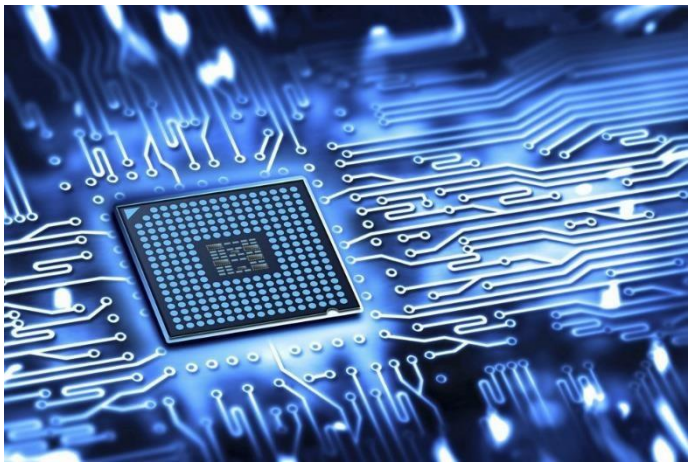


CSE112: Computer Architecture

Major Task Phase2

Presented to: Dr. Tamer Mostafa Abdelkader

Eng. Ahmed Fawzy

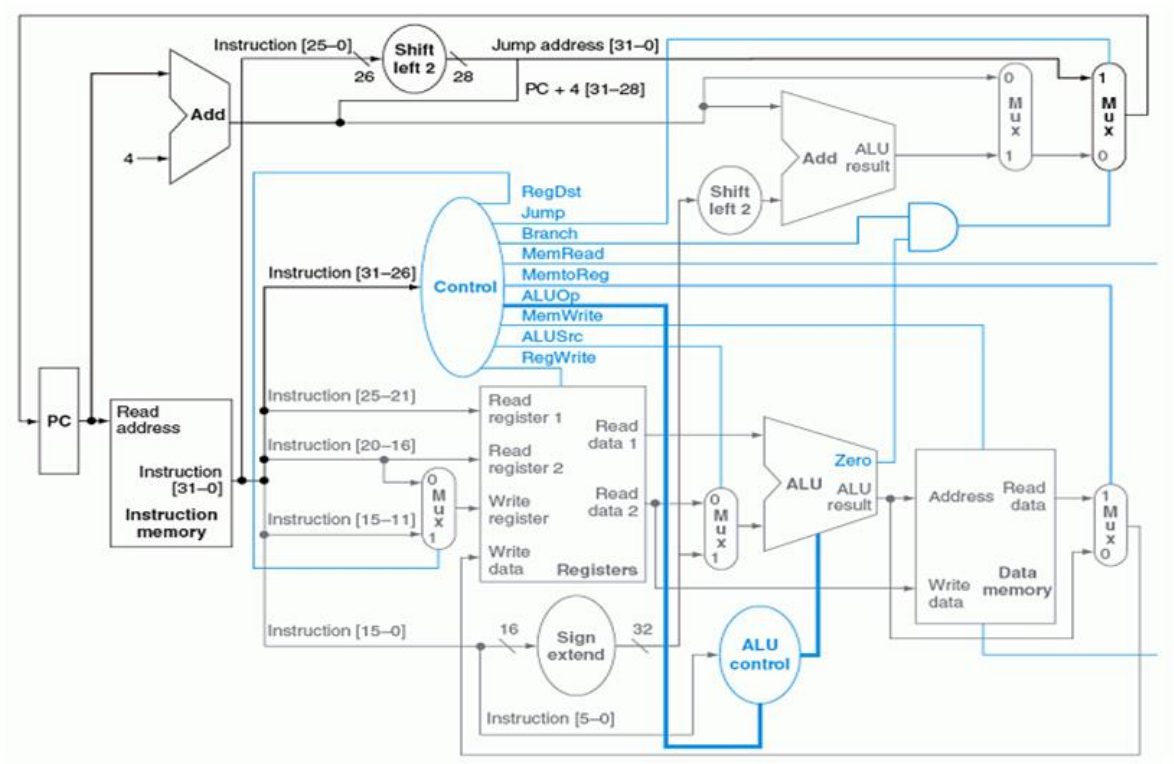
Eng. Mohamed Atef

Built by:

Mazen Mohamed Elsaied (**Team leader**)	20P5893
George Emad Welson	20P3831
Farida Mohamed El-Husseiny	20P6022
Zeina Hesham Reda	2000320
Hassan Mohamed El-Tobgy	20P6173

1.Description

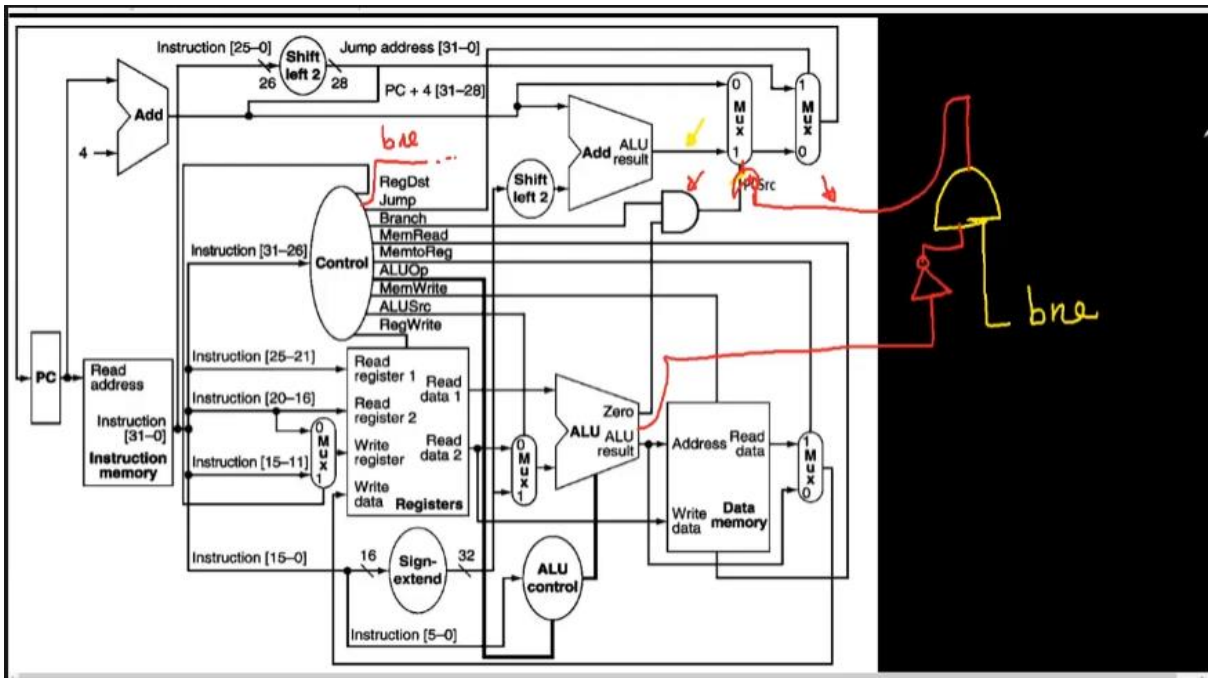
This project describes an emulation of a 32-bit Single-cycle MIPS processor using a hardware description language (VHDL). The MIPS processor is separated into five stages: instruction fetch, instruction decode, execution, data memory and write back. The control unit controls the operations performed in these stages. All the modules in the design are coded in VHDL, as it is very useful tool with its concept of concurrency to cope with the parallelism of digital hardware. The top-level module connects all the stages into a higher level. Once detecting the particular approaches for input, output, main block and different modules, the VHDL descriptions are run through a VHDL simulator, followed by the timing analysis for the validation, functionality and performance of the designated design that demonstrate the effectiveness of the design.



