

CS 342000 | CS343000 - Spring 2022
Instructor: Professor Izidor Gertner
Spring 2022 – 5/22/22
Single Cycle CPU - Azwad Shameem

Table of Contents

Objective	3
Components	4
Package	4
Instruction Memory	7
AdderSub	8
Extender	8
Mux	9
Bitwise Operations	10
Adder	11
Equal	11
Data Memory	12
Register File	13
Instruction Register	14
PC Register	15
Control Unit	16
Single Cycle CPU	19
Simulation.....	21
MIPS in Mars Simulator.....	32
Conclusion.....	34

Objective

The objective of the Single Cycle CPU lab was to write a program to compute the sum of five integers. Furthermore, we had to prove the correctness of the simulation using MIPS instructions in MARS Simulator. In order to simulate MIPS instructions in VHDL we needed to create data memory, instruction memory, 3-ported register file, IR-Instruction Register, Program Counter, AdderSub, Extender, and multiplexers components. Not to mention it was also imperative to implement a control unit that generated the control signals and regulated what action should be taken. Therefore, by completing the previous steps it became possible to simulate the cycle that an instruction takes when executing in the CPU such as reading the instruction from memory, breaking it down to specific parts and then executing the instruction and also determining the address of PC.

Components Package

```

library IEEE;
use IEEE.STD_LOGIC_1164.all;

package Shameen_5_22_22_Package is
    component Shameen_5_22_22_InstructionMemory is
        port (
            Shameen_5_22_22_Clock : IN STD_LOGIC;
            Shameen_5_22_22_Opcode : OUT STD_LOGIC_VECTOR(31 DOWNTO 0);
            Shameen_5_22_22_Address : IN STD_LOGIC_VECTOR(4 DOWNTO 0);
            Shameen_5_22_22_Funct : OUT STD_LOGIC_VECTOR(4 DOWNTO 0);
            Shameen_5_22_22_Mux1 : OUT STD_LOGIC_VECTOR(15 DOWNTO 0);
            Shameen_5_22_22_Instruction : OUT STD_LOGIC_VECTOR(31 DOWNTO 0)
        );
    end component;
    component Shameen_5_22_22_InstructionPort is
        port (
            Shameen_5_22_22_Address : IN STD_LOGIC_VECTOR(5 DOWNTO 0);
            Shameen_5_22_22_Clock : IN STD_LOGIC;
            Shameen_5_22_22_Opcode : OUT STD_LOGIC_VECTOR(5 DOWNTO 0);
            Shameen_5_22_22_Regist : Shameen_5_22_22_Extop, Shameen_5_22_22_ALUSRC, Shameen_5_22_22_ALUCLTR, Shameen_5_22_22_Mem,
            Shameen_5_22_22_Wren : IN STD_LOGIC;
            Shameen_5_22_22_Q : OUT STD_LOGIC_VECTOR(31 DOWNTO 0)
        );
    end component;
    component Shameen_5_22_22_ControlUnit is
        port (
            Shameen_5_22_22_Opcode : IN STD_LOGIC_VECTOR(5 DOWNTO 0);
            Shameen_5_22_22_Regist : Shameen_5_22_22_Extop, Shameen_5_22_22_ALUSRC, Shameen_5_22_22_ALUCLTR, Shameen_5_22_22_Mem,
            Shameen_5_22_22_Wren : IN STD_LOGIC;
            Shameen_5_22_22_Extop : OUT STD_LOGIC;
            Shameen_5_22_22_Input : IN STD_LOGIC_VECTOR(15 DOWNTO 0);
            Shameen_5_22_22_Extop : IN STD_LOGIC;
            Shameen_5_22_22_Input : IN STD_LOGIC_VECTOR(15 DOWNTO 0)
        );
    end component;
    component Shameen_5_22_22_RegisterFile is
        port (
            Shameen_5_22_22_Clock : IN STD_LOGIC;
            Shameen_5_22_22_Input : IN STD_LOGIC_VECTOR(15 DOWNTO 0);
            Shameen_5_22_22_Output : OUT STD_LOGIC_VECTOR(31 DOWNTO 0)
        );
    end component;
    component Shameen_5_22_22_InstructionRegister is
        port (
            Shameen_5_22_22_Clock : IN STD_LOGIC;
            Shameen_5_22_22_Input : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
            Shameen_5_22_22_Output : OUT STD_LOGIC_VECTOR(31 DOWNTO 0)
        );
    end component;
    component Shameen_5_22_22_Mux2_1_5bit is
        port (
            Shameen_5_22_22_Switch : IN STD_LOGIC;
            Shameen_5_22_22_A : Shameen_5_22_22_B : IN STD_LOGIC_VECTOR(4 DOWNTO 0);
            Shameen_5_22_22_Output : OUT STD_LOGIC_VECTOR(4 DOWNTO 0)
        );
    end component;
    component Shameen_5_22_22_TriplePortRam is
        port (
            Shameen_5_22_22_Clock : IN STD_LOGIC;
            Shameen_5_22_22_Address1 : IN STD_LOGIC_VECTOR(4 DOWNTO 0);
            Shameen_5_22_22_Address2 : IN STD_LOGIC_VECTOR(4 DOWNTO 0);
            Shameen_5_22_22_Address3 : IN STD_LOGIC_VECTOR(4 DOWNTO 0);
            Shameen_5_22_22_Data : OUT STD_LOGIC_VECTOR(31 DOWNTO 0)
        );
    end component;
end package;

```

Figure 1: Package VHDL code. This package contains all of the components used.

```

component Shameen_5_22_22_TriplePortRam is
    port (
        Shameen_5_22_22_Clock : IN STD_LOGIC;
        Shameen_5_22_22_Address1 : IN STD_LOGIC_VECTOR(4 DOWNTO 0);
        Shameen_5_22_22_Address2 : IN STD_LOGIC_VECTOR(4 DOWNTO 0);
        Shameen_5_22_22_Address3 : IN STD_LOGIC_VECTOR(4 DOWNTO 0);
        Shameen_5_22_22_Data : OUT STD_LOGIC_VECTOR(31 DOWNTO 0)
    );
end component;
end package;

```

Figure 2: Package VHDL code continued.

```

Quartus Prime Lite Edition - C:/Users/Azwad/Desktop/CPU/Shameem_5_22_22_CPU/Shameem_5_22_22_CPU - Shameem_5_22_22_CPU
File Edit View Project Assignments Processing Tools Window Help
Project Navigator Files Compilation Report - Shameem_5_22_22_CPU Shameem_5_22_22_Package.vhd IP Catalog
Search altera.com
Files
Shameem_5_22_22_CPU.vhd
Shameem_5_22_22_Package.vhd
Shameem_5_22_22_InstructionMemory.vhd
Shameem_5_22_22_InstructionPort.vhd
Shameem_5_22_22_DualPortRAM.vhd
Shameem_5_22_22_ControlUnit.vhd
Shameem_5_22_22_Extender.vhd
Shameem_5_22_22_InstructionRegister.vhd
Shameem_5_22_22_Mux2_1_Slot.vhd
Shameem_5_22_22_RegisterFile.vhd
Shameem_5_22_22_DualPortRAM.vhd
Shameem_5_22_22_TriplePortRAM.vhd
Shameem_5_22_22_Mux2_1.vhd
Tasks Compilation
Compilation Task
Compile Design
Analysis & Synthesis
Filter Place & Route
Assembler (Generate program)
Timing Analysis
EDA Netlist Writer
Edit Settings
Shameem_5_22_22_TriplePortRAM IS
PORT
  IN component: shameem_5_22_22_Clock: in STD_LOGIC := "1";
  IN component: shameem_5_22_22_Data: in STD_LOGIC_VECTOR(31 DOWNTO 0);
  IN component: shameem_5_22_22_RdAddress: in STD_LOGIC_VECTOR(4 DOWNTO 0);
  OUT component: shameem_5_22_22_WrAddress: out STD_LOGIC_VECTOR(4 DOWNTO 0);
  OUT component: shameem_5_22_22_WrData: out STD_LOGIC_VECTOR(31 DOWNTO 0);
  OUT component: shameem_5_22_22_O1: out STD_LOGIC_VECTOR(31 DOWNTO 0);
  OUT component: shameem_5_22_22_O2: out STD_LOGIC_VECTOR(31 DOWNTO 0);
end component;
component: shameem_5_22_22_Mux2_1 IS
PORT
  IN component: shameem_5_22_22_Switch: in STD_LOGIC;
  IN component: shameem_5_22_22_A, shameem_5_22_22_B: in STD_LOGIC_VECTOR(31 DOWNTO 0);
  OUT component: shameem_5_22_22_Output: out STD_LOGIC_VECTOR(31 DOWNTO 0);
end component;
component: shameem_5_22_22_ALU IS
generic(N : integer := 32);
PORT
  IN component: shameem_5_22_22_ALUOp: in STD_LOGIC;
  IN component: shameem_5_22_22_OpCode: in STD_LOGIC_VECTOR(3 DOWNTO 0);
  IN component: shameem_5_22_22_Funct: in STD_LOGIC_VECTOR(0 DOWNTO 0);
  IN component: shameem_5_22_22_A, shameem_5_22_22_B: in STD_LOGIC_VECTOR(N-1 DOWNTO 0);
  OUT component: shameem_5_22_22_Sum: out STD_LOGIC_VECTOR(N-1 DOWNTO 0);
  OUT component: shameem_5_22_22_Overflow, shameem_5_22_22_Zero, shameem_5_22_22_Negative, shameem_5_22_22_PCsre, shameem_5_22_22_Branch: out STD_LOGIC;
end component;
component: shameem_5_22_22_DotMemory IS
PORT
  IN component: shameem_5_22_22_Clock, shameem_5_22_22_MemOp: in std_logic;
  IN component: shameem_5_22_22_Address: in std_logic_vector(11 DOWNTO 0);
  OUT component: shameem_5_22_22_Output: out std_logic_vector(31 DOWNTO 0);
end component;
component: shameem_5_22_22_DotMemory
  generic(N : integer := 32);
  PORT
    IN component: shameem_5_22_22_Clock, shameem_5_22_22_MemOp: in std_logic;
    IN component: shameem_5_22_22_Address: in std_logic_vector(11 DOWNTO 0);
    OUT component: shameem_5_22_22_Output: out std_logic_vector(31 DOWNTO 0);
  end component;
  generic(N : integer := 32);
  PORT
    IN component: shameem_5_22_22_Clock, shameem_5_22_22_MemOp: in std_logic;
    IN component: shameem_5_22_22_Address: in std_logic_vector(11 DOWNTO 0);
    OUT component: shameem_5_22_22_Output: out std_logic_vector(31 DOWNTO 0);
  end component;
end component;
component: shameem_5_22_22_Adder IS
PORT
  IN component: shameem_5_22_22_DataA, shameem_5_22_22_DataB: in STD_LOGIC_VECTOR(31 DOWNTO 0);
  OUT component: shameem_5_22_22_Result: out STD_LOGIC_VECTOR(31 DOWNTO 0);
end component;
component: shameem_5_22_22_Mux3_1 IS
PORT
  IN component: shameem_5_22_22_PCsre, shameem_5_22_22_Branch: in STD_LOGIC;
  IN component: shameem_5_22_22_A, shameem_5_22_22_B, shameem_5_22_22_C: in std_logic_vector(31 DOWNTO 0);
  OUT component: shameem_5_22_22_Output: out std_logic_vector(31 DOWNTO 0);
end component;
component: shameem_5_22_22_PC_Register IS
PORT
  IN component: shameem_5_22_22_Clock: in STD_LOGIC;
  IN component: shameem_5_22_22_PcInput: in std_logic_vector(31 DOWNTO 0);
  OUT component: shameem_5_22_22_PcOutput: out std_logic_vector(31 DOWNTO 0);
end component;
component: shameem_5_22_22_AddSub IS
generic(N : Integer := 32);
PORT
  IN component: shameem_5_22_22_Operation: in STD_LOGIC;
  IN component: shameem_5_22_22_A, shameem_5_22_22_B: in std_logic_vector(31 DOWNTO 0);
  OUT component: shameem_5_22_22_Result: out std_logic_vector(31 DOWNTO 0);
end component;

```

Messages System Processing (404)

Figure 3: Package VHDL code continued.

```

Quartus Prime Lite Edition - C:/Users/Azwad/Desktop/CPU/Shameem_5_22_22_CPU/Shameem_5_22_22_CPU - Shameem_5_22_22_CPU
File Edit View Project Assignments Processing Tools Window Help
Project Navigator Files Compilation Report - Shameem_5_22_22_CPU Shameem_5_22_22_Package.vhd IP Catalog
Search altera.com
Files
Shameem_5_22_22_CPU.vhd
Shameem_5_22_22_Package.vhd
Shameem_5_22_22_InstructionMemory.vhd
Shameem_5_22_22_InstructionPort.vhd
Shameem_5_22_22_DualPortRAM.vhd
Shameem_5_22_22_ControlUnit.vhd
Shameem_5_22_22_Extender.vhd
Shameem_5_22_22_InstructionRegister.vhd
Shameem_5_22_22_Mux2_1_Slot.vhd
Shameem_5_22_22_RegisterFile.vhd
Shameem_5_22_22_DualPortRAM.vhd
Shameem_5_22_22_TriplePortRAM.vhd
Shameem_5_22_22_Mux2_1.vhd
Tasks Compilation
Compilation Task
Compile Design
Analysis & Synthesis
Filter Place & Route
Assembler (Generate program)
Timing Analysis
EDA Netlist Writer
Edit Settings
Shameem_5_22_22_TriplePortRAM IS
PORT
  IN component: shameem_5_22_22_Clock: in STD_LOGIC := "1";
  IN component: shameem_5_22_22_Data: in STD_LOGIC_VECTOR(31 DOWNTO 0);
  IN component: shameem_5_22_22_RdAddress: in STD_LOGIC_VECTOR(4 DOWNTO 0);
  OUT component: shameem_5_22_22_WrAddress: out STD_LOGIC_VECTOR(4 DOWNTO 0);
  OUT component: shameem_5_22_22_WrData: out STD_LOGIC_VECTOR(31 DOWNTO 0);
  OUT component: shameem_5_22_22_O1: out STD_LOGIC_VECTOR(31 DOWNTO 0);
  OUT component: shameem_5_22_22_O2: out STD_LOGIC_VECTOR(31 DOWNTO 0);
end component;
component: shameem_5_22_22_Mux2_1 IS
PORT
  IN component: shameem_5_22_22_Switch: in STD_LOGIC;
  IN component: shameem_5_22_22_A, shameem_5_22_22_B: in STD_LOGIC_VECTOR(31 DOWNTO 0);
  OUT component: shameem_5_22_22_Output: out STD_LOGIC_VECTOR(31 DOWNTO 0);
end component;
component: shameem_5_22_22_ALU IS
generic(N : integer := 32);
PORT
  IN component: shameem_5_22_22_ALUOp: in STD_LOGIC;
  IN component: shameem_5_22_22_OpCode: in STD_LOGIC_VECTOR(3 DOWNTO 0);
  IN component: shameem_5_22_22_Funct: in STD_LOGIC_VECTOR(0 DOWNTO 0);
  IN component: shameem_5_22_22_A, shameem_5_22_22_B: in STD_LOGIC_VECTOR(N-1 DOWNTO 0);
  OUT component: shameem_5_22_22_Sum: out STD_LOGIC_VECTOR(N-1 DOWNTO 0);
  OUT component: shameem_5_22_22_Overflow, shameem_5_22_22_Zero, shameem_5_22_22_Negative, shameem_5_22_22_PCsre, shameem_5_22_22_Branch: out STD_LOGIC;
end component;
component: shameem_5_22_22_DotMemory IS
PORT
  IN component: shameem_5_22_22_Clock, shameem_5_22_22_MemOp: in std_logic;
  IN component: shameem_22_22_Address: in std_logic_vector(11 DOWNTO 0);
  OUT component: shameem_5_22_22_Output: out std_logic_vector(31 DOWNTO 0);
end component;
component: shameem_5_22_22_DotMemory
  generic(N : integer := 32);
  PORT
    IN component: shameem_5_22_22_Clock, shameem_5_22_22_MemOp: in std_logic;
    IN component: shameem_5_22_22_Address: in std_logic_vector(11 DOWNTO 0);
    OUT component: shameem_5_22_22_Output: out std_logic_vector(31 DOWNTO 0);
  end component;
  generic(N : integer := 32);
  PORT
    IN component: shameem_5_22_22_Clock, shameem_5_22_22_MemOp: in std_logic;
    IN component: shameem_5_22_22_Address: in std_logic_vector(11 DOWNTO 0);
    OUT component: shameem_5_22_22_Output: out std_logic_vector(31 DOWNTO 0);
  end component;
end component;
component: shameem_5_22_22_Adder IS
PORT
  IN component: shameem_5_22_22_DataA, shameem_5_22_22_DataB: in STD_LOGIC_VECTOR(31 DOWNTO 0);
  OUT component: shameem_5_22_22_Result: out STD_LOGIC_VECTOR(31 DOWNTO 0);
end component;
component: shameem_5_22_22_Mux3_1 IS
PORT
  IN component: shameem_5_22_22_PCsre, shameem_5_22_22_Branch: in STD_LOGIC;
  IN component: shameem_5_22_22_A, shameem_5_22_22_B, shameem_5_22_22_C: in std_logic_vector(31 DOWNTO 0);
  OUT component: shameem_5_22_22_Output: out std_logic_vector(31 DOWNTO 0);
end component;
component: shameem_5_22_22_PC_Register IS
PORT
  IN component: shameem_5_22_22_Clock: in STD_LOGIC;
  IN component: shameem_5_22_22_PcInput: in std_logic_vector(31 DOWNTO 0);
  OUT component: shameem_5_22_22_PcOutput: out std_logic_vector(31 DOWNTO 0);
end component;
component: shameem_5_22_22_AddSub IS
generic(N : Integer := 32);
PORT
  IN component: shameem_5_22_22_Operation: in STD_LOGIC;
  IN component: shameem_5_22_22_A, shameem_5_22_22_B: in std_logic_vector(31 DOWNTO 0);
  OUT component: shameem_5_22_22_Result: out std_logic_vector(31 DOWNTO 0);
end component;

