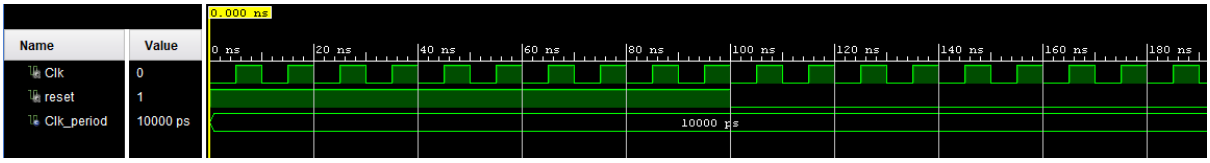
CS2022 PROJECT 2- MICROCODED INSTRUCTION SET PROCESSOR

Aniket Agarwal 17317437

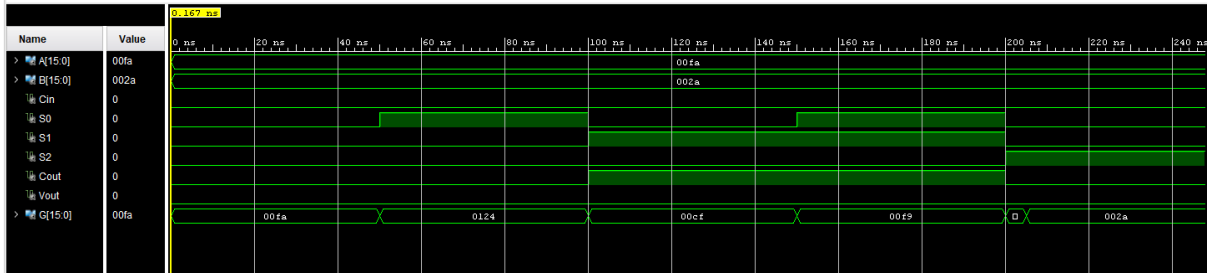
Contents:

1. Results of Test Benches

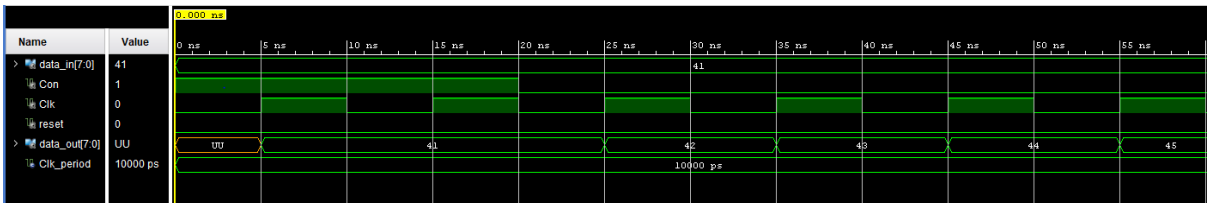
1.1 PROCESSOR FINAL FILE



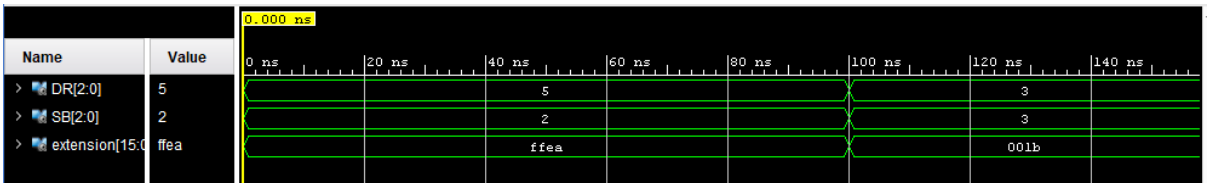
1.1 ARITHEMATIC LOGICAL UNIT



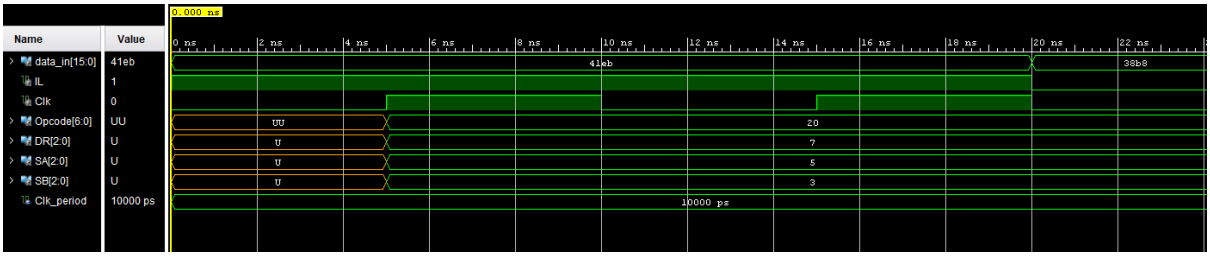
1.2 CONTROL ADDRESS REGISTER



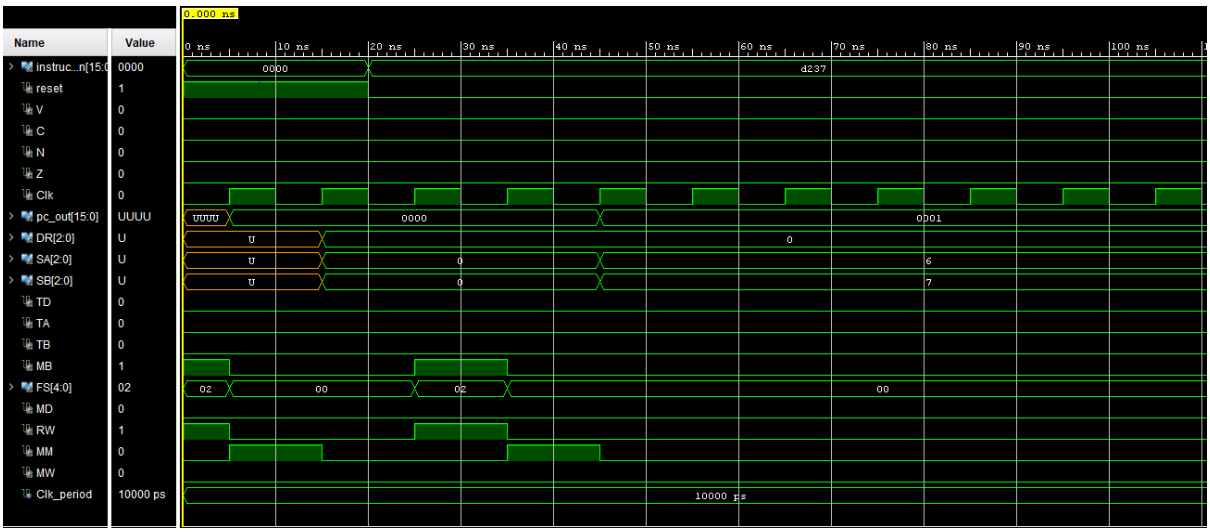
1.3 EXTEND



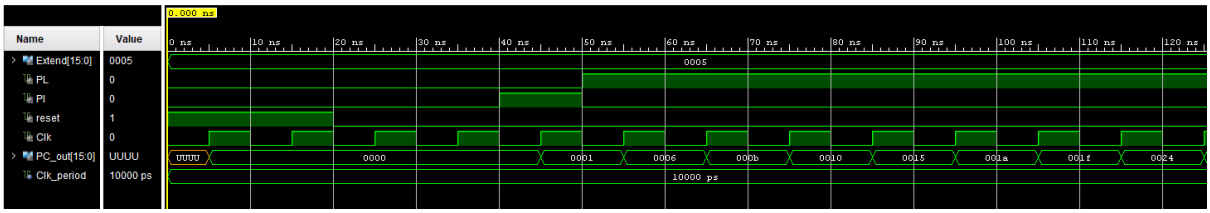
1.4 INSTRUCTION REGISTER



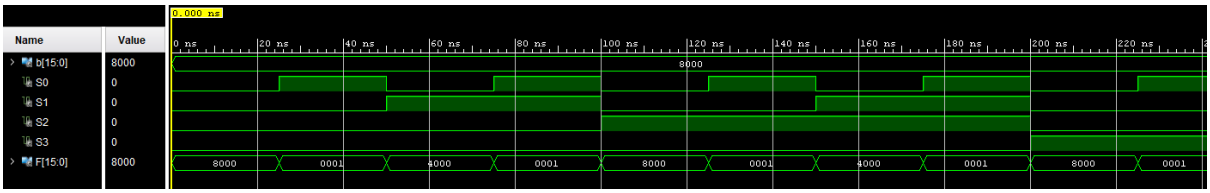
1.5 MICROPROGRAMMED CONTROL



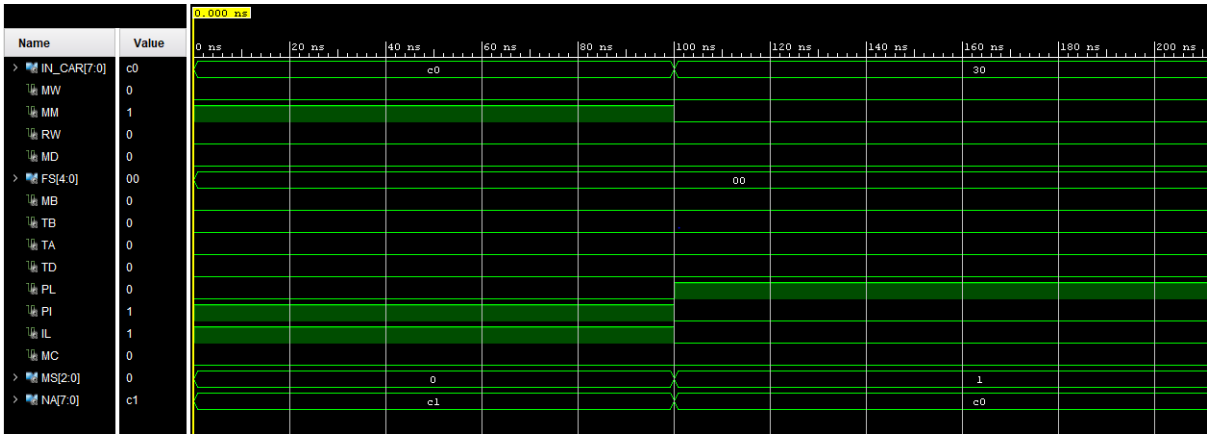
1.6 PROGRAM COUNTER



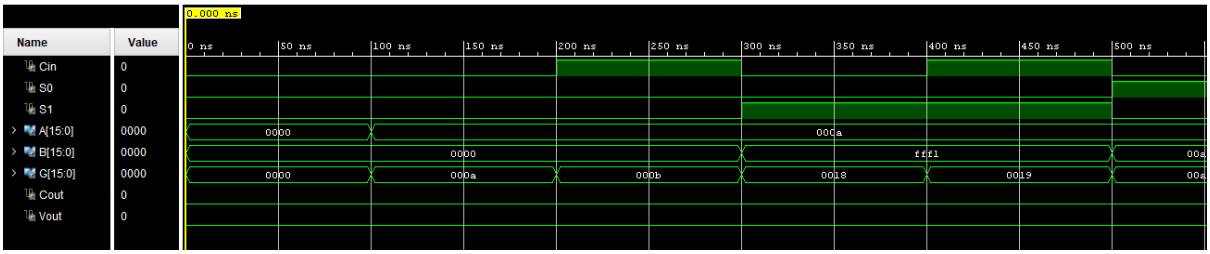
1.7 BARREL SHIFTER



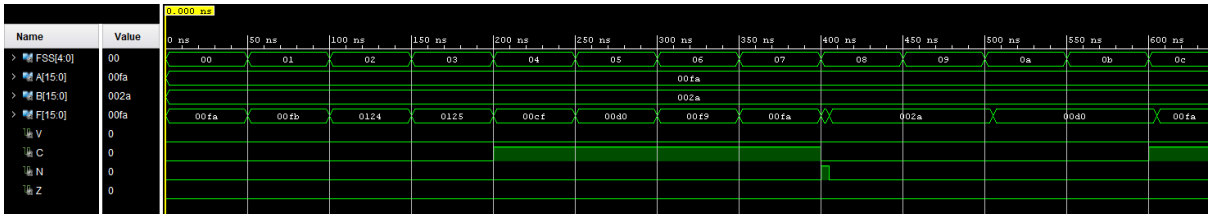
1.8 CONTROL MEMORY



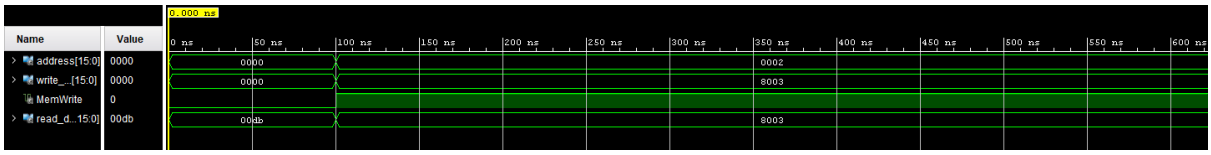
1.9 ARITHEMATIC CIRCUIT



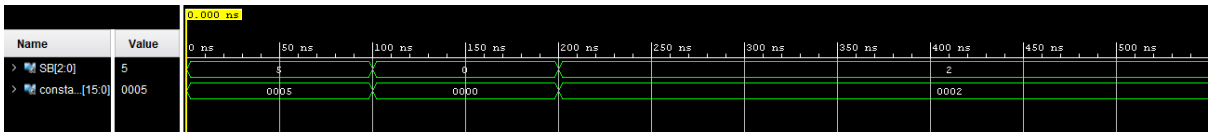
1.10 FUNCTIONAL UNIT



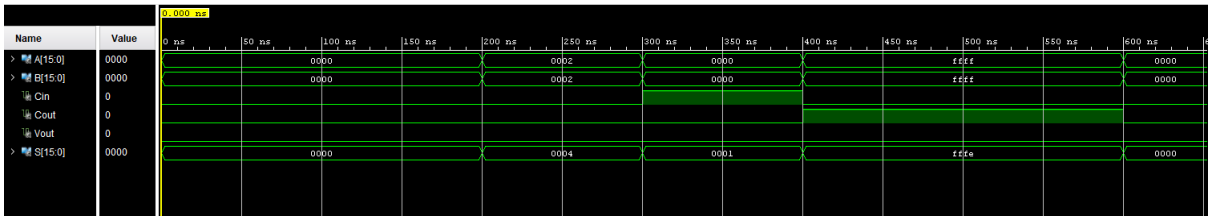
1.11 MEMORY



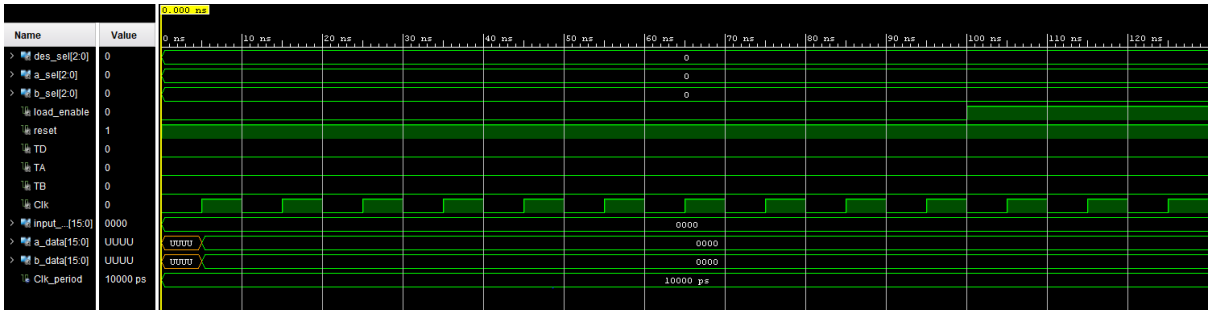
1.12 ZERO FILL



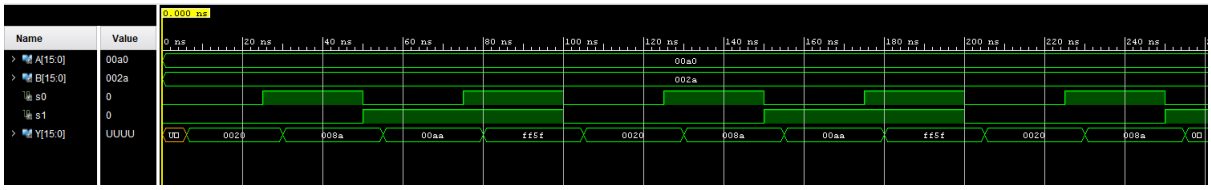
1.13 RIPPLE ADDER



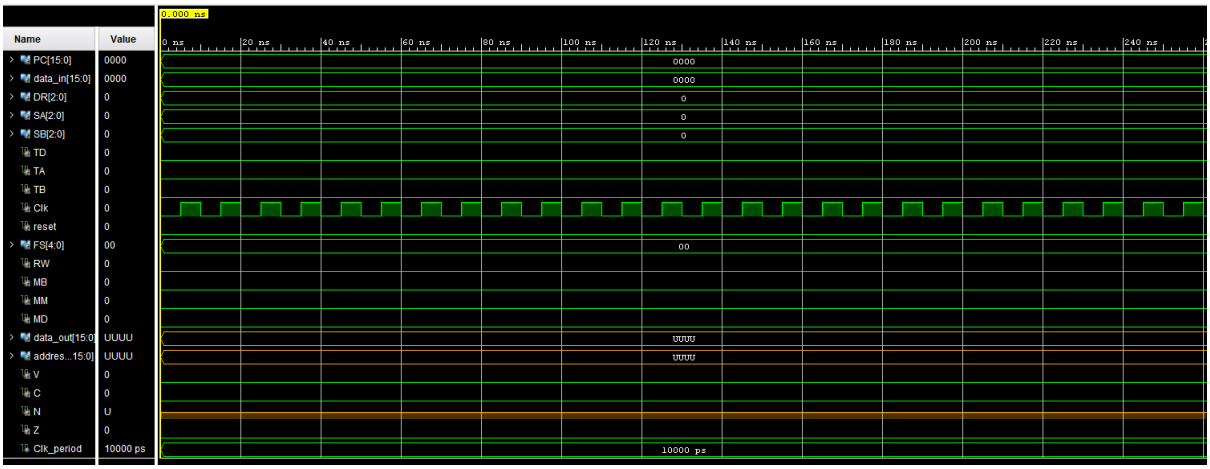
1.14 REGISTER FILE



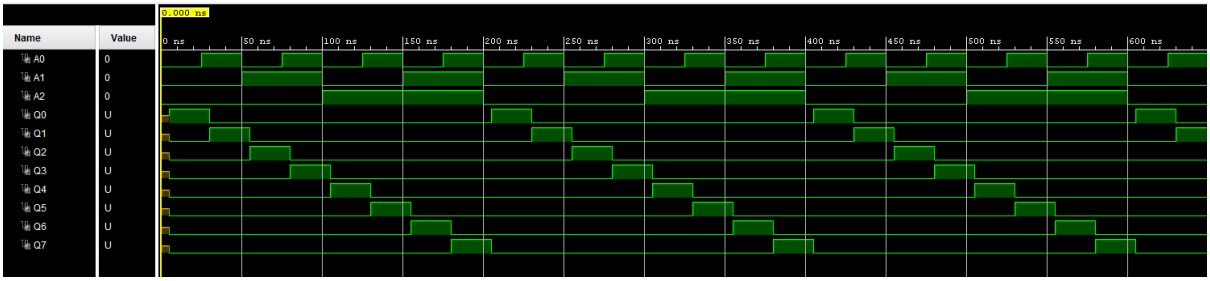
1.15 LOGIC CIRCUIT



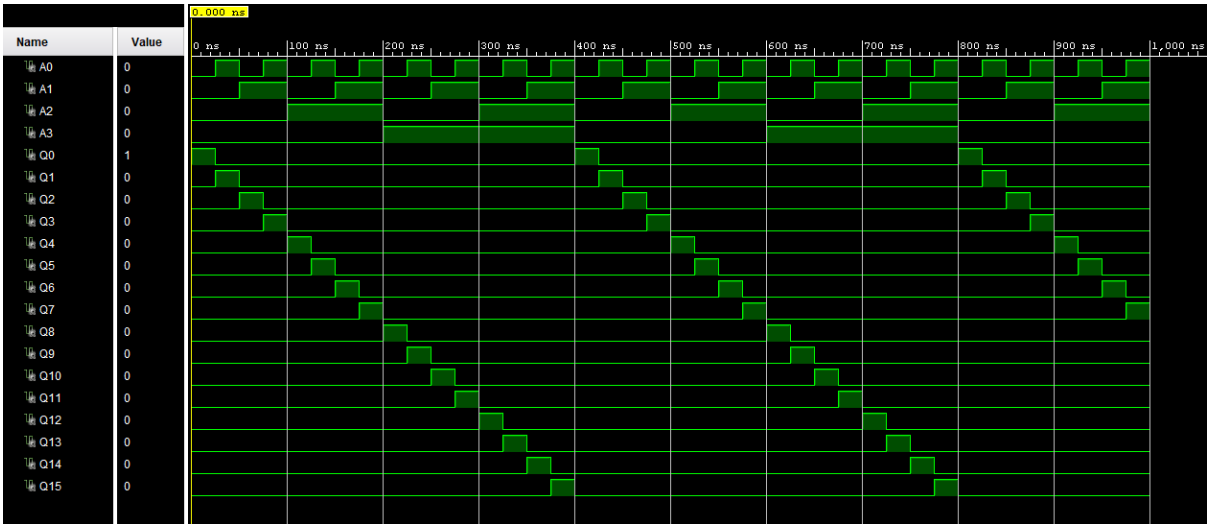
1.16 DATAPATH



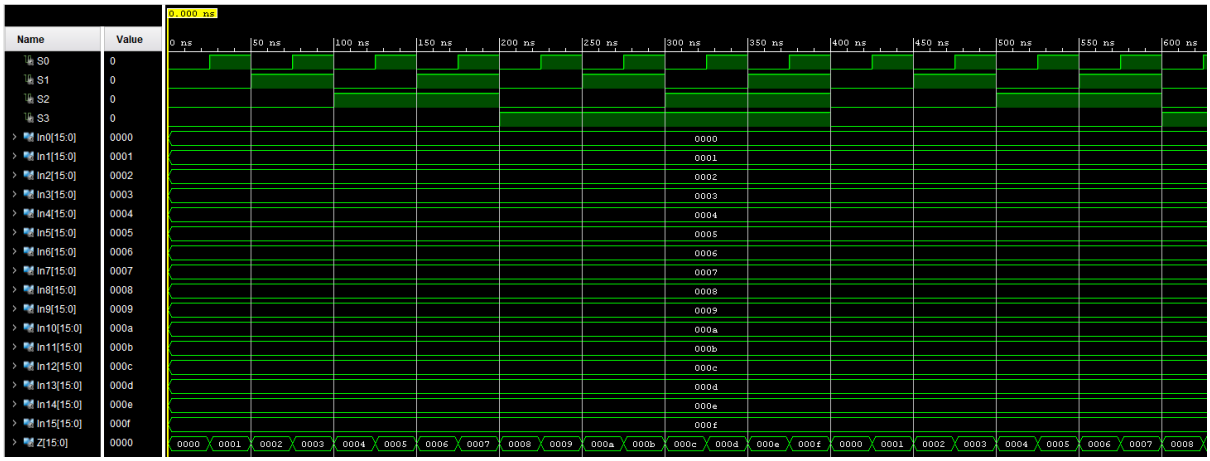
1.17 DECODER_3TO8



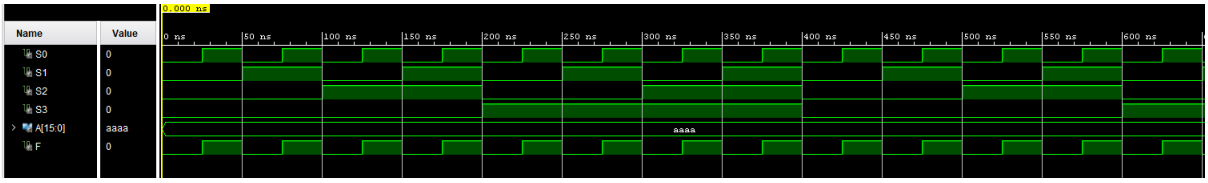
1.18 DECODER_4TO16



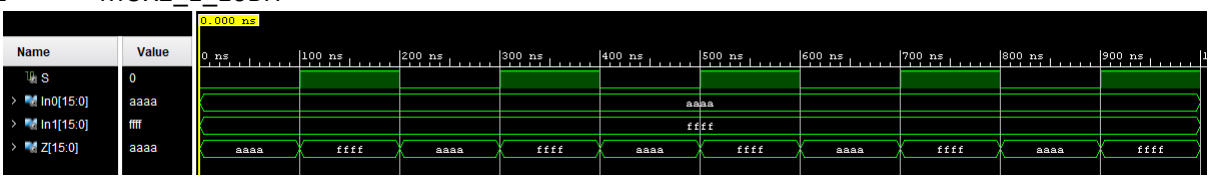
1.19 MUX16_16BIT



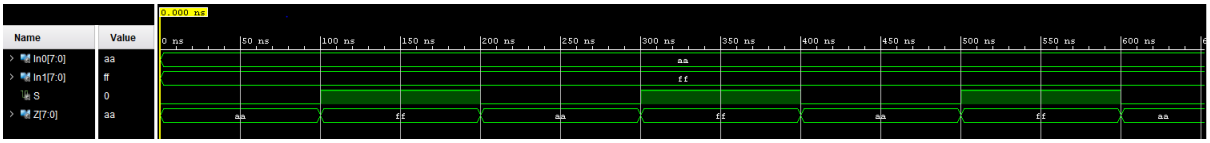
1.20 MUX16_1BIT



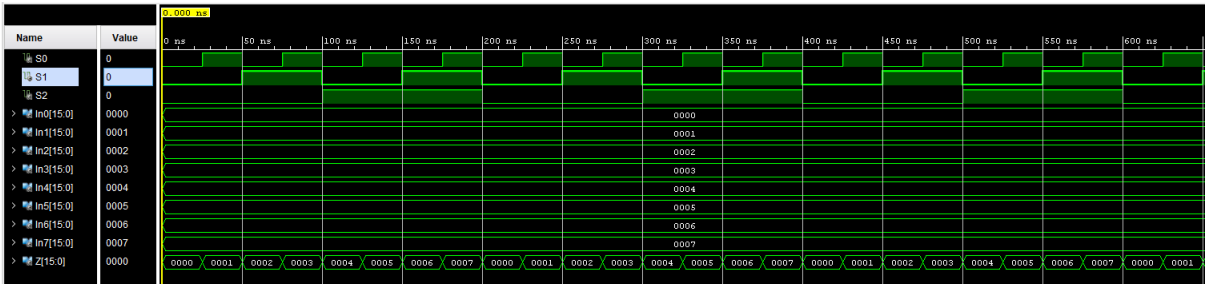
1.21 MUX2_1_16BIT



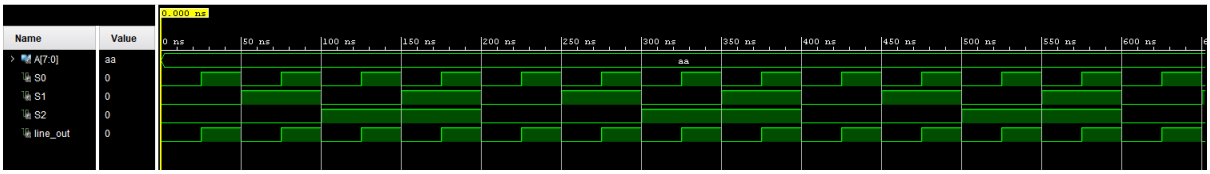
1.22 MUX2_1_8BIT



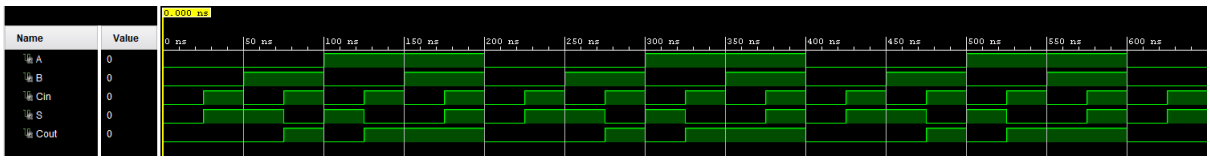
1.23 MUX8_16BIT



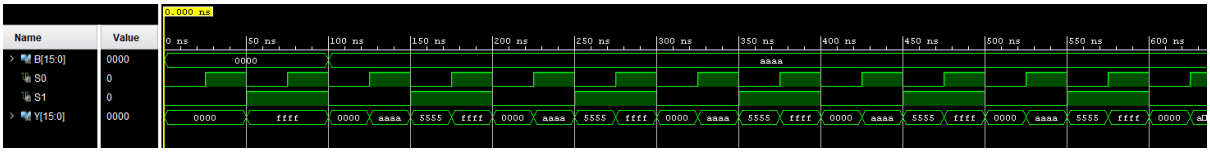
1.24 MUX8_1BIT



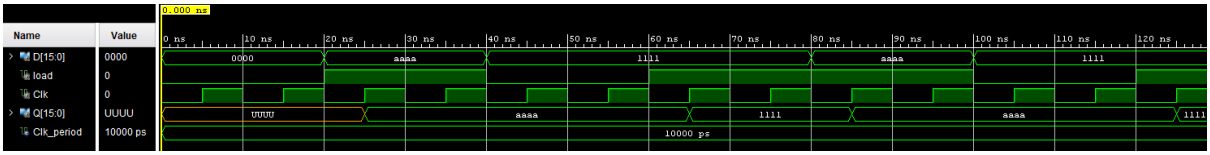
1.25 FULL ADDER



1.26 MUXB_Y_16BIT

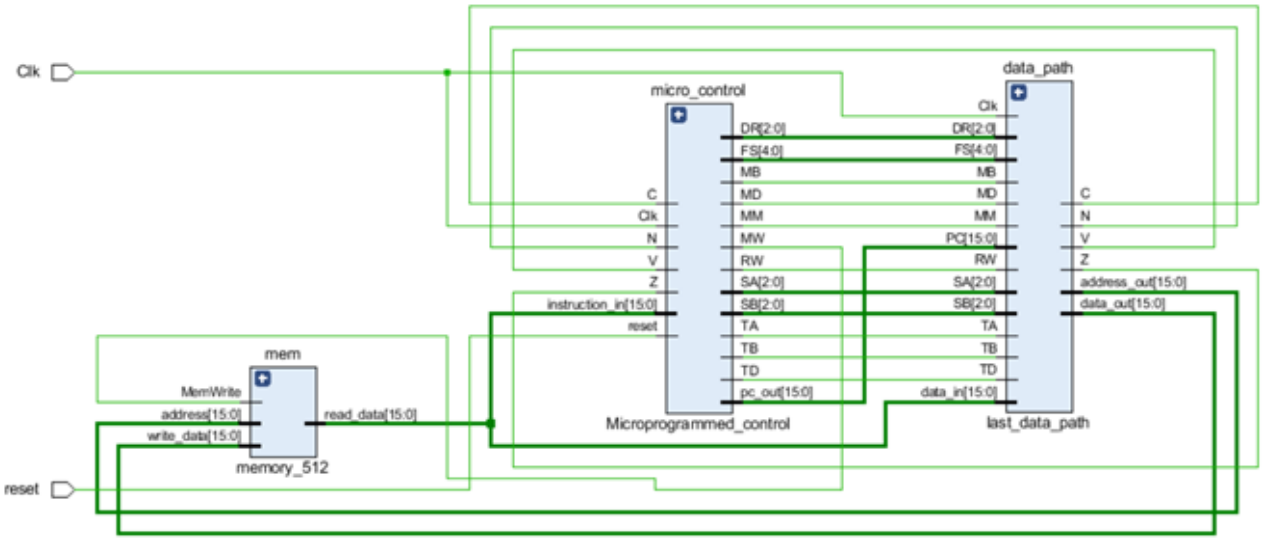


1.27 REGISTER

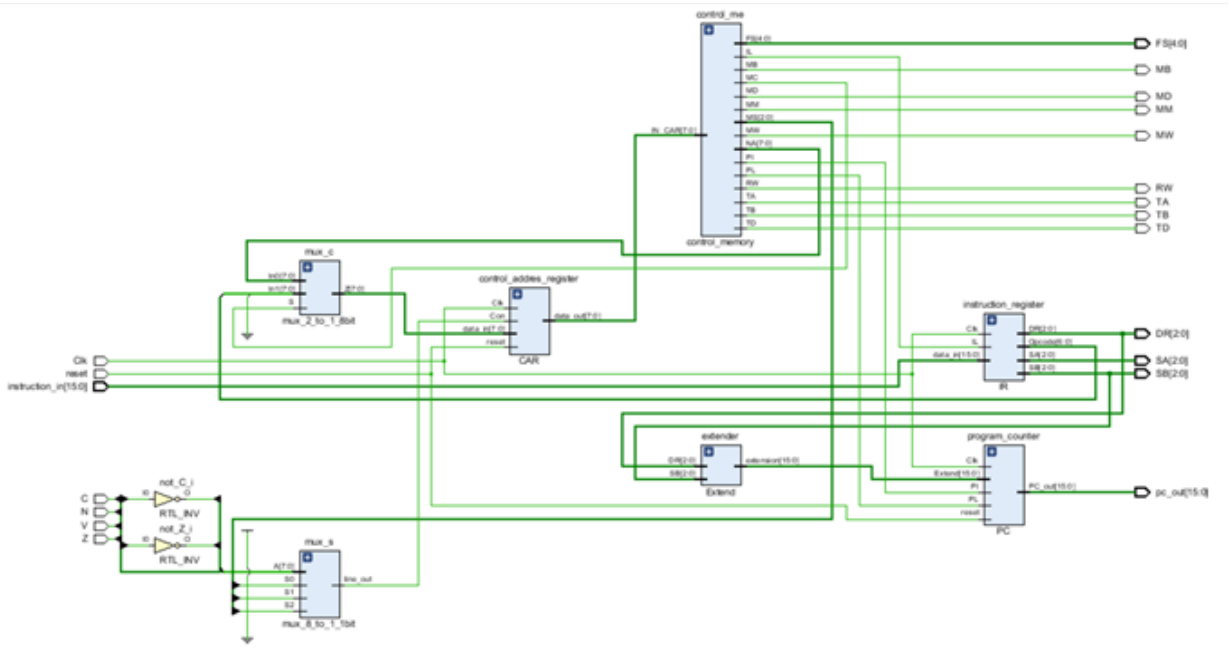


SCHEMATICS

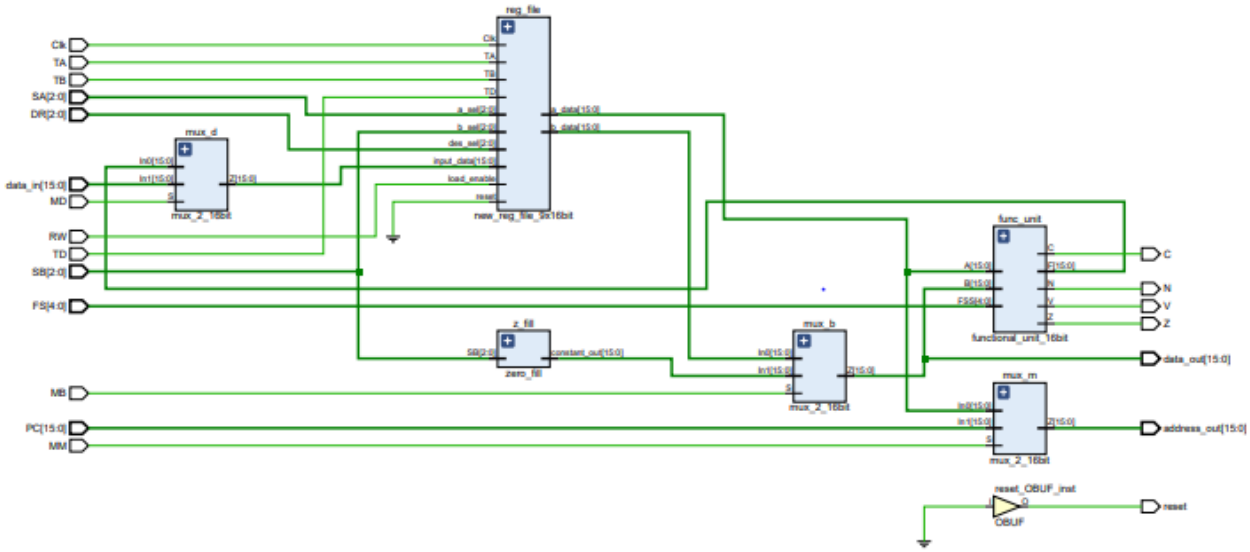
COMPLETE PROJECT



MICROPROGRAMMED CONTROLLER



REGISTER FILE



GENERAL EXPLANATION OF WHAT IS HAPPENING

Setup:

When turned on, the PC (program counter) and CAR (control address register) are both reset.