2.Modifications

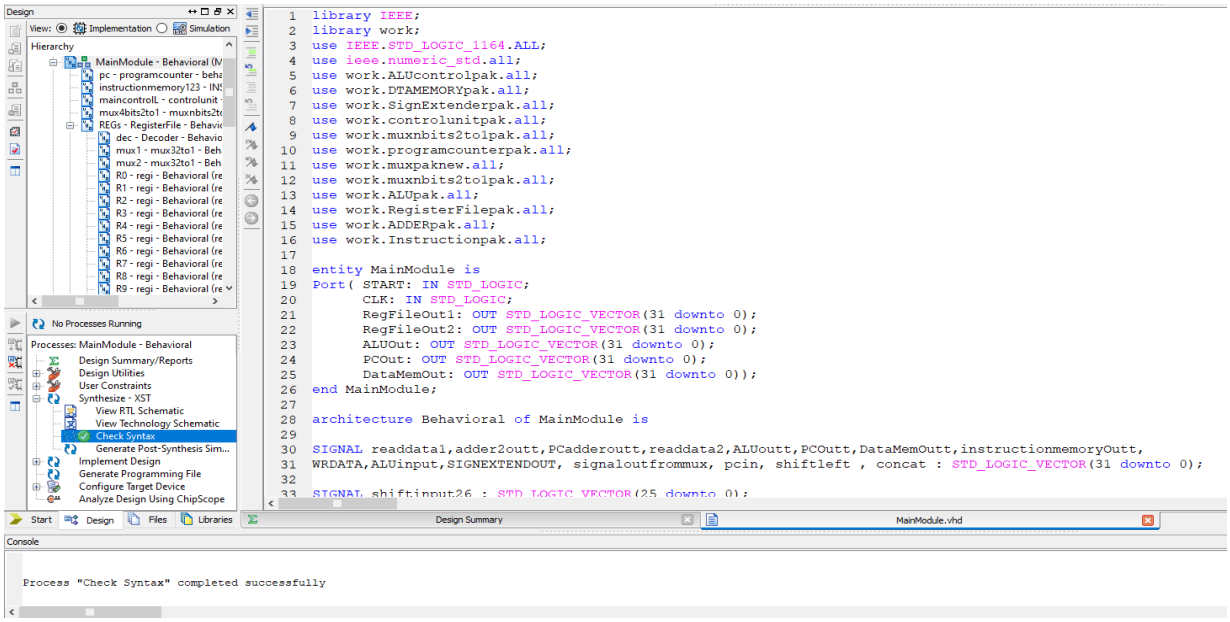
2.1 Removing Clock from instruction memory