```

Messages System Processing (404)

Figure 4: Package VHDL code continued.

```

Quartus Prime Lite Edition - C:/Users/Azwad/Desktop/CPU/Shameem_5_22_22_CPU/Shameem_5_22_22_CPU - Shameem_5_22_22_CPU
File Edit View Project Assignments Processing Tools Window Help
Search altera.com
Project Navigator Files Compilation Report - Shameem_5_22_22_CPU Shameem_5_22_22_Package.vhd IP Catalog
Shameem_5_22_22_CPU
126     Shameem_5_22_22_output : out STD_logic_vector(31 downto 0);
127   );
128 end component;
129
130 component Shameem_5_22_22_PC_Register is
131   PORT (
132     Shameem_5_22_22_clock: in STD_LOGIC;
133     Shameem_5_22_22_PCInput: in STD_logic_vector(31 downto 0);
134     Shameem_5_22_22_PCOOutput: out STD_logic_vector(31 downto 0)
135   );
136 end component;
137
138 component Shameem_5_22_22_AddSub is
139   generic (N : Integer := 32);
140   port (
141     Shameem_5_22_22_Operation: in STD_LOGIC;
142     Shameem_5_22_22_A, Shameem_5_22_22_B: in STD_LOGIC_VECTOR(N-1 downto 0);
143     Shameem_5_22_22_Carry, Shameem_5_22_22_Overflow, Shameem_5_22_22_Zero, Shameem_5_22_22_Negative: out STD_LOGIC
144   );
145 end component;
146
147 component Shameem_5_22_22_BitwiseOperations is
148   generic (N: Integer := 32);
149   port (
150     Shameem_5_22_22_Funct: in STD_LOGIC_VECTOR(5 downto 0);
151     Shameem_5_22_22_Input1, Shameem_5_22_22_Input2: in STD_LOGIC_VECTOR(N-1 downto 0);
152   );
153 end component;
154
155 component Shameem_5_22_22_Equal is
156   PORT (
157     dataaa : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
158     databb : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
159     aebl : OUT STD_LOGIC
160   );
161 end component;
162
163 END PACKAGE;
164
165 end Shameem_5_22_22_Package;

```

Tasks

- Compilation
- Task
- Compile Design
- Analysis & Synthesis
- Filter (Place & Route)
- Assembler (Generate program)
- Timing Analysis
- EDA Netlist Writer
- Edit Settings

Messages

Type ID Message

- 332140 No Removal paths to report
- 332140 No Removal paths to report
- 332101 Design is fully constrained for setup requirements
- 332101 Design is fully constrained for hold requirements
- Quartus Prime Timing Analyzer was successful. 0 errors, 6 warnings
- 293000 quartus Prime Full Compilation was successful. 0 errors, 182 warnings

Figure 5: Package VHDL code continued.

Instruction Memory

```

Quartus Prime Lite Edition - C:/Users/Azwad/Desktop/CPU/Shameem_5_22_22_CPU/Shameem_5_22_22_CPU - Shameem_5_22_22_CPU

File Edit View Project Assignments Processing Tool Window Help
Project Navigator Files Compilation Report - Shameem_5_22_22_CPU Shameem_5_22_22_InstructionMemory.vhd IP Catalog
Search altera.com
Shameem_5_22_22_CPU
Shameem_5_22_22_InstructionMemory.vhd
IP Catalog
Search altera.com
Search altera.com
Project Directory
No Selection Available
Library
Basic Functions
DSP
Interface Protocols
Memory Interfaces and Controllers
Processors and Peripherals
University Program
Search for Partner IP

USE IEEE.Numeric_Std.all;
entity Shameem_5_22_22_InstructionMemory is
    port (
        Shameen_5_22_22_Clock : in STD_LOGIC_VECTOR(31 downto 0);
        Shameen_5_22_22_InstructionAddress : in STD_LOGIC_VECTOR(5 downto 0);
        Shameen_5_22_22_WR : out STD_LOGIC_VECTOR(31 downto 0);
        Shameen_5_22_22_R : out STD_LOGIC_VECTOR(4 downto 0);
        Shameen_5_22_22_Sh : out STD_LOGIC_VECTOR(15 downto 0);
        Shameen_5_22_22_W : out STD_LOGIC_VECTOR(15 downto 0);
        Shameen_5_22_22_Instruction : out STD_LOGIC_VECTOR(31 downto 0)
    );
end entity Shameem_5_22_22_InstructionMemory;

architecture arch of Shameem_5_22_22_InstructionMemory is
begin
    process
        variable Shameen_5_22_22_Instructions : STD_LOGIC_VECTOR(31 downto 0);
        variable Shameen_5_22_22_Address : STD_LOGIC_VECTOR(5 downto 0);
        signal Shameen_5_22_22_R : STD_LOGIC_VECTOR(31 downto 0);
        begin
            loop
                Shameen_5_22_22_Address <= Shameen_5_22_22_InstructionAddress(5 downto 0);
                RamPort : Shameen_5_22_22_InstructionPort map(Shameen_5_22_22_Address, Shameen_5_22_22_Clock, shameen_5_22_22_data, "0", shameen_5_22_22_instructions);
                Shameen_5_22_22_R <= Shameen_5_22_22_Instructions(31 downto 16);
                Shameen_5_22_22_Sh <= Shameen_5_22_22_Instructions(15 downto 10);
                Shameen_5_22_22_W <= Shameen_5_22_22_Instructions(10 downto 5);
                Shameen_5_22_22_WR <= Shameen_5_22_22_Instructions(5 downto 0);
                Shameen_5_22_22_Instruction <= Shameen_5_22_22_Instructions(31 downto 0);
            end loop;
        end process;
end arch;

```

Messages

- 332140 No Removal paths to report
- 332140 No Minimum Pulse widths paths to report
- 332100 Design is fully constrained for setup requirements
- 332100 Design is fully constrained for hold requirements
- Quartus Prime Timing Analyzer was successful. 0 errors, 6 warnings
- 293000 Quartus Prime Full Compilation was successful. 0 errors, 182 warnings

Figure 6: Instruction Memory VHDL code.

```

Quartus Prime Lite Edition - C:/Users/Azwad/Desktop/CPU/Shameem_5_22_22_CPU/Shameem_5_22_22_CPU - Shameem_5_22_22_CPU

File Edit View Project Assignments Processing Tool Window Help
Project Navigator Files Compilation Report - Shameem_5_22_22_CPU Shameem_5_22_22_InstructionPort.vhd IP Catalog
Search altera.com
Shameem_5_22_22_CPU
Shameem_5_22_22_InstructionPort.vhd
IP Catalog
Search altera.com
Search altera.com
Project Directory
No Selection Available
Library
Basic Functions
DSP
Interface Protocols
Memory Interfaces and Controllers
Processors and Peripherals
University Program
Search for Partner IP

USE IEEE.std_logic_1164.all;
LIBRARY altera_mf;
USE altera_mf.altera_mf_components.all;
entity Shameem_5_22_22_InstructionPort is
    PORT (
        address : IN STD_LOGIC_VECTOR(5 DOWNTO 0);
        clock : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
        data : OUT STD_LOGIC_VECTOR(31 DOWNTO 0);
        wren : IN STD_LOGIC;
        q : OUT STD_LOGIC_VECTOR(31 DOWNTO 0)
    );
end Shameem_5_22_22_InstructionPort;

architecture SYN_OF_shameem_5_22_22_instructionport of Shameem_5_22_22_InstructionPort is
    SIGNAL sub_wire0 : STD_LOGIC_VECTOR(31 DOWNTO 0);
begin
    sub_wire0 <- sub_wire0(31 DOWNTO 0);
    altSyncram_component : altSyncram
        generic map(
            clock_enable_a=>"BYPASS",
            init_file=>"Shameem_5_22_22_RamInstructions.mif",
            pm_hinit=>"Cyclone V",
            pm_type=>"altSyncram",
            operation_mode=>"SINGLE_PORT",
            outdata_reg_a=>"UNREGISTERED",
            power_up_uninitialized=>"FALSE",
            read_during_write_mode_port_a=>"NEW_DATA_NO_NBE_READ",
            width_a=>32,
            width_bytew_a=>1
        )
        port map(
            address_a=>address,
            clock=>clock,
            data=>data,
            q=>q,
            wren=>wren
        );
end SYN_OF_shameem_5_22_22_instructionport;

```

Messages

Figure 6: Instruction Memory Port VHDL code.

AdderSub

```

library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity shameem_5_22_22_AddSub is
    generic(N : integer := 32);
    port (
        Shemeen_5_22_22_A : in STD_LOGIC_VECTOR(N-1 downto 0);
        Shemeen_5_22_22_B : in STD_LOGIC_VECTOR(N-1 downto 0);
        Shemeen_5_22_22_Cout : out STD_LOGIC_VECTOR(N-1 downto 0);
        Shemeen_5_22_22_Result : out STD_LOGIC_VECTOR(N-1 downto 0);
        Shemeen_5_22_22_ZeroCheck : out STD_LOGIC_VECTOR(N-1 downto 0);
        Shemeen_5_22_22_Negative : out STD_LOGIC
    );
end entity shameem_5_22_22_AddSub;

architecture arch of Shemeen_5_22_22_AddSub is
begin
    process (Shemeen_5_22_22_A, Shemeen_5_22_22_B, Shemeen_5_22_22_TmpThree, Shemeen_5_22_22_SumTmp : STD_LOGIC_VECTOR(N-1 downto 0));
    signal Shemeen_5_22_22_Cout, Shemeen_5_22_22_ZeroCheck : STD_LOGIC_VECTOR(N-1 downto 0);
    begin
        Shemeen_5_22_22_C1 : Shemeen_5_22_22_C2 : STD_LOGIC;
        begin
            process (Shemeen_5_22_22_A, Shemeen_5_22_22_B, Shemeen_5_22_22_Operation, Shemeen_5_22_22_Cout, Shemeen_5_22_22_Tmp, Shemeen_5_22_22_TmpTwo)
            begin
                Shemeen_5_22_22_Out : STD_LOGIC;
                for i in 0 to N-1 loop
                    Shemeen_5_22_22_B(i) <= Shemeen_5_22_22_A(i) xor Shemeen_5_22_22_Operation;
                    Shemeen_5_22_22_TmpTwo(i) <= Shemeen_5_22_22_Tmp(i) xor Shemeen_5_22_22_A(i);
                    Shemeen_5_22_22_ZeroCheck(i) <- Shemeen_5_22_22_ZeroCheck(i) and (not(Shemeen_5_22_22_TmpTwo(i) xor Shemeen_5_22_22_Cout(i)));
                    Shemeen_5_22_22_TmpThree(i) <- Shemeen_5_22_22_Cout(i) and Shemeen_5_22_22_TmpTwo(i);
                end loop;
                for i in 0 to N-1 loop
                    Shemeen_5_22_22_PerCheck(i+1) <- Shemeen_5_22_22_ZeroCheck(i) and (not Shemeen_5_22_22_SumTmp(i));
                end loop;
                Shemeen_5_22_22_Carry <- Shemeen_5_22_22_Carry(N-1);
                Shemeen_5_22_22_Ero <- Shemeen_5_22_22_ZeroCheck(N) and (not(Shemeen_5_22_22_Carry(N-1) xor Shemeen_5_22_22_Cout(N-1)));
                Shemeen_5_22_22_C1 <- Shemeen_5_22_22_A(N-1) or (Shemeen_5_22_22_B(N-1) xor Shemeen_5_22_22_Operation); -- A B check;
                Shemeen_5_22_22_C2 <- Shemeen_5_22_22_TmpThree(N-1) xor Shemeen_5_22_22_Cout(N-1); -- right most bit of the sum
                Shemeen_5_22_22_Negative <- Shemeen_5_22_22_C1 and (Shemeen_5_22_22_C2 xor Shemeen_5_22_22_C3);
                Shemeen_5_22_22_Result <- Shemeen_5_22_22_SumTmp;
            end process;
        end arch;
    end;

```

Figure 7: AdderSub VHDL code. Used to compute instructions Add, Addi, Addu, Addiu, Sub, Subu.

Extender

```

library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity shameem_5_22_22_Extender is
    port (
        Shemeen_5_22_22_Extop : in STD_LOGIC;
        Shemeen_5_22_22_Input : in STD_LOGIC_VECTOR(15 downto 0);
        Shemeen_5_22_22_Output : out STD_LOGIC_VECTOR(31 downto 0)
    );
end entity shameem_5_22_22_Extender;

architecture arch of shameem_5_22_22_Extender is
begin
    process (Shemeen_5_22_22_Input, Shemeen_5_22_22_Extop)
    begin
        if (Shemeen_5_22_22_Extop = '0') then
            Shemeen_5_22_22_Output <- STD_LOGIC_VECTOR(resize(unsigned(Shemeen_5_22_22_Input), 32));
        elsif (Shemeen_5_22_22_Extop = '1') then
            Shemeen_5_22_22_Output <- STD_LOGIC_VECTOR(resize(signed(Shemeen_5_22_22_Input), 32));
        end if;
    end process;
end arch;

```

Figure 7: Extender VHDL code.