The PC specifies the address of the next instruction in Memory M, while the CAR specifies the address of the next set of machine code instructions in Control Memory. The PI signal increments the value of the PC, and the PL signal uses DR || SB (concatenated) as the next address.

When the PC is reset it points to the first microprogram in memory. Memory contains the addresses of the instructions that are stored in control memory. For example, instruction 0x002E may correspond to the memory address in Control Memory of the instruction ADI. The address accessed is determined by the PC.

Execution:

This microprogram (instruction) address is loaded into the IR (Instruction Register), where it is split into a 7-bit opcode (the first 7 bits) that corresponds to the instruction to be accessed in Control Memory, and three 3-bit values (the last 9 bits) that can either be used for register selection or as values (for immediate values or PC offset). They decide the Destination Register.

Opcode	DA	SA	SB
7 bit	3 bit	3 bit	3 bit

For example

```
ADD r0, r1, r2 =
```

0000011	000	001	010

```
= 0000011000001010    = 0x060A
```

The opcode is loaded into MUX C, and, as the control memory is in the Instruction Fetch State, it is loaded from MUX C into the CAR and then used to select the next set of machine code in the Control Memory. The CAR determines what the next address the Control Memory uses to access the next instruction is.

The Control Memory accesses the instruction to be accessed (as dictated by CAR) based on the instruction address based in. An instruction could take more than one clock cycle (e.g Load).
The Control Memory controls the inputs to all of the multiplexers, Functional Unit, Register File, PC, IR, CAR and Memory.

Values coming from either the output of the Function unit or Memory (as selected by MUX D) can be loaded into registers in the Register File if RW (Read=0/Write=1) is set to high.

The Register File contains an array of registers which holds data. A temporary register (R8) is also here for multi-cycle instructions so that the data the user may be storing is not disrupted.

The destination register is selected by DR coming from IR, or TD (for the temp register) coming from Control Memory.

The Function Unit takes in values from the Register File (selected by SA, SB, TA or TB) or immediate values and performs arithmetic logic or shifting operations on those values. It contains an ALU & a Shifter. The status bits from the Function Unit are used for conditional instructions in the microprogrammed control.

Values from the Register File can also be stored in Memory M if MW(Memory Write) is set.

The processor is pipelined so that a number of operations can be happening simultaneously, i.e. instruction fetching, execution of a previous instruction and writing the results to the Register File.

2. VHDL Component Source Code

2.1 PROCESSOR FINAL

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

entity processor_final is
        Port(
                        Clk : in STD_LOGIC;
                        reset : in STD_LOGIC
                        );
end processor_final;

architecture Behavioral of processor_final is

COMPONENT Microprogrammed_control is
   Port ( instruction_in : in  STD_LOGIC_VECTOR (15 downto 0);
                        reset : in STD_LOGIC;
        V : in  STD_LOGIC;
        C : in  STD_LOGIC;
        N : in  STD_LOGIC;
        Z : in  STD_LOGIC;
                        Clk : in STD_LOGIC;
        pc_out : out  STD_LOGIC_VECTOR (15 downto 0);
        DR : out  STD_LOGIC_VECTOR (2 downto 0);
        SA : out  STD_LOGIC_VECTOR (2 downto 0);
        SB : out  STD_LOGIC_VECTOR (2 downto 0);
        TD : out  STD_LOGIC;
        TA : out  STD_LOGIC;
        TB : out  STD_LOGIC;
        MB : out  STD_LOGIC;
        FS : out  STD_LOGIC_VECTOR (4 downto 0);
        MD : out  STD_LOGIC;
        RW : out  STD_LOGIC;
        MM : out  STD_LOGIC;
        MW : out  STD_LOGIC);
end COMPONENT;


COMPONENT last_data_path is
   Port ( PC : in  STD_LOGIC_VECTOR (15 downto 0);
                        data_in : in  STD_LOGIC_VECTOR (15 downto 0);
        DR : IN  std_logic_vector(2 downto 0);
        SA : IN  std_logic_vector(2 downto 0);
        SB : IN  std_logic_vector(2 downto 0);
```

```
        TD : IN  std_logic;
        TA : IN  std_logic;
        TB : IN  std_logic;
        Clk : in  STD_LOGIC;
        FS : IN  std_logic_vector(4 downto 0);
        RW : IN  std_logic;
        MB : IN  std_logic;
        MM : IN  std_logic;
        MD : IN  std_logic;
        data_out : out  STD_LOGIC_VECTOR (15 downto 0);
        address_out : out  STD_LOGIC_VECTOR (15 downto 0);
        V : out  STD_LOGIC;
        C : out  STD_LOGIC;
        N : out  STD_LOGIC;
        Z : out  STD_LOGIC);
end COMPONENT;


COMPONENT memory_512
   PORT(
      address : IN  std_logic_vector(15 downto 0);
      write_data : IN  std_logic_vector(15 downto 0);
      MemWrite : IN  std_logic;
      read_data : OUT  std_logic_vector(15 downto 0)
     );
   END COMPONENT;

         --signals
         signal V_signal : std_logic;
         signal C_signal : std_logic;
         signal N_signal : std_logic;
         signal Z_signal : std_logic;
         signal TD_signal : std_logic;
         signal TA_signal : std_logic;
         signal TB_signal : std_logic;
         signal MB_signal : std_logic;
         signal MD_signal : std_logic;
         signal RW_signal : std_logic;
         signal MM_signal : std_logic;
         signal MW_signal : std_logic;
         signal DR_signal : std_logic_vector(2 downto 0);
         signal SA_signal : std_logic_vector(2 downto 0);
         signal SB_signal : std_logic_vector(2 downto 0);
         signal FS_signal : std_logic_vector(4 downto 0);
```

```
        signal datapath_address_out : std_logic_vector(15 downto 0);
        signal datapath_data_out : std_logic_vector(15 downto 0);
        signal memory_data_out : std_logic_vector(15 downto 0);
        signal micro_pc_out : std_logic_vector(15 downto 0);

begin

micro_control : Microprogrammed_control PORT MAP (
        instruction_in => memory_data_out,
        reset => reset,
        V => V_signal,
        C => C_signal,
        N => N_signal,
        Z => Z_signal,
        Clk => Clk,
        pc_out => micro_pc_out,
        DR => DR_signal,
        SA => SA_signal,
        SB => SB_signal,
        TD => TD_signal,
        TA => TA_signal,
        TB => TB_signal,
        MB => MB_signal,
        FS => FS_signal,
        MD => MD_signal,
        RW => RW_signal,
        MM => MM_signal,
        MW => MW_signal
      );


data_path : last_data_path PORT MAP (
        PC => micro_pc_out,
        data_in => memory_data_out,
        DR => DR_signal,
        SA => SA_signal,
        SB => SB_signal,
        TD => TD_signal,
        TA => TA_signal,
        TB => TB_signal,
        Clk => Clk,
        FS => FS_signal,
        RW => RW_signal,
        MB => MB_signal,
```

```
            MM => MM_signal,
            MD => MD_signal,
            data_out => datapath_data_out,
            address_out => datapath_address_out,
            V => V_signal,
            C => C_signal,
            N => N_signal,
            Z => Z_signal
           );


    mem : memory_512 PORT MAP (
            address => datapath_address_out,
            write_data => datapath_data_out,
            MemWrite => MW_signal,
            read_data => memory_data_out
           );


    end Behavioral;
```

2.2 ARITHEMATIC LOGICAL UNIT

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

entity ALU_unit_16bit is
   Port ( A : in  STD_LOGIC_VECTOR (15 downto 0);
          B : in  STD_LOGIC_VECTOR (15 downto 0);
          Cin : in  STD_LOGIC;
          S0 : in  STD_LOGIC;
          S1 : in  STD_LOGIC;
          S2 : in  STD_LOGIC;
          Cout : out  STD_LOGIC;
          Vout : out  STD_LOGIC;
          G : out  STD_LOGIC_VECTOR (15 downto 0));
end ALU_unit_16bit;

architecture Behavioral of ALU_unit_16bit is


   COMPONENT arithmetic_circuit_16bit
   PORT(
      Cin : IN  std_logic;
      S0 : IN  std_logic;
```

```
        S1 : IN  std_logic;
        A : IN  std_logic_vector(15 downto 0);
        B : IN  std_logic_vector(15 downto 0);
        G : OUT  std_logic_vector(15 downto 0);
        Cout : OUT  std_logic;
        Vout : OUT  std_logic
        );
    END COMPONENT;

    COMPONENT logic_circuit_16bit
    PORT(
        A : IN  std_logic_vector(15 downto 0);
        B : IN  std_logic_vector(15 downto 0);
        s0 : IN  std_logic;
        s1 : IN  std_logic;
        Y : OUT  std_logic_vector(15 downto 0)
        );
    END COMPONENT;


    COMPONENT mux_2_16bit
    PORT(
        S : IN  std_logic;
        In0 : IN  std_logic_vector(15 downto 0);
        In1 : IN  std_logic_vector(15 downto 0);
        Z : OUT  std_logic_vector(15 downto 0)
        );
    END COMPONENT;


      -- signals
      signal arith_out : std_logic_vector(15 downto 0);
      signal logic_out : std_logic_vector(15 downto 0);

begin
--portmaps:

--arithmetic circuit
ac: arithmetic_circuit_16bit PORT MAP (
        Cin => Cin,
        S0 => S0,
        S1 => S1,
        A => A,
        B => B,
```

```
        G => arith_out,
        Cout => Cout,
        Vout => Vout
       );

--logical circuit
lc: logic_circuit_16bit PORT MAP (
        A => A,
        B => B,
        s0 => s0,
        s1 => s1,
        Y => logic_out
       );

--2 to 1 multiplexer
  mux: mux_2_16bit PORT MAP (
        S => s2,
        In0 => arith_out,
        In1 => logic_out,
        Z => G
       );

end Behavioral;
```

2.3 CONTROL ADDRESS REGISTER

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


entity CAR is
   Port ( data_in : in  STD_LOGIC_VECTOR (7 downto 0);
                        reset : in STD_LOGIC;
          Con : in  STD_LOGIC;
                        Clk : in STD_LOGIC;
          data_out : out  STD_LOGIC_VECTOR (7 downto 0));
end CAR;
```

```
architecture Behavioral of CAR is

    COMPONENT ripple_adder_16bit
    PORT(
            A : IN std_logic_vector(15 downto 0);
            B : IN std_logic_vector(15 downto 0);
            Cin : IN std_logic;
            S : OUT std_logic_vector(15 downto 0);
            Cout : OUT std_logic;
            Vout : OUT std_logic
            );
    END COMPONENT;

    -- 16 bit Register
    COMPONENT reg16
    PORT(
            D : IN std_logic_vector(15 downto 0);
            load : IN std_logic;
            Clk : IN std_logic;
            Q : OUT std_logic_vector(15 downto 0)
            );
    END COMPONENT;

signal increment : std_logic_vector(7 downto 0);
signal unused : std_logic_vector (7 downto 0);
signal current_val : std_logic_vector (15 downto 0);
signal into_reg : std_logic_vector (15 downto 0);

begin
```

```
    -- ripple_adder_16bit
    adder: ripple_adder_16bit PORT MAP(
            A(7 downto 0) => current_val(7 downto 0),
            A(15 downto 8) => x"00",
            B => x"0001",
            Cin => '0',
            S(7 downto 0) => increment,
            s(15 downto 8) => unused
    );

    -- register
            reg: reg16 PORT MAP(
                    D => into_reg,
                    load => '1',
                    Clk => Clk,
                    Q => current_val
            );

--into_reg(7 downto 0) <= x"C0" when reset='1' else data_in when Con = '1' else
increment;
into_reg(7 downto 0) <= data_in when Con = '1' else increment;
into_reg(15 downto 8) <= x"00";
data_out <= current_val(7 downto 0);

end Behavioral;
```

2.4 EXTEND

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

entity Extend is
   Port ( DR : in  STD_LOGIC_VECTOR (2 downto 0);
          SB : in  STD_LOGIC_VECTOR (2 downto 0);
          extension : out  STD_LOGIC_VECTOR (15 downto 0));
end Extend;

architecture Behavioral of Extend is

begin
      extension(15 downto 6) <= "0000000000" when (DR(2)='0') else "1111111111";
      extension(5 downto 3) <= DR;
      extension(2 downto 0) <= SB;
end Behavioral;
```

2.5 INSTRUCTION REGISTER

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

entity IR is
   Port ( data_in : in  STD_LOGIC_VECTOR (15 downto 0);
          IL : in  STD_LOGIC;
                       Clk : in STD_LOGIC;
          Opcode : out  STD_LOGIC_VECTOR (6 downto 0);
          DR : out  STD_LOGIC_VECTOR (2 downto 0);
          SA : out  STD_LOGIC_VECTOR (2 downto 0);
```

```
        SB : out  STD_LOGIC_VECTOR (2 downto 0));
end IR;

architecture Behavioral of IR is

    -- 16 bit Register
    COMPONENT reg16
    PORT(
            D : IN std_logic_vector(15 downto 0);
            load : IN std_logic;
            Clk : IN std_logic;
            Q : OUT std_logic_vector(15 downto 0)
            );
    END COMPONENT;

signal data_out : std_logic_vector(15 downto 0);

begin

    -- register
    reg: reg16 PORT MAP(
            D => data_in,
            load => IL,
            Clk => Clk,
            Q => data_out
    );

    Opcode<=data_out(15 downto 9);
    DR<=data_out(8 downto 6);
```

```
    SA<=data_out(5 downto 3);
    SB<=data_out(2 downto 0);

end Behavioral;
```

2.6 MICROPROGRAMMED CONTROL

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

entity Microprogrammed_control is
   Port ( instruction_in : in  STD_LOGIC_VECTOR (15 downto 0);
                         reset : in STD_LOGIC;
          V : in  STD_LOGIC;
          C : in  STD_LOGIC;
          N : in  STD_LOGIC;
          Z : in  STD_LOGIC;
                         Clk : in STD_LOGIC;
          pc_out : out  STD_LOGIC_VECTOR (15 downto 0);
          DR : out  STD_LOGIC_VECTOR (2 downto 0);
          SA : out  STD_LOGIC_VECTOR (2 downto 0);
          SB : out  STD_LOGIC_VECTOR (2 downto 0);
          TD : out  STD_LOGIC;
          TA : out  STD_LOGIC;
          TB : out  STD_LOGIC;
          MB : out  STD_LOGIC;
          FS : out  STD_LOGIC_VECTOR (4 downto 0);
          MD : out  STD_LOGIC;
          RW : out  STD_LOGIC;
          MM : out  STD_LOGIC;
```

```
        MW : out  STD_LOGIC);
end Microprogrammed_control;

architecture Behavioral of Microprogrammed_control is

COMPONENT PC
   PORT(
      Extend : IN  std_logic_vector(15 downto 0);
      reset : IN  std_logic;
      PI : IN  std_logic;
      PL : IN  std_logic;
      Clk : IN  std_logic;
      PC_out : OUT  std_logic_vector(15 downto 0)
     );
   END COMPONENT;


COMPONENT mux_8_to_1_1bit
     PORT(
                  A : IN  std_logic_vector(7 downto 0);
                  S0 : IN std_logic;
                  S1 : IN std_logic;
                  S2 : IN std_logic;
                  line_out : OUT std_logic
            );
            END COMPONENT;


COMPONENT Extend
```

```
  PORT(
     DR : IN  std_logic_vector(2 downto 0);
     SB : IN  std_logic_vector(2 downto 0);
     extension : OUT  std_logic_vector(15 downto 0)
    );
  END COMPONENT;


COMPONENT IR
  PORT(
     data_in : IN  std_logic_vector(15 downto 0);
     IL : IN  std_logic;
                  Clk : IN std_logic;
     Opcode : OUT  std_logic_vector(6 downto 0);
     DR : OUT  std_logic_vector(2 downto 0);
     SA : OUT  std_logic_vector(2 downto 0);
     SB : OUT  std_logic_vector(2 downto 0)
    );
  END COMPONENT;


COMPONENT mux_2_to_1_8bit
  PORT(
     S : IN  std_logic;
     In0 : IN  std_logic_vector(7 downto 0);
     In1 : IN  std_logic_vector(7 downto 0);
     Z : OUT  std_logic_vector(7 downto 0)
    );
  END COMPONENT;
```

```
COMPONENT CAR
   PORT(
      data_in : IN  std_logic_vector(7 downto 0);
                    reset : IN std_logic;
                    Clk : IN std_logic;
      Con : IN  std_logic;
      data_out : OUT  std_logic_vector(7 downto 0)
     );
   END COMPONENT;


COMPONENT control_memory
   PORT(
      MW : OUT  std_logic;
      MM : OUT  std_logic;
      RW : OUT  std_logic;
      MD : OUT  std_logic;
      FS : OUT  std_logic_vector(4 downto 0);
      MB : OUT  std_logic;
      TB : OUT  std_logic;
      TA : OUT  std_logic;
      TD : OUT  std_logic;
      PL : OUT  std_logic;
      PI : OUT  std_logic;
      IL : OUT  std_logic;
      MC : OUT  std_logic;
      MS : OUT  std_logic_vector(2 downto 0);
```

```
        NA : OUT  std_logic_vector(7 downto 0);
        IN_CAR : IN  std_logic_vector(7 downto 0)
       );
   END COMPONENT;

                        --signals
  signal MC : std_logic;
  signal IL : std_logic;
  signal PI : std_logic;
  signal PL : std_logic;
  signal not_Z : std_logic;
  signal not_C : std_logic;
  signal mux_s_out : std_logic;
  signal MS : std_logic_vector(2 downto 0);
  signal DR_signal : std_logic_vector(2 downto 0);
  signal SA_signal : std_logic_vector(2 downto 0);
  signal SB_signal : std_logic_vector(2 downto 0);
  signal NA : std_logic_vector(7 downto 0);
  signal mux_c_out : std_logic_vector(7 downto 0);
  signal car_out : std_logic_vector(7 downto 0);
  signal extend_out : std_logic_vector(15 downto 0);
  signal ir_opcode_out : std_logic_vector(6 downto 0);

begin

extender : Extend PORT MAP (
        DR => DR_signal,
        SB => SB_signal,
        extension => extend_out
```

```
        );



program_counter : PC PORT MAP (
        Extend => extend_out,
                        reset => reset,
                        Clk => Clk,
        PL => PL,
        PI => PI,
        PC_out => pc_out
       );

mux_s : mux_8_to_1_1bit PORT MAP (
        A(0) => '0',
        A(1) => '1',
        A(2) => C,
        A(3) => V,
        A(4) => Z,
        A(5) => N,
        A(6) => not_C,
        A(7) => not_Z,
        S0 => MS(0),
        S1 => MS(1),
        S2 => MS(2),
        line_out => mux_s_out
       );

instruction_register : IR PORT MAP (
        data_in => instruction_in,
```

```
        IL => IL,
                        Clk => Clk,
        Opcode => ir_opcode_out,
        DR => DR_signal,
        SA => SA_signal,
        SB => SB_signal
      );

mux_c : mux_2_to_1_8bit PORT MAP (
        In0 => NA,
                        In1(7) => '0',
        In1(6 downto 0) => ir_opcode_out,
        S => MC,
        Z => mux_c_out
      );


control_addres_register : CAR PORT MAP (
        data_in => mux_c_out,
                        reset => reset,
                        Clk => Clk,
        Con => mux_s_out,
        data_out => car_out
      );

control_me : control_memory PORT MAP (
        MW => MW,
        MM => MM,
        RW => RW,
```

```
        MD => MD,
        FS => FS,
        MB => MB,
        TB => TB,
        TA => TA,
        TD => TD,
        PL => PL,
        PI => PI,
        IL => IL,
        MC => MC,
        MS => MS,
        NA => NA,
        IN_CAR => car_out
       );

DR <= DR_signal;
SB <= SB_signal;
SA <= SA_signal;
not_C <= not C;
not_Z <= not Z;
end Behavioral;
```