2.1 Adding Branch if not equal

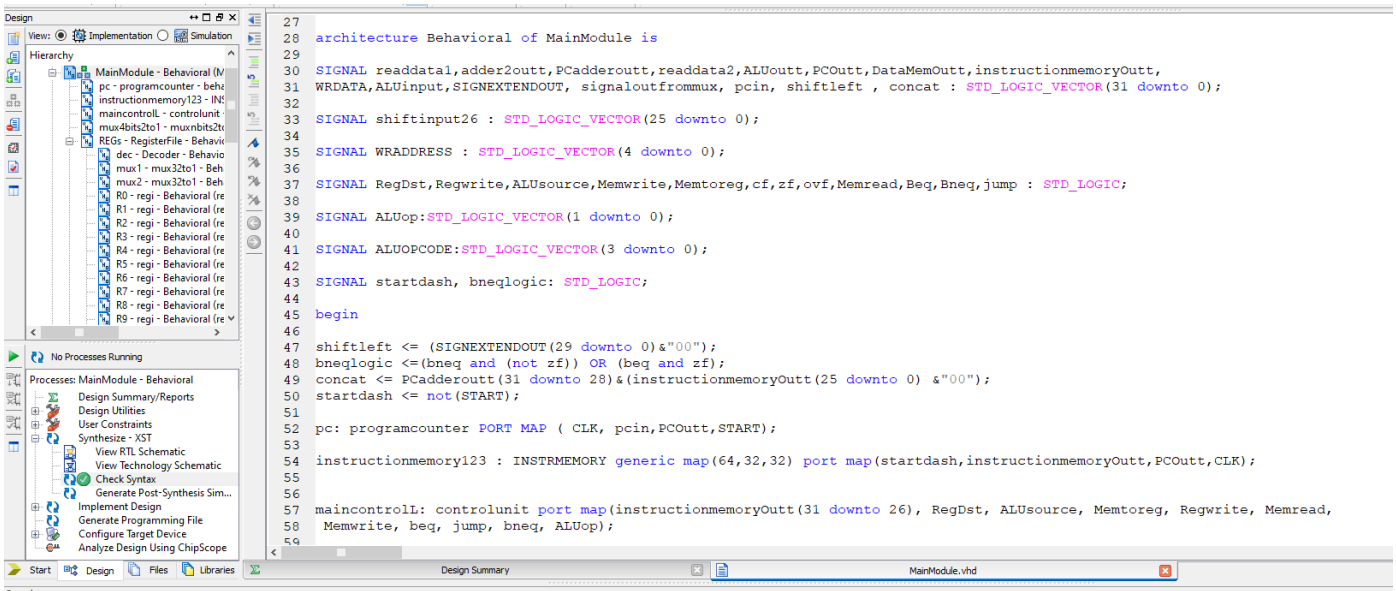


3. Implementation

3.1 Main module

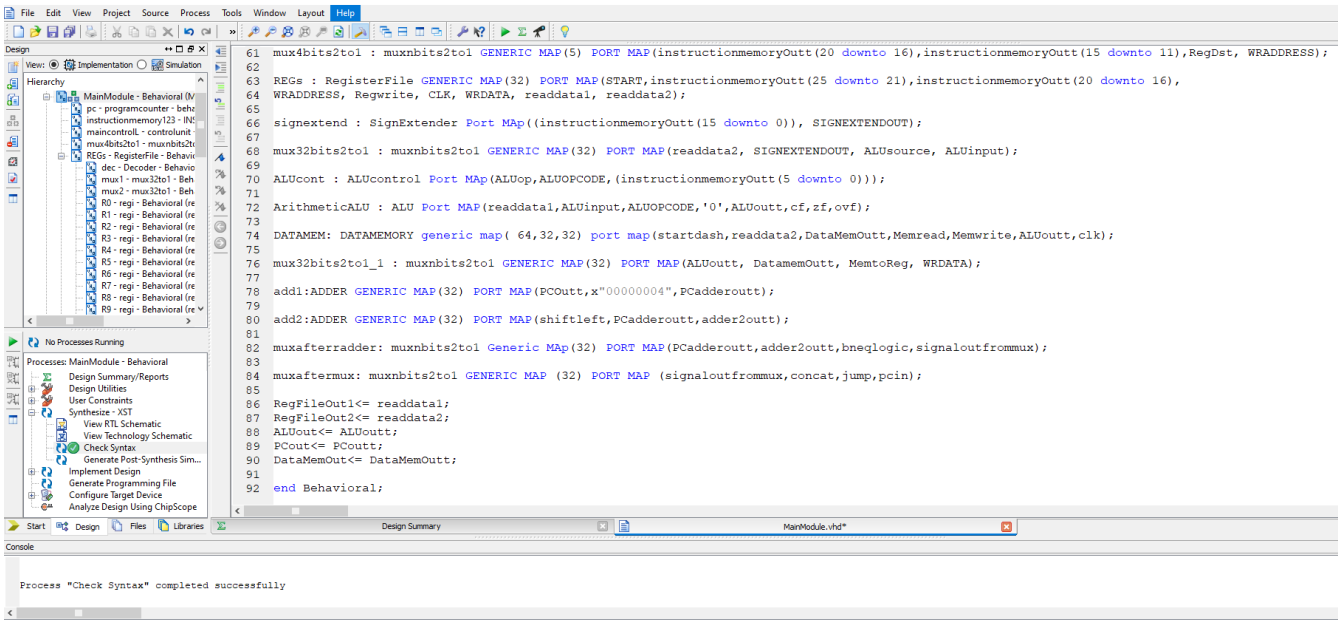



```
1  library IEEE;
2  library work;
3  use IEEE.STD_LOGIC_1164.ALL;
4  use ieee.numeric_std.all;
5  use work.ALUcontrolpak.all;
6  use work.DTAMEMORYpak.all;
7  use work.SignExtenderpak.all;
8  use work.controlunitpak.all;
9  use work.muxnbits2tolpak.all;
10 use work.programcounterpak.all;
11 use work.muxpaknew.all;
12 use work.muxnbits2tolpak.all;
13 use work.ALUpak.all;
14 use work.RegisterFilepak.all;
15 use work.ADDERpak.all;
16 use work.Instructionpak.all;
17
18 entity MainModule is
19 Port( START: IN STD_LOGIC;
20       CLK: IN STD_LOGIC;
21       RegFileOut1: OUT STD_LOGIC_VECTOR(31 downto 0);
22       RegFileOut2: OUT STD_LOGIC_VECTOR(31 downto 0);
23       ALUOut: OUT STD_LOGIC_VECTOR(31 downto 0);
24       PCOut: OUT STD_LOGIC_VECTOR(31 downto 0);
25       DataMemOut: OUT STD_LOGIC_VECTOR(31 downto 0));
26 end MainModule;
27
28 architecture Behavioral of MainModule is
29
30 SIGNAL readdata1,adder2outt,PCadderoutt,readdata2,ALUoutt,PCOutt,DataMemOutt,instructionmemoryOutt,
31 WRDATA,ALUinput,SIGNEXTENDOUT, signaloutfrommux, pcin, shiftleft , concat : STD_LOGIC_VECTOR(31 downto 0);
32
33 SIGNAL shiftinput26 : STD_LOGIC_VECTOR(25 downto 0);
```



```
architecture Behavioral of MainModule is

SIGNAL readdata1,adder2outt,PCadderoutt,readdata2,ALUoutt,PCOutt,DataMemOutt,instructionmemoryOutt,
WRDATA,ALUinput,SIGNEXTENDOUT, signaloutfrommux, pcin, shiftleft , concat : STD_LOGIC_VECTOR(31 downto 0);

SIGNAL shiftinput26 : STD_LOGIC_VECTOR(25 downto 0);

SIGNAL WRADDRESS : STD_LOGIC_VECTOR(4 downto 0);

SIGNAL RegDst,Regwrite,ALUsource,Memwrite,Memtoreg,cf,zf,ovf,Memread,Beq,Bneq,jump : STD_LOGIC;

SIGNAL ALUop:STD_LOGIC_VECTOR(1 downto 0);

SIGNAL ALUOPCODE:STD_LOGIC_VECTOR(3 downto 0);

SIGNAL startdash, bneqlogic: STD_LOGIC;

begin

shiftleft <= (SIGNEXTENDOUT(29 downto 0)&"00");
bneqlogic <=(bneq and (not zf)) OR (beq and zf);
concat <= PCadderoutt(31 downto 28)&(instructionmemoryOutt(25 downto 0) &"00");
startdash <= not(START);

pc: programcounter PORT MAP ( CLK, pcin,PCOutt,START);

instructionmemory123 : INSTRMEMORY generic map(64,32,32) port map(startdash,instructionmemoryOutt,PCOutt,CLK);


maincontrolL: controlunit port map(instructionmemoryOutt(31 downto 26), RegDst, ALUsource, Memtoreg, Regwrite, Memread,
 Memwrite, beq, jump, bneq, ALUop);
```

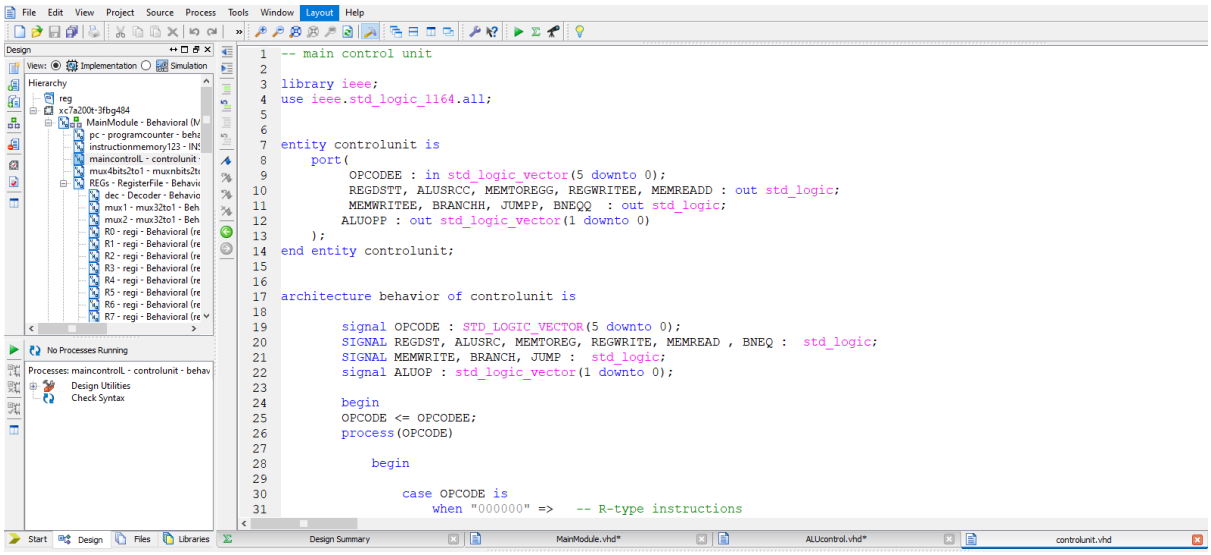
```
Process "Check Syntax" completed successfully
```



