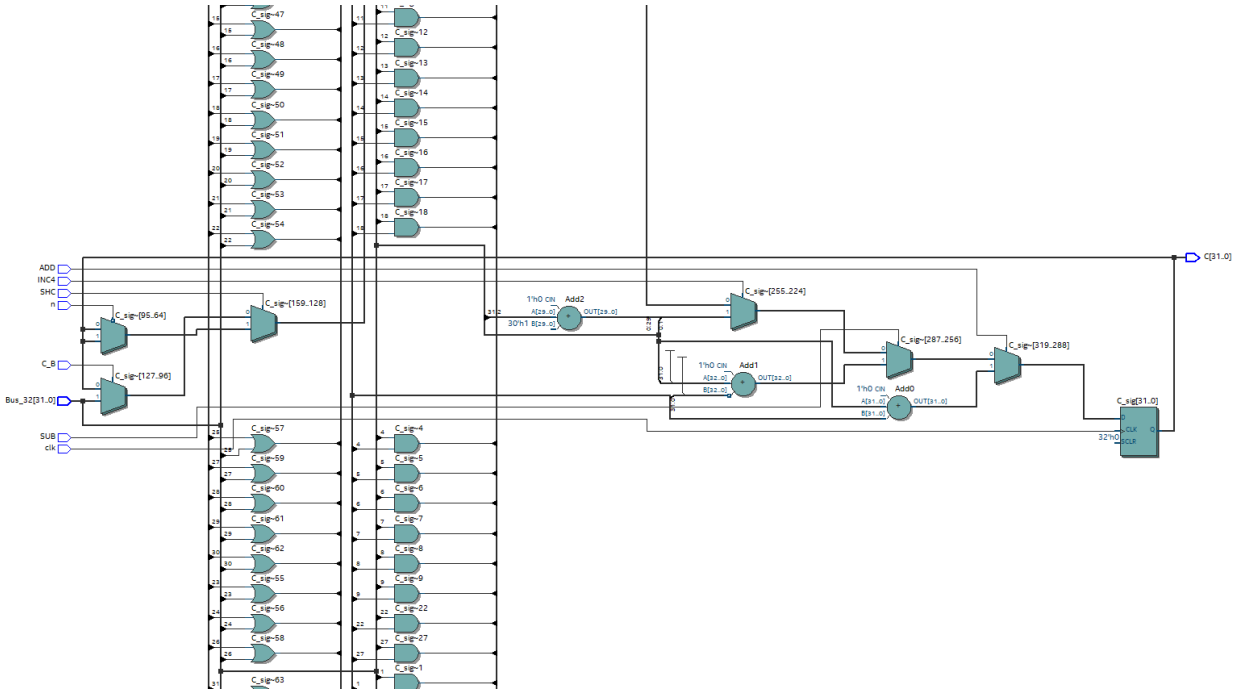
This section will focus on the RTN and simulation results obtained from each and every component.