2.7 PROGRAM COUNTER

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

entity PC is
    Port ( Extend : in  STD_LOGIC_VECTOR (15 downto 0);
                          reset : in STD_LOGIC;
```

```
                        Clk : in STD_LOGIC;
        PL : in  STD_LOGIC;
        PI : in  STD_LOGIC;
        PC_out : out  STD_LOGIC_VECTOR (15 downto 0));
end PC;

architecture Behavioral of PC is

    COMPONENT ripple_adder_16bit
    PORT(
            A : IN std_logic_vector(15 downto 0);
            B : IN std_logic_vector(15 downto 0);
            Cin : IN std_logic;
            S : OUT std_logic_vector(15 downto 0);
            Cout : OUT std_logic;
            Vout : OUT std_logic
            );
    END COMPONENT;

    -- 16 bit Register
    COMPONENT reg16
    PORT(
            D : IN std_logic_vector(15 downto 0);
            load : IN std_logic;
            Clk : IN std_logic;
            Q : OUT std_logic_vector(15 downto 0)
            );
    END COMPONENT;
```

```
signal pi_or_pl_or_reset : std_logic;
signal load_result : std_logic_vector(15 downto 0);
signal increment_result : std_logic_vector(15 downto 0);
signal next_PC : std_logic_vector (15 downto 0);
signal last_PC : std_logic_vector (15 downto 0);

begin

-- ripple_adder_16bit
adderLoad: ripple_adder_16bit PORT MAP(
    A => last_pc,
    B(15 downto 0) => Extend,
    Cin => '0',
    S => load_result
);

-- ripple_adder_16bit
adderIncrement: ripple_adder_16bit PORT MAP(
    A => last_PC,
    B => x"0001",
    Cin => '0',
    S => increment_result
);

-- register
reg: reg16 PORT MAP(
    D => next_PC,
    load => pi_or_pl_or_reset,
```

```
    Clk => Clk,
    Q => last_PC
);

    next_PC <= x"0000" when reset='1' else increment_result when PI='1' else
                                          load_result when PL='1' else x"0000";
    PC_out <= last_PC;
    pi_or_pl_or_reset <= (PI or PL or reset);

end Behavioral;
```

2.8 BARREL SHIFTER

```
--THIS BARRELL SHIFTER HAS BEEN ALTERED TO ONLY SHIFT ZERO BITS, 1 BIT LEFT OR
1 BIT RIGHT

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity barrel_shifter_16bit is
   Port ( B : in  STD_LOGIC_VECTOR (15 downto 0);
          S0 : in  STD_LOGIC;
          S1 : in  STD_LOGIC;
          S2 : in  STD_LOGIC;
          S3 : in  STD_LOGIC;
          F : out  STD_LOGIC_VECTOR (15 downto 0));
end barrel_shifter_16bit;

architecture Behavioral of barrel_shifter_16bit is
          -- 16 to 1 line multiplexer
          COMPONENT mux16_1bit
          PORT(
                    S0 : IN std_logic;
                    S1 : IN std_logic;
                    S2 : IN std_logic;
                    S3 : IN std_logic;
                    A : IN std_logic_vector(15 downto 0);
                    F : OUT std_logic
                    );
```

```
        END COMPONENT;

        signal fs_shift : std_logic_vector(3 downto 0);

begin
-- port maps ;-)

-- mux 0
mux : for i in 0 to 15 generate

mux0to16: mux16_1bit PORT MAP(
        S0 => fs_shift(0),
        S1 => fs_shift(1),
        S2 => fs_shift(2),
        S3 => fs_shift(3),
        A(0) => B(i+0),
        A(1) => B((i+15) mod 16),
        A(2) => B((i+14) mod 16),
        A(3) => B((i+13) mod 16),
        A(4) => B((i+12) mod 16),
        A(5) => B((i+11) mod 16),
        A(6) => B((i+10) mod 16),
        A(7) => B((i+9) mod 16),
        A(8) => B((i+8) mod 16),
        A(9) => B((i+7) mod 16),
        A(10) => B((i+6) mod 16),
        A(11) => B((i+5) mod 16),
        A(12) => B((i+4) mod 16),
        A(13) => B((i+3) mod 16),
        A(14) => B((i+2) mod 16),
        A(15) => B((i+1) mod 16),
        F => F(i)
);
end generate;

fs_shift <= "0000" when (S0='0' and S1='0') else "1111" when (S0='0' and S1='1') else
"0001" when S0='1';

end Behavioral;
```

2.9 CONTROL MEMORY

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;
```

```
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity control_memory is
Port (
        MW : out std_logic;
        MM : out std_logic;
        RW : out std_logic;
        MD : out std_logic;
        FS : out std_logic_vector(4 downto 0);
        MB : out std_logic;
        TB : out std_logic;
        TA : out std_logic;
        TD : out std_logic;
        PL : out std_logic;
        PI : out std_logic;
        IL : out std_logic;
        MC : out std_logic;
        MS : out std_logic_vector(2 downto 0);
        NA : out std_logic_vector(7 downto 0);
        IN_CAR : in std_logic_vector(7 downto 0));
end control_memory;

architecture Behavioral of control_memory is
type mem_array is array(0 to 255) of std_logic_vector(27 downto 0);

----------------------------------------------------------------------------------------------

--Instruction       Address       Next Address . MS          MC.IL  PI      PL
    TD.TA  TB      MB     . FS .MD       RW     MM     MW          Code
```

```
--          ADI                 00          IF -> C0            001     0       0
    0       0       0       0       0       1       00010   0       1       0       0       =
    0xC020224

--          LD                  01          IF -> C0            001     0
    0       0       0       0       0       0       0       00000   1       1       0       0
    =       0xC02000C

--          ST                  02          IF -> C0            001     0
    0       0       0       0       0       0       0       00000   0       0       0       1
    =       0xC020001

--          INC                 03          IF -> C0            001     0       0
    0       0       0       0       0       0       00001   0       1       0       0       =
    0xC020014

--          NOT                 04          IF -> C0            001     0       0
    0       0       0       0       0       0       01110   0       1       0       0       =
    0xC0200E4

--          ADD                 05          IF -> C0            001     0       0
    0       0       0       0       0       0       00010   0       1       0       0       =
    0xC020024


--          LRI                 06          LRI2-> 86           001     0       0
    0       0       0       0       0       0       00000   1       1       0       0       =
    0x862000C

--          SR                  07          SR2 -> 87           111     0
    0       0       0       1       0       0       0       00000   0       1       0       0
    =       0x87E1004


--          BEQ                 09          B  -> 30            100     0       0
    0       1  1  1  1       0  10000       0       1       0       0       =       0x0081D04

--          Catch               0A          IF -> C0            001     0       0
    0       0       0       0       0       0       00000   0       0       0       0       =
    0xC020000


--          BNZ                 0B          B  -> 30            111     0       0
    0       0       0       0       0       0       00000   0       0       0       0       =
    0x30E0000
```

-- Catch 0C IF -> C0 001 0 0
0 0 0 0 0 0 00000 0 0 0 0 =
0xC020000

-- CMP 0D IF -> C0 001 0 0
0 0 1 0 0 0 00101 0 1 0 0 =
0xC021054

-- LR 29 IF -> C0 001 0 0
0 0 0 0 0 0 00000 1 1 0 0 =
0xC02000C

-- B 30 IF -> C0 001 0
0 0 1 0 0 0 0 00000 0 0 0 0
= 0xC022000

-- LRI2 86 IF -> C0 001 0 0
0 0 0 1 0 0 00000 1 1 0 0 =
0xC02080C

-- SR2 87 SR3 -> 89 001 0 0
0 0 0 0 0 0 10100 0 1 0 0 =
0x8820144

-- SR3 88 SR2 -> 87 100 0 0
0 0 1 1 0 0 00110 0 1 0 0 =
0x8781864

-- Catch 89 IF -> C0 001 0 0
0 0 0 0 0 0 00000 0 0 0 0 =
0xC020000

-- IF C0 EXO -> C1 000 0
1 1 0 0 0 0 0 00000 0 0 1 0
= 0xC10C002

-- EXO C1 -- -> 00 001 1
0 0 0 0 0 0 0 00000 0 0 0 0
= 0x0030000

```
begin

memory_m: process(IN_CAR)
variable control_mem : mem_array:=(

X"C020224", -- 0x00        ADI:    Add immediate constant
X"C02000C", -- 0x01        LD:     Load
X"C020001", -- 0x02        ST:     Store
X"C020014", -- 0x03        INC:    Increment
X"C0200E4", -- 0x04        NOT:    Not
X"C020024", -- 0x05        ADD:    Add
X"862000C", -- 0x06        LRI:    Load into immediate register Micro operation 1
X"87E1004", -- 0x07        SR:     Shift Right
    Micro-operation 1
X"C020000", -- 0x08        Catch (to fall through to if shifted right by zero)
X"3080000", -- 0x09        BEQ:  Branch if equal
X"C020000", -- 0x0A        Catch (to fall through to if branch not taken)
X"30E0000", -- 0x0B        BNZ:    Branch if not zero
X"C020000", -- 0x0C        Catch (to fall through to if branch not taken)
X"C021054", -- 0x0D        CMP:  Compare
X"0000000", -- 0x0E
X"0000000", -- 0x0F

X"0000000", -- 0x10
X"0000000", -- 0x11
X"0000000", -- 0x12
X"0000000", -- 0x13
X"0000000", -- 0x14
X"0000000", -- 0x15
X"0000000", -- 0x16
```

```
X"0000000", -- 0x17
X"0000000", -- 0x18
X"0000000", -- 0x19
X"0000000", -- 0x1A
X"0000000", -- 0x1B
X"0000000", -- 0x1C
X"0000000", -- 0x1D
X"0000000", -- 0x1E
X"0000000", -- 0x1F

X"0000000", -- 0x20
X"0000000", -- 0x21
X"0000000", -- 0x22
X"0000000", -- 0x23
X"0000000", -- 0x24
X"0000000", -- 0x25
X"0000000", -- 0x26
X"0000000", -- 0x27
X"0000000", -- 0x28
X"C02000C", -- 0x29        LR: Load
X"0000000", -- 0x2A
X"0000000", -- 0x2B
X"0000000", -- 0x2C
X"0000000", -- 0x2D
X"0000000", -- 0x2E
X"0000000", -- 0x2F

X"C022000", -- 0x30        B: Unconditional Branch
X"0000000", -- 0x31
```

X"0000000", -- 0x32

X"0000000", -- 0x33

X"0000000", -- 0x34

X"0000000", -- 0x35

X"0000000", -- 0x36

X"0000000", -- 0x37

X"0000000", -- 0x38

X"0000000", -- 0x39

X"0000000", -- 0x3A

X"0000000", -- 0x3B

X"0000000", -- 0x3C

X"0000000", -- 0x3D

X"0000000", -- 0x3E

X"0000000", -- 0x3F

X"0000000", -- 0x40

X"0000000", -- 0x41

X"0000000", -- 0x42

X"0000000", -- 0x43

X"0000000", -- 0x44

X"0000000", -- 0x45

X"0000000", -- 0x46

X"0000000", -- 0x47

X"0000000", -- 0x48

X"0000000", -- 0x49

X"0000000", -- 0x4A

X"0000000", -- 0x4B

X"0000000", -- 0x4C

X"0000000", -- 0x4D

X"0000000", -- 0x4E

X"0000000", -- 0x4F

X"0000000", -- 0x50

X"0000000", -- 0x51

X"0000000", -- 0x52

X"0000000", -- 0x53

X"0000000", -- 0x54

X"0000000", -- 0x55

X"0000000", -- 0x56

X"0000000", -- 0x57

X"0000000", -- 0x58

X"0000000", -- 0x59

X"0000000", -- 0x5A

X"0000000", -- 0x5B

X"0000000", -- 0x5C

X"0000000", -- 0x5D

X"0000000", -- 0x5E

X"0000000", -- 0x5F

X"0000000", -- 0x60

X"0000000", -- 0x61

X"0000000", -- 0x62

X"0000000", -- 0x63

X"0000000", -- 0x64

X"0000000", -- 0x65

X"0000000", -- 0x66

X"0000000", -- 0x67

X"0000000", -- 0x68

X"0000000", -- 0x69

X"0000000", -- 0x6A

X"0000000", -- 0x6B

X"0000000", -- 0x6C

X"0000000", -- 0x6D

X"0000000", -- 0x6E

X"0000000", -- 0x6F

X"0000000", -- 0x70

X"0000000", -- 0x71

X"0000000", -- 0x72

X"0000000", -- 0x73

X"0000000", -- 0x74

X"0000000", -- 0x75

X"0000000", -- 0x76

X"0000000", -- 0x77

X"0000000", -- 0x78

X"0000000", -- 0x79

X"0000000", -- 0x7A

X"0000000", -- 0x7B

X"0000000", -- 0x7C

X"0000000", -- 0x7D

X"0000000", -- 0x7E

X"0000000", -- 0x7F

X"0000000", -- 0x80

X"0000000", -- 0x81

X"0000000", -- 0x82

X"0000000", -- 0x83

```
X"0000000", -- 0x84
X"0000000", -- 0x85
X"C02080C", -- 0x86     LRI2:    Load Immediate Register Micro-operation 2
X"8820144", -- 0x87     SRM2: Shift Register   Micro-operation 2
X"8781864", -- 0x88     SRM3: Shift Register   Micro-operation 3
X"C020000", -- 0x89     Catch (to fall through to if finished shifting right)
X"0000000", -- 0x8A
X"0000000", -- 0x8B
X"0000000", -- 0x8C
X"0000000", -- 0x8D
X"0000000", -- 0x8E
X"0000000", -- 0x8F

X"0000000", -- 0x90
X"0000000", -- 0x91
X"0000000", -- 0x92
X"0000000", -- 0x93
X"0000000", -- 0x94
X"0000000", -- 0x95
X"0000000", -- 0x96
X"0000000", -- 0x97
X"0000000", -- 0x98
X"0000000", -- 0x99
X"0000000", -- 0x9A
X"0000000", -- 0x9B
X"0000000", -- 0x9C
X"0000000", -- 0x9D
X"0000000", -- 0x9E
X"0000000", -- 0x9F
```

X"0000000", -- 0xA0

X"0000000", -- 0xA1

X"0000000", -- 0xA2

X"0000000", -- 0xA3

X"0000000", -- 0xA4

X"0000000", -- 0xA5

X"0000000", -- 0xA6

X"0000000", -- 0xA7

X"0000000", -- 0xA8

X"0000000", -- 0xA9

X"0000000", -- 0xAA

X"0000000", -- 0xAB

X"0000000", -- 0xAC

X"0000000", -- 0xAD

X"0000000", -- 0xAE

X"0000000", -- 0xAF

X"0000000", -- 0xB0

X"0000000", -- 0xB1

X"0000000", -- 0xB2

X"0000000", -- 0xB3

X"0000000", -- 0xB4

X"0000000", -- 0xB5

X"0000000", -- 0xB6

X"0000000", -- 0xB7

X"0000000", -- 0xB8

X"0000000", -- 0xB9

X"0000000", -- 0xBA

```
X"0000000", -- 0xBB
X"0000000", -- 0xBC
X"0000000", -- 0xBD
X"0000000", -- 0xBE
X"0000000", -- 0xBF

X"C10C002", -- 0xC0      Instruction fetch
X"0030000", -- 0xC1      EXO
X"0000000", -- 0xC2
X"0000000", -- 0xC3
X"0000000", -- 0xC4
X"0000000", -- 0xC5
X"0000000", -- 0xC6
X"0000000", -- 0xC7
X"0000000", -- 0xC8
X"0000000", -- 0xC9
X"0000000", -- 0xCA
X"0000000", -- 0xCB
X"0000000", -- 0xCC
X"0000000", -- 0xCD
X"0000000", -- 0xCE
X"0000000", -- 0xCF

X"0000000", -- 0xD0
X"0000000", -- 0xD1
X"0000000", -- 0xD2
X"0000000", -- 0xD3
X"0000000", -- 0xD4
X"0000000", -- 0xD5
```