Mux

```

library IEEE;
use IEEE.STD.LOGIC_1164.all;
use IEEE.NUMERIC_STD.all;

entity Shameem_5_22_22_Mux2_1 is
    PORT (
        Shameen_5_22_22_Switch : in STD_LOGIC;
        Shameen_5_22_22_A : in STD_LOGIC_VECTOR(31 downto 0);
        Shameen_5_22_22_B : in STD_LOGIC_VECTOR(31 downto 0);
        Shameen_5_22_22_Output : out STD_LOGIC_VECTOR(31 downto 0)
    );
end entity Shameem_5_22_22_Mux2_1;

architecture arch of Shameem_5_22_22_Mux2_1 is
begin
    process (Shameen_5_22_22_Switch, Shameen_5_22_22_A, Shameen_5_22_22_B)
    begin
        if (Shameen_5_22_22_Switch = '0') then
            Shameen_5_22_22_Output <= Shameen_5_22_22_A;
        elsif (Shameen_5_22_22_Switch = '1') then
            Shameen_5_22_22_Output <= Shameen_5_22_22_B;
        end if;
    end process;
end architecture;

```

Figure 8: Mux 2:1 VHDL code.

```

library IEEE;
use IEEE.STD.LOGIC_1164.all;
use IEEE.NUMERIC_STD.all;

entity Shameem_5_22_22_Mux2_1_5bit is
    PORT (
        Shameen_5_22_22_Switch : in STD_LOGIC;
        Shameen_5_22_22_A : in STD_LOGIC_VECTOR(4 downto 0);
        Shameen_5_22_22_B : in STD_LOGIC_VECTOR(4 downto 0);
        Shameen_5_22_22_Output : out STD_LOGIC_VECTOR(4 downto 0)
    );
end entity Shameem_5_22_22_Mux2_1_5bit;

architecture arch of Shameem_5_22_22_Mux2_1_5bit is
begin
    process (Shameen_5_22_22_Switch, Shameen_5_22_22_A, Shameen_5_22_22_B)
    begin
        if (Shameen_5_22_22_Switch = '0') then
            Shameen_5_22_22_Output <= Shameen_5_22_22_A;
        elsif (Shameen_5_22_22_Switch = '1') then
            Shameen_5_22_22_Output <= Shameen_5_22_22_B;
        end if;
    end process;
end architecture;

```

Figure 9: Mux 2:1 5-bit VHDL code. Used for input decide RW from RD or RT.

```

library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.Numeric_std.all;
entity Shameem_5_22_22_Mux3_1 is
  port (
    shameen_5_22_22_Pcsrc: STD_LOGIC;
    shameen_5_22_22_branch: STD_LOGIC;
    shameen_5_22_22_A: STD_LOGIC;
    shameen_5_22_22_B: STD_LOGIC;
    shameen_5_22_22_C: STD_LOGIC_VECTOR(31 downto 0);
    shameen_5_22_22_Output: STD_LOGIC_VECTOR(31 downto 0)
  );
end entity Shameem_5_22_22_Mux3_1;
architecture arch of Shameem_5_22_22_Mux3_1 is
begin
  process(shameen_5_22_22_Pcsrc, shameen_5_22_22_branch, shameen_5_22_22_A, shameen_5_22_22_B)
  begin
    if (shameen_5_22_22_Pcsrc = '0' AND shameen_5_22_22_branch = '0') then
      shameen_5_22_22_Output <= shameen_5_22_22_A;
    elsif (shameen_5_22_22_Pcsrc = '1' AND shameen_5_22_22_branch = '0') then
      shameen_5_22_22_Output <= shameen_5_22_22_B;
    else
      shameen_5_22_22_Output <= shameen_5_22_22_C;
    end if;
  end process;
end architecture;

```

Figure 10: Mux 3:1 VHDL code. Used for input decide address for regular instructions, BNE/BEQ and jump.

Bitwise Operations

```

library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.Numeric_std.all;
entity Shameem_5_22_22_BitwiseOperations is
  port (
    Shameem_5_22_22_Funct: STD_LOGIC_VECTOR(5 downto 0);
    Shameem_5_22_22_Input1: STD_LOGIC_VECTOR(N-1 downto 0);
    Shameem_5_22_22_Input2: STD_LOGIC_VECTOR(N-1 downto 0);
    Shameem_5_22_22_Output: STD_LOGIC_VECTOR(10 downto 0)
  );
end entity Shameem_5_22_22_BitwiseOperations;
architecture arch of Shameem_5_22_22_BitwiseOperations is
  signal Shameem_5_22_22_tmp: STD_LOGIC_VECTOR(10 downto 0);
  signal shift: STD_LOGIC_VECTOR(10 downto 0);
begin
  process(Shameem_5_22_22_Funct, Shameem_5_22_22_Input1, Shameem_5_22_22_Input2, Shameem_5_22_22_tmp)
  begin
    if (Shameem_5_22_22_Funct = "100100" OR Shameem_5_22_22_Funct = "100001") then
      Shameem_5_22_22_tmp <= Shameem_5_22_22_Input1;
    elsif (Shameem_5_22_22_Funct = "100101" OR Shameem_5_22_22_Funct = "101000") then
      Shameem_5_22_22_tmp <= NOT(Shameem_5_22_22_Input1) OR Shameem_5_22_22_Input2;
    elsif (Shameem_5_22_22_Funct = "100110") then
      Shameem_5_22_22_tmp <= Shameem_5_22_22_Input1 NOR Shameem_5_22_22_Input2;
    elsif (Shameem_5_22_22_Funct = "100011") then
      Shameem_5_22_22_tmp <= NOT(Shameem_5_22_22_Input1) AND Shameem_5_22_22_Input2;
    elsif (Shameem_5_22_22_Funct = "000100") then
      Shameem_5_22_22_tmp <= STD_LOGIC_VECTOR(SHIFT_left(unsigned(Shameem_5_22_22_Input1), to_integer(unsigned(Shameem_5_22_22_Input2(0 downto 6))))) ;
    elsif (Shameem_5_22_22_Funct = "000101") then
      Shameem_5_22_22_tmp <= STD_LOGIC_VECTOR(SHIFT_right(unsigned(Shameem_5_22_22_Input1), to_integer(unsigned(Shameem_5_22_22_Input2(10 downto 0))))) ;
    end if;
    Shameem_5_22_22_Output <= Shameem_5_22_22_tmp;
  end process;
end architecture;

```

Figure 11: Bitwise Operations VHDL code. Used for instructions such as AND, OR, NOR, ORI, ANDI, SRL, SRL, SLL.

Adder

```

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.all;
LIBRARY lpm;
USE lpm.all;
ENTITY shameem_5_22_22_Adder IS
  PORT (
    dataa : IN STD.LOGIC_VECTOR (31 DOWNTO 0);
    datab : IN STD.LOGIC_VECTOR (31 DOWNTO 0);
    result : OUT STD.LOGIC_VECTOR (31 DOWNTO 0)
  );
END shameem_5_22_22_Adder;

ARCHITECTURE SYN OF shameem_5_22_22_Adder IS
  SIGNAL sub_wire0 : STD.LOGIC_VECTOR (31 DOWNTO 0);

  COMPONENT lpm_add_sub
  GENERIC
    lpm_direction : STRING;
    lpm_hint : STRING;
    lpm_representation : STRING;
    lpm_type : STRING;
    lpm_width : NATURAL;
  PORT
    dataa : IN STD.LOGIC_VECTOR (31 DOWNTO 0);
    datab : IN STD.LOGIC_VECTOR (31 DOWNTO 0);
    result : OUT STD.LOGIC_VECTOR (31 DOWNTO 0);
  END COMPONENT;
BEGIN
  result <- sub_wire0(31 DOWNTO 0);
  LPM_ADD_SUB_component : LPM_ADD_SUB
  GENERIC MAP (
    lpm_direction => "ADD",
    lpm_hint => "ONE_INPUT",
    lpm_representation => "SIGNED",
    lpm_type => "LPM_ADD_SUB",
    lpm_width => 32
  )
  PORT MAP (
    dataa => sub_wire0,
    datab => sub_wire0,
    result => result
  );
END;

```

Figure 12: Adder VHDL code. Used for address change per instruction.

Equal

```

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.all;
LIBRARY lpm;
USE lpm.all;
ENTITY shameem_5_22_22_Equal IS
  PORT (
    dataa : IN STD.LOGIC_VECTOR (31 DOWNTO 0);
    datab : IN STD.LOGIC_VECTOR (31 DOWNTO 0);
    aeb : OUT STD.LOGIC;
  );
END shameem_5_22_22_Equal;

ARCHITECTURE SYN OF shameem_5_22_22_equal IS
  SIGNAL sub_wire0 : STD.LOGIC;

  COMPONENT lpm_compare
  GENERIC
    lpm_representation : STRING;
    lpm_type : STRING;
    lpm_width : NATURAL;
  PORT
    dataa : IN STD.LOGIC_VECTOR (31 DOWNTO 0);
    datab : IN STD.LOGIC_VECTOR (31 DOWNTO 0);
    aeb : OUT STD.LOGIC;
  END COMPONENT;
BEGIN
  aeb <- sub_wire0;
  LPM_COMPARE_component : LPM_COMPARE
  GENERIC MAP (
    lpm_representation => "SIGNED",
    lpm_type => "LPM_COMPARE",
    lpm_width => 32
  )
  PORT MAP (
    dataa => dataa,
    datab => datab,
    aeb => sub_wire0
  );
END;

```

Figure 12: Equal VHDL code. Used BNE and BEQ to check condition to branch.

Data Memory

The screenshot shows the Quartus Prime Lite Edition software interface. The main window displays the VHDL code for the `Shameem_5_22_22_DataMemory.vhd` file. The code defines an entity `Shameem_5_22_22_DataMemory` with a port containing address, data, and control signals. It also includes an architecture `arch` with a `port map` clause. The IP Catalog panel on the right lists various Altera IP components. The bottom pane shows the message log with two informational messages related to signal removal and pulse width.

```

library IEEE;
use IEEE.STD.LOGIC_1164.all;

entity Shameem_5_22_22_DataMemory is
    port (
        Shameem_5_22_22_address : in STD_logic_vector(31 downto 0);
        Shameem_5_22_22_data : in STD_logic_vector(31 downto 0);
        Shameem_5_22_22_wren : in STD_logic_vector(31 downto 0);
        Shameem_5_22_22_q : out STD_logic_vector(31 downto 0)
    );
end entity;

architecture arch of Shameem_5_22_22_DataMemory is
begin
    Shameem_5_22_22_address <=> Shameem_5_22_22_Address;
    Shameem_5_22_22_clock <=> Shameem_5_22_22_Clock;
    Shameem_5_22_22_data <=> Shameem_5_22_22_Data;
    Shameem_5_22_22_wren <=> Shameem_5_22_22_Wren;
    Shameem_5_22_22_q <=> Shameem_5_22_22_Q;
end architecture;
end component;

signal Shameem_5_22_22_Tmp : STD_logic_vector(4 downto 0);
begin
    Shameem_5_22_22_Tmp <=> Shameem_5_22_22_Address(4 downto 0);
    Shameem_5_22_22_Data_Memory_Port : Shameem_5_22_22_DataMemoryPort
    port map(Shameem_5_22_22_Tmp, Shameem_5_22_22_Clock, Shameem_5_22_22_Wren, Shameem_5_22_22_Output);
end arch;

```

Figure 13: Data Memory VHDL code.

The screenshot shows the Quartus Prime Lite Edition software interface. The main window displays the VHDL code for the `Shameem_5_22_22_DataMemoryPort.vhd` file. The code defines an entity `Shameem_5_22_22_DataMemoryPort` with a port containing address, data, and control signals. It also includes an architecture `SYN` with a `port map` clause. The IP Catalog panel on the right lists various Altera IP components. The bottom pane shows the message log with two informational messages related to signal removal and pulse width.

```

entity Shameem_5_22_22_DataMemoryPort is
    port (
        Shameem_5_22_22_Address : IN STD_logic_VECTOR(4 DOWNTO 0);
        Shameem_5_22_22_Clock : IN STD_logic := '1';
        Shameem_5_22_22_Data : IN STD_logic_VECTOR(31 DOWNTO 0);
        Shameem_5_22_22_Wren : IN STD_logic_VECTOR(31 DOWNTO 0);
        Shameem_5_22_22_Q : OUT STD_logic_VECTOR(31 DOWNTO 0)
    );
end entity;

architecture SYN of Shameem_5_22_22_DataMemoryPort is
    signal Shameem_5_22_22_temp : STD_logic_VECTOR(31 DOWNTO 0);
begin
    Shameem_5_22_22_Q <= shameem_5_22_22_temp(31 DOWNTO 0);
    altSyncram_component : altSyncram
        generic map(
            clock_enable_input_a => "BYPASS",
            clock_enable_output_a => "BYPASS",
            device_name => "Shameem_5_22_22_DataMemory.mif",
            intended_device_family => "Cyclone V",
            ip_type => "altSyncram",
            numwords_a => 32,
            width_a => 32,
            width_byteena_a => 1
        )
        port map(
            address_a => shameem_5_22_22_address,
            data_a => shameem_5_22_22_data,
            data_a_q => Shameem_5_22_22_Data,
            wren_a => Shameem_5_22_22_wren,
            q_a => shameem_5_22_22_temp
        );
end architecture;

```

Figure 14: Data Memory Port VHDL code. This is used in Data Memory.

Register File

```

library IEEE;
use IEEE.STD.LOGIC_1164.all;
use STD.TEXTIO.all;
use work.Shameem_5_22_22_Package.all;

entity Shameem_5_22_22_RegisterFile is
    port (
        Shemeen_5_22_22_Clock : in STD_LOGIC;
        Shemeen_5_22_22_RA : Shemeen_5_22_22_Reg;
        Shemeen_5_22_22_BusW : in STD_LOGIC_VECTOR(4 downto 0);
        Shemeen_5_22_22_BusR : out STD_LOGIC_VECTOR(31 downto 0)
    );
end Shameem_5_22_22_RegisterFile;

architecture arch of Shameem_5_22_22_RegisterFile is
begin
    map : process (Shemeen_5_22_22_RA) begin
        Shemeen_5_22_22_BusR := Shemeen_5_22_22_TriplePortRAM port map(Shemeen_5_22_22_Clock, Shemeen_5_22_22_BusW, Shemeen_5_22_22_RA, Shemeen_5_22_22_BusR, Shemeen_5_22_22_BusR);
    end process;
end arch;

```

The screenshot shows the Quartus Prime Lite Edition interface with the project "Shameem_5_22_22_CPU" open. The "Files" tab is selected, displaying the VHDL file "Shameem_5_22_22_RegisterFile.vhd". The code is pasted above. The "Messages" panel at the bottom shows two messages: "332140 No Removal paths to report" and "332140 No Minimum Pulse width paths to report". The "IP Catalog" panel on the right is visible.

Figure 15: Register File VHDL code.

```

library IEEE;
use IEEE.STD.LOGIC_1164.all;
entity Shameem_5_22_22_TriplePortRam is
    PORT(
        Shemeen_5_22_22_Clock : in STD_LOGIC := '1';
        Shemeen_5_22_22_Rdaddress1 : in STD_LOGIC_VECTOR(4 DOWNTO 0);
        Shemeen_5_22_22_Wraddress1 : in STD_LOGIC_VECTOR(4 DOWNTO 0);
        Shemeen_5_22_22_Rdaddress2 : in STD_LOGIC_VECTOR(4 DOWNTO 0);
        Shemeen_5_22_22_Wraddress2 : in STD_LOGIC_VECTOR(4 DOWNTO 0);
        Shemeen_5_22_22_Wren : in STD_LOGIC := '0';
        Shemeen_5_22_22_Data1 : out STD_LOGIC_VECTOR(31 DOWNTO 0);
        Shemeen_5_22_22_Data2 : out STD_LOGIC_VECTOR(31 DOWNTO 0)
    );
end Shameem_5_22_22_TriplePortRam;

architecture arch of Shameem_5_22_22_TriplePortRam is
component Shameem_5_22_22_DualPortedRam is
    PORT(
        clock : in STD_LOGIC := '1';
        Raddress : in STD_LOGIC_VECTOR(4 DOWNTO 0);
        Rdaddress : in STD_LOGIC_VECTOR(4 DOWNTO 0);
        Wraddress : in STD_LOGIC_VECTOR(4 DOWNTO 0);
        Q : out STD_LOGIC_VECTOR(31 DOWNTO 0)
    );
end component;
begin
    map : process (Shemeen_5_22_22_Rdaddress1, Shemeen_5_22_22_Wraddress1, Shemeen_5_22_22_Wren, Shemeen_5_22_22_Data1, Shemeen_5_22_22_Rdaddress2, Shemeen_5_22_22_Wraddress2, Shemeen_5_22_22_Wren, Shemeen_5_22_22_Data2)
    begin
        Shemeen_5_22_22_Data1 <= Shameem_5_22_22_DualPortedRam port map (Shemeen_5_22_22_Clock, Shemeen_5_22_22_Rdaddress1, Shemeen_5_22_22_Wraddress1, Shemeen_5_22_22_Wren, Shemeen_5_22_22_Data1, Shemeen_5_22_22_Rdaddress2, Shemeen_5_22_22_Wraddress2, Shemeen_5_22_22_Wren, Shemeen_5_22_22_Data2);
    end process;
end arch;

```

The screenshot shows the Quartus Prime Lite Edition interface with the project "Shameem_5_22_22_CPU" open. The "Files" tab is selected, displaying the VHDL file "Shameem_5_22_22_TriplePortRam.vhd". The code is pasted above. The "Messages" panel at the bottom shows two messages: "332140 No Removal paths to report" and "332140 No Minimum Pulse width paths to report". The "IP Catalog" panel on the right is visible.