Components RTL

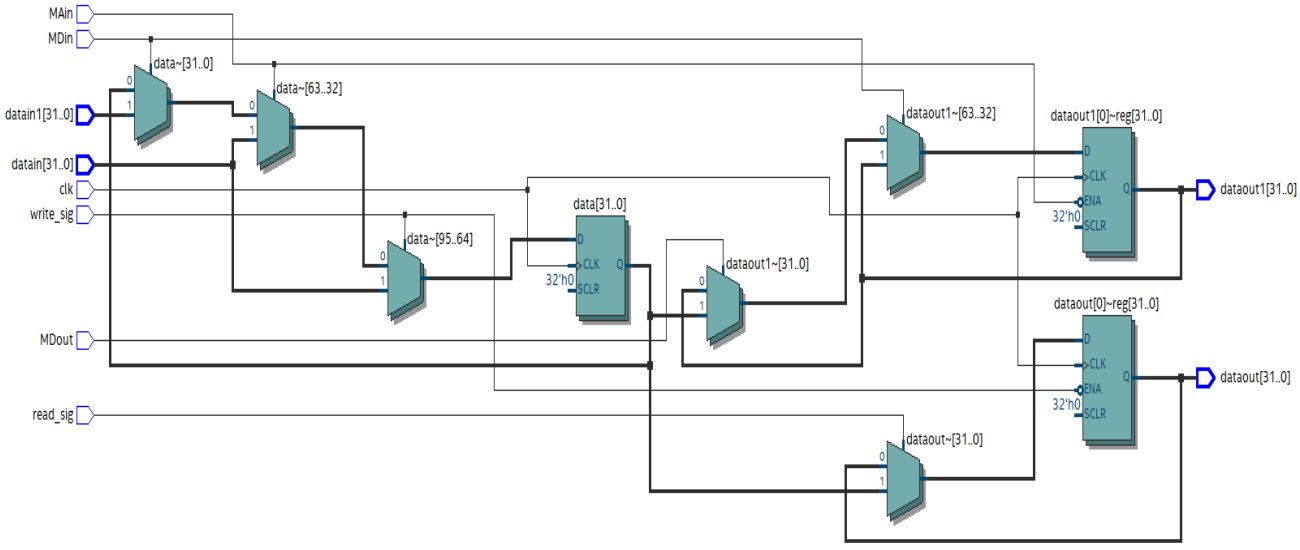
ALU Unit Component



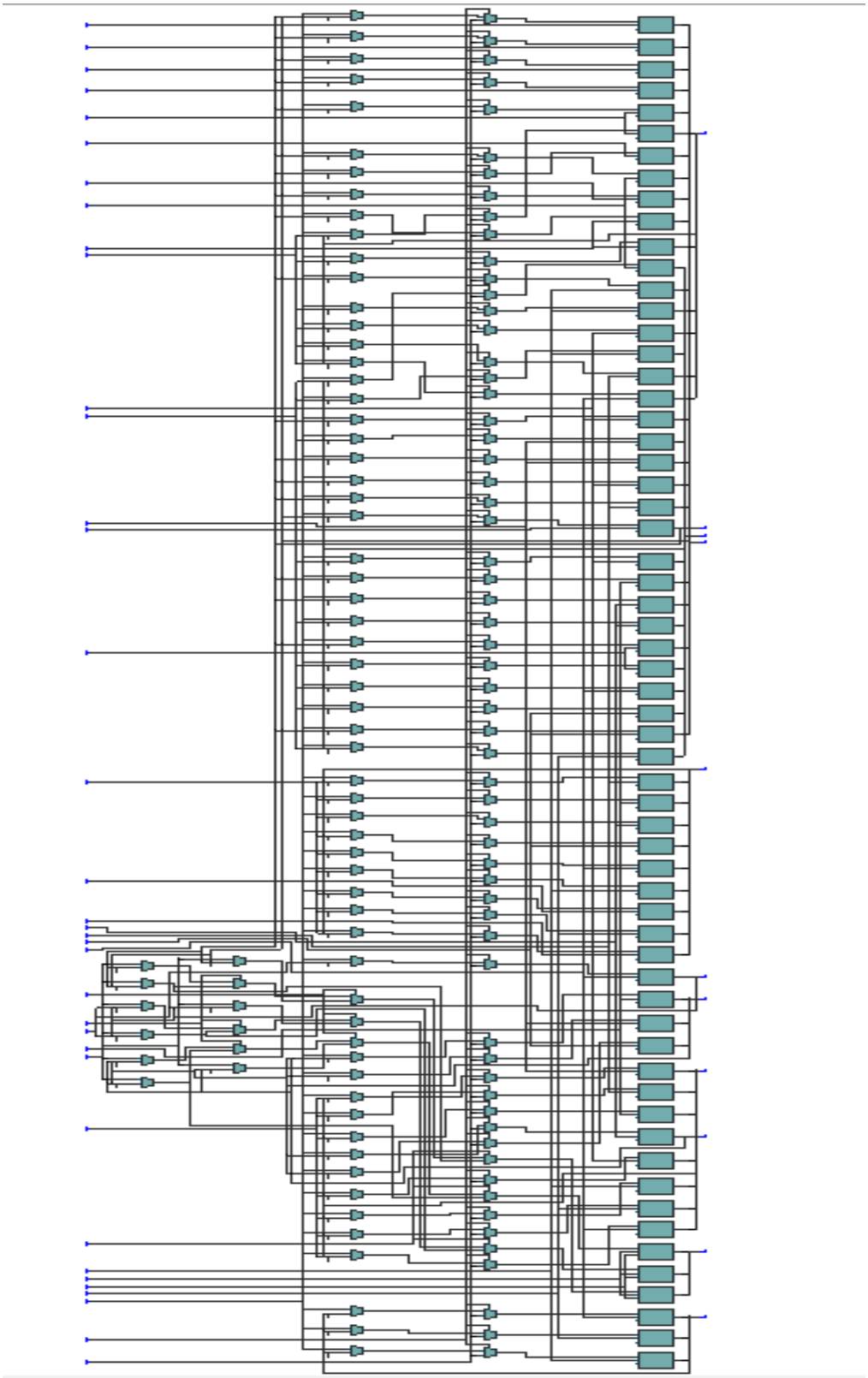
Closer Look at ALU Component RTL Viewer:



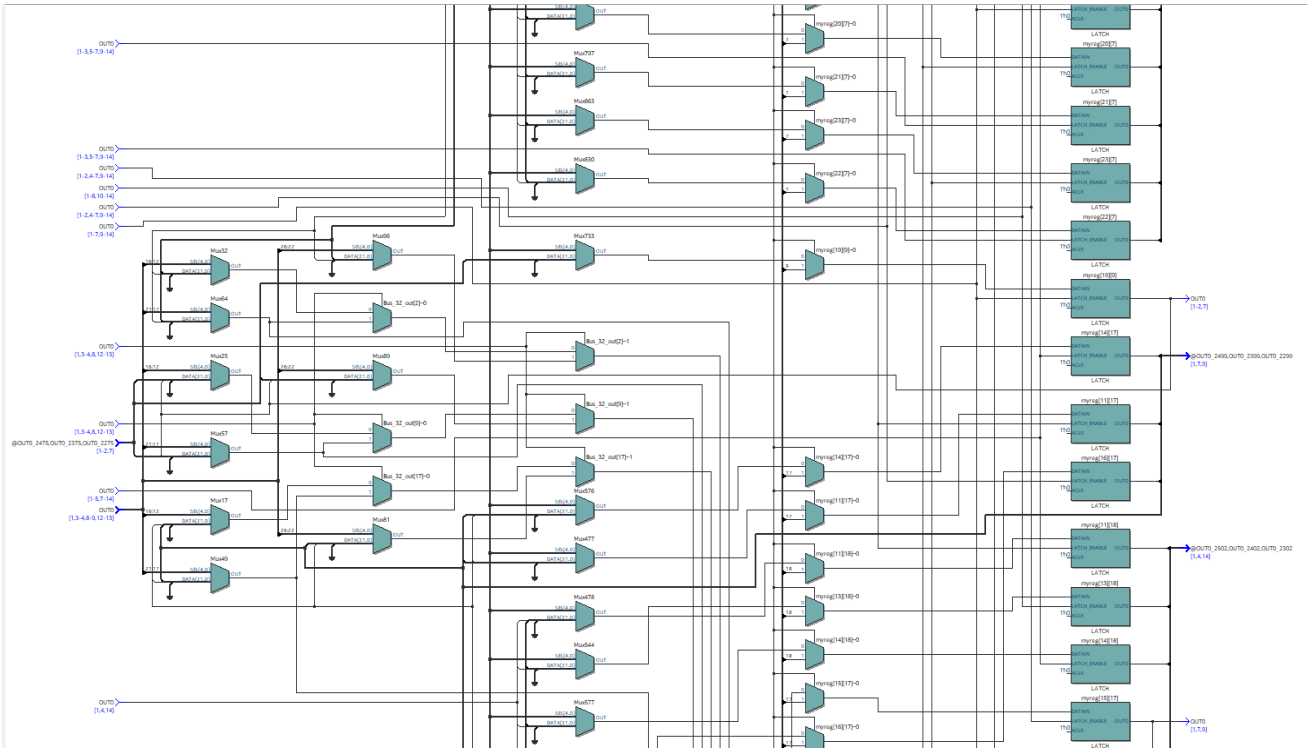
Memory Unit Component



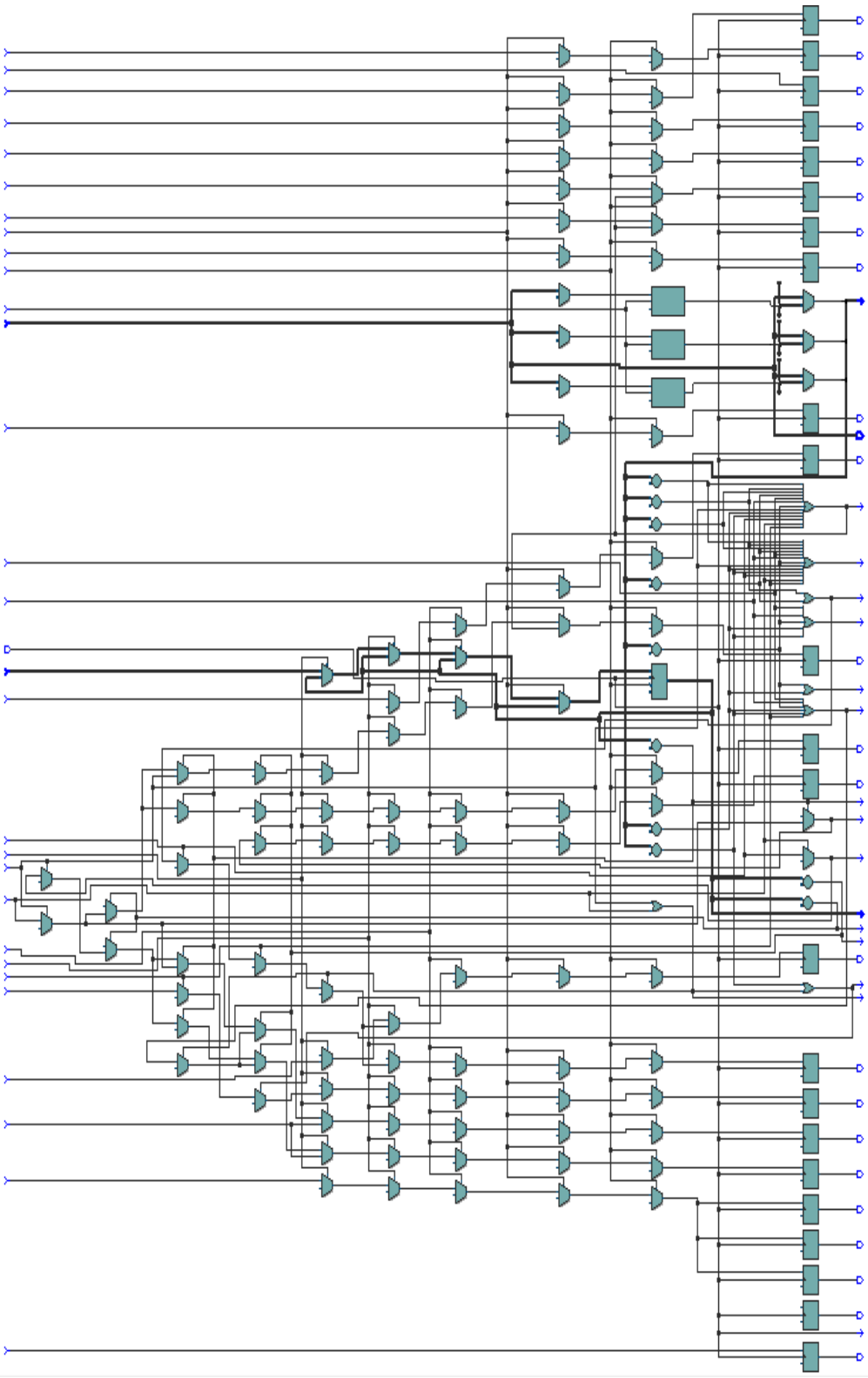
Register File Component



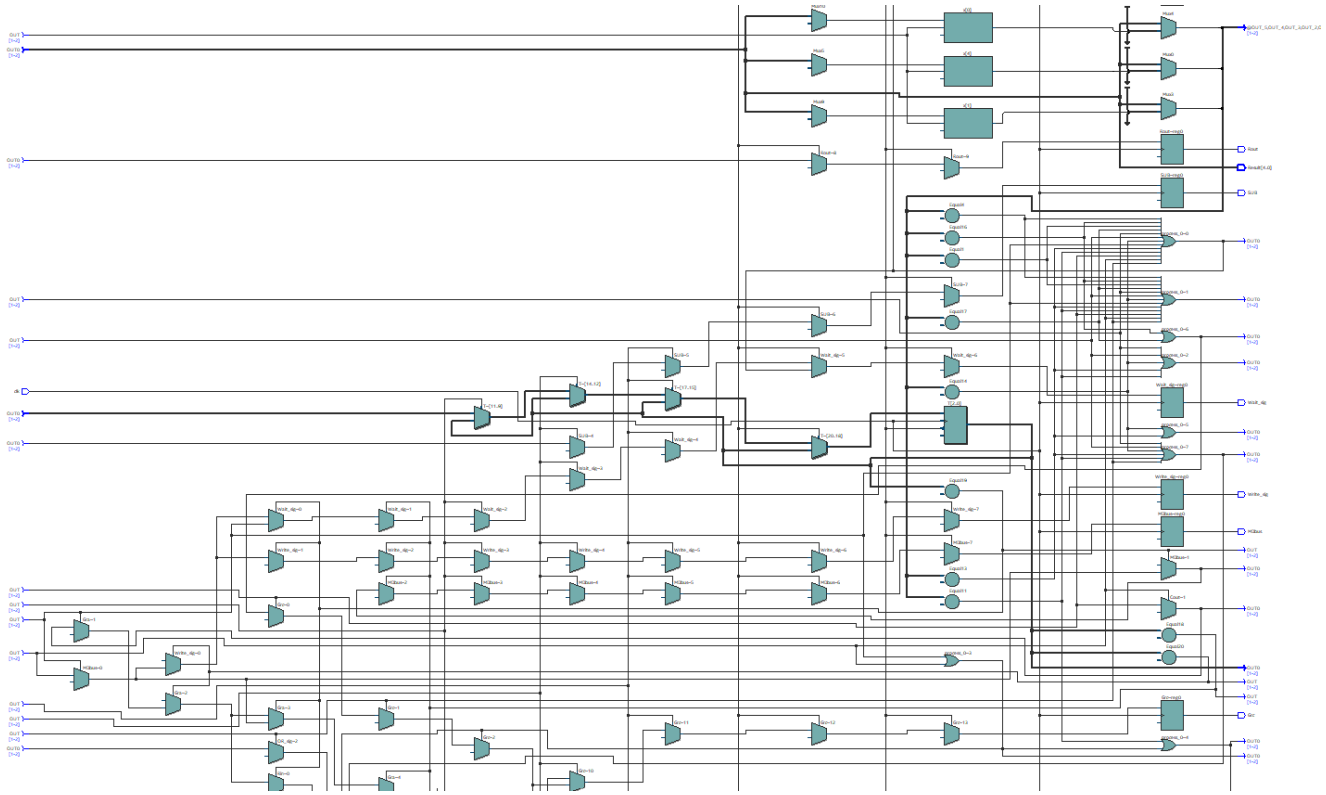
Closer Look at Register File Component RTL Viewer:



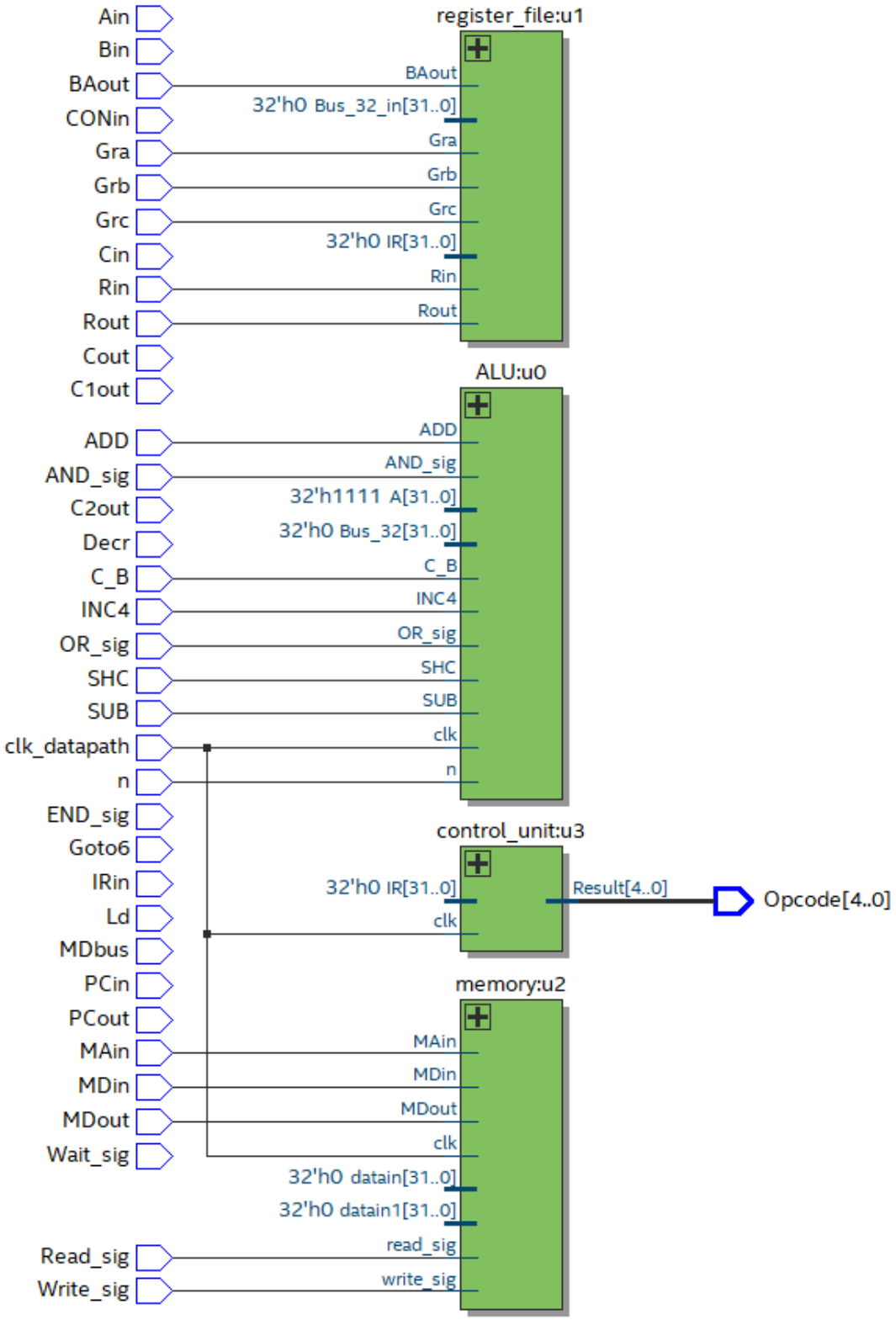
Control Unit Component



Closer Look at Control Unit Component RTL Viewer:

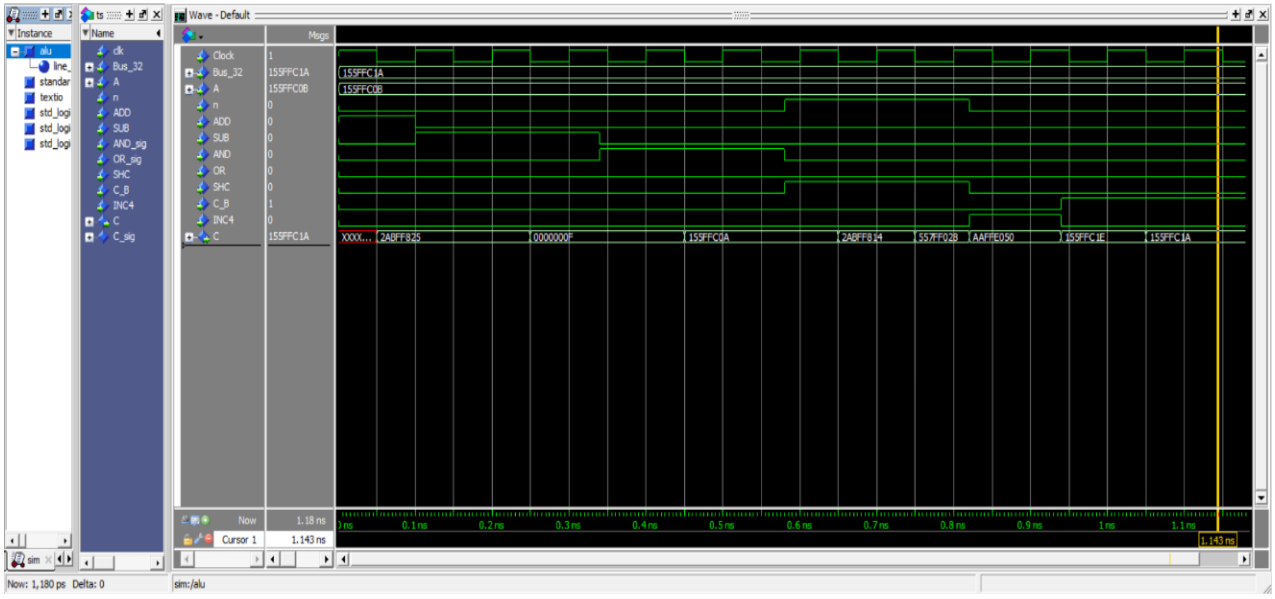


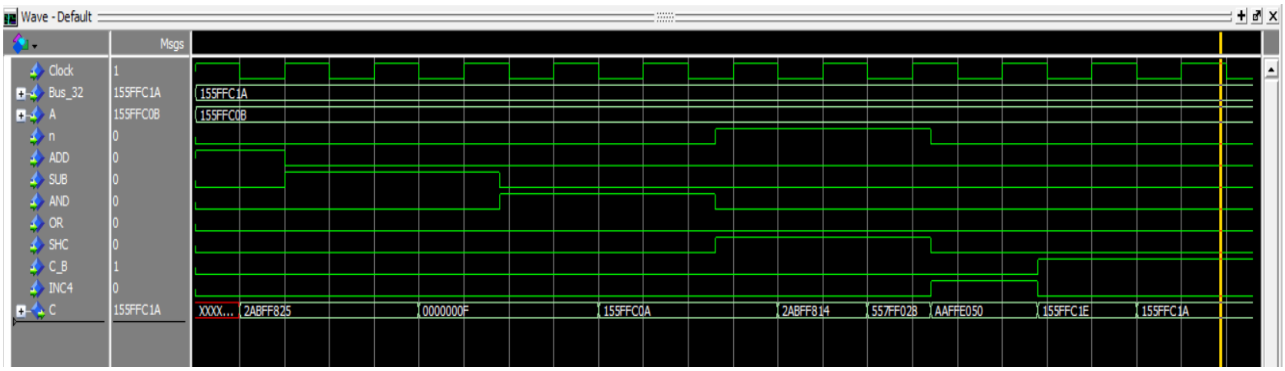
Data Path Component



Components Simulation

ALU Unit Component





The following inputs were initialized:

Bus_32: 00010101 01011111 11111100 00011010 (binary) = 155FFC1A (hexadecimal)

A: 00010101 01011111 11111100 00001011 (binary) = 155FFC0B (hexadecimal)

At 1st clock pulse: (Addition operation)

When ADD=1, we get output C= 2ABFF825 (hexadecimal)

At 2nd clock pulse: (Subtraction operation)

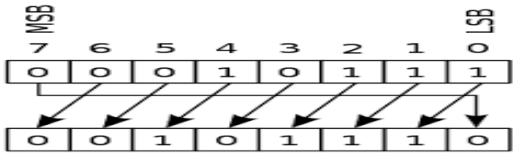
When SUB=1, we get output C= F (hexadecimal)