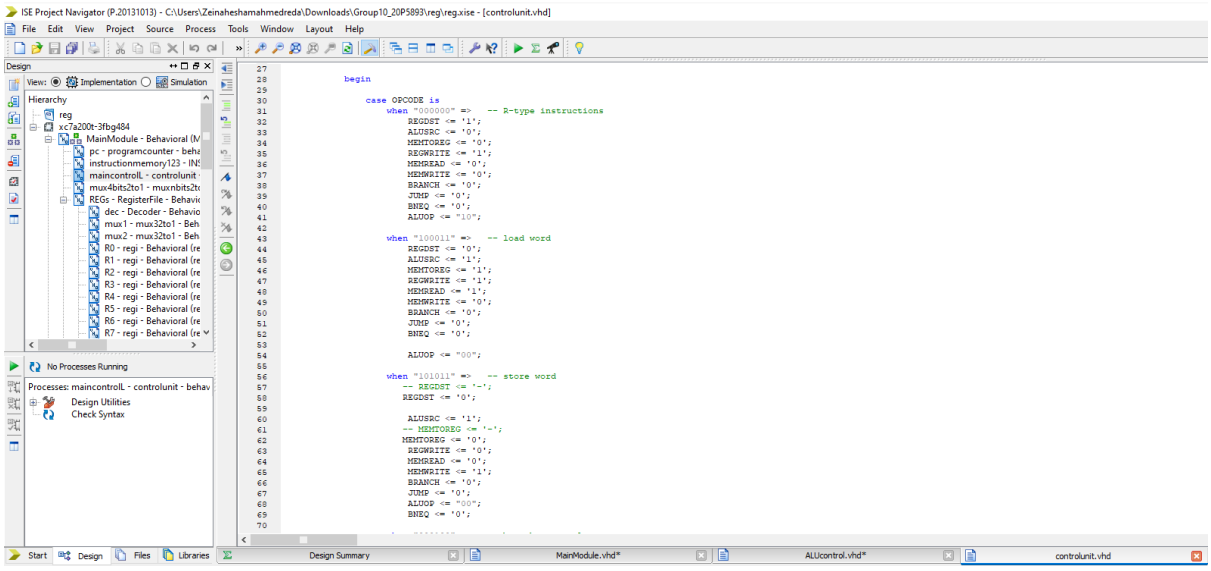
```
mux4bits2to1 : muxnbits2to1 GENERIC MAP(5) PORT MAP(instructionmemoryOutt(20 downto 16),instructionmemoryOutt(15 downto 11),RegDst, WRADDRESS);

REGs : RegisterFile GENERIC MAP(32) PORT MAP(START,instructionmemoryOutt(25 downto 21),instructionmemoryOutt(20 downto 16),
WRADDRESS, Regwrite, CLK, WRDATA, readdata1, readdata2);

signextend : SignExtender Port MAp((instructionmemoryOutt(15 downto 0)), SIGNEXTENDOUT);

mux32bits2to1 : muxnbits2to1 GENERIC MAP(32) PORT MAP(readdata2, SIGNEXTENDOUT, ALUsource, ALUinput);

ALUcont : ALUcontrol Port MAp(ALUop,ALUOPCODE,(instructionmemoryOutt(5 downto 0)));

ArithmeticALU : ALU Port MAP(readdata1,ALUinput,ALUOPCODE,'0',ALUoutt,cf,zf,ovf);

DATAMEM: DATAMEMORY generic map( 64,32,32) port map(startdash,readdata2,DataMemOutt,Memread,Memwrite,ALUoutt,clk);

mux32bits2to1_1 : muxnbits2to1 GENERIC MAP(32) PORT MAP(ALUoutt, DatamemOutt, MemtoReg, WRDATA);

add1:ADDER GENERIC MAP(32) PORT MAP(PCOutt,x"00000004",PCadderoutt);

add2:ADDER GENERIC MAP(32) PORT MAP(shiftleft,PCadderoutt,adder2outt);

muxafterradder: muxnbits2to1 Generic MAp(32) PORT MAP(PCadderoutt,adder2outt,bneqlogic,signaloutfrommux);

muxaftermux: muxnbits2to1 GENERIC MAP (32) PORT MAP (signaloutfrommux,concat,jump,pcin);

RegFileOut1<= readdata1;
RegFileOut2<= readdata2;
ALUout<= ALUoutt;
PCout<= PCoutt;
DataMemOut<= DataMemOutt;

end Behavioral;
```

```
Process "Check Syntax" completed successfully
```

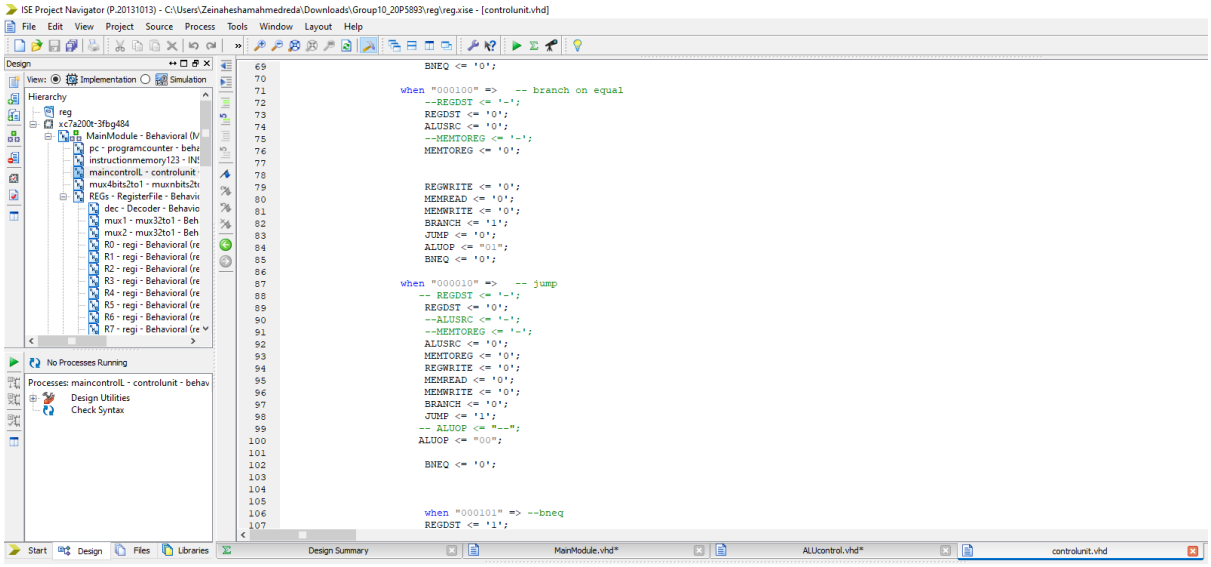
3.2 Main Control Unit

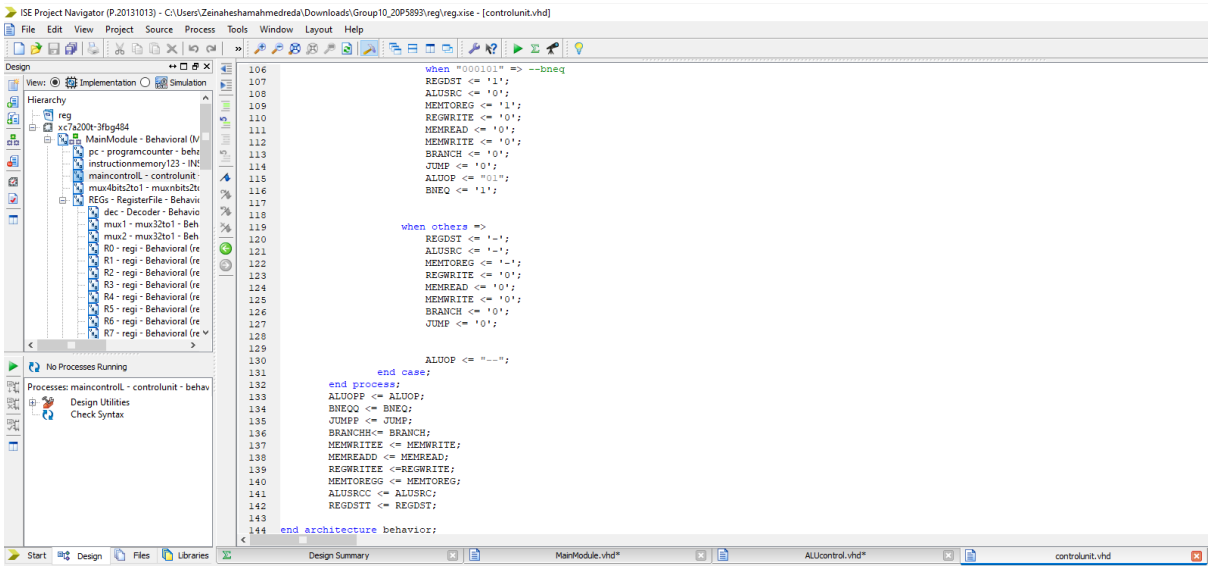


```
-- main control unit

library ieee;
use ieee.std_logic_1164.all;


entity controlunit is
    port(
          OPCODEE : in std_logic_vector(5 downto 0);
          REGDSTT, ALUSRCC, MEMTOREGG, REGWRITEE, MEMREADD : out std_logic;
          MEMWRITEE, BRANCHH, JUMPP, BNEQQ  : out std_logic;
         ALUOPP : out std_logic_vector(1 downto 0)
    );
end entity controlunit;


architecture behavior of controlunit is

        signal OPCODE : STD_LOGIC_VECTOR(5 downto 0);
        SIGNAL REGDST, ALUSRC, MEMTOREG, REGWRITE, MEMREAD , BNEQ :  std_logic;
        SIGNAL MEMWRITE, BRANCH, JUMP :  std_logic;
        signal ALUOP : std_logic_vector(1 downto 0);

        begin
        OPCODE <= OPCODEE;
        process(OPCODE)

            begin

                case OPCODE is
                    when "000000" =>   -- R-type instructions
```