Figure 16: Triple Port Ram VHDL code.

```

library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity Shameen_5_22_22_DualPortedRam is
    PORT (
        clock : IN STD_LOGIC;
        data : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
        address : IN STD_LOGIC_VECTOR(4 DOWNTO 0);
        wren : IN STD_LOGIC;
        q : OUT STD_LOGIC_VECTOR(31 DOWNTO 0)
    );
end Shameen_5_22_22_DualPortedRam;

ARCHITECTURE SYN OF Shameen_5_22_22_DualPortedRam IS
    SIGNAL sub_wire0 : STD_LOGIC_VECTOR (31 DOWNTO 0);
begin
    q <= sub_wire0(31 DOWNTO 0);
    altsyncram_component : altsyncram
        generic map (
            address_aclr_a => "NONE",
            address_reg_a => "CLOCKED",
            address_reg_b => "BYPASS",
            clock_enable_input_a => "BYPASS",
            clock_enable_output_b => "BYPASS",
            init_file => "empty.Ram32x32.mif",
            intended_device_family => "Cyclone V",
            output_aclr_a => "NONE"
        )
        port map (
            address_a => address,
            data_a => data,
            numwords_a => 32,
            operation_mode => "DUAL_PORT",
            outdata_aclr_b => "NONE",
            outdata_reg_b => "REGISTERED",
            power_up_uninitialized => "FALSE",
            read_during_write_mode_mixed_ports => "OLD_DATA",
            width_a => 32,
            width_b => 32,
            width_byteena_a => 1
        );
    PORT MAP (
        address_a => address,
        address_b => address,
        data_a => data,
        data_b => data,
        wren_a => wren,
        wren_b => wren
    );
end;

```

Figure 16: Dual Port Ram VHDL code. Used in Triple Ported Ram.

Instruction Register

```

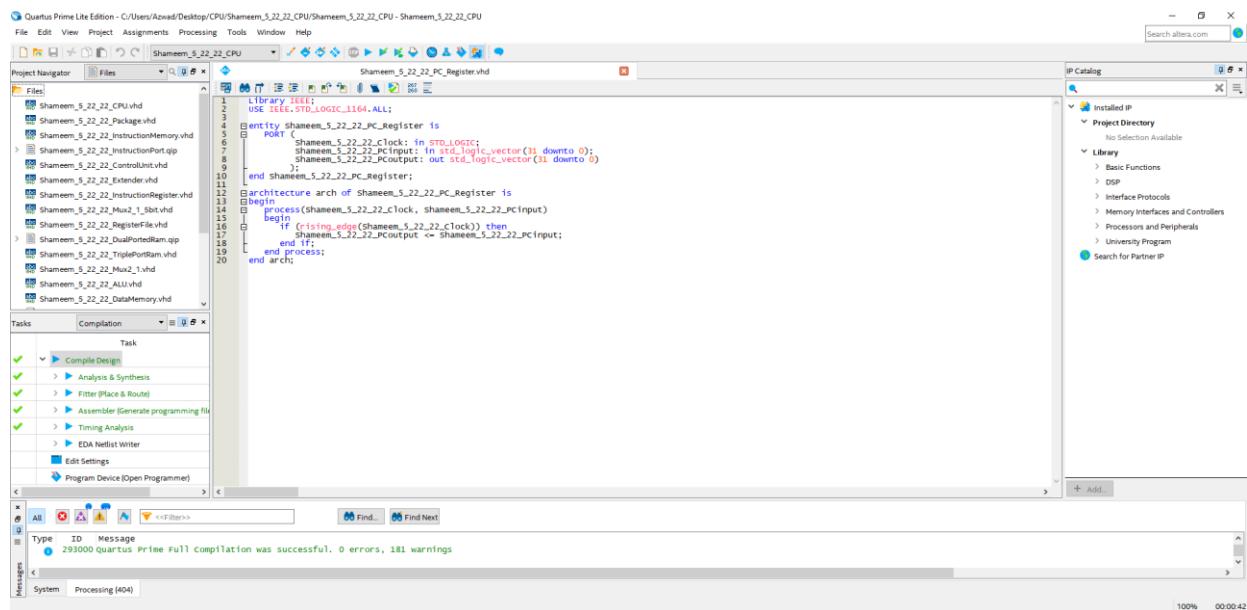
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity Shameen_5_22_22_InstructionRegister is
    PORT (
        Shameen_5_22_22_Clock : IN STD_LOGIC;
        Shameen_5_22_22_Input : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
        Shameen_5_22_22_Output : OUT STD_LOGIC_VECTOR(31 DOWNTO 0)
    );
end Shameen_5_22_22_InstructionRegister;

ARCHITECTURE arch OF Shameen_5_22_22_InstructionRegister IS
begin
    process (Shameen_5_22_22_Clock, Shameen_5_22_22_Input)
    begin
        if rising_edge(Shameen_5_22_22_Clock) then
            Shameen_5_22_22_Output <- Shameen_5_22_22_Input;
        end if;
    end process;
end arch;

```

Figure 17: Instruction Register VHDL code.

PC Register



The screenshot shows the Quartus Prime Lite Edition interface with the following details:

- Title Bar:** Quartus Prime Lite Edition - C:/Users/Azwad/Desktop/CPU/Shameem_5_22_22_CPU/Shameem_5_22_22_CPU - Shameem_5_22_22_CPU
- Project Navigator:** Shows files like shameem_5_22_22_CPU.vhd, shameem_5_22_22_Package.vhd, etc.
- Editor Area:** Displays the VHDL code for the PC Register:

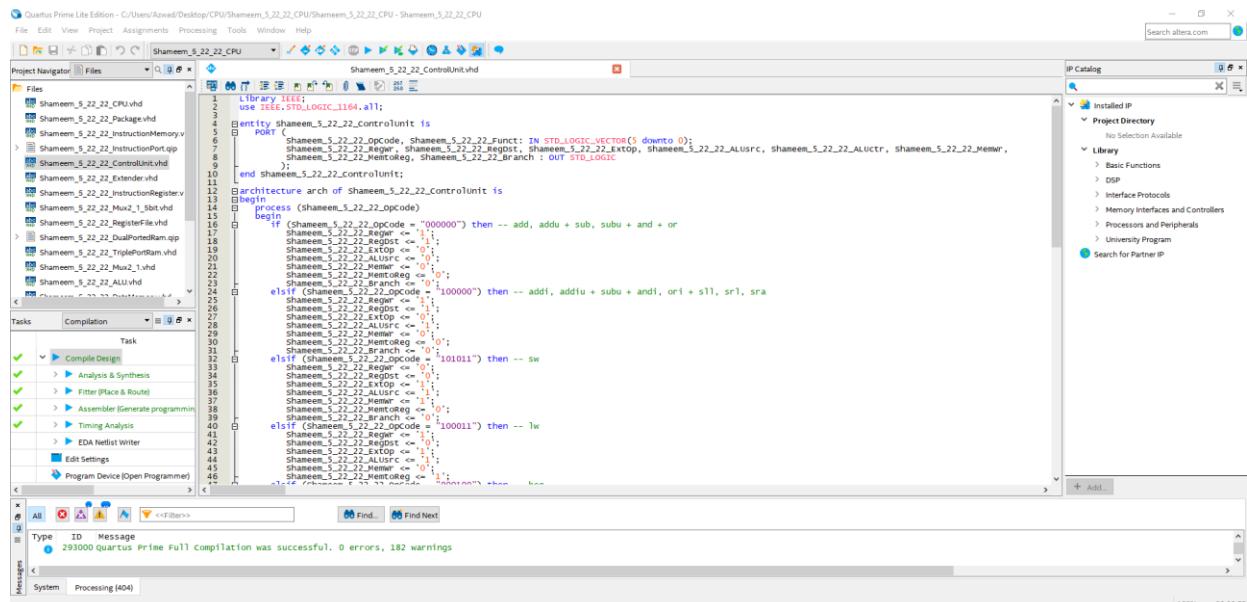
```

1  LIBRARY IEEE;
2  USE IEEE.STD.LOGIC_1164.ALL;
3
4  entity shameem_5_22_22_PC_Register is
5    port (
6      shameen_5_22_22_Clock: in STD_LOGIC;
7      shameen_5_22_22_PcInput: in STD_LOGIC_VECTOR(31 downto 0);
8      shameen_5_22_22_PcOutput: out STD_LOGIC_VECTOR(31 downto 0)
9    );
10 end shameem_5_22_22_PC_Register;
11
12 architecture arch of shameem_5_22_22_PC_Register is
13 begin
14   process (shameen_5_22_22_Clock, shameen_5_22_22_PcInput)
15   begin
16     if (rising_edge(shameen_5_22_22_Clock)) then
17       shameen_5_22_22_PcOutput <= shameen_5_22_22_PcInput;
18     end if;
19   end process;
20 end arch;

```
- IP Catalog:** Shows the Installed IP section with no selection available.
- Tasks:** Shows successful compilation tasks: Compile Design, Analysis & Synthesis, Filter Place & Route, Assembler, Timing Analysis, EDA Netlist Writer, Edit Settings, and Program Device (Open Programmer).
- Messages:** Shows a successful compilation message: "293000 Quartus Prime Full Compilation was successful. 0 errors, 181 warnings".

Figure 18: PC Register VHDL code.

Control Unit



```

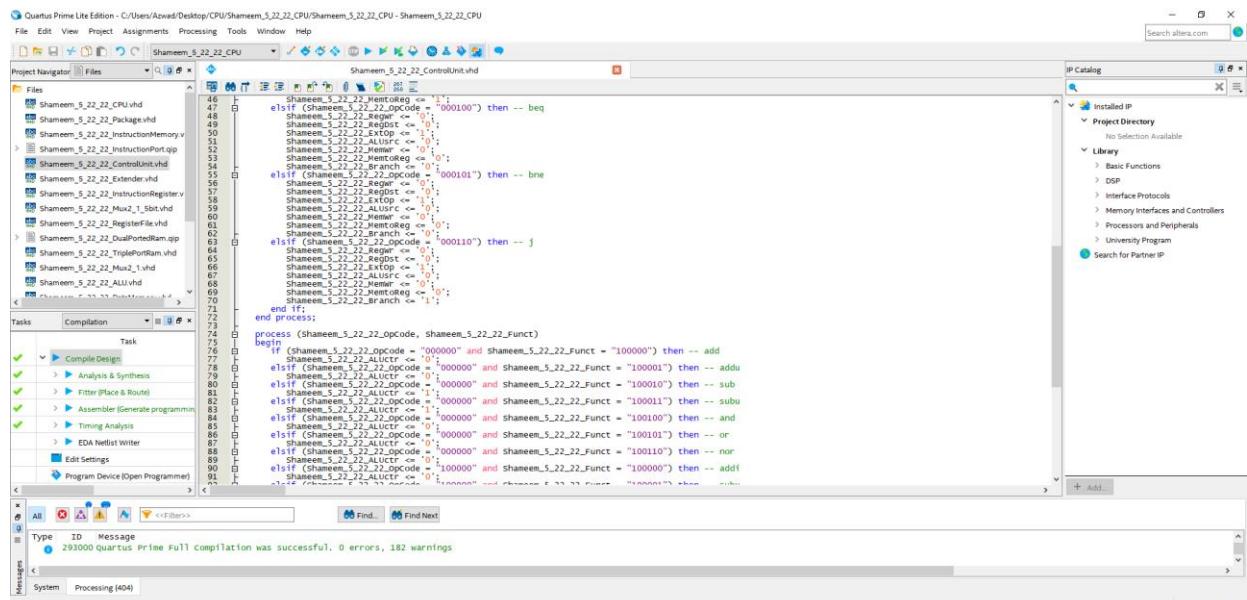
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity shameen_5_22_22_ControlUnit is
    port(
        shameen_5_22_22_opcode : STD_LOGIC_VECTOR(5 downto 0);
        shameen_5_22_22_ALUsrc : STD_LOGIC_VECTOR(1 down to 0);
        shameen_5_22_22_ExTop : STD_LOGIC_VECTOR(1 down to 0);
        shameen_5_22_22_Hemter : STD_LOGIC_VECTOR(1 down to 0);
        shameen_5_22_22_Branch : STD_LOGIC_VECTOR(1 down to 0);
        shameen_5_22_22_ALUctrl : STD_LOGIC_VECTOR(3 down to 0)
    );
end entity shameen_5_22_22_ControlUnit;

architecture arch of shameen_5_22_22_ControlUnit is
begin
    process (shameen_5_22_22_opcode)
        begin
            if (shameen_5_22_22_opcode = "000000") then -- add, addu + sub, subu + and + or
                shameen_5_22_22_ALUsrc <= "1";
                shameen_5_22_22_ExTop <= "1";
                shameen_5_22_22_Hemter <= "0";
                shameen_5_22_22_ALUctrl <= "0000";
                shameen_5_22_22_Branch <= "0";
            elsif (shameen_5_22_22_opcode = "100000") then -- addi, addiu + subu + andi, ori + sll, srl, sra
                shameen_5_22_22_ExTop <= "1";
                shameen_5_22_22_Hemter <= "0";
                shameen_5_22_22_ALUctrl <= "0000";
                shameen_5_22_22_Branch <= "0";
            elsif (shameen_5_22_22_opcode = "101011") then -- sw
                shameen_5_22_22_ExTop <= "0";
                shameen_5_22_22_Hemter <= "1";
                shameen_5_22_22_ALUsrc <= "1";
                shameen_5_22_22_ALUctrl <= "0000";
                shameen_5_22_22_Branch <= "0";
            elsif (shameen_5_22_22_opcode = "100011") then -- lw
                shameen_5_22_22_ExTop <= "1";
                shameen_5_22_22_Hemter <= "1";
                shameen_5_22_22_ALUsrc <= "0";
                shameen_5_22_22_ALUctrl <= "0000";
                shameen_5_22_22_Branch <= "0";
            else
                shameen_5_22_22_ExTop <= "0";
                shameen_5_22_22_Hemter <= "0";
                shameen_5_22_22_ALUsrc <= "0";
                shameen_5_22_22_ALUctrl <= "0000";
                shameen_5_22_22_Branch <= "0";
            end if;
        end process;
    end architecture;