At 3rd clock pulse: (AND operation)

When AND=1, we get output C= 155FFC0A (hexadecimal)

In case of Shift Circular operation:

When SHC=1, n=1 (at each clock pulse the LSB from the existing output shifts in the below manner)



Before 4th clock pulse: 00010101010111111111110000001010 = 155FFC0A (hexadecimal)

After 4th clock pulse: 00101010101111111111100000010100 = 2ABFF814 (hexadecimal)

After 5th clock pulse: 01010101011111111111000000101000 = 557FF028 (hexadecimal)

After 6th clock pulse: 10101010111111111110000001010000 = AAFFE050 (hexadecimal)

At 7th clock pulse: (INC4 operation)

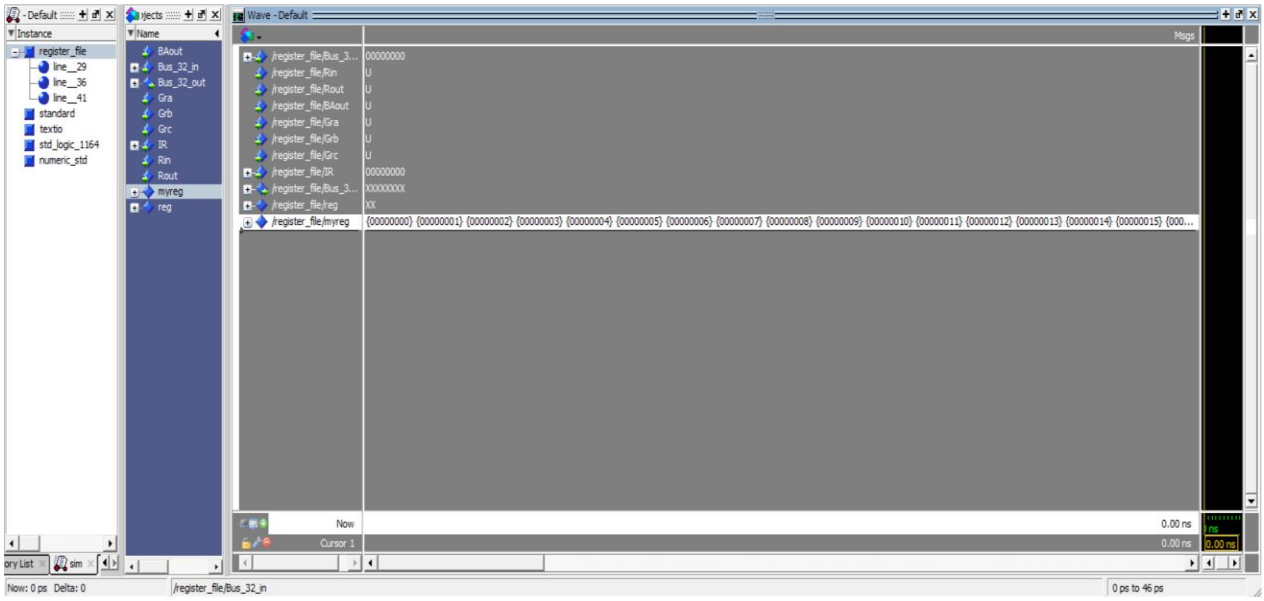
When INC4=1, we get output C= 00010101010111111111110000011010 = 155FFC1E (hexadecimal)

At 8th clock pulse: (C_B operation)

When C_B =1, we get output C= 155FFC1A (hexadecimal)

Register File Component

The beginning phase when we can see the signal initialised for the 24 General Purpose Registers

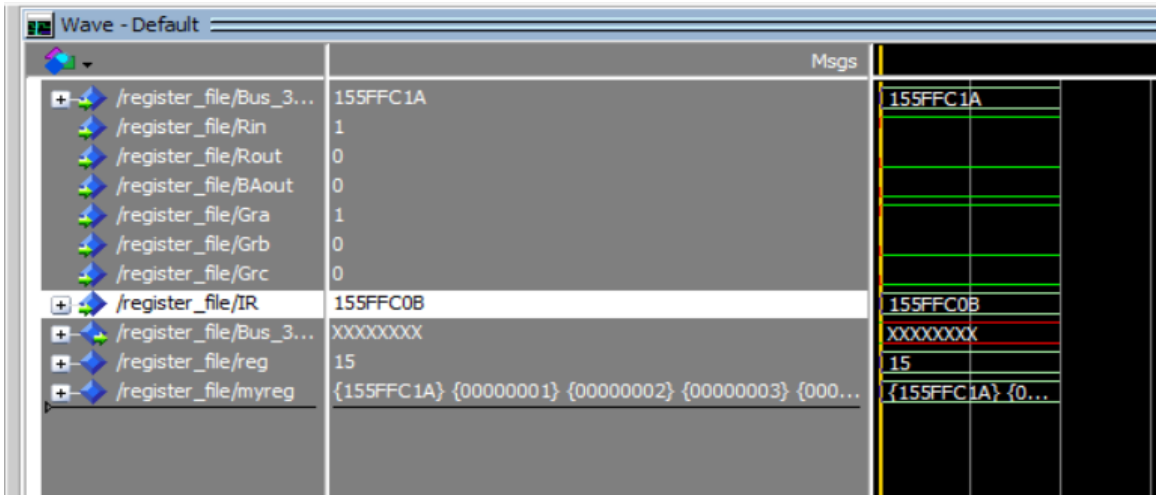


The following inputs were initialized:

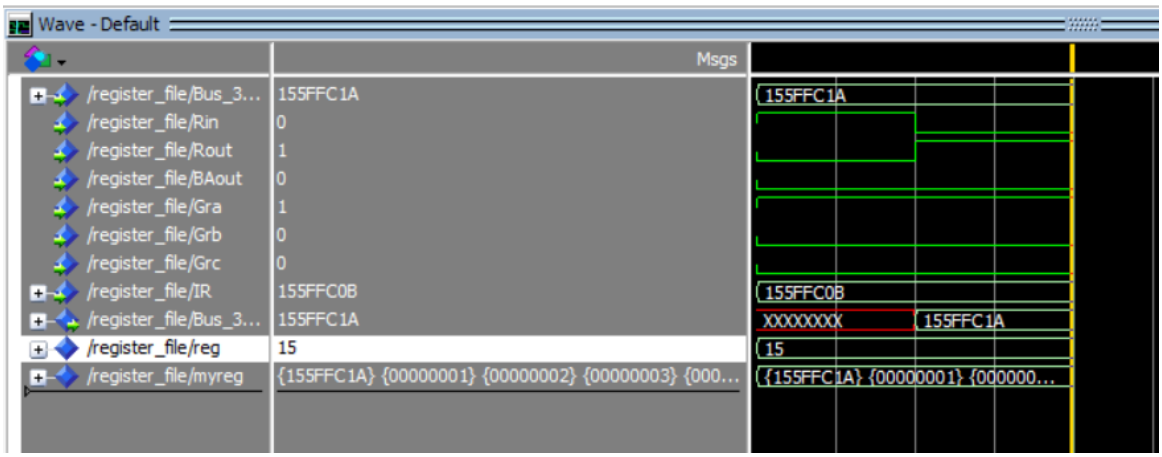
Bus_32_in: 00010101 01011111 11111100 00011010 (binary) = 155FFC1A (hexadecimal)