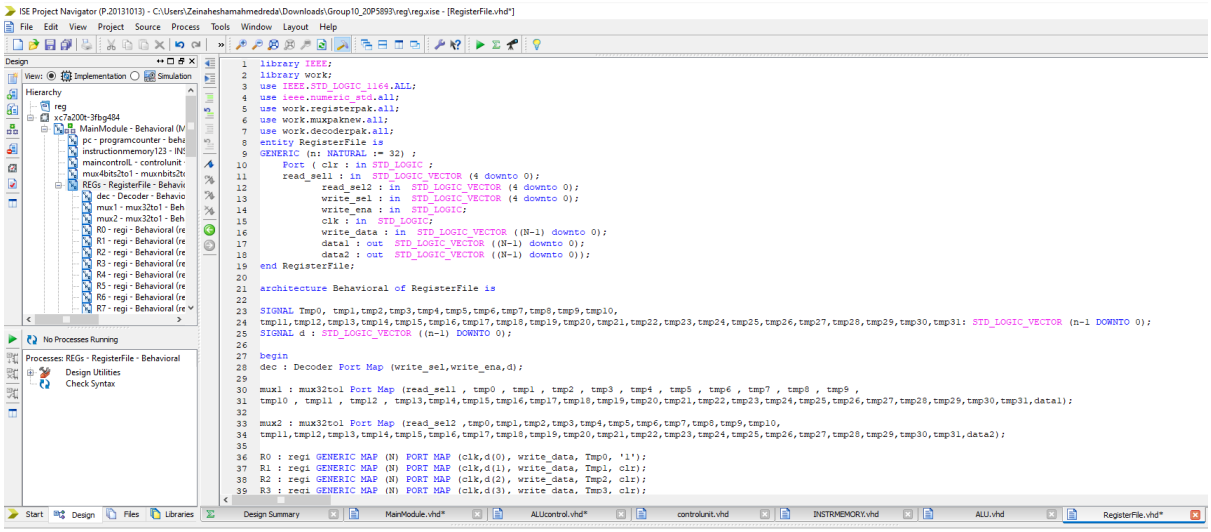


```
        begin

            case OPCODE is
                when "000000" =>   -- R-type instructions
                    REGDST <= '1';
                    ALUSRC <= '0';
                    MEMTOREG <= '0';
                    REGWRITE <= '1';
                    MEMREAD <= '0';
                    MEMWRITE <= '0';
                    BRANCH <= '0';
                    JUMP <= '0';
                    BNEQ <= '0';
                    ALUOP <= "10";

                when "100011" =>   -- load word
                    REGDST <= '0';
                    ALUSRC <= '1';
                    MEMTOREG <= '1';
                    REGWRITE <= '1';
                    MEMREAD <= '1';
                    MEMWRITE <= '0';
                    BRANCH <= '0';
                    JUMP <= '0';
                    BNEQ <= '0';

                    ALUOP <= "00";

                when "101011" =>   -- store word
                   -- REGDST <= '-';
                  REGDST <= '0';

                    ALUSRC <= '1';
                  -- MEMTOREG <= '-';
                  MEMTOREG <= '0';
                   REGWRITE <= '0';
                   MEMREAD <= '0';
                   MEMWRITE <= '1';
                   BRANCH <= '0';
                   JUMP <= '0';
                   ALUOP <= "00";
                   BNEQ <= '0';
```

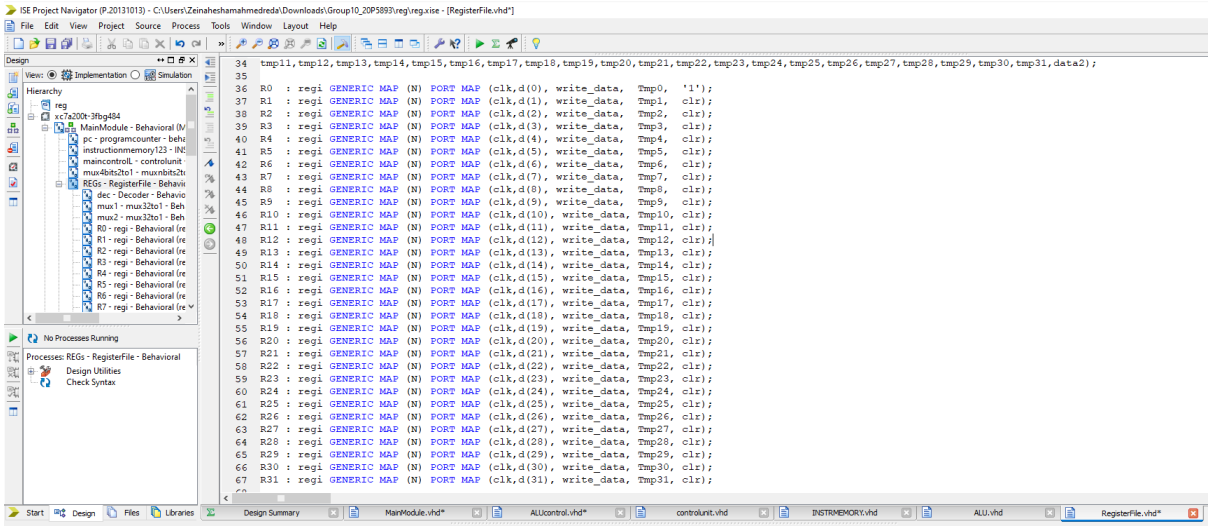

ISE Project Navigator (P.20131013) - C:\Users\Zeinaheshamahmedreda\Downloads\Group10_20P5893\reg\reg.xise - [controlunit.vhd]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
reg
xc7a200t-3fbg484
MainModule - Behavioral (M
pc - programcounter - beha
instructionmemory123 - INS
maincontrolL - controlunit -
mux4bits2to1 - muxnbits2t
REGs - RegisterFile - Behavic
dec - Decoder - Behavio
mux1 - mux32to1 - Beh
mux2 - mux32to1 - Beh
R0 - regi - Behavioral (re
R1 - regi - Behavioral (re
R2 - regi - Behavioral (re
R3 - regi - Behavioral (re
R4 - regi - Behavioral (re
R5 - regi - Behavioral (re
R6 - regi - Behavioral (re
R7 - regi - Behavioral (re
No Processes Running
Processes: maincontrolL - controlunit - behav
Design Utilities
Check Syntax
Start
Design
Files
Libraries
Design Summary
MainModule.vhd*
ALUcontrol.vhd*
controlunit.vhd
69 BNEQ <= '0';
70
71 when "000100" => -- branch on equal
72 --REGDST <= '-';
73 REGDST <= '0';
74 ALUSRC <= '0';
75 --MEMTOREG <= '-';
76 MEMTOREG <= '0';
77
78
79 REGWRITE <= '0';
80 MEMREAD <= '0';
81 MEMWRITE <= '0';
82 BRANCH <= '1';
83 JUMP <= '0';
84 ALUOP <= "01";
85 BNEQ <= '0';
86
87 when "000010" => -- jump
88 -- REGDST <= '-';
89 REGDST <= '0';
90 --ALUSRC <= '-';
91 --MEMTOREG <= '-';
92 ALUSRC <= '0';
93 MEMTOREG <= '0';
94 REGWRITE <= '0';
95 MEMREAD <= '0';
96 MEMWRITE <= '0';
97 BRANCH <= '0';
98 JUMP <= '1';
99 -- ALUOP <= "--";
100 ALUOP <= "00";
101
102 BNEQ <= '0';
103
104
105
106 when "000101" => --bneq
107 REGDST <= '1';