```

Figure 19: Control Unit VHDL code.



```

        elsif (shameen_5_22_22_opcode = "000100") then -- beq
            shameen_5_22_22_ExTop <= "1";
            shameen_5_22_22_RegOp <= "1";
            shameen_5_22_22_ExTop <= "1";
            shameen_5_22_22_ALUsrc <= "0";
            shameen_5_22_22_Hemter <= "0";
            shameen_5_22_22_ALUctrl <= "0000";
            shameen_5_22_22_Branch <= "0";
        elsif (shameen_5_22_22_opcode = "000101") then -- bne
            shameen_5_22_22_RegOp <= "0";
            shameen_5_22_22_ExTop <= "1";
            shameen_5_22_22_ExTop <= "1";
            shameen_5_22_22_Hemter <= "0";
            shameen_5_22_22_Hemter <= "0";
            shameen_5_22_22_ALUsrc <= "0";
            shameen_5_22_22_ALUctrl <= "0000";
            shameen_5_22_22_Branch <= "1";
        elsif (shameen_5_22_22_opcode = "000110") then -- j
            shameen_5_22_22_ExTop <= "0";
            shameen_5_22_22_RegOp <= "0";
            shameen_5_22_22_ExTop <= "0";
            shameen_5_22_22_Hemter <= "0";
            shameen_5_22_22_Hemter <= "0";
            shameen_5_22_22_ALUsrc <= "0";
            shameen_5_22_22_ALUctrl <= "0000";
            shameen_5_22_22_Branch <= "1";
        end if;
    end process;
    process (shameen_5_22_22_opcode, shameen_5_22_22_Funct)
    begin
        if (shameen_5_22_22_opcode = "000000" and shameen_5_22_22_Funct = "100000") then -- add
            shameen_5_22_22_ALUctrl <= "0100";
            shameen_5_22_22_ALUsrc <= "000000";
            shameen_5_22_22_ExTop <= "1";
            shameen_5_22_22_Hemter <= "0";
            shameen_5_22_22_Branch <= "0";
        elsif (shameen_5_22_22_OPCODE = "000000" and shameen_5_22_22_Funct = "100001") then -- addu
            shameen_5_22_22_ALUctrl <= "0101";
            shameen_5_22_22_ALUsrc <= "000000";
            shameen_5_22_22_ExTop <= "1";
            shameen_5_22_22_Hemter <= "0";
            shameen_5_22_22_Branch <= "0";
        elsif (shameen_5_22_22_OPCODE = "000000" and shameen_5_22_22_Funct = "100010") then -- sub
            shameen_5_22_22_ALUctrl <= "1000";
            shameen_5_22_22_ALUsrc <= "000000";
            shameen_5_22_22_ExTop <= "1";
            shameen_5_22_22_Hemter <= "0";
            shameen_5_22_22_Branch <= "0";
        elsif (shameen_5_22_22_OPCODE = "000000" and shameen_5_22_22_Funct = "100011") then -- subu
            shameen_5_22_22_ALUctrl <= "1001";
            shameen_5_22_22_ALUsrc <= "000000";
            shameen_5_22_22_ExTop <= "1";
            shameen_5_22_22_Hemter <= "0";
            shameen_5_22_22_Branch <= "0";
        elsif (shameen_5_22_22_OPCODE = "000000" and shameen_5_22_22_Funct = "100100") then -- and
            shameen_5_22_22_ALUctrl <= "0010";
            shameen_5_22_22_ALUsrc <= "000000";
            shameen_5_22_22_ExTop <= "1";
            shameen_5_22_22_Hemter <= "0";
            shameen_5_22_22_Branch <= "0";
        elsif (shameen_5_22_22_OPCODE = "000000" and shameen_5_22_22_Funct = "100101") then -- andu
            shameen_5_22_22_ALUctrl <= "0011";
            shameen_5_22_22_ALUsrc <= "000000";
            shameen_5_22_22_ExTop <= "1";
            shameen_5_22_22_Hemter <= "0";
            shameen_5_22_22_Branch <= "0";
        elsif (shameen_5_22_22_OPCODE = "000000" and shameen_5_22_22_Funct = "100110") then -- or
            shameen_5_22_22_ALUctrl <= "0110";
            shameen_5_22_22_ALUsrc <= "000000";
            shameen_5_22_22_ExTop <= "1";
            shameen_5_22_22_Hemter <= "0";
            shameen_5_22_22_Branch <= "0";
        elsif (shameen_5_22_22_OPCODE = "000000" and shameen_5_22_22_Funct = "100100") then -- oru
            shameen_5_22_22_ALUctrl <= "0111";
            shameen_5_22_22_ALUsrc <= "000000";
            shameen_5_22_22_ExTop <= "1";
            shameen_5_22_22_Hemter <= "0";
            shameen_5_22_22_Branch <= "0";
        end if;
    end process;

```

Figure 20: Control Unit VHDL code continued.

```

Quartus Prime Lite Edition - C:/Users/Azwad/Desktop/CPU/Shameem_5_22_22,CPU/Shameem_5_22_22,CPU - Shameem_5_22_22,CPU
File Edit View Project Assignments Processing Tools Window Help
Project Navigator Files Shemeen_5_22_22_ControlUnit.vhd IP Catalog
Shemeen_5_22_22_ControlUnit.vhd Search altera.com
File Edit View Project Assignments Processing Tools Window Help
Process (Shameen_5_22_22_Opcode, Shameen_5_22_22_Funct)
begin
    if (Shameen_5_22_22_Opcode = "000000" and Shameen_5_22_22_Funct = "100000") then -- add
        Shameen_5_22_22_ALUOp <= "0";
        Shameen_5_22_22_ALUOp <= "1";
    elsif (Shameen_5_22_22_Opcode = "000000" and Shameen_5_22_22_Funct = "100001") then -- addu
        Shameen_5_22_22_ALUOp <= "0";
        Shameen_5_22_22_ALUOp <= "1";
    elsif (Shameen_5_22_22_Opcode = "000000" and Shameen_5_22_22_Funct = "100010") then -- sub
        Shameen_5_22_22_ALUOp <= "0";
        Shameen_5_22_22_ALUOp <= "1";
    elsif (Shameen_5_22_22_Opcode = "000000" and Shameen_5_22_22_Funct = "100011") then -- subu
        Shameen_5_22_22_ALUOp <= "0";
        Shameen_5_22_22_ALUOp <= "1";
    elsif (Shameen_5_22_22_Opcode = "000000" and Shameen_5_22_22_Funct = "100100") then -- and
        Shameen_5_22_22_ALUOp <= "0";
        Shameen_5_22_22_ALUOp <= "1";
    elsif (Shameen_5_22_22_Opcode = "000000" and Shameen_5_22_22_Funct = "100101") then -- or
        Shameen_5_22_22_ALUOp <= "0";
        Shameen_5_22_22_ALUOp <= "1";
    elsif (Shameen_5_22_22_Opcode = "000000" and Shameen_5_22_22_Funct = "100110") then -- nor
        Shameen_5_22_22_ALUOp <= "0";
        Shameen_5_22_22_ALUOp <= "1";
    elsif (Shameen_5_22_22_Opcode = "000000" and Shameen_5_22_22_Funct = "100000") then -- addi
        Shameen_5_22_22_ALUOp <= "0";
        Shameen_5_22_22_ALUOp <= "1";
    elsif (Shameen_5_22_22_Opcode = "000000" and Shameen_5_22_22_Funct = "100001") then -- subu
        Shameen_5_22_22_ALUOp <= "0";
        Shameen_5_22_22_ALUOp <= "1";
    elsif (Shameen_5_22_22_Opcode = "000000" and Shameen_5_22_22_Funct = "100010") then -- addiu
        Shameen_5_22_22_ALUOp <= "0";
        Shameen_5_22_22_ALUOp <= "1";
    elsif (Shameen_5_22_22_Opcode = "000000" and Shameen_5_22_22_Funct = "100011") then -- andi
        Shameen_5_22_22_ALUOp <= "0";
        Shameen_5_22_22_ALUOp <= "1";
    elsif (Shameen_5_22_22_Opcode = "000000" and Shameen_5_22_22_Funct = "101000") then -- ori
        Shameen_5_22_22_ALUOp <= "0";
        Shameen_5_22_22_ALUOp <= "1";
    elsif (Shameen_5_22_22_Opcode = "000000" and Shameen_5_22_22_Funct = "000000") then -- srl
        Shameen_5_22_22_ALUOp <= "0";
        Shameen_5_22_22_ALUOp <= "1";
    elsif (Shameen_5_22_22_Opcode = "000000" and Shameen_5_22_22_Funct = "000010") then -- srsl
        Shameen_5_22_22_ALUOp <= "0";
        Shameen_5_22_22_ALUOp <= "1";
    elsif (Shameen_5_22_22_Opcode = "101011") then -- sw
        Shameen_5_22_22_ALUOp <= "0";
        Shameen_5_22_22_ALUOp <= "1";
    elsif (Shameen_5_22_22_Opcode = "100011") then -- lw
        Shameen_5_22_22_ALUOp <= "0";
        Shameen_5_22_22_ALUOp <= "1";
    elsif (Shameen_5_22_22_Opcode = "000100") then -- beq
        Shameen_5_22_22_ALUOp <= "0";
        Shameen_5_22_22_ALUOp <= "1";
    elsif (Shameen_5_22_22_Opcode = "000101") then -- bne
        Shameen_5_22_22_ALUOp <= "0";
        Shameen_5_22_22_ALUOp <= "1";
    elsif (Shameen_5_22_22_Opcode = "000010") then -- j
        Shameen_5_22_22_ALUOp <= "0";
        Shameen_5_22_22_ALUOp <= "1";
    end if;
end process;
end arch;

```

Messages

Type ID Message

293000 Quartus Prime full compilation was successful. 0 errors, 182 warnings

Figure 21: Control Unit VHDL code continued.

ALU

```

        entity Shameem_5_22_22_ALU is
            generic (N : integer := 32);
            port (
                shaneem_5_22_22_ALUopcode : in STD_LOGIC;
                shaneem_5_22_22_Funct : in STD_LOGIC_VECTOR(5 downto 0);
                shaneem_5_22_22_A : in STD_LOGIC_VECTOR(N-1 downto 0);
                shaneem_5_22_22_B : in STD_LOGIC_VECTOR(N-1 downto 0);
                shaneem_5_22_22_Carry : out STD_LOGIC_VECTOR(N-1 downto 0);
                shaneem_5_22_22_Carry1, shaneem_5_22_22_overflow, shaneem_5_22_22_zero, shaneem_5_22_22_negative, shaneem_5_22_22_PCSrc,
                shaneem_5_22_22_branch : out STD_LOGIC
            );
        end entity Shameem_5_22_22_ALU;

        architecture arch of Shameem_5_22_22_ALU is
            signal shaneem_5_22_22_Sum : STD_LOGIC_VECTOR(31 downto 0);
            signal shaneem_5_22_22_Carry1, shaneem_5_22_22_overflow, shaneem_5_22_22_zero, shaneem_5_22_22_negative, shaneem_5_22_22_Equals : STD_LOGIC;
            begin
                Add_Addr_Addiu_Sub_subInstruction: process(shaneem_5_22_22_ALUopcode) begin
                    Add_Addr_Addiu_Sub_subInstruction: process(shaneem_5_22_22_ALUopcode) begin
                        Add_OR_NOR_instruction: process(shaneem_5_22_22_ALUopcode) begin
                            BNE_BEQ_instruction: process(shaneem_5_22_22_ALUopcode) begin
                                begin
                                    process (shaneem_5_22_22_OPCODE, shaneem_5_22_22_FUNCT) begin
                                        if ((shaneem_5_22_22_OPCODE = "000000" AND (shaneem_5_22_22_FUNCT = "100001" OR shaneem_5_22_22_FUNCT = "100100")) OR
                                            (shaneem_5_22_22_OPCODE = "100000" AND (shaneem_5_22_22_FUNCT = "100010" OR shaneem_5_22_22_FUNCT = "100110"))
                                            ) then
                                                shaneem_5_22_22_sum <= shaneem_5_22_22_B;
                                                shaneem_5_22_22_carry1 <= shaneem_5_22_22_a;
                                                shaneem_5_22_22_OVERFLOW <= shaneem_5_22_22_OVERFLOW;
                                                shaneem_5_22_22_zero <= shaneem_5_22_22_ZERO;
                                                shaneem_5_22_22_NEGATIVE <= shaneem_5_22_22_NEGATIVE;
                                            else
                                                shaneem_5_22_22_sum <= shaneem_5_22_22_B;
                                                shaneem_5_22_22_carry1 <= shaneem_5_22_22_a;
                                                shaneem_5_22_22_OVERFLOW <= shaneem_5_22_22_OVERFLOW;
                                                shaneem_5_22_22_zero <= shaneem_5_22_22_ZERO;
                                                shaneem_5_22_22_NEGATIVE <= shaneem_5_22_22_NEGATIVE;
                                            end if;
                                        end process;
                                    end if;
                                end if;
                            end process;
                        end if;
                    end process;
                end if;
            end process;
        end architecture;
    
```

The screenshot shows the Quartus Prime Lite Edition interface with the project navigation pane, a code editor containing the VHDL code for the ALU, and a message window indicating a successful compilation. The IP Catalog is also visible on the right side of the interface.

Figure 22: ALU VHDL code.

```

        entity Shameem_5_22_22_ALU is
            generic (N : integer := 32);
            port (
                shaneem_5_22_22_ALUopcode : in STD_LOGIC;
                shaneem_5_22_22_Funct : in STD_LOGIC_VECTOR(5 downto 0);
                shaneem_5_22_22_A : in STD_LOGIC_VECTOR(N-1 downto 0);
                shaneem_5_22_22_B : in STD_LOGIC_VECTOR(N-1 downto 0);
                shaneem_5_22_22_Carry : out STD_LOGIC_VECTOR(N-1 downto 0);
                shaneem_5_22_22_overflow, shaneem_5_22_22_zero, shaneem_5_22_22_negative, shaneem_5_22_22_Equals : STD_LOGIC;
                shaneem_5_22_22_PCSrc, shaneem_5_22_22_branch : out STD_LOGIC
            );
        end entity Shameem_5_22_22_ALU;

        architecture arch of Shameem_5_22_22_ALU is
            signal shaneem_5_22_22_Sum : STD_LOGIC_VECTOR(31 downto 0);
            signal shaneem_5_22_22_Carry1, shaneem_5_22_22_OVERFLOW, shaneem_5_22_22_ZERO, shaneem_5_22_22_NEGATIVE, shaneem_5_22_22_Equals : STD_LOGIC;
            begin
                lw_instruct: process(shaneem_5_22_22_OPCODE, shaneem_5_22_22_FUNCT) begin
                    lw_instruct: process(shaneem_5_22_22_OPCODE, shaneem_5_22_22_FUNCT) begin
                        beq_instruct: process(shaneem_5_22_22_OPCODE, shaneem_5_22_22_FUNCT) begin
                            beq_instruct: process(shaneem_5_22_22_OPCODE, shaneem_5_22_22_FUNCT) begin
                                begin
                                    process (shaneem_5_22_22_OPCODE, shaneem_5_22_22_FUNCT) begin
                                        if ((shaneem_5_22_22_OPCODE = "101011" AND (shaneem_5_22_22_FUNCT = "100001" OR shaneem_5_22_22_FUNCT = "100101")) OR
                                            (shaneem_5_22_22_OPCODE = "101010" AND (shaneem_5_22_22_FUNCT = "100010" OR shaneem_5_22_22_FUNCT = "100110"))
                                            ) then
                                                shaneem_5_22_22_sum <= shaneem_5_22_22_B;
                                                shaneem_5_22_22_carry1 <= shaneem_5_22_22_a;
                                                shaneem_5_22_22_OVERFLOW <= shaneem_5_22_22_OVERFLOW;
                                                shaneem_5_22_22_zero <= shaneem_5_22_22_ZERO;
                                                shaneem_5_22_22_NEGATIVE <= shaneem_5_22_22_NEGATIVE;
                                            else
                                                shaneem_5_22_22_sum <= shaneem_5_22_22_B;
                                                shaneem_5_22_22_carry1 <= shaneem_5_22_22_a;
                                                shaneem_5_22_22_OVERFLOW <= shaneem_5_22_22_OVERFLOW;
                                                shaneem_5_22_22_zero <= shaneem_5_22_22_ZERO;
                                                shaneem_5_22_22_NEGATIVE <= shaneem_5_22_22_NEGATIVE;
                                            end if;
                                        end process;
                                    end if;
                                end if;
                            end process;
                        end if;
                    end process;
                end if;
            end process;
        end architecture;
    
```

This screenshot continues the VHDL code for the ALU, showing the implementation of load (lw), store (sw), branch on equal (beq), and branch on not-equal (bne) instructions. It includes logic to update flags based on the results and utilize them for subsequent instructions.

Figure 23: ALU VHDL code continued.