IR: 00010101 01011111 11111100 00001011 (binary) = 155FFC0B (hexadecimal)

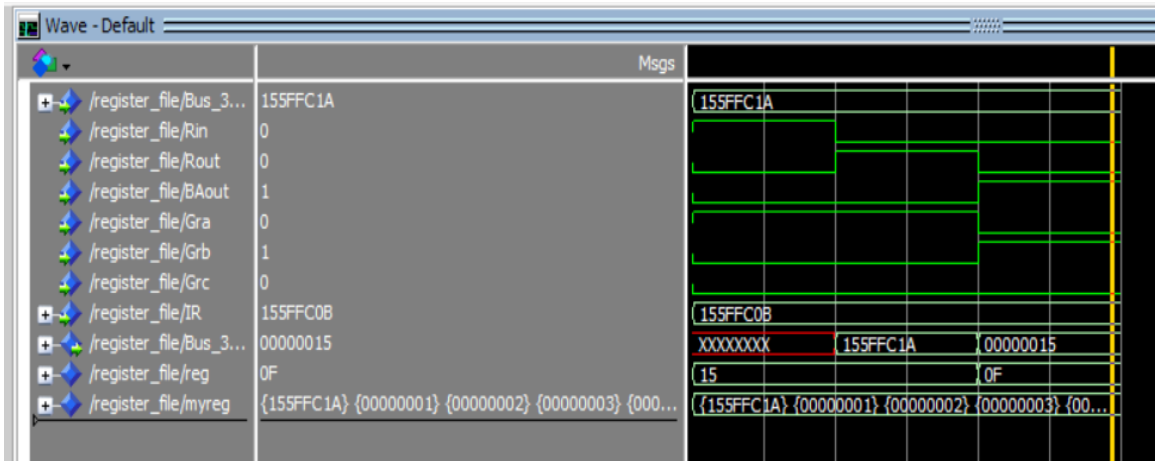
Condition 1: When Rin=1, Gra=1, we save the value of the (Bus_32_in) input to one of the register signals



Condition 2: When Rout=1, Gra=1, we get the value saved in the register as (Bus_32_out) output



Condition 3: When BAout=1, Grb=1 we get the value from IR(21 downto 17)= 01111 (binary)= 15 (decimal) as output for (Bus_32_out)



Memory Unit Component

The following inputs were initialized:

Datain: 155FFC1A (hexadecimal)

Datain1: 155FFC0B (hexadecimal)

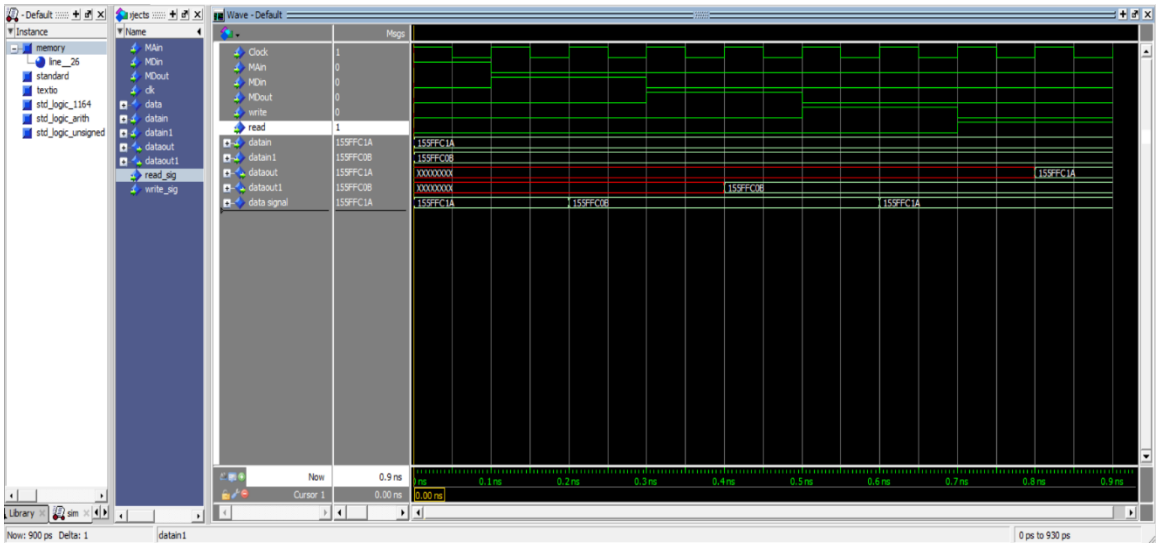
At rising edge of clock when **Main=1**, we get data signal = 155FFC1A (hexadecimal)

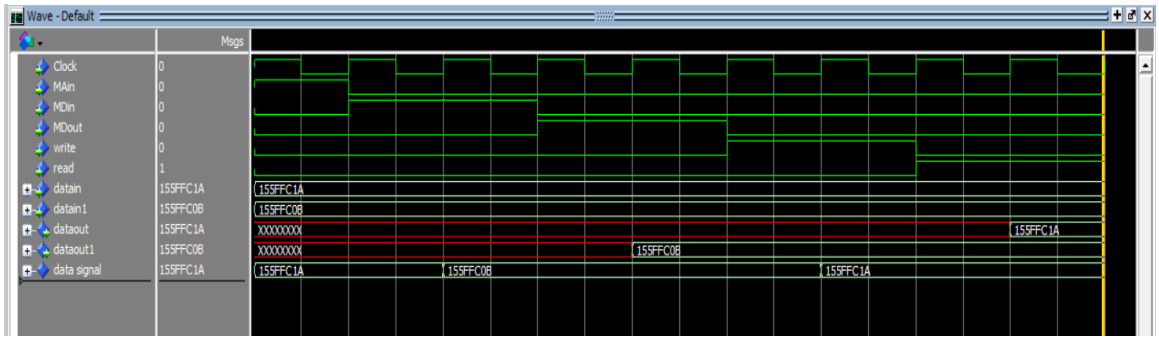
When MDin=1, we get data signal = 155FFC0B (hexadecimal)

When MDout=1, we get Dataout1 = 155FFC0B (previously saved as data signal)

When Write=1, we get data signal = Datain = 155FFC1A (hexadecimal)

When Read=1, we get Dataout = 155FFC1A (hexadecimal) (previously inputted using write signal)