X"0000000", -- 0xD6

X"0000000", -- 0xD7

X"0000000", -- 0xD8

X"0000000", -- 0xD9

X"0000000", -- 0xDA

X"0000000", -- 0xDB

X"0000000", -- 0xDC

X"0000000", -- 0xDD

X"0000000", -- 0xDE

X"0000000", -- 0xDF

X"0000000", -- 0xE0

X"0000000", -- 0xE1

X"0000000", -- 0xE2

X"0000000", -- 0xE3

X"0000000", -- 0xE4

X"0000000", -- 0xE5

X"0000000", -- 0xE6

X"0000000", -- 0xE7

X"0000000", -- 0xE8

X"0000000", -- 0xE9

X"0000000", -- 0xEA

X"0000000", -- 0xEB

X"0000000", -- 0xEC

X"0000000", -- 0xED

X"0000000", -- 0xEE

X"0000000", -- 0xEF

X"0000000", -- 0xF0

```
X"0000000", -- 0xF1
X"0000000", -- 0xF2
X"0000000", -- 0xF3
X"0000000", -- 0xF4
X"0000000", -- 0xF5
X"0000000", -- 0xF6
X"0000000", -- 0xF7
X"0000000", -- 0xF8
X"0000000", -- 0xF9
X"0000000", -- 0xFA
X"0000000", -- 0xFB
X"0000000", -- 0xFC
X"0000000", -- 0xFD
X"0000000", -- 0xFE
X"0000000"  -- 0xFF
);
variable addr : integer;
variable control_out : std_logic_vector(27 downto 0);

begin
    addr := conv_integer(IN_CAR);
    control_out := control_mem(addr);
    MW <= control_out(0);
    MM <= control_out(1);
    RW <= control_out(2);
    MD <= control_out(3);
    FS <= control_out(8 downto 4);
    MB <= control_out(9);
    TB <= control_out(10);
```

```
    TA <= control_out(11);
    TD <= control_out(12);
    PL <= control_out(13);
    PI <= control_out(14);
    IL <= control_out(15);
    MC <= control_out(16);
    MS <= control_out(19 downto 17);
    NA <= control_out(27 downto 20);
    end process;
end Behavioral;
```

2.10 ARITHMETIC CIRCUIT

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

    entity arithmetic_circuit_16bit is
       Port ( Cin : in  STD_LOGIC;
            S0 : in  STD_LOGIC;
            S1 : in  STD_LOGIC;
            A : in  STD_LOGIC_VECTOR (15 downto 0);
            B : in  STD_LOGIC_VECTOR (15 downto 0);
            G : out  STD_LOGIC_VECTOR (15 downto 0);
            Cout : out  STD_LOGIC;
            Vout : out  STD_LOGIC);
    end arithmetic_circuit_16bit;


    architecture Behavioral of arithmetic_circuit_16bit is
    -- components
            COMPONENT ripple_adder_16bit
            PORT(
                    A : IN std_logic_vector(15 downto 0);
                    B : IN std_logic_vector(15 downto 0);
                    Cin : IN std_logic;
                    S : OUT std_logic_vector(15 downto 0);
                    Cout : OUT std_logic;
                    Vout : OUT std_logic
                    );
```

```
        END COMPONENT;


        COMPONENT b_to_y_16bit
        PORT(
                B : IN std_logic_vector(15 downto 0);
                S0 : IN std_logic;
                S1 : IN std_logic;
                Y : OUT std_logic_vector(15 downto 0)
                );
        END COMPONENT;

    --signals
    signal from_logic_to_adder : std_logic_vector(15 downto 0);

    begin
    -- port maps ;-)

    -- ripple_adder_16bit
    adder: ripple_adder_16bit PORT MAP(
            A => A,
            B => from_logic_to_adder,
            Cin => Cin,
            S => G,
            Cout => Cout,
            Vout => Vout
    );

    -- b_to_y_16bit
    bitslice_logic: b_to_y_16bit PORT MAP(
            B => B,
            S0 => S0,
            S1 => S1,
            Y => from_logic_to_adder
    );

    end Behavioral;
```

2.11 FUNCTIONAL UNIT

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;
```

```
entity functional_unit_16bit is
    Port ( FSS : in  STD_LOGIC_VECTOR (4 downto 0);
           A : in  STD_LOGIC_VECTOR (15 downto 0);
           B : in  STD_LOGIC_VECTOR (15 downto 0);
           F : out  STD_LOGIC_VECTOR (15 downto 0);
           V : out  STD_LOGIC;
           C : out  STD_LOGIC;
           N : out  STD_LOGIC;
           Z : out  STD_LOGIC);
end functional_unit_16bit;

architecture Behavioral of functional_unit_16bit is


    COMPONENT ALU_unit_16bit
    PORT(
        A : IN  std_logic_vector(15 downto 0);
        B : IN  std_logic_vector(15 downto 0);
        Cin : IN  std_logic;
        S0 : IN  std_logic;
        S1 : IN  std_logic;
        S2 : IN  std_logic;
        Cout : OUT  std_logic;
        Vout : OUT  std_logic;
        G : OUT  std_logic_vector(15 downto 0)
       );
    END COMPONENT;

    COMPONENT barrel_shifter_16bit
```

```
PORT(
    B : IN  std_logic_vector(15 downto 0);
    S0 : IN  std_logic;
    S1 : IN  std_logic;
    S2 : IN  std_logic;
    S3 : IN  std_logic;
    F : OUT  std_logic_vector(15 downto 0)
   );
END COMPONENT;


COMPONENT mux_2_16bit
PORT(
    S : IN  std_logic;
    In0 : IN  std_logic_vector(15 downto 0);
    In1 : IN  std_logic_vector(15 downto 0);
    Z : OUT  std_logic_vector(15 downto 0)
   );
END COMPONENT;


--Inputs
signal S0 : std_logic := '0';
signal S1 : std_logic := '0';
signal S2 : std_logic := '0';
signal S3 : std_logic := '0';
   signal In0 : std_logic_vector(15 downto 0) := (others => '0');
   signal In1 : std_logic_vector(15 downto 0) := (others => '0');
```

```
      --Outputs
   signal Cout : std_logic;
   signal Vout : std_logic;
   signal alu_out : std_logic_vector(15 downto 0);
   signal shift_out : std_logic_vector(15 downto 0);

begin
--ALU
   aluu: ALU_unit_16bit PORT MAP (
          A => A,
          B => B,
          Cin => FSS(0),--carry only used for some operations determined by FS
          S0 => FSS(1),
          S1 => FSS(2),
          S2 => FSS(3),
          Cout => C,
          Vout => V,
          G => alu_out
        );
--shifter
  shifter: barrel_shifter_16bit PORT MAP (
          B => B,
          S0 => FSS(3),
          S1 => FSS(2),
          S2 => FSS(1),
          S3 => FSS(0),
          F => shift_out
        );
--mux
```

```
 mux: mux_2_16bit PORT MAP (
        S => FSS(4),
        In0 => alu_out,
        In1 => shift_out,
        Z => F
       );

Z <= '1' when (alu_out=x"0000") else '0';
N <= alu_out(15);
end Behavioral;
```

2.12 MEMORY

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

entity memory_512 is
   Port ( address : in  STD_LOGIC_VECTOR (15 downto 0);
          write_data : in  STD_LOGIC_VECTOR (15 downto 0);
                       --Clk : in STD_LOGIC;
          MemWrite : in  STD_LOGIC;
          read_data : out  STD_LOGIC_VECTOR (15 downto 0));
end memory_512;

architecture Behavioral of memory_512 is

type mem_array is array(0 to 511) of std_logic_vector(15 downto 0);
-- define type, for memory arrays
signal snl : unsigned(15 downto 0);
```

```
signal clk11 : std_logic;

begin

snl <= unsigned(address);

mem_process: process (address, write_data)

-- initialize data memory, X denotes hexadecimal number

variable data_mem : mem_array := (


--                              PC                                      Instructions
                        Pseudocode
    Codewords                                   Hex

--

                                                Opcode  DR  SA  SB


X"00DB", --0000                                 ADI R3, R3, #3
    R3 = (0 + 3)  = 3                                   0000000 011 011 011
            00DB

X"0ADB", --0001                                 ADD R3, R3, R3
            R3 = (3 + 3)  = 6                                       0000101 011 011
011                 0ADB

X"0121",    --0010                              ADI R4, R4, #1
    R4 = (0 + 1)  = 1                                   0000000 100 100 001
    0121

X"0F24",    --0011                              SR      R4, R4, R4
            R4 = (1 >> 1) = 0x8000                      0000111 100 100 100
    0F24

X"0ADB", --0100                                 ADD R3, R3, R3
            R3 = (6 + 6)  = 12                          0000101 011 011 011
    0ADB

X"03D8", --0101                                 LD  R7, Memory[R3]                  R7
= Memory[12] = 0x00FF                       0000001 111 011 000         03D8

X"08D8", --0110                                 NOT R3, R3
            R3 = (~12)      = 0xFFF3                    0000100 011 011 000
    08D8
```

X"06D8", --0111 while INC R3, R3
4x(R3 = R3 + 1) = 0 0000011 011 011 000
06D8

X"17C6", --1000 BNZ while

0001011 111 000 110 17C6

X"3A00", --1001 stop B stop

0011101 000 000 000 3A00

X"0000", -- State now
= [R3==0 R4==0x8000 R7==0xFF]

X"0000", --

X"00FF", --(Memory[12])

X"0000", --

X"0000", --

X"0000", --

X"0000", X"0000", X"0000",X"0000", --x0010 - x001F

X"0000", X"0000", X"0000",X"0000",

X"0000", X"0000", X"0000",X"0000",

X"0000", X"0000", X"0000",X"0000",

X"0000", X"0000", X"0000",X"0000", --x0020 - x002F

X"0000", X"0000", X"0000",X"0000",

X"0000", X"0000", X"0000",X"0000",

X"0000", X"0000", X"0000",X"0000",

X"0000", X"0000", X"0000",X"0000", --x0030 - x003F

X"0000", X"0000", X"0000",X"0000",

X"0000", X"0000",X"0000",X"0000",

X"0000", X"0000", X"0000",X"0000",

```
X"0000", X"0000", X"0000",X"0000",          --x0040 - x004F
X"0000", X"0000", X"0000",X"0000",
 X"0000", X"0000",X"0000",X"0000",
X"0000", X"0000", X"0000",X"0000",

X"0000", X"0000", X"0000",X"0000",          --x0050 - x005F
X"0000", X"0000", X"0000",X"0000",
 X"0000", X"0000",X"0000",X"0000",
X"0000", X"0000", X"0000",X"0000",

X"0000", X"0000", X"0000",X"0000",          --x0060 - x006F
X"0000", X"0000", X"0000",X"0000",
 X"0000", X"0000",X"0000",X"0000",
X"0000", X"0000", X"0000",X"0000",

X"0000", X"0000", X"0000",X"0000",          --x0070 - x007F
X"0000", X"0000", X"0000",X"0000",
 X"0000", X"0000",X"0000",X"0000",
X"0000", X"0000", X"0000",X"0000",

X"0000", X"0000", X"0000",X"0000",          --x0080 - x008F
X"0000", X"0000", X"0000",X"0000",
 X"0000", X"0000",X"0000",X"0000",
X"0000", X"0000", X"0000",X"0000",

X"0000", X"0000", X"0000",X"0000",          --x0090 - x009F
X"0000", X"0000", X"0000",X"0000",
 X"0000", X"0000",X"0000",X"0000",
X"0000", X"0000", X"0000",X"0000",
```

X"0000", X"0000", X"0000",X"0000", --x00A0 - x00AF
X"0000", X"0000", X"0000",X"0000",
X"0000", X"0000",X"0000",X"0000",
X"0000", X"0000", X"0000",X"0000",

X"0000", X"0000", X"0000",X"0000", --x00A0 - x00AF
X"0000", X"0000", X"0000",X"0000",
X"0000", X"0000",X"0000",X"0000",
X"0000", X"0000", X"0000",X"0000",

X"0000", X"0000", X"0000",X"0000", --x00B0 - x00BF
X"0000", X"0000", X"0000",X"0000",
X"0000", X"0000",X"0000",X"0000",
X"0000", X"0000", X"0000",X"0000",

X"0000", X"0000", X"0000",X"0000", --x00C0 - x00CF
X"0000", X"0000", X"0000",X"0000",
X"0000", X"0000",X"0000",X"0000",
X"0000", X"0000", X"0000",X"0000",

X"0000", X"0000", X"0000",X"0000", --x00D0 - x00DF
X"0000", X"0000", X"0000",X"0000",
X"0000", X"0000",X"0000",X"0000",
X"0000", X"0000", X"0000",X"0000",

X"0000", X"0000", X"0000",X"0000", --x00F0 - x00FF
X"0000", X"0000", X"0000",X"0000",
X"0000", X"0000",X"0000",X"0000",

```
X"0000", X"0000", X"0000",X"0000",

X"0000", X"0000", X"0000",X"0000",          --x0100 - x010F
X"0000", X"0000", X"0000",X"0000",
 X"0000", X"0000",X"0000",X"0000",
X"0000", X"0000", X"0000",X"0000",

X"0000", X"0000", X"0000",X"0000",          --x0110 - x011F
X"0000", X"0000", X"0000",X"0000",
 X"0000", X"0000",X"0000",X"0000",
X"0000", X"0000", X"0000",X"0000",

X"0000", X"0000", X"0000",X"0000",          --x0120 - x012F
X"0000", X"0000", X"0000",X"0000",
 X"0000", X"0000",X"0000",X"0000",
X"0000", X"0000", X"0000",X"0000",

X"0000", X"0000", X"0000",X"0000",          --x0130 - x013F
X"0000", X"0000", X"0000",X"0000",
 X"0000", X"0000",X"0000",X"0000",
X"0000", X"0000", X"0000",X"0000",

X"0000", X"0000", X"0000",X"0000",          --x0140 - x014F
X"0000", X"0000", X"0000",X"0000",
 X"0000", X"0000",X"0000",X"0000",
X"0000", X"0000", X"0000",X"0000",

X"0000", X"0000", X"0000",X"0000",          --x0150 - x015F
X"0000", X"0000", X"0000",X"0000",
```

```
 X"0000", X"0000",X"0000",X"0000",
X"0000", X"0000", X"0000",X"0000",

X"0000", X"0000", X"0000",X"0000",          --x0160 - x016F
X"0000", X"0000", X"0000",X"0000",
 X"0000", X"0000",X"0000",X"0000",
X"0000", X"0000", X"0000",X"0000",

X"0000", X"0000", X"0000",X"0000",          --x0170 - x017F
X"0000", X"0000", X"0000",X"0000",
 X"0000", X"0000",X"0000",X"0000",
X"0000", X"0000", X"0000",X"0000",

X"0000", X"0000", X"0000",X"0000",          --x0180 - x018F
X"0000", X"0000", X"0000",X"0000",
 X"0000", X"0000",X"0000",X"0000",
X"0000", X"0000", X"0000",X"0000",

X"0000", X"0000", X"0000",X"0000",          --x0190 - x019F
X"0000", X"0000", X"0000",X"0000",
 X"0000", X"0000",X"0000",X"0000",
X"0000", X"0000", X"0000",X"0000",

X"0000", X"0000", X"0000",X"0000",          --x01A0 - x01AF
X"0000", X"0000", X"0000",X"0000",
 X"0000", X"0000",X"0000",X"0000",
X"0000", X"0000", X"0000",X"0000",

X"0000", X"0000", X"0000",X"0000",          --x01A0 - x01AF
```

```
X"0000", X"0000", X"0000",X"0000",
 X"0000", X"0000",X"0000",X"0000",
X"0000", X"0000", X"0000",X"0000",

X"0000", X"0000", X"0000",X"0000",          --x01B0 - x01BF
X"0000", X"0000", X"0000",X"0000",
 X"0000", X"0000",X"0000",X"0000",
X"0000", X"0000", X"0000",X"0000",

X"0000", X"0000", X"0000",X"0000",          --x01C0 - x01CF
X"0000", X"0000", X"0000",X"0000",
 X"0000", X"0000",X"0000",X"0000",
X"0000", X"0000", X"0000",X"0000",

X"0000", X"0000", X"0000",X"0000",          --x01D0 - x01DF
X"0000", X"0000", X"0000",X"0000",
 X"0000", X"0000",X"0000",X"0000",
X"0000", X"0000", X"0000",X"0000",

X"0000", X"0000", X"0000",X"0000",          --x01F0 - x01FF
X"0000", X"0000", X"0000",X"0000",
 X"0000", X"0000",X"0000",X"0000",
X"0000", X"0000", X"0000",X"0000"  );
variable addr:integer;

begin -- the following type conversion function is in std_logic_arith

snl <= unsigned(address);
addr:= to_integer(snl(6 downto 0));
```

```
if MemWrite ='1' then
data_mem(addr):= write_data;
end if;
read_data<=data_mem(addr);

end process;

end Behavioral;
```

2.13 ZERO-FILL

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

entity zero_fill is
   Port ( SB : in  STD_LOGIC_VECTOR (2 downto 0);
      constant_out : out STD_LOGIC_VECTOR (15 downto 0));
end zero_fill;

architecture Behavioral of zero_fill is

begin
constant_out(2 downto 0) <= SB;
constant_out(15 downto 3) <= "0000000000000";
end Behavioral;
```