Single Cycle CPU

Quartus Prime Lite Edition - C:/Users/Azwad/Desktop/CPU/Shameem_5_22_22_CPU/Shameem_5_22_22_CPU - Shameem_5_22_22_CPU

File Edit View Project Assignments Processing Tools Window Help

Shameem_5_22_22_CPU.vhd

```

1  Library IEEE;
2  use IEEE.STD_LOGIC_1164.all;
3  use work.Shameem_5_22_22_Package.all;
4
5  generic (N : INTEGER := 32);
6  entity Shameem_5_22_22_CPU is
7    PORT (
8      shameem_5_22_22_clock: in STD_LOGIC;
9    );
10 end Shameem_5_22_22_CPU;
11
12 architecture arch of shameem_5_22_22_CPU is
13   constant shameem_5_22_22_csr: STD_LOGIC_VECTOR(31 downto 0) := x"00000000";
14   signal Shameem_5_22_22_opcode: STD_LOGIC_VECTOR(5 downto 0);
15   signal Shameem_5_22_22_shamt: STD_LOGIC_VECTOR(4 downto 0);
16   signal Shameem_5_22_22_funct: STD_LOGIC_VECTOR(5 downto 0);
17   signal Shameem_5_22_22_rs: STD_LOGIC_VECTOR(4 downto 0);
18   signal Shameem_5_22_22_instruction: Shameem_5_22_22_Instructions: STD_LOGIC_VECTOR(31 downto 0);
19   signal Shameem_5_22_22_RegDst: Shameem_5_22_22_RegDst; Shameem_5_22_22_Extop, Shameem_5_22_22_ALUsrc,
20   Shameem_5_22_22_ALUctr, Shameem_5_22_22_MemW, Shameem_5_22_22_MemToReg, Shameem_5_22_22_Branch : STD_LOGIC;
21   signal Shameem_5_22_22_MemR: STD_LOGIC_VECTOR(4 downto 0);
22   signal Shameem_5_22_22_Mux2_1: STD_LOGIC_VECTOR(31 downto 0);
23   signal Shameem_5_22_22_ALUOutput: STD_LOGIC_VECTOR(31 downto 0);
24   signal Shameem_5_22_22_Carry, Shameem_5_22_22_overflow, Shameem_5_22_22_Zero : STD_LOGIC;
25   signal Shameem_5_22_22_DataFromOutput: STD_LOGIC_VECTOR(31 downto 0);
26   signal Shameem_5_22_22_Imm16: STD_LOGIC_VECTOR(15 downto 0); Shameem_5_22_22_PC : STD_LOGIC_VECTOR(31 downto 0);
27
28 begin
29   shameem_5_22_22_instruction_memory: Shameem_5_22_22_instructionMemory
30     port map (Shameem_5_22_22_clock, Shameem_5_22_22_csr&PC, Shameem_5_22_22_opcode, Shameem_5_22_22_RS, Shameem_5_22_22_RT, Shameem_5_22_22_RD,
31     Shameem_5_22_22_Shamt, Shameem_5_22_22_Funct, Shameem_5_22_22_Imm16, Shameem_5_22_22_instruction);
32   shameem_5_22_22_Control_Unit: Shameem_5_22_22_ControlUnit
33     port map (Shameem_5_22_22_opcode, Shameem_5_22_22_Funct, Shameem_5_22_22_RegWr, Shameem_5_22_22_RegDst, Shameem_5_22_22_Extop, Shameem_5_22_22_ALUsrc,
34     Shameem_5_22_22_Extop, Shameem_5_22_22_MemToReg, Shameem_5_22_22_Branch);
35   shameem_5_22_22_Instruction_Unit: Shameem_5_22_22_Instruction
36     port map (Shameem_5_22_22_clock, Shameem_5_22_22_instruction, Shameem_5_22_22_Instructions);
37   shameem_5_22_22_Memory: Shameem_5_22_22_Memory
38     port map (Shameem_5_22_22_Address, Shameem_5_22_22_Data, Shameem_5_22_22_BusA, Shameem_5_22_22_BusB, Shameem_5_22_22_BusC);
39   shameem_5_22_22_Extop: Shameem_5_22_22_Extop
40     port map (Shameem_5_22_22_Clock, Shameem_5_22_22_RegWr, Shameem_5_22_22_RegDst, Shameem_5_22_22_RT, Shameem_5_22_22_RD, Shameem_5_22_22_RW);
41   shameem_5_22_22_ALU: Shameem_5_22_22_ALU
42     port map (Shameem_5_22_22_ALUctr, Shameem_5_22_22_RS, Shameem_5_22_22_BusA, Shameem_5_22_22_BusB);
43   shameem_5_22_22_ALU: Shameem_5_22_22_ALU
44     port map (Shameem_5_22_22_ALUctr, Shameem_5_22_22_RS, Shameem_5_22_22_BusA, Shameem_5_22_22_BusB, Shameem_5_22_22_BusC);
45   shameem_5_22_22_ALU: Shameem_5_22_22_ALU
46     port map (Shameem_5_22_22_ALUctr, Shameem_5_22_22_RS, Shameem_5_22_22_BusA, Shameem_5_22_22_BusB, Shameem_5_22_22_BusC);

```

IP Catalog

Installed IP
Project Directory
No Selection Available

Library
Basic Functions
DSP
Interface Protocols
Memory Interfaces and Controllers
Processors and Peripherals
University Program

Search for Partner IP

Tasks Compilation

Task

- ✓ ▶ Compile Design
- ✓ ▶ Analysis & Synthesis
- ✓ ▶ Fitter (Place & Route)
- ✓ ▶ Assembler (Generate program)
- ✓ ▶ Timing Analysis
- ✓ ▶ EDA Netlist Writer
- Edit Settings
- Program Device (Open Programmer)

All Find Find Next

Type ID Message

293000 quartus Prime Full compilation was successful. 0 errors, 182 warnings

Messages System Processing (404)

100% 00:00:53

Figure 24: Single Cycle CPU VHDL code.

Quartus Prime Lite Edition - C:/Users/Azwad/Desktop/CPU/Shameem_5_22_22_CPU/Shameem_5_22_22_CPU - Shameem_5_22_22_CPU

File Edit View Project Assignments Processing Tools Window Help

Project Navigator Files Search altera.com

Shameem_5_22_22_CPU.vhd

```

15   signal Shameem_5_22_22_Shamt: STD_LOGIC_VECTOR(4 downto 0);
16   signal Shameem_5_22_22_Funct: STD_LOGIC_VECTOR(5 downto 0);
17   signal Shameem_5_22_22_Imm16: STD_LOGIC_VECTOR(15 downto 0);
18   signal Shameem_5_22_22_PCSrc: Shameem_5_22_22_BegOp.Shameem_5_22_22_RegDst, Shameem_5_22_22_Extop, Shameem_5_22_22_ALUSrc,
19   signal Shameem_5_22_22_ALUctr, Shameem_5_22_22_Memr, Shameem_5_22_22_MemReq, Shameem_5_22_22_Branch : STD_LOGIC;
20   signal Shameem_5_22_22_RS: Shameem_5_22_22_RT, Shameem_5_22_22_RD, Shameem_5_22_22_RW: STD_LOGIC_VECTOR(4 downto 0);
21   signal Shameem_5_22_22_BusA: Shameem_5_22_22_BusB, Shameem_5_22_22_MuxBusB, Shameem_5_22_22_BusW : STD_LOGIC_VECTOR(31 downto 0);
22   signal Shameem_5_22_22_ALUOutput: STD_LOGIC_VECTOR(31 downto 0);
23   signal Shameem_5_22_22_Carry, Shameem_5_22_22_Overflow, Shameem_5_22_22_Negative, Shameem_5_22_22_Zero : STD_LOGIC;
24   signal Shameem_5_22_22_DataMemoryOutput: STD_LOGIC_VECTOR(31 downto 0);
25   signal Shameem_5_22_22(nextAddress_1, Shameem_5_22_22_nextAddress_2, Shameem_5_22_22_PC : STD_LOGIC_VECTOR(31 downto 0);
26   signal Shameem_5_22_22_ExtendedImm16 : STD_LOGIC_VECTOR(31 downto 0);
27 begin
28   Shameem_5_22_22_Instruction_Memory: Shameem_5_22_22_InstructionMemory
29     port map (Shameem_5_22_22_Clock, Shameem_5_22_22_cinPC, Shameem_5_22_22_Opcode, Shameem_5_22_22_RS, Shameem_5_22_22_RT, Shameem_5_22_22_RD,
30     port map (Shameem_5_22_22_ControlUnit, Shameem_5_22_22_ControlUnit);
31   Shameem_5_22_22_Control_Unit: Shameem_5_22_22_ControlUnit
32     port map (Shameem_5_22_22_Opcode, Shameem_5_22_22_Funct, Shameem_5_22_22_RegWr, Shameem_5_22_22_Extop, Shameem_5_22_22_ALUSrc,
33     port map (Shameem_5_22_22_ALUctr, Shameem_5_22_22_Memr, Shameem_5_22_22_MemReq, Shameem_5_22_22_Branch);
34   Shameem_5_22_22_InstructionRegister: Shameem_5_22_22_InstructionRegister
35     port map (Shameem_5_22_22_Clock, Shameem_5_22_22_Instruction, Shameem_5_22_22_Instructions);
36   Shameem_5_22_22_InputRW: Shameem_5_22_22_MUX2_1abit
37     port map (Shameem_5_22_22_BusA, Shameem_5_22_22_BusB, Shameem_5_22_22_RS, Shameem_5_22_22_RT, Shameem_5_22_22_RD, Shameem_5_22_22_RW);
38   Shameem_5_22_22_DataMemory: Shameem_5_22_22_DataMemory
39     port map (Shameem_5_22_22_BusA, Shameem_5_22_22_BusB, Shameem_5_22_22_ExtendedImm16, Shameem_5_22_22_MuxBusB);
40   Shameem_5_22_22_ALU_MUX: Shameem_5_22_22_MUX2_1abit
41     port map (Shameem_5_22_22_ALUctr, Shameem_5_22_22_ALU)
42     port map (Shameem_5_22_22_ALUOutput, Shameem_5_22_22_BusA, Shameem_5_22_22_MuxBusB,
43     port map (Shameem_5_22_22_ALUctr, Shameem_5_22_22_Opcode, Shameem_5_22_22_Funct, Shameem_5_22_22_BusA, Shameem_5_22_22_MuxBusB,
44     port map (Shameem_5_22_22_DataMemory, Shameem_5_22_22_DataMemory
45     port map (Shameem_5_22_22_BusA, Shameem_5_22_22_BusB, Shameem_5_22_22_Overflow, Shameem_5_22_22_Zero, Shameem_5_22_22_Negative);
46   Shameem_5_22_22_BusMUX: Shameem_5_22_22_MUX2_1
47     port map (Shameem_5_22_22_BusA, Shameem_5_22_22_BusB, Shameem_5_22_22_BusW);
48   Shameem_5_22_22_BranchAddress: Shameem_5_22_22_BranchAddress
49     port map (Shameem_5_22_22_BranchAddress, Adder: Shameem_5_22_22_ALUOutput, Shameem_5_22_22_DataMemoryOutput, Shameem_5_22_22_BusW);
50   Shameem_5_22_22_BranchAddress: Shameem_5_22_22_BranchAddress
51     port map (Shameem_5_22_22_BranchAddress, Adder: Shameem_5_22_22_ALUOutput, Shameem_5_22_22_DataMemoryOutput, Shameem_5_22_22_BusW);
52   Shameem_5_22_22_BranchAddress: Shameem_5_22_22_BranchAddress
53     port map (Shameem_5_22_22_BranchAddress, Adder: Shameem_5_22_22_ALUOutput, Shameem_5_22_22_DataMemoryOutput, Shameem_5_22_22_BusW);
54   Shameem_5_22_22_BranchAddress: Shameem_5_22_22_BranchAddress
55     port map (Shameem_5_22_22_BranchAddress, Adder: Shameem_5_22_22_ALUOutput, Shameem_5_22_22_DataMemoryOutput, Shameem_5_22_22_BusW);
56   Shameem_5_22_22_BranchAddress: Shameem_5_22_22_BranchAddress
57     port map (Shameem_5_22_22_BranchAddress, Adder: Shameem_5_22_22_ALUOutput, Shameem_5_22_22_DataMemoryOutput, Shameem_5_22_22_BusW);
58   Shameem_5_22_22_BranchAddress: Shameem_5_22_22_BranchAddress
59     port map (Shameem_5_22_22_BranchAddress, Adder: Shameem_5_22_22_ALUOutput, Shameem_5_22_22_DataMemoryOutput, Shameem_5_22_22_BusW);
60 end arch;
```

Tasks Compilation + Add...

Type ID Message

293000 Quartus Prime Full Compilation was successful. 0 errors, 182 warnings

System Processing (404)

100% 00:05

Figure 25: Single Cycle CPU VHDL code continued.

1. Perform the ultimate test of your design by inputting integers x_1, x_2, x_3, x_4, x_5 , into DATA Memory, and computing the following expression

$$Z = \sum_{k=1}^5 X_k$$

Simulation

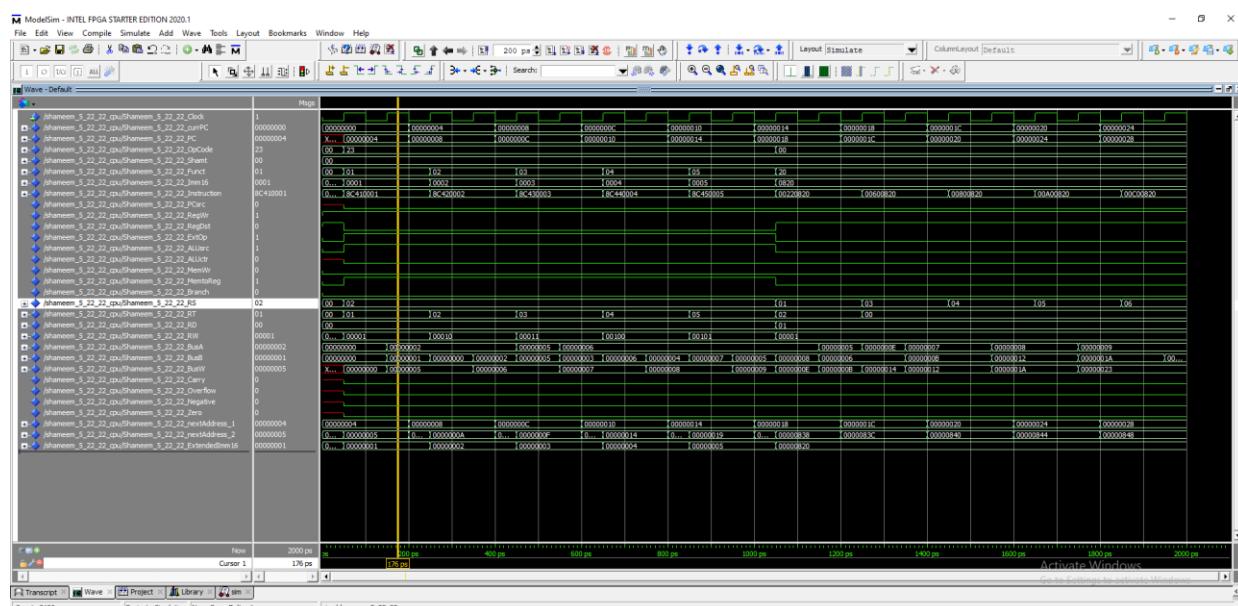


Figure 26: Simulation Waveform

This is the simulation of using 5 load word instructions and then 5 add instructions and finally one save word instruction. We will be verifying every instruction one by one in a zoomed in snippet.