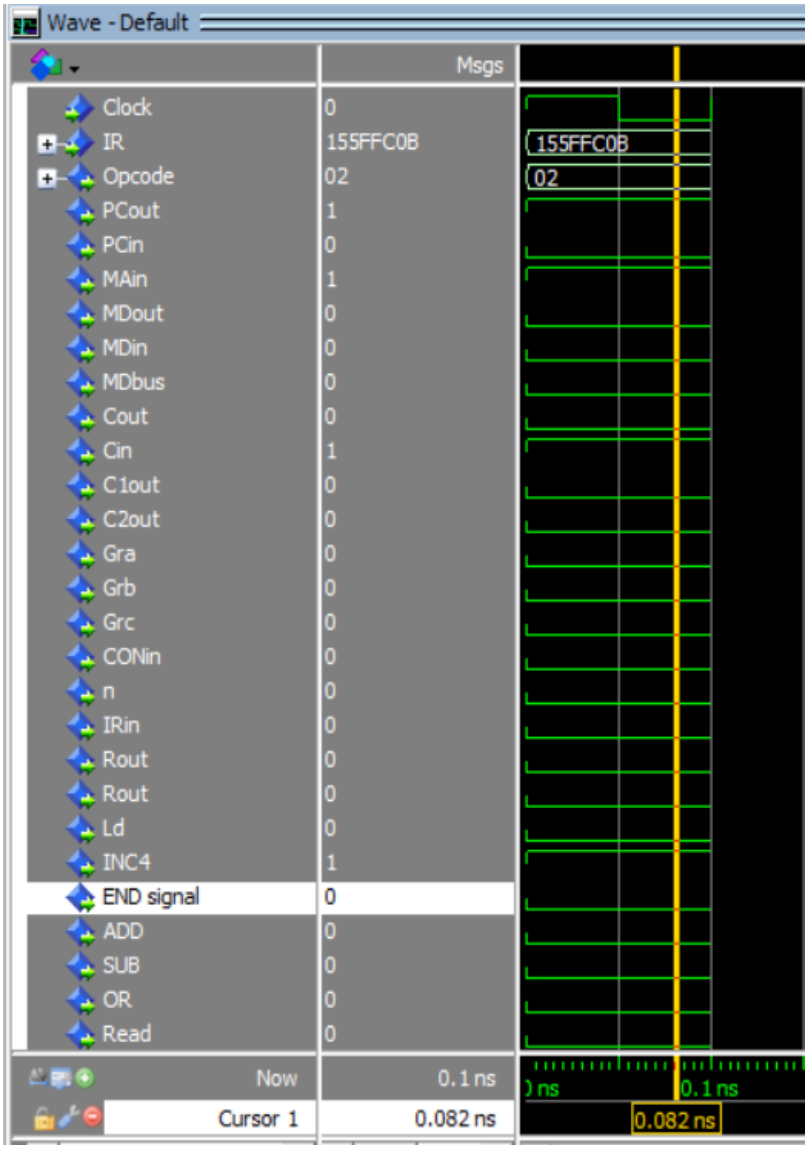


Control Unit Component

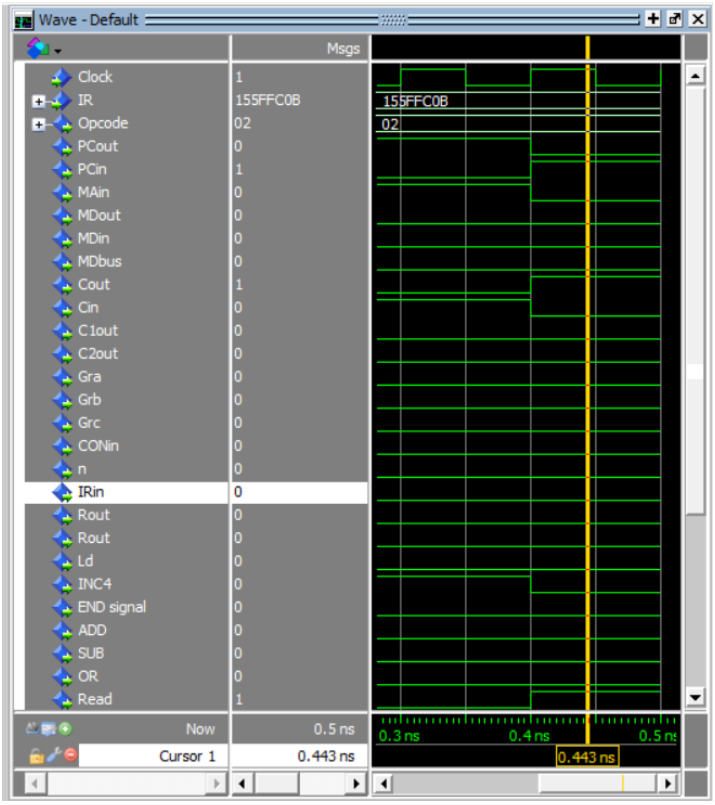
The following inputs were initialized:

IR: 00010101 01011111 11111100 00001011 (binary) = 155FFC0B (hexadecimal)

The following outputs were obtain PCout=1, Main=1, INC4=1, Cin=1, Opcode=2 (decimal) and initially step T=0 (instruction fetch step 1)

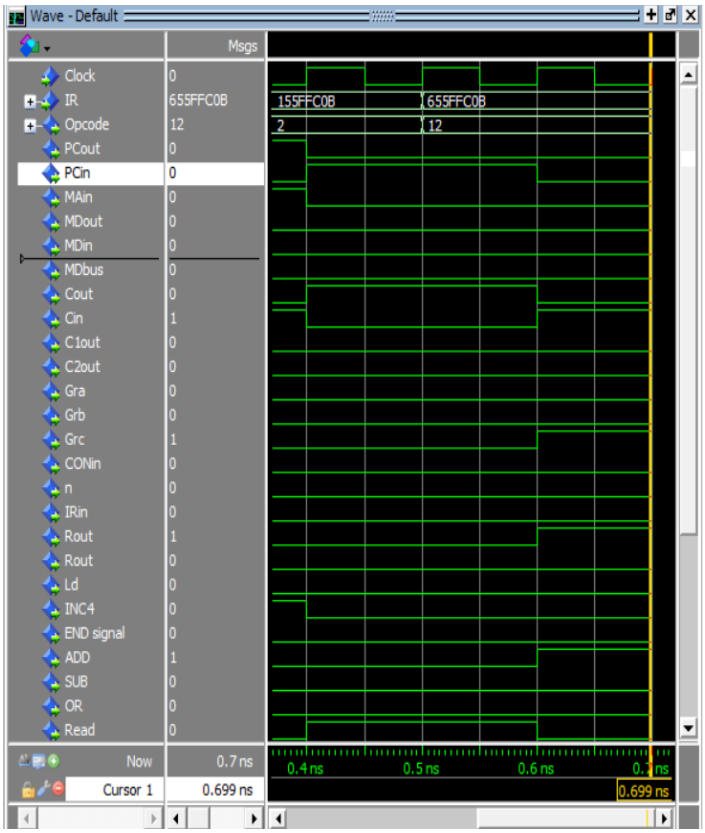


To verify the simulation, we manually initialized step T=1 (instruction fetch step 2) to match with control unit outputs Read=1, Wait=1, Cout=1, PCin=1, Opcode=2 (decimal)