2.14 RIPPLE ADDER

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

```
entity ripple_adder_16bit is
    Port ( A : in  STD_LOGIC_VECTOR (15 downto 0);
           B : in  STD_LOGIC_VECTOR (15 downto 0);
           Cin : in  STD_LOGIC;
           Cout : out  STD_LOGIC;
           Vout : out  STD_LOGIC;
           S : out  STD_LOGIC_VECTOR (15 downto 0));
end ripple_adder_16bit;

architecture Behavioral of ripple_adder_16bit is
-- components
	COMPONENT full_adder
	PORT(
		A : IN std_logic;
		B : IN std_logic;
		Cin : IN std_logic;
		S : OUT std_logic;
		Cout : OUT std_logic
		);
	END COMPONENT;

	-- signals
	-- each c(i) is the carry going into adder(i) except c(0) which is the final carry
	signal c : std_logic_vector(15 downto 0);

begin
-- port maps ;-)
```

```
-- adder 0
add00: full_adder PORT MAP(
    A => A(0),
    B => B(0),
    Cin => Cin,
    S => S(0),
    Cout => c(1)
);
-- adder 1
add01: full_adder PORT MAP(
    A => A(1),
    B => B(1),
    Cin => c(1),
    S => S(1),
    Cout => c(2)
);
-- adder 2
add02: full_adder PORT MAP(
    A => A(2),
    B => B(2),
    Cin => c(2),
    S => S(2),
    Cout => c(3)
);
-- adder 3
add03: full_adder PORT MAP(
    A => A(3),
    B => B(3),
    Cin => c(3),
```

```
        S => S(3),
        Cout => c(4)
    );
    -- adder 4
    add04: full_adder PORT MAP(
        A => A(4),
        B => B(4),
        Cin => c(4),
        S => S(4),
        Cout => c(5)
    );
    -- adder 5
    add05: full_adder PORT MAP(
        A => A(5),
        B => B(5),
        Cin => c(5),
        S => S(5),
        Cout => c(6)
    );
    -- adder 6
    add06: full_adder PORT MAP(
        A => A(6),
        B => B(6),
        Cin => c(6),
        S => S(6),
        Cout => c(7)
    );
    -- adder 7
    add07: full_adder PORT MAP(
```

```
        A => A(7),
        B => B(7),
        Cin => c(7),
        S => S(7),
        Cout => c(8)
    );
    -- adder 8
    add08: full_adder PORT MAP(
        A => A(8),
        B => B(8),
        Cin => c(8),
        S => S(8),
        Cout => c(9)
    );
    -- adder 9
    add09: full_adder PORT MAP(
        A => A(9),
        B => B(9),
        Cin => c(9),
        S => S(9),
        Cout => c(10)
    );
    -- adder 10
    add10: full_adder PORT MAP(
        A => A(10),
        B => B(10),
        Cin => c(10),
        S => S(10),
        Cout => c(11)
```

```
);
-- adder 11
add11: full_adder PORT MAP(
    A => A(11),
    B => B(11),
    Cin => c(11),
    S => S(11),
    Cout => c(12)
);
-- adder 12
add12: full_adder PORT MAP(
    A => A(12),
    B => B(12),
    Cin => c(12),
    S => S(12),
    Cout => c(13)
);
-- adder 13
add13: full_adder PORT MAP(
    A => A(13),
    B => B(13),
    Cin => c(13),
    S => S(13),
    Cout => c(14)
);
-- adder 14
add14: full_adder PORT MAP(
    A => A(14),
    B => B(14),
```

```
    Cin => c(14),
    S => S(14),
    Cout => c(15)
);
-- adder 15
add15: full_adder PORT MAP(
    A => A(15),
    B => B(15),
    Cin => c(15),
    S => S(15),
    Cout => c(0)
);
Vout <= (c(15) xor c(0));
Cout <= c(0);
end Behavioral;
```

2.15 REGISTER FILE

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

entity new_reg_file_9x16bit is
   Port ( des_sel : in  STD_LOGIC_VECTOR (2 downto 0);
          a_sel : in  STD_LOGIC_VECTOR (2 downto 0);
          b_sel : in  STD_LOGIC_VECTOR (2 downto 0);
                        TD : in STD_LOGIC;
                        TA : in STD_LOGIC;
                        TB : in STD_LOGIC;
          load_enable : in  STD_LOGIC;
                        reset : in std_logic;
```

```
        Clk : in  STD_LOGIC;
        input_data : in  STD_LOGIC_VECTOR (15 downto 0);
        a_data : out  STD_LOGIC_VECTOR (15 downto 0);
        b_data : out  STD_LOGIC_VECTOR (15 downto 0));
end new_reg_file_9x16bit;

architecture Behavioral of new_reg_file_9x16bit is
-- components
    -- 16 bit Register for register file
    COMPONENT reg16
    PORT(
            D : IN std_logic_vector(15 downto 0);
            load : IN std_logic;
            Clk : IN std_logic;
            Q : OUT std_logic_vector(15 downto 0)
            );
    END COMPONENT;

            -- 3 to 8 Decoder
    COMPONENT decoder_4to16
    PORT(
            A0 : IN std_logic;
            A1 : IN std_logic;
            A2 : IN std_logic;
            A3 : IN std_logic;
            Q0 : OUT std_logic;
            Q1 : OUT std_logic;
            Q2 : OUT std_logic;
            Q3 : OUT std_logic;
```

```
        Q4 : OUT std_logic;
        Q5 : OUT std_logic;
        Q6 : OUT std_logic;
        Q7 : OUT std_logic;
        Q8 : OUT std_logic;
        Q9 : OUT std_logic;
        Q10 : OUT std_logic;
        Q11 : OUT std_logic;
        Q12 : OUT std_logic;
        Q13 : OUT std_logic;
        Q14 : OUT std_logic;
        Q15 : OUT std_logic
        );
END COMPONENT;

-- 2 to 1 line multiplexer
COMPONENT mux_2_16bit
PORT(
        In0 : IN std_logic_vector(15 downto 0);
        In1 : IN std_logic_vector(15 downto 0);
        s : IN std_logic;
        Z : OUT std_logic_vector(15 downto 0)
        );
END COMPONENT;

-- 16 to 1 line multiplexer
COMPONENT mux16_16bit
PORT(
        In0 : IN std_logic_vector(15 downto 0);
```

```
        In1 : IN std_logic_vector(15 downto 0);
        In2 : IN std_logic_vector(15 downto 0);
        In3 : IN std_logic_vector(15 downto 0);
        In4 : IN std_logic_vector(15 downto 0);
        In5 : IN std_logic_vector(15 downto 0);
        In6 : IN std_logic_vector(15 downto 0);
        In7 : IN std_logic_vector(15 downto 0);
        In8 : IN std_logic_vector(15 downto 0);
        In9 : IN std_logic_vector(15 downto 0);
        In10 : IN std_logic_vector(15 downto 0);
        In11 : IN std_logic_vector(15 downto 0);
        In12 : IN std_logic_vector(15 downto 0);
        In13 : IN std_logic_vector(15 downto 0);
        In14 : IN std_logic_vector(15 downto 0);
        In15 : IN std_logic_vector(15 downto 0);
        S0 : IN std_logic;
        S1 : IN std_logic;
        S2 : IN std_logic;
        S3 : IN std_logic;
        Z : OUT std_logic_vector(15 downto 0)
        );
END COMPONENT;

-- signals
signal d_data : std_logic_vector(15 downto 0);
signal load_reg0, load_reg1, load_reg2, load_reg3,
load_reg4, load_reg5, load_reg6, load_reg7, load_reg8, load_reg9, load_reg10,
load_reg11, load_reg12, load_reg13, load_reg14, load_reg15 : std_logic;
signal reg0_q, reg1_q, reg2_q, reg3_q,
```

```
    reg4_q, reg5_q, reg6_q, reg7_q, reg8_q, reg9_q, reg10_q,
    reg11_q, reg12_q, reg13_q, reg14_q, reg15_q : std_logic_vector(15 downto 0);
    signal write_reg0, write_reg1, write_reg2, write_reg3,
    write_reg4, write_reg5, write_reg6, write_reg7, write_reg8, write_reg9,
write_reg10,
    write_reg11, write_reg12, write_reg13, write_reg14, write_reg15 : std_logic;

begin
-- port maps ;-)

-- register 0
reg00: reg16 PORT MAP(
    D => d_data,
    load => write_reg0,
    Clk => Clk,
    Q => reg0_q
);
-- register 1
reg01: reg16 PORT MAP(
    D => d_data,
    load => write_reg1,
    Clk => Clk,
    Q => reg1_q
);
-- register 2
reg02: reg16 PORT MAP(
    D => d_data,
    load => write_reg2,
    Clk => Clk,
    Q => reg2_q
```

```
);
-- register 3
reg03: reg16 PORT MAP(
      D => d_data,
      load => write_reg3,
      Clk => Clk,
      Q => reg3_q
);
-- register 4
reg04: reg16 PORT MAP(
      D => d_data,
      load => write_reg4,
      Clk => Clk,
      Q => reg4_q
);
-- register 5
reg05: reg16 PORT MAP(
      D => d_data,
      load => write_reg5,
      Clk => Clk,
      Q => reg5_q
);
-- register 6
reg06: reg16 PORT MAP(
      D => d_data,
      load => write_reg6,
      Clk => Clk,
      Q => reg6_q
);
```

```
-- register 7
reg07: reg16 PORT MAP(
      D => d_data,
      load => write_reg7,
      Clk => Clk,
      Q => reg7_q
);
-- register 8
reg08: reg16 PORT MAP(
      D => d_data,
      load => write_reg8,
      Clk => Clk,
      Q => reg8_q
);

-- Destination register decoder
des_decoder_4to16: decoder_4to16 PORT MAP(
      A0 => des_sel(0),
      A1 => des_sel(1),
      A2 => des_sel(2),
      A3 => TD,
      Q0 => load_reg0,
      Q1 => load_reg1,
      Q2 => load_reg2,
      Q3 => load_reg3,
      Q4 => load_reg4,
      Q5 => load_reg5,
      Q6 => load_reg6,
      Q7 => load_reg7,
```

```
        Q8 => load_reg8,
        Q9 => load_reg9,
        Q10 => load_reg10,
        Q11 => load_reg11,
        Q12 => load_reg12,
        Q13 => load_reg13,
        Q14 => load_reg14,
        Q15 => load_reg15
);



-- 16 to 1 source register multiplexer
a_data_mux: mux16_16bit PORT MAP(
        In0 => reg0_q,
        In1 => reg1_q,
        In2 => reg2_q,
        In3 => reg3_q,
        In4 => reg4_q,
        In5 => reg5_q,
        In6 => reg6_q,
        In7 => reg7_q,
        In8 => reg8_q,
        In9 => reg9_q,
        In10 => reg10_q,
        In11 => reg11_q,
        In12 => reg12_q,
        In13 => reg13_q,
        In14 => reg14_q,
        In15 => reg15_q,
```

```
    S0 => a_sel(0),
    S1 => a_sel(1),
    S2 => a_sel(2),
    S3 => TA,
    Z => a_data
);

-- 16 to 1 source register multiplexer
b_data_mux: mux16_16bit PORT MAP(
    In0 => reg0_q,
    In1 => reg1_q,
    In2 => reg2_q,
    In3 => reg3_q,
    In4 => reg4_q,
    In5 => reg5_q,
    In6 => reg6_q,
    In7 => reg7_q,
    In8 => reg8_q,
    In9 => reg9_q,
    In10 => reg10_q,
    In11 => reg11_q,
    In12 => reg12_q,
    In13 => reg13_q,
    In14 => reg14_q,
    In15 => reg15_q,
    S0 => b_sel(0),
    S1 => b_sel(1),
    S2 => b_sel(2),
    S3 => TB,
```

```
    Z => b_data
);

    write_reg0 <= (load_reg0 and load_enable) or reset;
    write_reg1 <= (load_reg1 and load_enable) or reset;
    write_reg2 <= (load_reg2 and load_enable) or reset;
    write_reg3 <= (load_reg3 and load_enable) or reset;
    write_reg4 <= (load_reg4 and load_enable) or reset;
    write_reg5 <= (load_reg5 and load_enable) or reset;
    write_reg6 <= (load_reg6 and load_enable) or reset;
    write_reg7 <= (load_reg7 and load_enable) or reset;
    write_reg8 <= ( TD and load_enable) or reset;
    write_reg9 <= (load_reg9 and load_enable) or reset;
    write_reg10 <= (load_reg10 and load_enable) or reset;
    write_reg11 <= (load_reg11 and load_enable) or reset;
    write_reg12 <= (load_reg12 and load_enable) or reset;
    write_reg13 <= (load_reg13 and load_enable) or reset;
    write_reg14 <= (load_reg14 and load_enable) or reset;
    write_reg15 <= (load_reg15 and load_enable) or reset;

    d_data <= x"0000" when reset='1' else input_data;

end Behavioral;
```

2.16 LOGIC CIRCUIT

```
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 logic_circuit_16bit is
Port (
 A: in STD_LOGIC_VECTOR(15 downto 0);
 B: in STD_LOGIC_VECTOR(15 downto 0);
 s0,s1: in STD_LOGIC;
 Y: out STD_LOGIC_VECTOR(15 downto 0));
end logic_circuit_16bit;

architecture Behavioral of logic_circuit_16bit is

COMPONENT mux4_16bits
Port(
In0, In1, In2, In3: in std_logic_vector(15 downto 0);
 S0, S1 : in std_logic;
 Z : out std_logic_vector(15 downto 0));
 End Component;

signal sig1,sig2,sig3,sig4: std_logic_vector(15 downto 0);
begin

sig1(0) <= A(0) and B(0) after 1 ns;
sig1(1) <= A(1) and B(1) after 1 ns;
sig1(2) <= A(2) and B(2) after 1 ns;
sig1(3) <= A(3) and B(3) after 1 ns;
sig1(4) <= A(4) and B(4) after 1 ns;
sig1(5) <= A(5) and B(5) after 1 ns;
sig1(6) <= A(6) and B(6) after 1 ns;
```

```
sig1(7) <= A(7) and B(7) after 1 ns;
sig1(8) <= A(8) and B(8) after 1 ns;
sig1(9) <= A(9) and B(9) after 1 ns;
sig1(10) <=A(10) and B(10) after 1 ns;
sig1(11) <=A(11) and B(11) after 1 ns;
sig1(12) <=A(12) and B(12) after 1 ns;
sig1(13) <=A(13) and B(13) after 1 ns;
sig1(14) <=A(14) and B(14) after 1 ns;
sig1(15) <=A(15) and B(15) after 1 ns;
--sig1(8) <= In0(8) and In1(8) after 1 ns;

sig2(0) <= A(0) or B(0) after 1 ns;
sig2(1) <= A(1) or B(1) after 1 ns;
sig2(2) <= A(2) or B(2) after 1 ns;
sig2(3) <= A(3) or B(3) after 1 ns;
sig2(4) <= A(4) or B(4) after 1 ns;
sig2(5) <= A(5) or B(5) after 1 ns;
sig2(6) <= A(6) or B(6) after 1 ns;
sig2(7) <= A(7) or B(7) after 1 ns;
sig2(8) <= A(8) or B(8) after 1 ns;
sig2(9) <= A(9) or B(9) after 1 ns;
sig2(10) <=A(10)or B(10) after 1 ns;
sig2(11) <=A(11)or B(11) after 1 ns;
sig2(12) <=A(12)or B(12) after 1 ns;
sig2(13) <=A(13)or B(13) after 1 ns;
sig2(14) <=A(14)or B(14) after 1 ns;
sig2(15) <=A(15)or B(15) after 1 ns;
```

```
sig3(0) <= A(0)  xor B(0) after 1 ns;
sig3(1) <= A(1)  xor B(1) after 1 ns;
sig3(2) <= A(2)  xor B(2) after 1 ns;
sig3(3) <= A(3)  xor B(3) after 1 ns;
sig3(4) <= A(4)  xor B(4) after 1 ns;
sig3(5) <= A(5)  xor B(5) after 1 ns;
sig3(6) <= A(6)  xor B(6) after 1 ns;
sig3(7) <= A(7)  xor B(7) after 1 ns;
sig3(8) <= A(8)  xor B(8) after 1 ns;
sig3(9) <= A(9)  xor B(9) after 1 ns;
sig3(10) <=A(10) xor B(10) after 1 ns;
sig3(11) <=A(11) xor B(11) after 1 ns;
sig3(12) <=A(12) xor B(12) after 1 ns;
sig3(13) <=A(13) xor B(13) after 1 ns;
sig3(14) <=A(14) xor B(14) after 1 ns;
sig3(15) <=A(15) xor B(15) after 1 ns;


sig4(0) <= not A(0) after 1 ns;
sig4(1) <= not A(1) after 1 ns;
sig4(2) <= not A(2) after 1 ns;
sig4(3) <= not A(3) after 1 ns;
sig4(4) <= not A(4) after 1 ns;
sig4(5) <= not A(5) after 1 ns;
sig4(6) <= not A(6) after 1 ns;
sig4(7) <= not A(7) after 1 ns;
sig4(8) <= not A(8) after 1 ns;
sig4(9) <= not A(9) after 1 ns;
sig4(10) <=not A(10) after 1 ns;
```