ISE Project Navigator (P.20131013) - C:\Users\Zeinaheshamahmedreda\Downloads\Group10_20P5893\reg\reg.xise - [controlunit.vhd]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
reg
xc7a200t-3fbg484
MainModule - Behavioral (M
pc - programcounter - beha
instructionmemory123 - INS
maincontrolL - controlunit -
mux4bits2to1 - muxnbits2t
REGs - RegisterFile - Behavic
dec - Decoder - Behavio
mux1 - mux32to1 - Beh
mux2 - mux32to1 - Beh
R0 - regi - Behavioral (re
R1 - regi - Behavioral (re
R2 - regi - Behavioral (re
R3 - regi - Behavioral (re
R4 - regi - Behavioral (re
R5 - regi - Behavioral (re
R6 - regi - Behavioral (re
R7 - regi - Behavioral (re
No Processes Running
Processes: maincontrolL - controlunit - behav
Design Utilities
Check Syntax
Start
Design
Files
Libraries
Design Summary
MainModule.vhd*
ALUcontrol.vhd*
controlunit.vhd
106 when "000101" => --bneq
107 REGDST <= '1';
108 ALUSRC <= '0';
109 MEMTOREG <= '1';
110 REGWRITE <= '0';
111 MEMREAD <= '0';
112 MEMWRITE <= '0';
113 BRANCH <= '0';
114 JUMP <= '0';
115 ALUOP <= "01";
116 BNEQ <= '1';
117
118
119 when others =>
120 REGDST <= '-';
121 ALUSRC <= '-';
122 MEMTOREG <= '-';
123 REGWRITE <= '0';
124 MEMREAD <= '0';
125 MEMWRITE <= '0';
126 BRANCH <= '0';
127 JUMP <= '0';
128
129
130 ALUOP <= "--";
131 end case;
132 end process;
133 ALUOPP <= ALUOP;
134 BNEQQ <= BNEQ;
135 JUMPP <= JUMP;
136 BRANCHH<= BRANCH;
137 MEMWRITEE <= MEMWRITE;
138 MEMREADD <= MEMREAD;
139 REGWRITEE <=REGWRITE;
140 MEMTOREGG <= MEMTOREG;
141 ALUSRCC <= ALUSRC;
142 REGDSTT <= REGDST;
143
144 end architecture behavior;

3.3 Register File

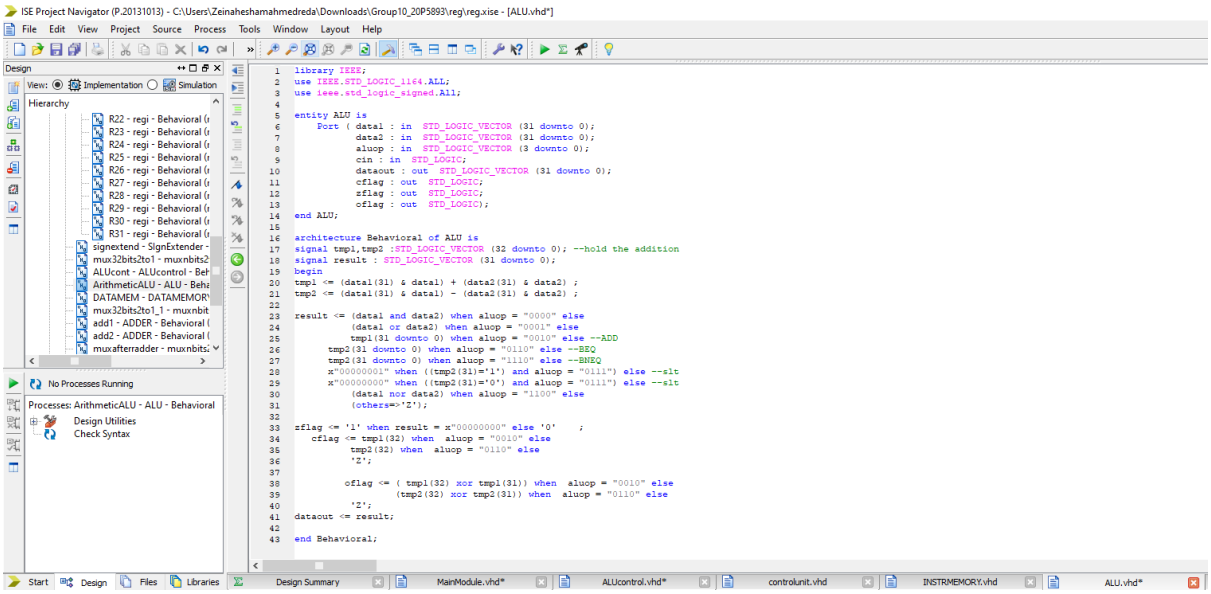


```
library IEEE;
library work;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;
use work.registerpak.all;
use work.muxpaknew.all;
use work.decoderpak.all;
entity RegisterFile is
GENERIC (n: NATURAL := 32) ;
    Port ( clr : in STD_LOGIC ;
    read_sel1 : in  STD_LOGIC_VECTOR (4 downto 0);
           read_sel2 : in  STD_LOGIC_VECTOR (4 downto 0);
           write_sel : in  STD_LOGIC_VECTOR (4 downto 0);
           write_ena : in  STD_LOGIC;
           clk : in  STD_LOGIC;
           write_data : in  STD_LOGIC_VECTOR ((N-1) downto 0);
           data1 : out  STD_LOGIC_VECTOR ((N-1) downto 0);
           data2 : out  STD_LOGIC_VECTOR ((N-1) downto 0));
end RegisterFile;

architecture Behavioral of RegisterFile is

SIGNAL Tmp0, tmp1,tmp2,tmp3,tmp4,tmp5,tmp6,tmp7,tmp8,tmp9,tmp10,
tmp11,tmp12,tmp13,tmp14,tmp15,tmp16,tmp17,tmp18,tmp19,tmp20,tmp21,tmp22,tmp23,tmp24,tmp25,tmp26,tmp27,tmp28,tmp29,tmp30,tmp31: STD_LOGIC_VECTOR (n-1 DOWNTO 0);
SIGNAL d : STD_LOGIC_VECTOR ((n-1) DOWNTO 0);

begin
dec : Decoder Port Map (write_sel,write_ena,d);

mux1 : mux32to1 Port Map (read_sel1 , tmp0 , tmp1 , tmp2 , tmp3 , tmp4 , tmp5 , tmp6 , tmp7 , tmp8 , tmp9 ,
tmp10 , tmp11 , tmp12 , tmp13,tmp14,tmp15,tmp16,tmp17,tmp18,tmp19,tmp20,tmp21,tmp22,tmp23,tmp24,tmp25,tmp26,tmp27,tmp28,tmp29,tmp30,tmp31,data1);

mux2 : mux32to1 Port Map (read_sel2 ,tmp0,tmp1,tmp2,tmp3,tmp4,tmp5,tmp6,tmp7,tmp8,tmp9,tmp10,
tmp11,tmp12,tmp13,tmp14,tmp15,tmp16,tmp17,tmp18,tmp19,tmp20,tmp21,tmp22,tmp23,tmp24,tmp25,tmp26,tmp27,tmp28,tmp29,tmp30,tmp31,data2);

R0 : regi GENERIC MAP (N) PORT MAP (clk,d(0), write_data, Tmp0, '1');
R1 : regi GENERIC MAP (N) PORT MAP (clk,d(1), write_data, Tmp1, clr);
R2 : regi GENERIC MAP (N) PORT MAP (clk,d(2), write_data, Tmp2, clr);
R3 : regi GENERIC MAP (N) PORT MAP (clk,d(3), write_data, Tmp3, clr);
```