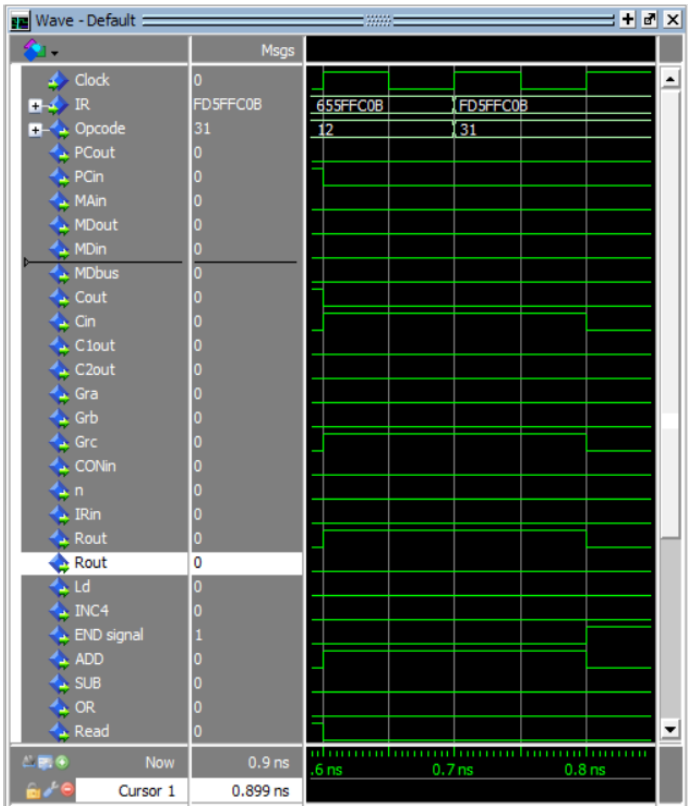
We initialised an input where IR: 01100101 01011111 11111100 00001011 (binary) = 655FFC0B (hexadecimal)

We get outputs Grc=1, Rout=1, ADD=1, Cin=1, Opcode=12 (decimal) which is ADD operation and set the step T=4 to verify the simulation.



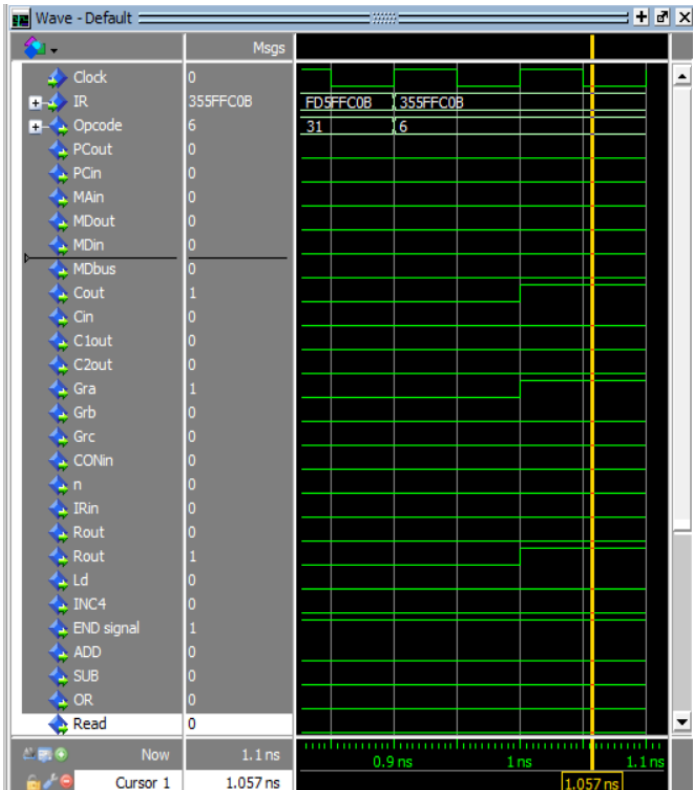
We initialised an input where IR: FD5FFC0B (hexadecimal)

We get all outputs zero other than End_signal=1, Opcode=31 (decimal) which is STOP operation and set the step T=4 to verify the simulation.



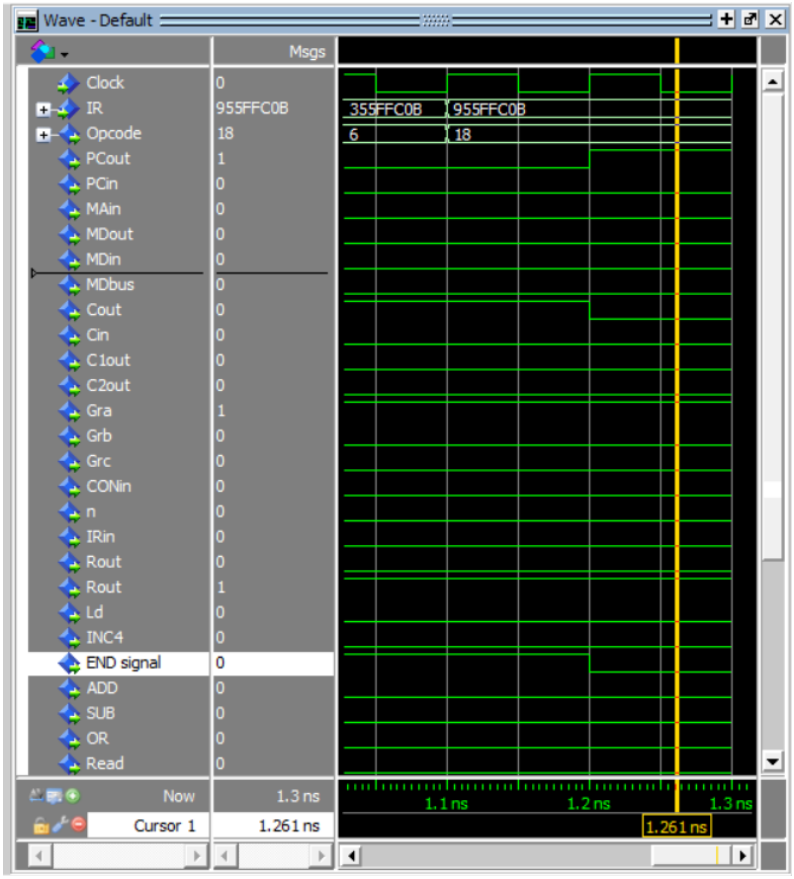
We initialised an input where IR: 355FFC0B (hexadecimal)

We get as outputs Cout=1, Gra=1, Rin=1, END_signal=1, Opcode=6 (decimal) which is LAR operation and set the step T=5 to verify the simulation.



We initialised an input where IR: 955FFC0B (hexadecimal)

We get as outputs PCout=1, Gra=1, Rin=1, Opcode=18 (decimal) which is brlpl operation and set the step T=4 to verify the simulation.



Conclusion

The objective of this project have been achieved by implementing implement 1-bus processor with a hardwired control unit that have specific characteristic using VHDL Code. Also, the components' functionality was tested by running the RTL Viewer and simulate each component. Furthermore, we have tried to connect all components as a one system. By achieving these goals, the team members have learnt:

- The importance of RTN Concrete (Control Signals) within the CPU.
- Learnt how to program all the components and debug errors.
- Understand how all the components are connected to reach other and how we program the top level.
- Main program syntax was learnt for example, during the use of "while" condition the 'process' is not required.