```
sig4(11) <=not A(11) after 1 ns;
sig4(12) <=not A(12) after 1 ns;
sig4(13) <=not A(13) after 1 ns;
sig4(14) <=not A(14) after 1 ns;
sig4(15) <=not A(15) after 1 ns;

mux4_16bit : mux4_16bits PORT MAP(
S0 => s0,
S1 => s1,
In0 => sig1,
In1 => sig2,
In2 => sig3,
In3 => sig4,
Z => Y
);

end Behavioral;
```

2.17 DATAPATH

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

entity last_data_path is
   Port ( PC : in  STD_LOGIC_VECTOR (15 downto 0);
                        data_in : in  STD_LOGIC_VECTOR (15 downto 0);
          DR : IN  std_logic_vector(2 downto 0);
          SA : IN  std_logic_vector(2 downto 0);
          SB : IN  std_logic_vector(2 downto 0);
          TD : IN  std_logic;
          TA : IN  std_logic;
```

```
        TB : IN  std_logic;
        Clk : in  STD_LOGIC;
              reset : OUT STD_LOGIC;
        FS : IN  std_logic_vector(4 downto 0);
        RW : IN  std_logic;
        MB : IN  std_logic;
        MM : IN  std_logic;
        MD : IN  std_logic;
        data_out : out  STD_LOGIC_VECTOR (15 downto 0);
        address_out : out  STD_LOGIC_VECTOR (15 downto 0);
        V : out  STD_LOGIC;
        C : out  STD_LOGIC;
        N : out  STD_LOGIC;
        Z : out  STD_LOGIC);
end last_data_path;

architecture Behavioral of last_data_path is

   COMPONENT new_reg_file_9x16bit
   PORT(
      des_sel : IN  std_logic_vector(2 downto 0);
      a_sel : IN  std_logic_vector(2 downto 0);
      b_sel : IN  std_logic_vector(2 downto 0);
                  TD : IN std_logic;
                  TA : IN std_logic;
                  TB : IN std_logic;
      load_enable : IN  std_logic;
                  reset :  in std_logic;
      Clk : IN  std_logic;
```

```
    input_data : IN  std_logic_vector(15 downto 0);
    a_data : OUT  std_logic_vector(15 downto 0);
    b_data : OUT  std_logic_vector(15 downto 0)
   );
END COMPONENT;

   COMPONENT zero_fill
   PORT(
                    SB : IN std_logic_vector(2 downto 0);
                    constant_out : out std_logic_vector(15 downto 0)
                    );
   END COMPONENT;

COMPONENT functional_unit_16bit
PORT(
    FSS : IN  std_logic_vector(4 downto 0);
    A : IN  std_logic_vector(15 downto 0);
    B : IN  std_logic_vector(15 downto 0);
    F : OUT  std_logic_vector(15 downto 0);
    V : OUT  std_logic;
    C : OUT  std_logic;
    N : OUT  std_logic;
    Z : OUT  std_logic
   );
END COMPONENT;

COMPONENT mux_2_16bit
PORT(
    S : IN  std_logic;
```

```
        In0 : IN  std_logic_vector(15 downto 0);
        In1 : IN  std_logic_vector(15 downto 0);
        Z : OUT  std_logic_vector(15 downto 0)
       );
    END COMPONENT;

                --signals
   signal Cout : std_logic;
   signal Vout : std_logic;
   signal reset1 : std_logic := '0';
   signal mux_b_out : std_logic_vector(15 downto 0);
   signal mux_d_out : std_logic_vector(15 downto 0);
   signal zero_fill_out : std_logic_vector(15 downto 0);
   signal functional_unit_out : std_logic_vector(15 downto 0);
   signal d_data : std_logic_vector(15 downto 0);
   signal a_reg_file_out : std_logic_vector(15 downto 0);
   signal b_reg_file_out : std_logic_vector(15 downto 0);

begin

reg_file: new_reg_file_9x16bit PORT MAP (
        des_sel(2 downto 0) => DR,
                        TD => TD,
        a_sel(2 downto 0) => SA,
                        TA => TA,
        b_sel(2 downto 0) => SB,
                        TB => TB,
        load_enable => RW,
                        reset => reset1,
```

```
        -- reset => reset,
                    --reset1 => reset,
        Clk => Clk,
        input_data => mux_d_out,
        a_data => a_reg_file_out,
        b_data => b_reg_file_out
      );

z_fill: zero_fill PORT MAP (
                    SB => SB,
                    constant_out => zero_fill_out
                    );

func_unit: functional_unit_16bit PORT MAP (
        FSS => FS,
        A => a_reg_file_out,
        B => mux_b_out,
        F => functional_unit_out,
        V => V,
        C => C,
        N => N,
        Z => Z
      );

mux_b: mux_2_16bit PORT MAP (
        S => MB,
        In0 => b_reg_file_out,
        In1 => zero_fill_out,
        Z => mux_b_out
```

```
        );

mux_d: mux_2_16bit PORT MAP (
          S => MD,
          In0 => functional_unit_out,
          In1 => data_in,
          Z => mux_d_out
        );

mux_m: mux_2_16bit PORT MAP (
          S => MM,
          In0 => a_reg_file_out,
          In1 => PC,
          Z => address_out
        );
reset <= reset1;
data_out <= mux_b_out;
end Behavioral;
```

2.18 DECODER3_8

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

entity decoder_3to8 is
    Port ( A0 : in  STD_LOGIC;
           A1 : in  STD_LOGIC;
           A2 : in  STD_LOGIC;
           Q0 : out  STD_LOGIC;
```

```
        Q1 : out  STD_LOGIC;
        Q2 : out  STD_LOGIC;
        Q3 : out  STD_LOGIC;
        Q4 : out  STD_LOGIC;
        Q5 : out  STD_LOGIC;
        Q6 : out  STD_LOGIC;
        Q7 : out  STD_LOGIC);
end decoder_3to8;

architecture Behavioral of decoder_3to8 is

begin
Q0 <= ( (not A0) and (not A1) and (not A2) ) after 5 ns;
Q1 <= ( A0 and (not A1) and (not A2) ) after 5 ns;
Q2 <= ( (not A0) and A1 and (not A2) ) after 5 ns;
Q3 <= ( A0 and A1 and (not A2) ) after 5 ns;
Q4 <= ( (not A0) and (not A1) and A2 ) after 5 ns;
Q5 <= ( A0 and (not A1) and A2 ) after 5 ns;
Q6 <= ( (not A0) and A1 and A2 ) after 5 ns;
Q7 <= ( A0 and A1 and A2 ) after 5 ns;
end Behavioral;
```

2.19 DECODER4_16

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

entity decoder_4to16 is
   Port ( A0 : in  STD_LOGIC;
          A1 : in  STD_LOGIC;
```

```
        A2 : in  STD_LOGIC;
        A3 : in  STD_LOGIC;
        Q0 : out  STD_LOGIC;
        Q1 : out  STD_LOGIC;
        Q2 : out  STD_LOGIC;
        Q3 : out  STD_LOGIC;
        Q4 : out  STD_LOGIC;
        Q5 : out  STD_LOGIC;
        Q6 : out  STD_LOGIC;
        Q7 : out  STD_LOGIC;
        Q8 : out  STD_LOGIC;
        Q9 : out  STD_LOGIC;
        Q10 : out  STD_LOGIC;
        Q11 : out  STD_LOGIC;
        Q12 : out  STD_LOGIC;
        Q13 : out  STD_LOGIC;
        Q14 : out  STD_LOGIC;
        Q15 : out  STD_LOGIC);
end decoder_4to16;

architecture Behavioral of decoder_4to16 is

begin
Q0 <= ( (not A0) and (not A1) and (not A2) and (not A3) );
Q1 <= ( A0 and (not A1) and (not A2)  and (not A3));
Q2 <= ( (not A0) and A1 and (not A2)  and (not A3));
Q3 <= ( A0 and A1 and (not A2)  and (not A3));
Q4 <= ( (not A0) and (not A1) and A2  and (not A3));
Q5 <= ( A0 and (not A1) and A2  and (not A3));
```

```
Q6 <= ( (not A0) and A1 and A2  and (not A3));
Q7 <= ( A0 and A1 and A2  and (not A3));
Q8 <= ( (not A0) and (not A1) and (not A2) and A3);
Q9 <= ( A0 and (not A1) and (not A2)  and A3);
Q10 <= ( (not A0) and A1 and (not A2)  and A3);
Q11 <= ( A0 and A1 and (not A2)  and A3);
Q12 <= ( (not A0) and (not A1) and A2  and A3);
Q13 <= ( A0 and (not A1) and A2  and A3);
Q14 <= ( (not A0) and A1 and A2  and A3);
Q15 <= ( A0 and A1 and A2  and A3);
end Behavioral;
```

2.20 MUX16_16BIT

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

entity mux16_16bit is
   Port ( S0 : in  STD_LOGIC;
          S1 : in  STD_LOGIC;
          S2 : in  STD_LOGIC;
          S3 : in  STD_LOGIC;
          In0 : in  STD_LOGIC_VECTOR (15 downto 0);
          In1 : in  STD_LOGIC_VECTOR (15 downto 0);
          In2 : in  STD_LOGIC_VECTOR (15 downto 0);
          In3 : in  STD_LOGIC_VECTOR (15 downto 0);
          In4 : in  STD_LOGIC_VECTOR (15 downto 0);
          In5 : in  STD_LOGIC_VECTOR (15 downto 0);
          In6 : in  STD_LOGIC_VECTOR (15 downto 0);
          In7 : in  STD_LOGIC_VECTOR (15 downto 0);
```

```
        In8 : in  STD_LOGIC_VECTOR (15 downto 0);
        In9 : in  STD_LOGIC_VECTOR (15 downto 0);
        In10 : in  STD_LOGIC_VECTOR (15 downto 0);
        In11 : in  STD_LOGIC_VECTOR (15 downto 0);
        In12 : in  STD_LOGIC_VECTOR (15 downto 0);
        In13 : in  STD_LOGIC_VECTOR (15 downto 0);
        In14 : in  STD_LOGIC_VECTOR (15 downto 0);
        In15 : in  STD_LOGIC_VECTOR (15 downto 0);
        Z : out  STD_LOGIC_VECTOR (15 downto 0));
end mux16_16bit;

architecture Behavioral of mux16_16bit is

begin
Z <= In0 when (S0='0' and S1='0' and S2='0' and S3='0') else
In1 when (S0='1' and S1='0' and S2='0' and S3='0') else
In2 when (S0='0' and S1='1' and S2='0' and S3='0') else
In3 when (S0='1' and S1='1' and S2='0' and S3='0') else
In4 when (S0='0' and S1='0' and S2='1' and S3='0') else
In5 when (S0='1' and S1='0' and S2='1' and S3='0') else
In6 when (S0='0' and S1='1' and S2='1' and S3='0') else
In7 when (S0='1' and S1='1' and S2='1' and S3='0') else
In8 when (S0='0' and S1='0' and S2='0' and S3='1') else
In9 when (S0='1' and S1='0' and S2='0' and S3='1') else
In10 when (S0='0' and S1='1' and S2='0' and S3='1') else
In11 when (S0='1' and S1='1' and S2='0' and S3='1') else
In12 when (S0='0' and S1='0' and S2='1' and S3='1') else
In13 when (S0='1' and S1='0' and S2='1' and S3='1') else
In14 when (S0='0' and S1='1' and S2='1' and S3='1') else
```

```
In15 when (S0='1' and S1='1' and S2='1' and S3='1') else
x"0000";
end Behavioral;
```

2.21 MUX16_1BIT

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

entity mux16_1bit is
    Port ( S0 : in  STD_LOGIC;
                S1 : in  STD_LOGIC;
                S2 : in  STD_LOGIC;
                S3 : in  STD_LOGIC;
           A : in  STD_LOGIC_VECTOR (15 downto 0);
           F : out  STD_LOGIC);
end mux16_1bit;

architecture Behavioral of mux16_1bit is

begin
F <= A(0) when (S3='0' and S2='0' and S1='0' and S0='0') else
A(1) when (S3='0' and S2='0' and S1='0' and S0='1') else
A(2) when (S3='0' and S2='0' and S1='1' and S0='0') else
A(3) when (S3='0' and S2='0' and S1='1' and S0='1') else
A(4) when (S3='0' and S2='1' and S1='0' and S0='0') else
A(5) when (S3='0' and S2='1' and S1='0' and S0='1') else
A(6) when (S3='0' and S2='1' and S1='1' and S0='0') else
A(7) when (S3='0' and S2='1' and S1='1' and S0='1') else
A(8) when (S3='1' and S2='0' and S1='0' and S0='0') else
A(9) when (S3='1' and S2='0' and S1='0' and S0='1') else
```

```
A(10) when (S3='1' and S2='0' and S1='1' and S0='0') else
A(11) when (S3='1' and S2='0' and S1='1' and S0='1') else
A(12) when (S3='1' and S2='1' and S1='0' and S0='0') else
A(13) when (S3='1' and S2='1' and S1='0' and S0='1') else
A(14) when (S3='1' and S2='1' and S1='1' and S0='0') else
A(15) when (S3='1' and S2='1' and S1='1' and S0='1');
end Behavioral;
```

2.22 MUX2_16BIT

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

entity mux_2_16bit is
   Port ( S : in  STD_LOGIC;
          In0 : in  STD_LOGIC_VECTOR (15 downto 0);
          In1 : in  STD_LOGIC_VECTOR (15 downto 0);
          Z : out  STD_LOGIC_VECTOR (15 downto 0));
end mux_2_16bit;

architecture Behavioral of mux_2_16bit is

begin
Z <= In0 when S='0' else
In1 when S='1' else
x"0000";
end Behavioral;
```

2.23 MUX2_8BIT

```
library IEEE;
```

```
use IEEE.STD_LOGIC_1164.ALL;

entity mux_2_to_1_8bit is
   Port ( In0 : in  STD_LOGIC_VECTOR (7 downto 0);
          In1 : in  STD_LOGIC_VECTOR (7 downto 0);
                      S : in STD_LOGIC;
          Z : out  STD_LOGIC_VECTOR (7 downto 0));
end mux_2_to_1_8bit;

architecture Behavioral of mux_2_to_1_8bit is

begin
Z <= In0 when S='0' else
In1 when S='1' else
x"00";
end Behavioral;
```

2.24 MUX8_16BIT

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

entity mux8_16bit is
   Port ( S0 : in  STD_LOGIC;
          S1 : in  STD_LOGIC;
          S2 : in  STD_LOGIC;
          In0 : in  STD_LOGIC_VECTOR (15 downto 0);
          In1 : in  STD_LOGIC_VECTOR (15 downto 0);
          In2 : in  STD_LOGIC_VECTOR (15 downto 0);
```

```
        In3 : in  STD_LOGIC_VECTOR (15 downto 0);
        In4 : in  STD_LOGIC_VECTOR (15 downto 0);
        In5 : in  STD_LOGIC_VECTOR (15 downto 0);
        In6 : in  STD_LOGIC_VECTOR (15 downto 0);
        In7 : in  STD_LOGIC_VECTOR (15 downto 0);
        Z : out  STD_LOGIC_VECTOR (15 downto 0));
end mux8_16bit;

architecture Behavioral of mux8_16bit is

begin
Z <= In0 when (S0='0' and S1='0' and S2='0') else
In1 when (S0='1' and S1='0' and S2='0') else
In2 when (S0='0' and S1='1' and S2='0') else
In3 when (S0='1' and S1='1' and S2='0') else
In4 when (S0='0' and S1='0' and S2='1') else
In5 when (S0='1' and S1='0' and S2='1') else
In6 when (S0='0' and S1='1' and S2='1') else
In7 when (S0='1' and S1='1' and S2='1') else
x"0000";
end Behavioral;
```

2.25 MUX8_1BIT

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

entity mux_8_to_1_1bit is
    Port ( A : in  STD_LOGIC_VECTOR (7 downto 0);
           S0 : in  STD_LOGIC;
```

```
        S1 : in  STD_LOGIC;
        S2 : in  STD_LOGIC;
        line_out : out  STD_LOGIC);
end mux_8_to_1_1bit;

architecture Behavioral of mux_8_to_1_1bit is

begin
line_out <= A(0) when (S2='0' and S1='0' and S0='0') else
A(1) when (S2='0' and S1='0' and S0='1') else
A(2) when (S2='0' and S1='1' and S0='0') else
A(3) when (S2='0' and S1='1' and S0='1') else
A(4) when (S2='1' and S1='0' and S0='0') else
A(5) when (S2='1' and S1='0' and S0='1') else
A(6) when (S2='1' and S1='1' and S0='0') else
A(7) when (S2='1' and S1='1' and S0='1') else '0';

end Behavioral;
```

2.26 FULL ADDER

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

entity full_adder is
   Port ( A : in  STD_LOGIC;
        B : in  STD_LOGIC;
        Cin : in  STD_LOGIC;
        S : out  STD_LOGIC;
        Cout : out  STD_LOGIC);
```

```
end full_adder;

architecture Behavioral of full_adder is

begin
S <= ((A xor B) xor Cin);
Cout <= ((Cin and (A xor B)) or (A and B));
end Behavioral;
```

2.27 MUXB_Y_16BIT

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

entity b_to_y_16bit is
   Port ( B : in  STD_LOGIC_VECTOR (15 downto 0);
          Y : out  STD_LOGIC_VECTOR (15 downto 0);
                   S0 : in std_logic;
                   S1 : in std_logic
                );
end b_to_y_16bit;

architecture Behavioral of b_to_y_16bit is

begin
Y <= x"0000" when (S0='0' and S1='0') else
B when (S0='1' and S1='0') else
(not B) when (S0='0' and S1='1') else
x"FFFF" when (S0='1' and S1='1');
end Behavioral;
```

2.28 REGISTER

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

entity reg16 is
   Port ( D : in  STD_LOGIC_VECTOR (15 downto 0);
          load : in  STD_LOGIC;
          Clk : in  STD_LOGIC;
          Q : out  STD_LOGIC_VECTOR (15 downto 0));
end reg16;

architecture Behavioral of reg16 is

begin
process(Clk)
begin
if (rising_edge(Clk)) then
          if load='1' then
                    Q<=D;
          end if;
end if;
end process;
end Behavioral;
```

3.