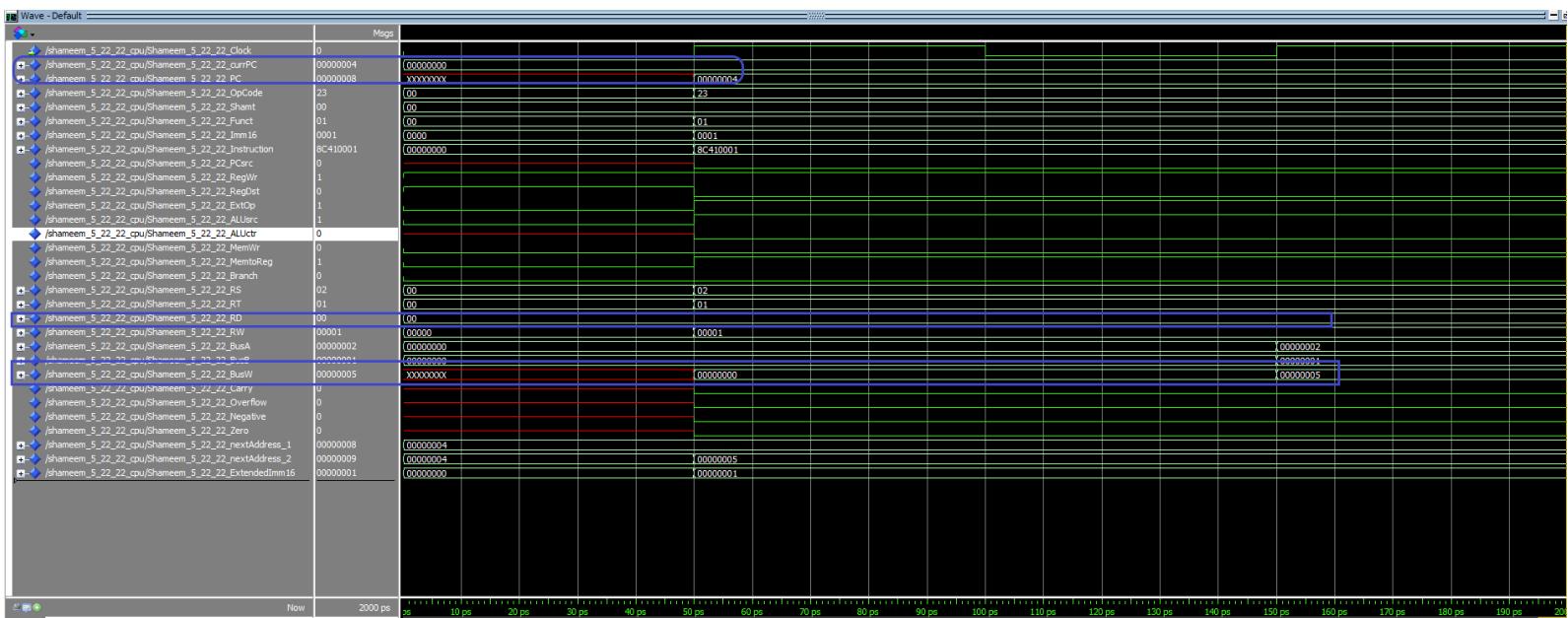


Figure 27: Simulation Waveform Zoomed in 100 – 200 ps - LW

First of all let's verify the program counter. On the top it is easily noticeable that the PC goes from 0x00000000 to 0x00000004 at 50 ps this is because the BNE, BEQ or J instructions are not called so we only add 4 to the current program counter. Next we see that at RW at 50 ps the register is the register at 00001. Then if you look at BusW it is noticeable that at 150 ps the number 0x00000005 is loaded into RW which is the register at 00001.

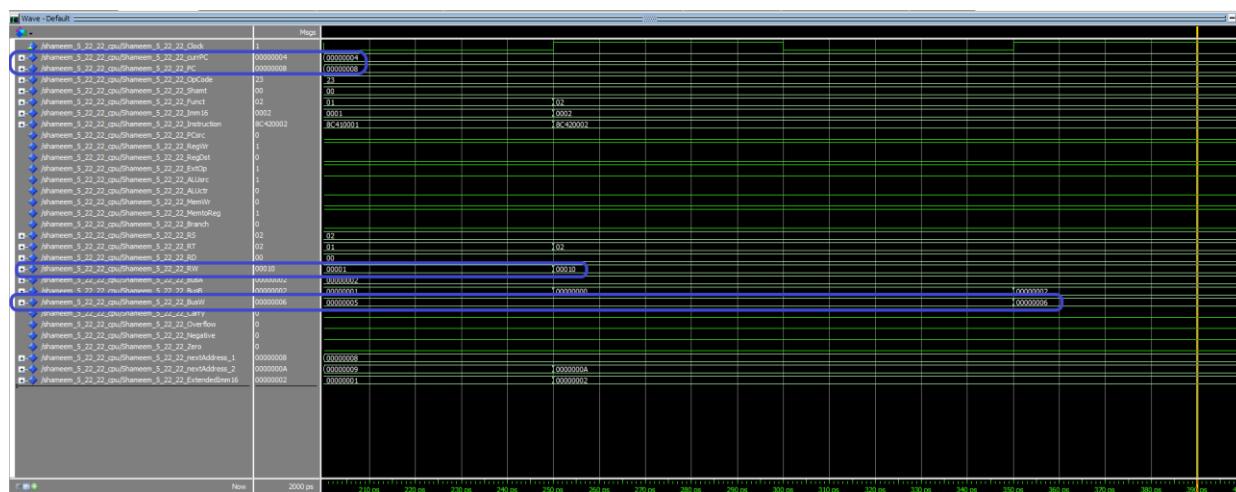


Figure 28: Simulation Waveform Zoomed in 200 - 400ps - LW

First of all, let's verify the program counter. On the top it is easily noticeable that the PC goes from 0x00000004 to 0x00000008 at 200 ps this is because the BNE, BEQ or J instructions are not called so we only add 4 to the current program counter. Next, we see that at RW at 250 ps the register is the register at 00010. Then if you look at BusW it is noticeable that at 150 ps the number 0x00000006 is loaded into RW which is the register at 00010.

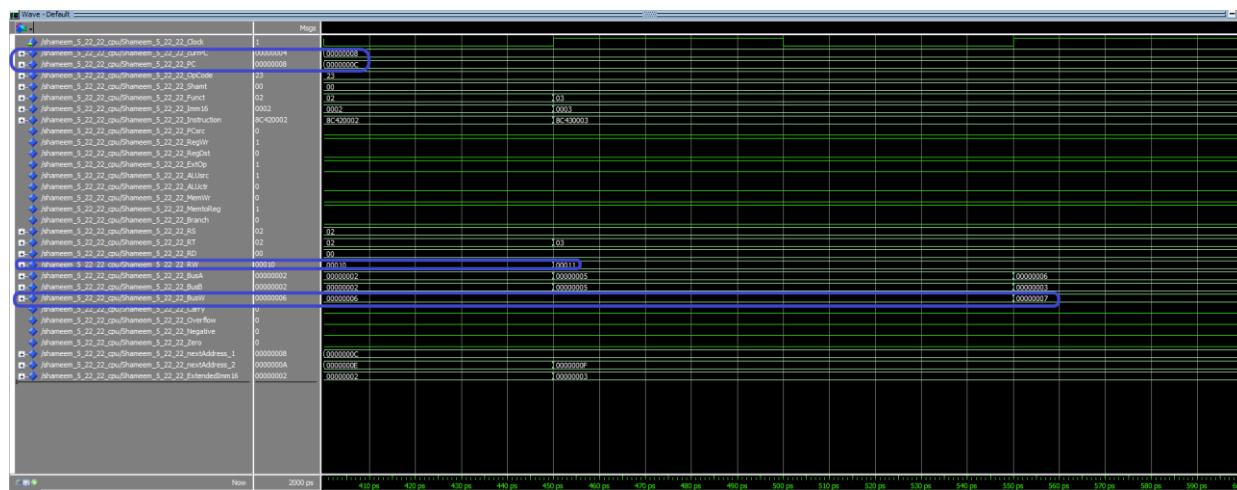


Figure 29: Simulation Waveform Zoomed in 400 - 600 ps - LW

First of all let's verify the program counter. On the top it is easily noticeable that the PC goes from 0x00000008 to 0x0000000C at 400 ps this is because the BNE, BEQ or J instructions are called so we only add 4 to the current program counter. Next we see that at RW at 450 ps the register is the register at 00011. Then if you look at BusW it is noticeable that at 550 ps the number 0x00000007 is loaded into RW which is the register at 00011.

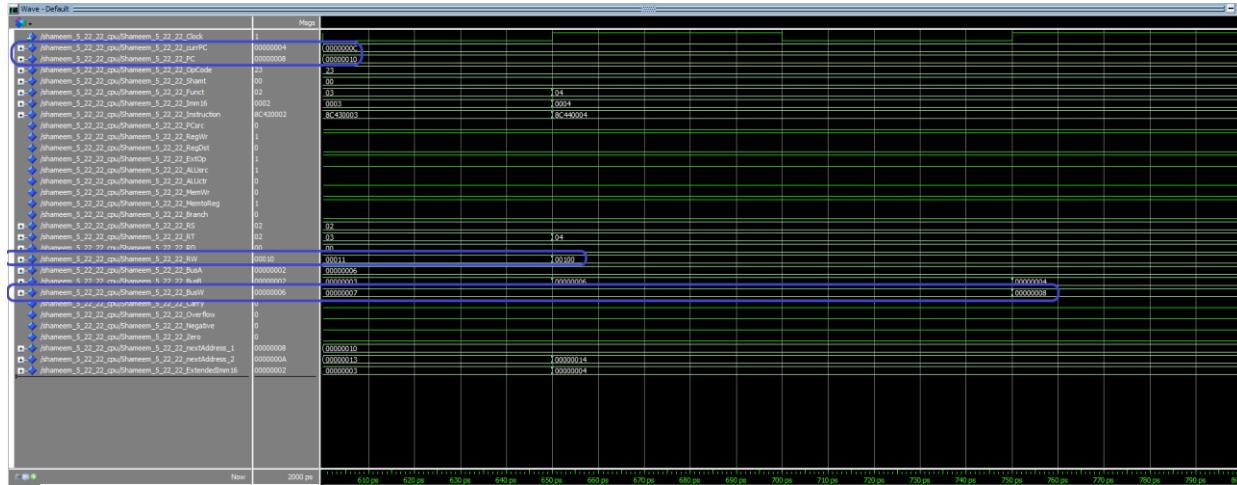


Figure 30: Simulation Waveform Zoomed in 600 - 800ps - LW

First of all let's verify the program counter. On the top it is easily noticeable that the PC goes from 0x0000000C to 0x00000010 at 600 ps this is because the BNE, BEQ or J instructions are called so we only add 4 to the current program counter. Next we see that at RW at 650 ps the register is the register at 00100. Then if you look at BusW it is noticeable that at 750 ps the number 0x00000008 is loaded into RW which is the register at 00100.

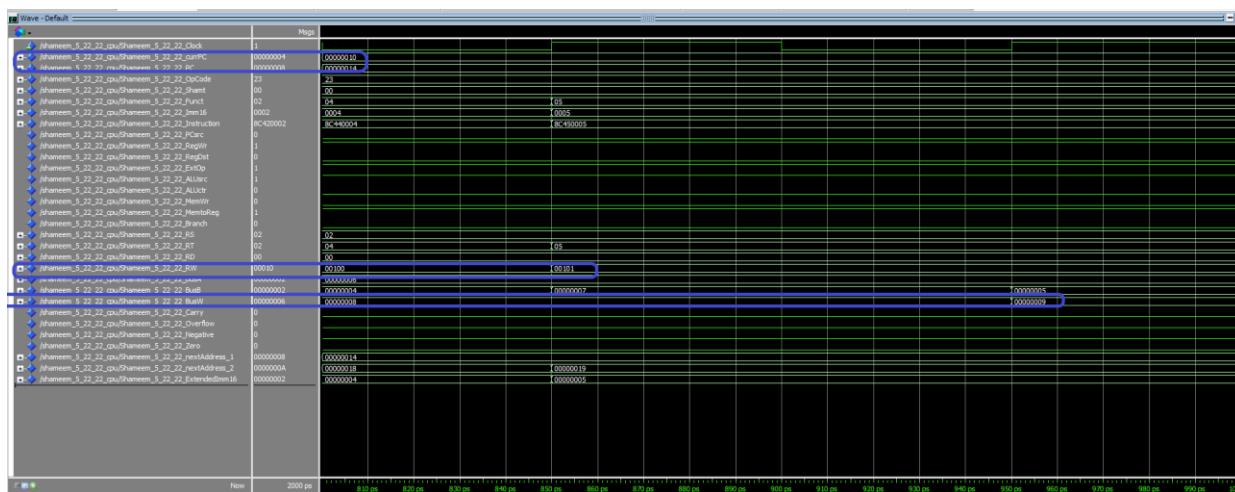


Figure 31: Simulation Waveform Zoomed in 800-1000ps - LW

First of all, let's verify the program counter. On the top it is easily noticeable that the PC goes from 0x00000010 to 0x00000014 at 800 ps this is because the BNE, BEQ or J instructions are called so we only add 4 to the current program counter. Next, we see that at RW at 850 ps the register is the register at 00101. Then if you look at BusW it is noticeable that at 950 ps the number 0x00000009 is loaded into RW which is the register at 00101. Therefore, at this point 5 instructions have been completed and 5 integers are stored in the registers 00001, 00010, 00011, 00100, 00101

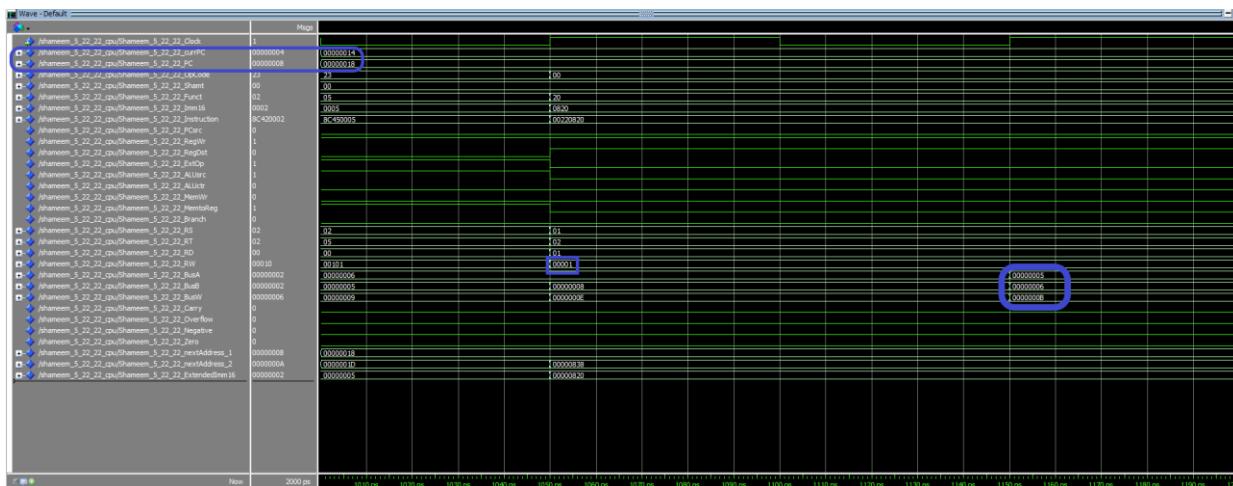


Figure 32: Simulation Waveform Zoomed in 1000 - 1200ps - Add

First of all, let's verify the program counter. On the top it is easily noticeable that the PC goes from 0x00000014 to 0x00000018 at 800 ps this is because the BNE, BEQ or J instructions are called so we only add 4 to the current program counter. Next, we see that RS is 1 which is register 00001, and RT is 2 which is register 00010 at 1050 ps. Then at 1150 ps it is noticeable that the numbers 0x00000005 and 0x00000006 we have loaded into the registers are adding and we get the result 0x0000000B in BusW. Furthermore, RW is 00001 from 1050 ps and after so the sum of addition will be saved in the register 00001.