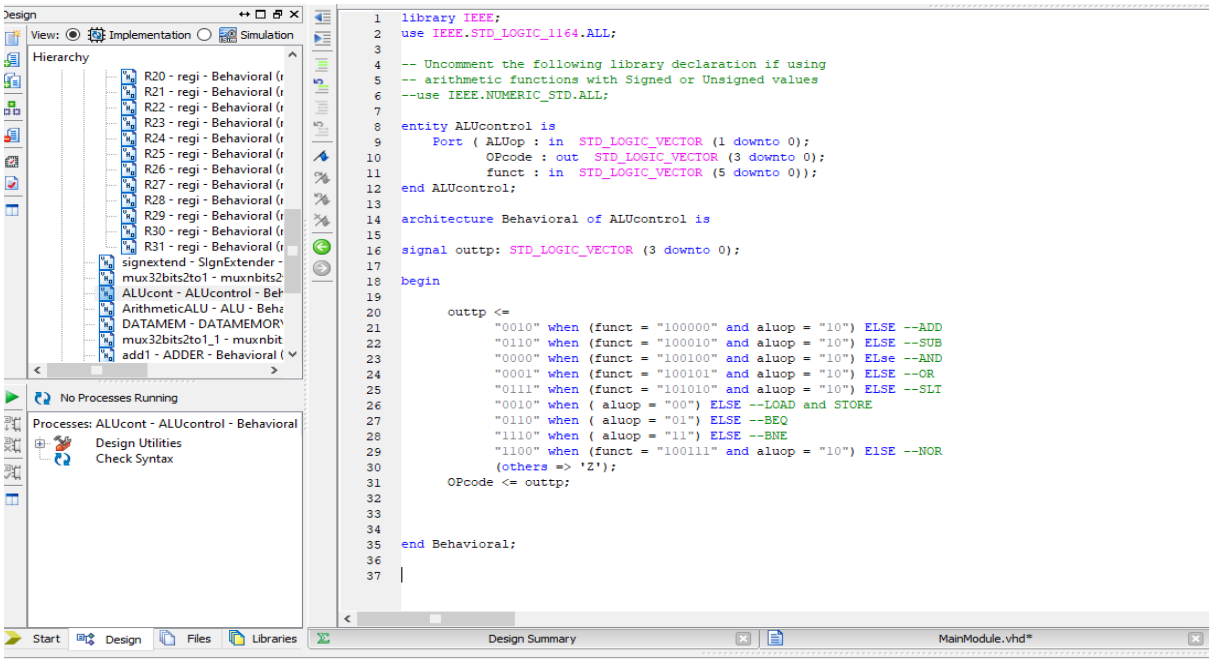
```
tmp11,tmp12,tmp13,tmp14,tmp15,tmp16,tmp17,tmp18,tmp19,tmp20,tmp21,tmp22,tmp23,tmp24,tmp25,tmp26,tmp27,tmp28,tmp29,tmp30,tmp31,data2);

R0  : regi GENERIC MAP (N) PORT MAP (clk,d(0), write_data,  Tmp0,  '1');
R1  : regi GENERIC MAP (N) PORT MAP (clk,d(1), write_data,  Tmp1,  clr);
R2  : regi GENERIC MAP (N) PORT MAP (clk,d(2), write_data,  Tmp2,  clr);
R3  : regi GENERIC MAP (N) PORT MAP (clk,d(3), write_data,  Tmp3,  clr);
R4  : regi GENERIC MAP (N) PORT MAP (clk,d(4), write_data,  Tmp4,  clr);
R5  : regi GENERIC MAP (N) PORT MAP (clk,d(5), write_data,  Tmp5,  clr);
R6  : regi GENERIC MAP (N) PORT MAP (clk,d(6), write_data,  Tmp6,  clr);
R7  : regi GENERIC MAP (N) PORT MAP (clk,d(7), write_data,  Tmp7,  clr);
R8  : regi GENERIC MAP (N) PORT MAP (clk,d(8), write_data,  Tmp8,  clr);
R9  : regi GENERIC MAP (N) PORT MAP (clk,d(9), write_data,  Tmp9,  clr);
R10 : regi GENERIC MAP (N) PORT MAP (clk,d(10), write_data, Tmp10, clr);
R11 : regi GENERIC MAP (N) PORT MAP (clk,d(11), write_data, Tmp11, clr);
R12 : regi GENERIC MAP (N) PORT MAP (clk,d(12), write_data, Tmp12, clr);
R13 : regi GENERIC MAP (N) PORT MAP (clk,d(13), write_data, Tmp13, clr);
R14 : regi GENERIC MAP (N) PORT MAP (clk,d(14), write_data, Tmp14, clr);
R15 : regi GENERIC MAP (N) PORT MAP (clk,d(15), write_data, Tmp15, clr);
R16 : regi GENERIC MAP (N) PORT MAP (clk,d(16), write_data, Tmp16, clr);
R17 : regi GENERIC MAP (N) PORT MAP (clk,d(17), write_data, Tmp17, clr);
R18 : regi GENERIC MAP (N) PORT MAP (clk,d(18), write_data, Tmp18, clr);
R19 : regi GENERIC MAP (N) PORT MAP (clk,d(19), write_data, Tmp19, clr);
R20 : regi GENERIC MAP (N) PORT MAP (clk,d(20), write_data, Tmp20, clr);
R21 : regi GENERIC MAP (N) PORT MAP (clk,d(21), write_data, Tmp21, clr);
R22 : regi GENERIC MAP (N) PORT MAP (clk,d(22), write_data, Tmp22, clr);
R23 : regi GENERIC MAP (N) PORT MAP (clk,d(23), write_data, Tmp23, clr);
R24 : regi GENERIC MAP (N) PORT MAP (clk,d(24), write_data, Tmp24, clr);
R25 : regi GENERIC MAP (N) PORT MAP (clk,d(25), write_data, Tmp25, clr);
R26 : regi GENERIC MAP (N) PORT MAP (clk,d(26), write_data, Tmp26, clr);
R27 : regi GENERIC MAP (N) PORT MAP (clk,d(27), write_data, Tmp27, clr);
R28 : regi GENERIC MAP (N) PORT MAP (clk,d(28), write_data, Tmp28, clr);
R29 : regi GENERIC MAP (N) PORT MAP (clk,d(29), write_data, Tmp29, clr);
R30 : regi GENERIC MAP (N) PORT MAP (clk,d(30), write_data, Tmp30, clr);
R31 : regi GENERIC MAP (N) PORT MAP (clk,d(31), write_data, Tmp31, clr);
```