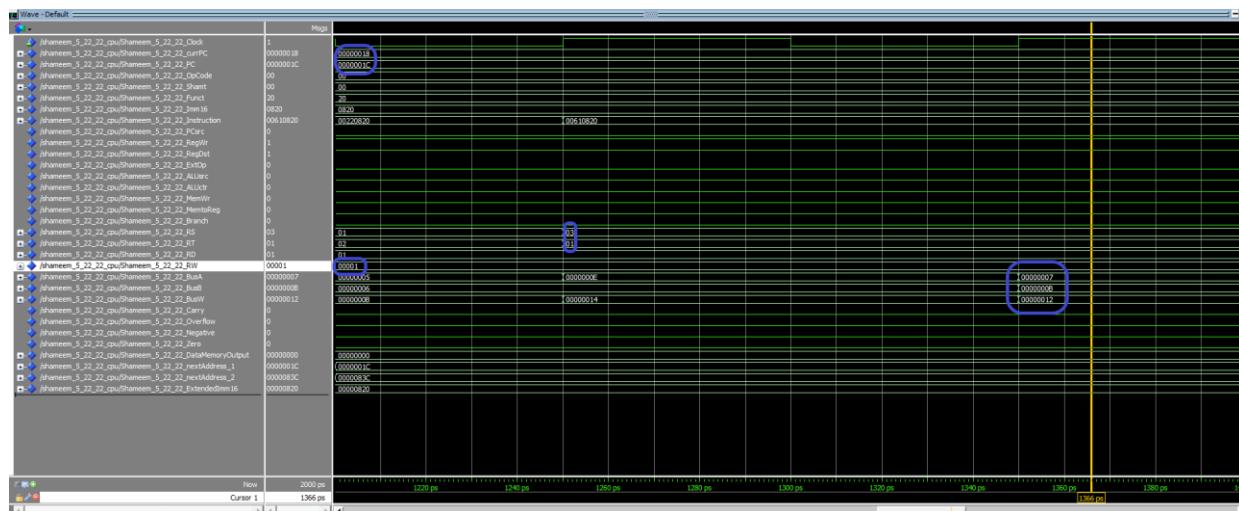


Figure 33: Simulation Waveform Zoomed in 1200 - 1400ps - Add

First of all, let's verify the program counter. On the top it is easily noticeable that the PC goes from 0x00000018 to 0x0000001C at 1200 ps this is because the BNE, BEQ or J instructions are called so we only add 4 to the current program counter. Next, we see that RS is 3 which is register 00011, and RT is 1 which is register 00001 at 1050 ps. Then at 1350 ps it is noticeable that the number 0x00000007 in BusA we have loaded is there and the number 0x0000000B in BusB, which is the sum we added for previously is now being added and gives us the value of 0x00000012 in BusW. Furthermore, RW is 00001 from 1200 ps and after so the sum of addition will be saved in the register 00001.

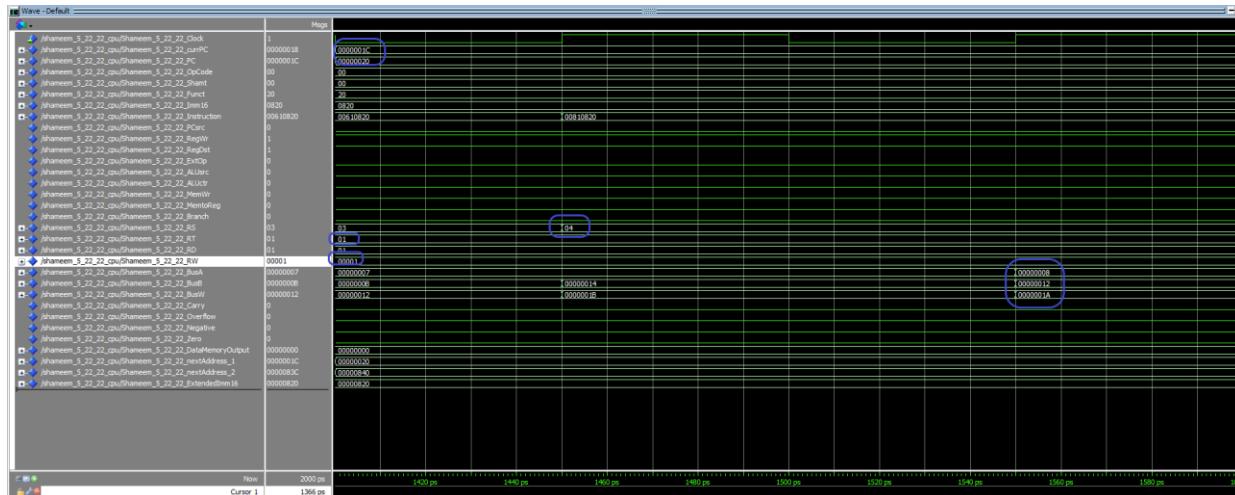


Figure 34: Simulation Waveform Zoomed in 1400 - 1600ps - Add

First of all, let's verify the program counter. On the top it is easily noticeable that the PC goes from 0x0000001C to 0x00000020 at 1400 ps this is because the BNE, BEQ or J instructions are called so we only add 4 to the current program counter. Next, we see that RS is 4 which is register 00100, and RT is 1 which is register 00001 at 1450 ps. Then at 1550 ps it is noticeable that the number 0x00000008 in BusA we have loaded is there and the number 0x00000012 in BusB, which is the sum we added for previously is now being added and gives us the value of 0x0000001A in BusW. Furthermore, RW is 00001 from 1400 ps and after so the sum of addition will be saved in the register 00001.

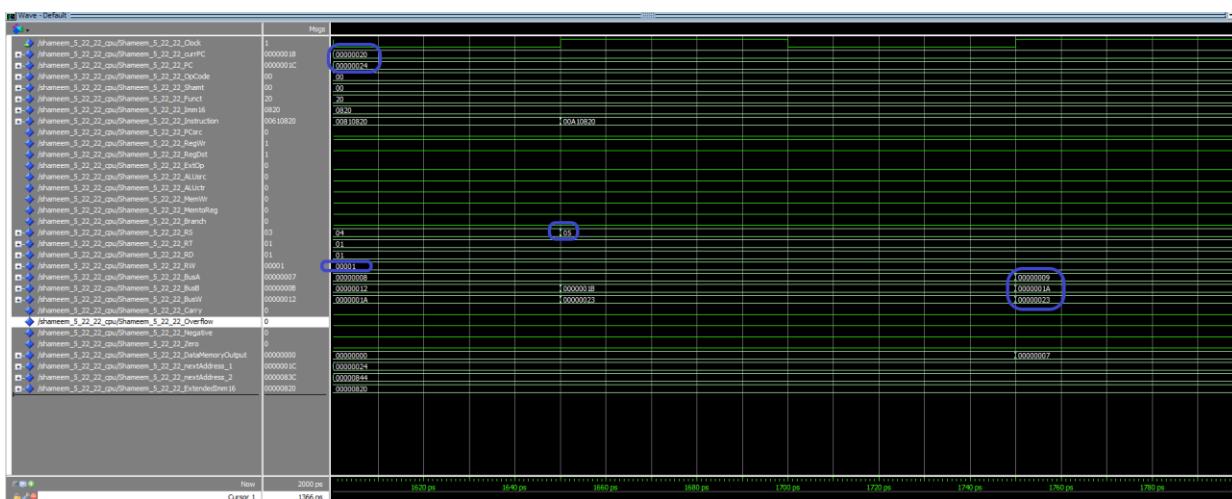


Figure 34: Simulation Waveform Zoomed in 1600 - 1800ps - Add

First of all, let's verify the program counter. On the top it is easily noticeable that the PC goes from 0x00000020 to 0x00000024 at 1600 ps this is because the BNE, BEQ or J instructions are called so we only add 4 to the current program counter. Next, we see that RS is 5 which is register 00101, and RT is 1 which is register 00001 at 1650 ps. Then in 1750 ps it is noticeable that the number 0x00000009 in BusA we have loaded is there and the number 0x0000001A in BusB, which is the sum we added for previously is now being added and gives us the value of 0x00000023 in BusW. Furthermore, RW is 00001 from 1600 ps and after so the sum of addition will be saved in the register 00001.

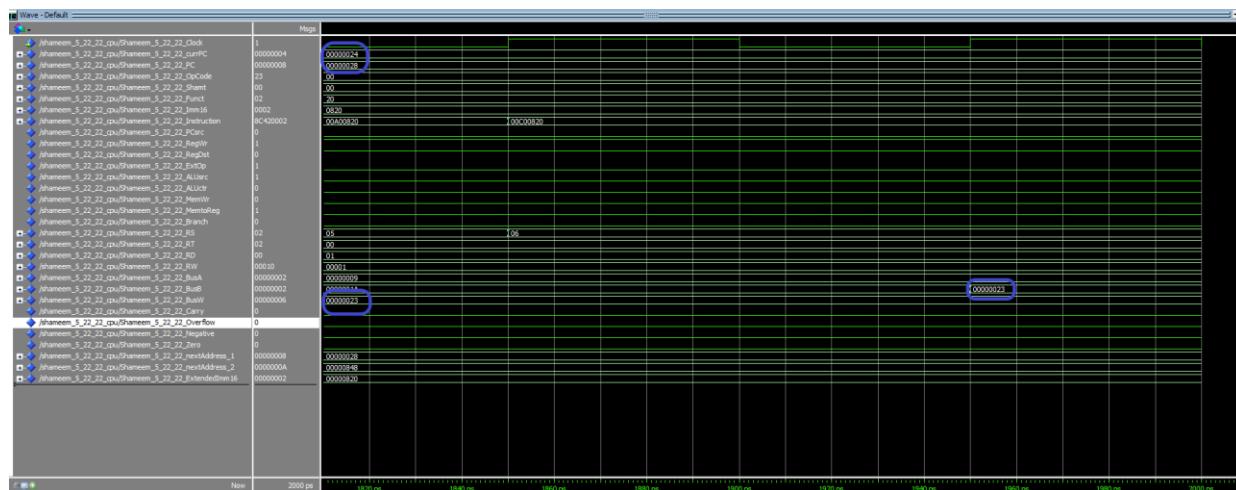


Figure 35: Simulation Waveform Zoomed in 1800 – 2000 ps - SW

First of all, let's verify the program counter. On the top it is easily noticeable that the PC goes from 0x00000024 to 0x00000028 at 1800 ps this is because the BNE, BEQ or J instructions are called so we only add 4 to the current program counter. We see at 1800 ps RW is 00001 and busW is 0x00000023. Furthermore in 1950 ps we see that BusB shows 0x00000023, which shows that the number has saved properly.

MIPS in Mars Simulator

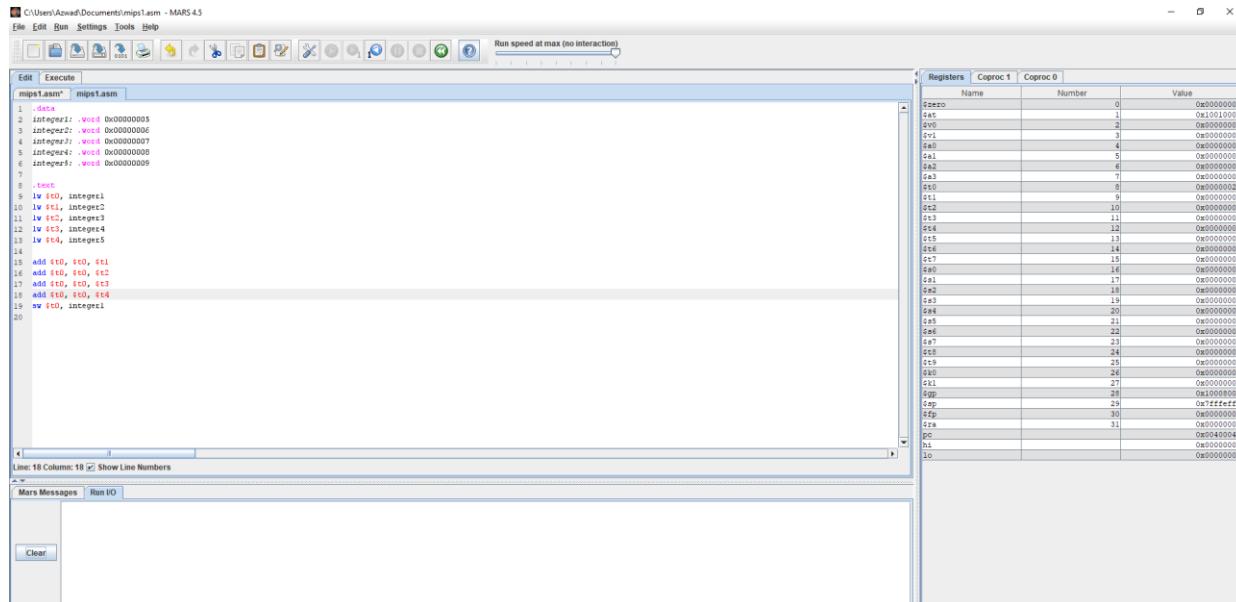


Figure 36: MIPS Code in MARS Simulator.

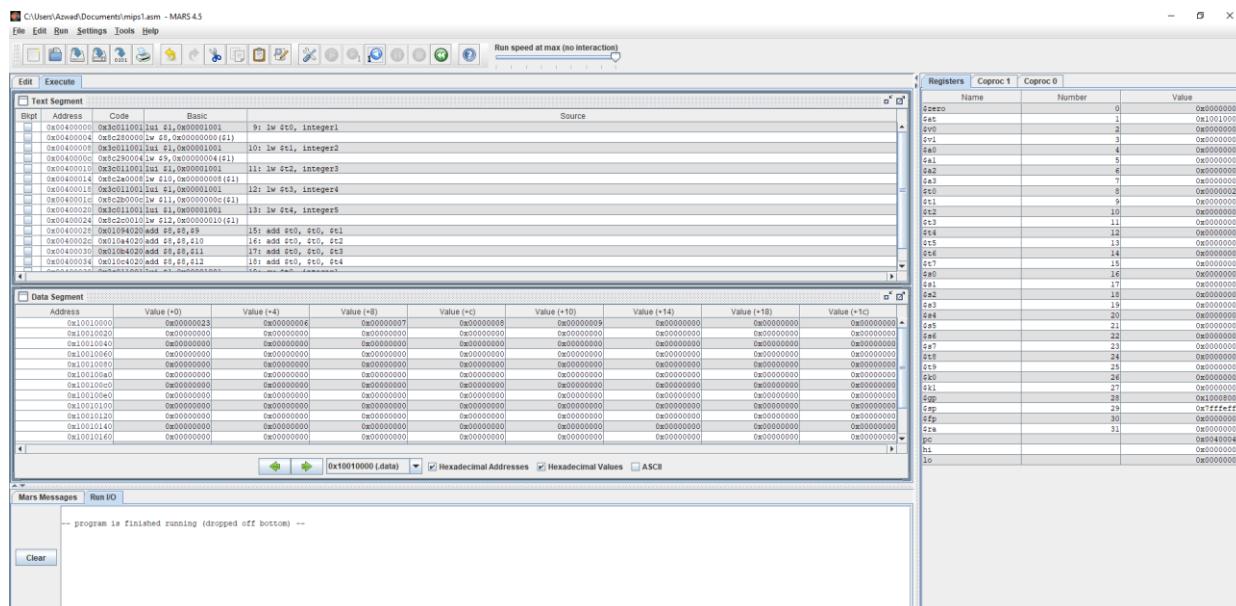


Figure 37: Executed in MARS Simulator.

At the Data Segment at 0x10010000, MARS Simulator also shows that

0x00000023 is the sum of the addition of the 5 numbers, just like the simulation waveforms.

Explanation

Let's recap we got a total of 0x23. To obtain this total we did $0x5 + 0x6 = 0xB$ then $0xB + 0x7 = 0x12$ then $0x8 + 0x12 = 0x1A$ then $0x9 + 0x1A = 0x23$. To verify if this is correct, we check the decimal version can add $5 + 6 + 7 + 8 + 9 = 35$. 35 is in decimal, but in hexadecimal 35(decimal) is 0x23. So far by checking decimal version the correctness stacks up to the hexadecimal version. This is furthered by the same numbers showing up in MIPS in MARS Simulator. Therefore, we can say that the simulation correctly computes the number in VHDL.

Conclusion

The Single Cycle CPU lab was a great assignment to test our VHDL skills to simulate MIPS instructions in VHDL. Specifically, the assignment was to write a program to compute the sum of five integers. Furthermore, we had to prove the correctness of the simulation using MIPS instructions in MARS Simulator. Therefore, by using VHDL and creating components to simulate the MIPS instruction to get the sum of five integers we learn a lot about how the CPU works and how we can use VHDL to simulate things that may be happening in the CPU of our computers.