3.4 ALU



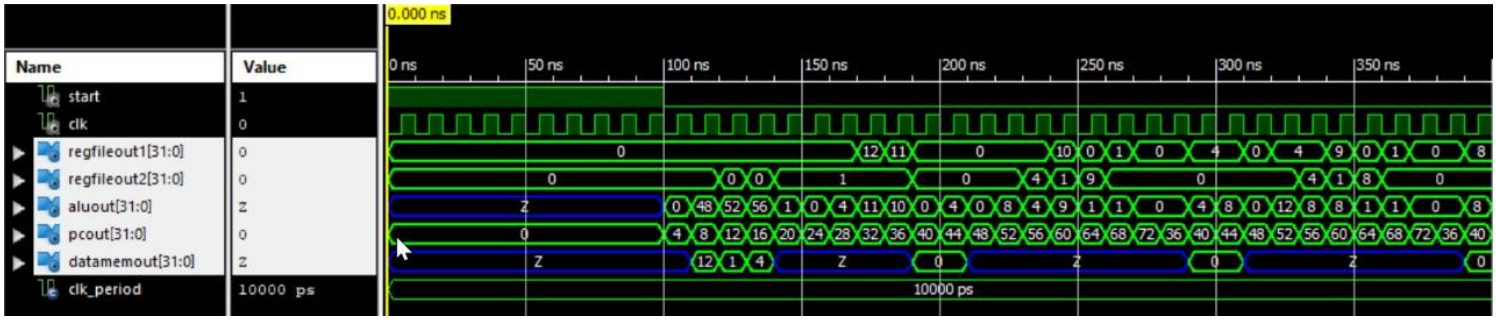
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_signed.All;

entity ALU is
    Port ( data1 : in  STD_LOGIC_VECTOR (31 downto 0);
           data2 : in  STD_LOGIC_VECTOR (31 downto 0);
           aluop : in  STD_LOGIC_VECTOR (3 downto 0);
           cin : in  STD_LOGIC;
           dataout : out  STD_LOGIC_VECTOR (31 downto 0);
           cflag : out  STD_LOGIC;
           zflag : out  STD_LOGIC;
           oflag : out  STD_LOGIC);
end ALU;

architecture Behavioral of ALU is
signal tmp1,tmp2 :STD_LOGIC_VECTOR (32 downto 0); --hold the addition
signal result : STD_LOGIC_VECTOR (31 downto 0);
begin
tmp1 <= (data1(31) & data1) + (data2(31) & data2) ;
tmp2 <= (data1(31) & data1) - (data2(31) & data2) ;

result <= (data1 and data2) when aluop = "0000" else
          (data1 or data2) when aluop = "0001" else
          tmp1(31 downto 0) when aluop = "0010" else --ADD
      tmp2(31 downto 0) when aluop = "0110" else --BEQ
      tmp2(31 downto 0) when aluop = "1110" else --BNEQ
      x"00000001" when ((tmp2(31)='1') and aluop = "0111") else --slt
      x"00000000" when ((tmp2(31)='0') and aluop = "0111") else --slt
          (data1 nor data2) when aluop = "1100" else
          (others=>'Z');

zflag <= '1' when result = x"00000000" else '0'    ;
   cflag <= tmp1(32) when  aluop = "0010" else
          tmp2(32) when  aluop = "0110" else
          'Z';

         oflag <= ( tmp1(32) xor tmp1(31)) when  aluop = "0010" else
                  (tmp2(32) xor tmp2(31)) when  aluop = "0110" else
           'Z';
dataout <= result;

end Behavioral;
```

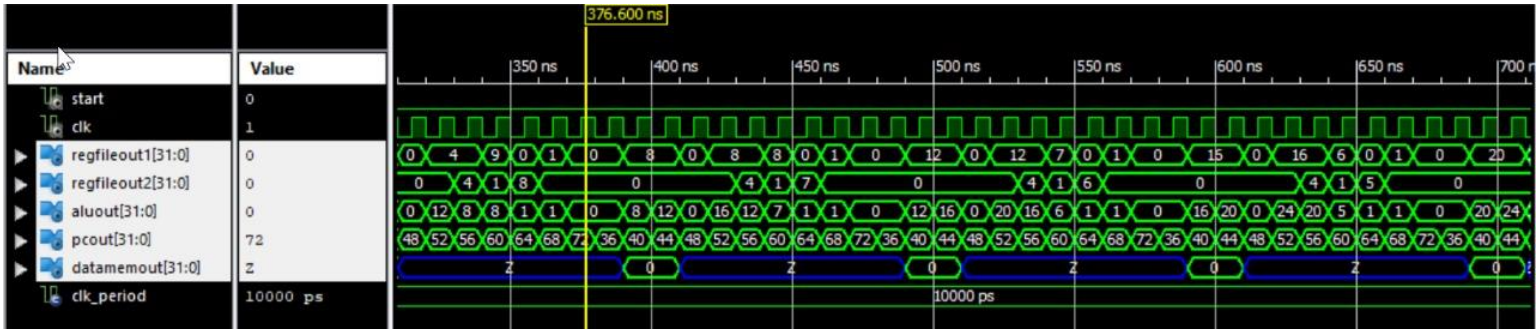
3.5 ALU Control

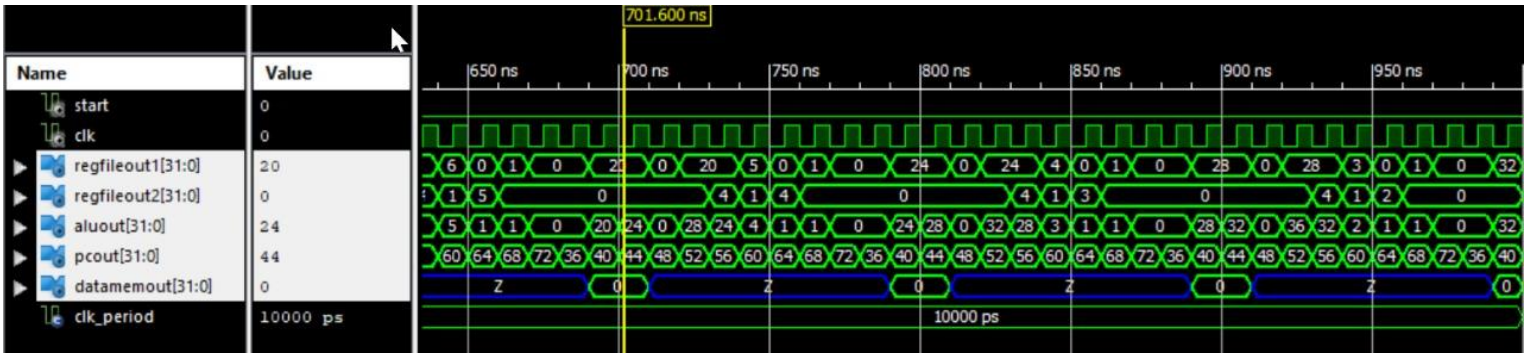


```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

entity ALUcontrol is
    Port ( ALUop : in  STD_LOGIC_VECTOR (1 downto 0);
           OPcode : out  STD_LOGIC_VECTOR (3 downto 0);
           funct : in  STD_LOGIC_VECTOR (5 downto 0));
end ALUcontrol;

architecture Behavioral of ALUcontrol is

signal outtp: STD_LOGIC_VECTOR (3 downto 0);

begin

      outtp <=
            "0010" when (funct = "100000" and aluop = "10") ELSE --ADD
            "0110" when (funct = "100010" and aluop = "10") ELSE --SUB
            "0000" when (funct = "100100" and aluop = "10") ELse --AND
            "0001" when (funct = "100101" and aluop = "10") ELSE --OR
            "0111" when (funct = "101010" and aluop = "10") ELSE --SLT
            "0010" when ( aluop = "00") ELSE --LOAD and STORE
            "0110" when ( aluop = "01") ELSE --BEQ
            "1110" when ( aluop = "11") ELSE --BNE
            "1100" when (funct = "100111" and aluop = "10") ELSE --NOR
            (others => 'Z');
      OPcode <= outtp;


end Behavioral;
```

3.6 Test Bench







4.Contribution

Name	Contribution
Mazen Mohamed Elsaied (**Team leader**) 20P5893	**20%**
George Emad Welson 20P3831	**20%**
Farida Mohamed El-Husseiny 20P6022	**20%**
Zeina Hesham Reda 2000320	**20%**
Hassan Mohamed El-Tobgy 20P6173	**20%**