3.1 PROCESSOR FINAL

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY processor_final_TestBench IS
```

```
END processor_final_TestBench;

ARCHITECTURE behavior OF processor_final_TestBench IS

   -- Component Declaration for the Unit Under Test (UUT)

   COMPONENT processor_final
   PORT(
        Clk : IN  std_logic;
        reset : IN  std_logic
       );
   END COMPONENT;



  --Inputs
  signal Clk : std_logic := '0';
  signal reset : std_logic := '0';

  -- Clock period definitions
  constant Clk_period : time := 10 ns;

BEGIN

     -- Instantiate the Unit Under Test (UUT)
  uut: Processor_final PORT MAP (
         Clk => Clk,
         reset => reset
        );
```

```
  -- Clock process definitions
  Clk_process :process
  begin
                Clk <= '0';
                wait for Clk_period/2;
                Clk <= '1';
                wait for Clk_period/2;
  end process;


  -- Stimulus process
  stim_proc: process
  begin
    reset <= '1';
    wait for 100 ns;
                reset <= '0';
    wait;
  end process;

END;
```

3.2 ARITHEMATIC LOGICAL UNIT

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY ALU_unit_16bit_TestBench IS
END ALU_unit_16bit_TestBench;

ARCHITECTURE behavior OF ALU_unit_16bit_TestBench IS
```

```
-- Component Declaration for the Unit Under Test (UUT)

 COMPONENT ALU_unit_16bit
 PORT(
     A : IN  std_logic_vector(15 downto 0);
     B : IN  std_logic_vector(15 downto 0);
     Cin : in  STD_LOGIC;
     S0 : IN  std_logic;
     S1 : IN  std_logic;
     S2 : IN  std_logic;
     Cout : OUT  std_logic;
     Vout : OUT  std_logic;
     G : OUT  std_logic_vector(15 downto 0)
    );
 END COMPONENT;


--Inputs
signal A : std_logic_vector(15 downto 0) := (others => '0');
signal B : std_logic_vector(15 downto 0) := (others => '0');
signal Cin : std_logic := '0';
signal S0 : std_logic := '0';
signal S1 : std_logic := '0';
signal S2 : std_logic := '0';

   --Outputs
signal Cout : std_logic;
signal Vout : std_logic;
signal G : std_logic_vector(15 downto 0);
```

```
  -- No clocks detected in port list. Replace <clock> below with
  -- appropriate port name


BEGIN

    -- Instantiate the Unit Under Test (UUT)
  uut: ALU_unit_16bit PORT MAP (
        A => A,
        B => B,
                        Cin => Cin,
        S0 => S0,
        S1 => S1,
        S2 => S2,
        Cout => Cout,
        Vout => Vout,
        G => G
      );

    --CYCLE TROUGH DIFFERENT INPUTS
    --  TO TEST EACH OPERATION
  stim_proc1: process
  begin
    wait for 50 ns;
              S0 <= not S0;
  end process;

  stim_proc2: process
  begin
```

```
    wait for 100 ns;
            S1 <= not S1;
  end process;

  stim_proc3: process
  begin
    wait for 200 ns;
            S2 <= not S2;
  end process;

    stim_proc4: process
  begin
            A <= x"00FA";
            B <= x"002A";
    wait for 400 ns;
  end process;

END;
```

3.3 CONTROL ADDRESS REGISTER

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY CAR_TestBench IS
END CAR_TestBench;

ARCHITECTURE behavior OF CAR_TestBench IS

   -- Component Declaration for the Unit Under Test (UUT)
```

```
 COMPONENT CAR
 PORT(
     data_in : IN  std_logic_vector(7 downto 0);
                  reset : IN std_logic;
                  Clk : IN std_logic;
     Con : IN  std_logic;
     data_out : OUT  std_logic_vector(7 downto 0)
    );
 END COMPONENT;


 --Inputs
 signal data_in : std_logic_vector(7 downto 0) := (others => '0');
 signal Con : std_logic := '0';
   signal Clk : std_logic := '0';
   signal reset : std_logic := '0';

   --Outputs
 signal data_out : std_logic_vector(7 downto 0);

 -- Clock period definitions
 constant Clk_period : time := 10 ns;

BEGIN

   -- Instantiate the Unit Under Test (UUT)
 uut: CAR PORT MAP (
      data_in => data_in,
                  reset => reset,
```

```
                    Clk => Clk,
        Con => Con,
        data_out => data_out
       );

-- Clock process definitions
  Clk_process :process
  begin
            Clk <= '0';
            wait for Clk_period/2;
            Clk <= '1';
            wait for Clk_period/2;
  end process;

            -- Stimulus process
  stim_proc0: process
  begin
            --Load the CAR with the input (condition == 1)
            data_in <= "01000001";
            Con <= '1';
            reset <='0';
            --data_out = 01000001
            wait for Clk_period*2;

            --Increment the CAR (condition == 0)
            data_in <= "01000001";
            Con <= '0';
            --data_out = 01000010
            wait;
```

```
  end process;

END;
```

3.4 EXTEND

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY Extend_TestBench IS
END Extend_TestBench;

ARCHITECTURE behavior OF Extend_TestBench IS

   -- Component Declaration for the Unit Under Test (UUT)

   COMPONENT Extend
   PORT(
        DR : IN  std_logic_vector(2 downto 0);
        SB : IN  std_logic_vector(2 downto 0);
        extension : OUT  std_logic_vector(15 downto 0)
       );
   END COMPONENT;


  --Inputs
  signal DR : std_logic_vector(2 downto 0) := (others => '0');
  signal SB : std_logic_vector(2 downto 0) := (others => '0');

     --Outputs
  signal extension : std_logic_vector(15 downto 0);
```

```
BEGIN

    -- Instantiate the Unit Under Test (UUT)
  uut: Extend PORT MAP (
       DR => DR,
       SB => SB,
       extension => extension
      );

  -- Stimulus process
  stim_proc: process
  begin
    -- Test with most significant  bit = 1
              DR <= "101";
              SB <= "010";
              -- Extension = 1111111111101010
    wait for 100 ns;

    -- Test with most significant  bit = 0
              DR <= "011";
              SB <= "011";
              -- Extension = 0000000000011011
    wait;
  end process;

END;
```

3.5 INSTRUCTION REGISTER

```
LIBRARY ieee;
```

```
USE ieee.std_logic_1164.ALL;

ENTITY IR_TestBench IS
END IR_TestBench;

ARCHITECTURE behavior OF IR_TestBench IS

   -- Component Declaration for the Unit Under Test (UUT)

   COMPONENT IR
   PORT(
        data_in : IN  std_logic_vector(15 downto 0);
        IL : IN  std_logic;
                        Clk : IN std_logic;
        Opcode : OUT  std_logic_vector(6 downto 0);
        DR : OUT  std_logic_vector(2 downto 0);
        SA : OUT  std_logic_vector(2 downto 0);
        SB : OUT  std_logic_vector(2 downto 0)
       );
   END COMPONENT;


  --Inputs
  signal data_in : std_logic_vector(15 downto 0) := (others => '0');
  signal IL : std_logic := '0';
  signal Clk : std_logic := '0';

     --Outputs
  signal Opcode : std_logic_vector(6 downto 0);
```

```
  signal DR : std_logic_vector(2 downto 0);
  signal SA : std_logic_vector(2 downto 0);
  signal SB : std_logic_vector(2 downto 0);

  -- Clock period definitions
  constant Clk_period : time := 10 ns;

BEGIN

    -- Instantiate the Unit Under Test (UUT)
  uut: IR PORT MAP (
       data_in => data_in,
       IL => IL,
                    Clk => Clk,
       Opcode => Opcode,
       DR => DR,
       SA => SA,
       SB => SB
      );

-- Clock process definitions
  Clk_process :process
  begin
            Clk <= '0';
            wait for Clk_period/2;
            Clk <= '1';
            wait for Clk_period/2;
  end process;
```

```
    -- Stimulus process
  stim_proc0: process
  begin
                --Test Data being loaded in.
                --Opcode = 0100000, DR = 111, SA = 101, SB = 011.
                --Data in = 0100000 111 101 011,  IL = 1.
                data_in <= "0100000111101011";
                IL <= '1';
                wait for Clk_period*2;

                --Test Data not being loaded in. (Therefore the Outputs are the same as the
last test)
                --Opcode = 0100000, DR = 111, SA = 101, SB = 011.
                --Data in = 0011100 010 111 000, IL = 1.
                data_in <= "0011100010111000";
                IL <= '0';
                wait;

  end process;

END;
```

3.6 MICROPROGRAMMED CONTROL

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY Microprogrammed_control_TestBench IS
END Microprogrammed_control_TestBench;

ARCHITECTURE behavior OF Microprogrammed_control_TestBench IS
```

```
-- Component Declaration for the Unit Under Test (UUT)

COMPONENT Microprogrammed_control
PORT(
    instruction_in : IN  std_logic_vector(15 downto 0);
    reset : IN  std_logic;
    V : IN  std_logic;
    C : IN  std_logic;
    N : IN  std_logic;
    Z : IN  std_logic;
    Clk : IN  std_logic;
    pc_out : OUT  std_logic_vector(15 downto 0);
    DR : OUT  std_logic_vector(2 downto 0);
    SA : OUT  std_logic_vector(2 downto 0);
    SB : OUT  std_logic_vector(2 downto 0);
    TD : OUT  std_logic;
    TA : OUT  std_logic;
    TB : OUT  std_logic;
    MB : OUT  std_logic;
    FS : OUT  std_logic_vector(4 downto 0);
    MD : OUT  std_logic;
    RW : OUT  std_logic;
    MM : OUT  std_logic;
    MW : OUT  std_logic
   );
END COMPONENT;


--Inputs
```

```
  signal instruction_in : std_logic_vector(15 downto 0) := (others => '0');
  signal reset : std_logic := '0';
  signal V : std_logic := '0';
  signal C : std_logic := '0';
  signal N : std_logic := '0';
  signal Z : std_logic := '0';
  signal Clk : std_logic := '0';

    --Outputs
  signal pc_out : std_logic_vector(15 downto 0);
  signal DR : std_logic_vector(2 downto 0);
  signal SA : std_logic_vector(2 downto 0);
  signal SB : std_logic_vector(2 downto 0);
  signal TD : std_logic;
  signal TA : std_logic;
  signal TB : std_logic;
  signal MB : std_logic;
  signal FS : std_logic_vector(4 downto 0);
  signal MD : std_logic;
  signal RW : std_logic;
  signal MM : std_logic;
  signal MW : std_logic;

  -- Clock period definitions
  constant Clk_period : time := 10 ns;

BEGIN

     -- Instantiate the Unit Under Test (UUT)
```

```
uut: Microprogrammed_control PORT MAP (
      instruction_in => instruction_in,
      reset => reset,
      V => V,
      C => C,
      N => N,
      Z => Z,
      Clk => Clk,
      pc_out => pc_out,
      DR => DR,
      SA => SA,
      SB => SB,
      TD => TD,
      TA => TA,
      TB => TB,
      MB => MB,
      FS => FS,
      MD => MD,
      RW => RW,
      MM => MM,
      MW => MW
    );

-- Clock process definitions
Clk_process :process
begin
            Clk <= '0';
            wait for Clk_period/2;
            Clk <= '1';
```

```
        wait for Clk_period/2;
end process;



stim_proc: process
begin
    reset <= '1';
    wait for 20 ns;

   -- Testing instruction = 1101001000110111
   reset <= '0';
   instruction_in <= x"D237";
   -- output should be
   -- pc 0000000000000001
   -- dr 000
   -- sa 110
   -- sb 111
   -- td 0
   -- ta 0
   -- tb 0
   -- mb 0
   -- fs 00000
   -- md 0
   -- rw 0
   -- mm 0
   -- mw 0

  wait;
```

```
  end process;

END;
```

3.7 PROGRAM COUNTER

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY PC_TestBench IS
END PC_TestBench;

ARCHITECTURE behavior OF PC_TestBench IS

   -- Component Declaration for the Unit Under Test (UUT)

   COMPONENT PC
   PORT(
       Extend : IN  std_logic_vector(15 downto 0);
                        reset : IN std_logic;
                        Clk : IN std_logic;
       PL : IN  std_logic;
       PI : IN  std_logic;
       PC_out : OUT  std_logic_vector(15 downto 0)
      );
   END COMPONENT;


  --Inputs
  signal Extend : std_logic_vector(15 downto 0) := (others => '0');
  signal PL : std_logic := '0';
```

```
  signal PI : std_logic := '0';
    signal reset : std_logic := '0';
    signal Clk : std_logic := '0';

    --Outputs
  signal PC_out : std_logic_vector(15 downto 0);


  -- Clock period definitions
  constant Clk_period : time := 10 ns;

BEGIN

    -- Instantiate the Unit Under Test (UUT)
  uut: PC PORT MAP (
       Extend => Extend,
                    reset => reset,
                    Clk => Clk,
       PL => PL,
       PI => PI,
       PC_out => PC_out
      );


-- Clock process definitions
  Clk_process :process
  begin
            Clk <= '0';
            wait for Clk_period/2;
```

```
		Clk <= '1';
		wait for Clk_period/2;
end process;

  -- Stimulus process
stim_proc0: process
begin
		--Test reset
		reset<='1';
		Extend <= x"0005";
		PI <= '0';
		PL <= '0';
		-- PC_out = 0x0000
		wait for Clk_period*2;


		--Test with PI = '0' and PL ='0'
		reset<='0';
		Extend <= x"0005";
		PI <= '0';
		PL <= '0';
		-- PC_out = 0x0000
		wait for Clk_period*2;

		--Test with PI = '1' and PL = '0'
		Extend <= x"0005";
		PI <= '1';
		PL <= '0';
		-- PC_out = 0x0001
```

```
        wait for Clk_period;

        --Test with PI = '0', PL = '1' and offset as 0x101
        Extend <= x"0005";
        PI <= '0';
        PL <= '1';
        -- PC_out = 0x06
        wait;

  end process;

END;
```

3.8 BARREL SHIFTER

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY barrel_shifter_16bit_TestBench IS
END barrel_shifter_16bit_TestBench;

ARCHITECTURE behavior OF barrel_shifter_16bit_TestBench IS

   -- Component Declaration for the Unit Under Test (UUT)

   COMPONENT barrel_shifter_16bit
   PORT(
      B : IN  std_logic_vector(15 downto 0);
      S0 : IN  std_logic;
      S1 : IN  std_logic;
      S2 : IN  std_logic;
```

```
        S3 : IN  std_logic;
        F : OUT  std_logic_vector(15 downto 0)
       );
  END COMPONENT;

  --Inputs
  signal b : std_logic_vector(15 downto 0) := (others => '0');
  signal S0 : std_logic := '0';
  signal S1 : std_logic := '0';
  signal S2 : std_logic := '0';
  signal S3 : std_logic := '0';

    --Outputs
  signal F : std_logic_vector(15 downto 0);
  -- No clocks detected in port list. Replace <clock> below with
  -- appropriate port name

BEGIN

    -- Instantiate the Unit Under Test (UUT)
  uut: barrel_shifter_16bit PORT MAP (
        B => B,
        S0 => S0,
        S1 => S1,
        S2 => S2,
        S3 => S3,
        F => F
       );
```

```
-- Stimulus process
stim_proc0: process
begin
  wait for 25 ns;
        S0 <= not S0;
end process;

stim_proc1: process
begin
  wait for 50 ns;
        S1 <= not S1;
end process;

stim_proc2: process
begin
  wait for 100 ns;
        S2 <= not S2;
end process;


stim_proc3: process
begin
  wait for 200 ns;
        S3 <= not S3;
end process;

   stim_proc4: process
begin
        B <= x"8000";
```

```
    wait for 400 ns;
            B <= x"C000";
    wait for 400 ns;
  end process;

END;
```

3.9 CONTROL MEMORY

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY control_memory_TestBench IS
END control_memory_TestBench;

ARCHITECTURE behavior OF control_memory_TestBench IS

   -- Component Declaration for the Unit Under Test (UUT)

   COMPONENT control_memory
   PORT(
      MW : OUT  std_logic;
      MM : OUT  std_logic;
      RW : OUT  std_logic;
      MD : OUT  std_logic;
      FS : OUT  std_logic_vector(4 downto 0);
      MB : OUT  std_logic;
      TB : OUT  std_logic;
      TA : OUT  std_logic;
      TD : OUT  std_logic;
      PL : OUT  std_logic;
```

```
     PI : OUT  std_logic;
     IL : OUT  std_logic;
     MC : OUT  std_logic;
     MS : OUT  std_logic_vector(2 downto 0);
     NA : OUT  std_logic_vector(7 downto 0);
     IN_CAR : IN  std_logic_vector(7 downto 0)
    );
 END COMPONENT;


--Inputs
signal IN_CAR : std_logic_vector(7 downto 0) := (others => '0');

   --Outputs
signal MW : std_logic;
signal MM : std_logic;
signal RW : std_logic;
signal MD : std_logic;
signal FS : std_logic_vector(4 downto 0);
signal MB : std_logic;
signal TB : std_logic;
signal TA : std_logic;
signal TD : std_logic;
signal PL : std_logic;
signal PI : std_logic;
signal IL : std_logic;
signal MC : std_logic;
signal MS : std_logic_vector(2 downto 0);
signal NA : std_logic_vector(7 downto 0);
```

```
BEGIN


    -- Instantiate the Unit Under Test (UUT)
  uut: control_memory PORT MAP (
        MW => MW,
        MM => MM,
        RW => RW,
        MD => MD,
        FS => FS,
        MB => MB,
        TB => TB,
        TA => TA,
        TD => TD,
        PL => PL,
        PI => PI,
        IL => IL,
        MC => MC,
        MS => MS,
        NA => NA,
        IN_CAR => IN_CAR
      );


    -- Stimulus process
  stim_proc0: process
  begin
            --Test that accessing IF address gives out the IF code:
            -- Next Address   MS          MC IL   PI      PL      TD TA   TB      MB      FS
MDRW      MM     MW
```

```
            --              C1                  001     0       1       1       0       0
    0       0       0       00000  0       0       1       0

            IN_CAR <= x"C0";

            wait for 100 ns;


            --Test that accessing B address gives out the B code:

            -- Next Address   MS          MC IL   PI      PL      TD TA  TB      MB      FS
MDRW    MM      MW

            --              C0                  001     0       0       0       1       0
    0       0       0       00000  0       0       0       0

            IN_CAR <= x"30";

            wait;


  end process;


END;
```

3.10 ARITHMETIC CIRCUIT

```
LIBRARY ieee;

USE ieee.std_logic_1164.ALL;


ENTITY arithmetic_circuit_16bit_TestBench IS

END arithmetic_circuit_16bit_TestBench;


ARCHITECTURE behavior OF arithmetic_circuit_16bit_TestBench IS


   -- Component Declaration for the Unit Under Test (UUT)


   COMPONENT arithmetic_circuit_16bit

   PORT(

       Cin : IN  std_logic;
```

```
        S0 : IN  std_logic;
        S1 : IN  std_logic;
        A : IN  std_logic_vector(15 downto 0);
        B : IN  std_logic_vector(15 downto 0);
        G : OUT  std_logic_vector(15 downto 0);
        Cout : OUT  std_logic;
                        Vout : OUT std_logic
       );
    END COMPONENT;


   --Inputs
   signal Cin : std_logic := '0';
   signal S0 : std_logic := '0';
   signal S1 : std_logic := '0';
   signal A : std_logic_vector(15 downto 0) := (others => '0');
   signal B : std_logic_vector(15 downto 0) := (others => '0');

      --Outputs
   signal G : std_logic_vector(15 downto 0);
   signal Cout : std_logic;
   signal Vout : std_logic;
   -- No clocks detected in port list. Replace <clock> below with
   -- appropriate port name

BEGIN

      -- Instantiate the Unit Under Test (UUT)
   uut: arithmetic_circuit_16bit PORT MAP (
```

```
      Cin => Cin,
      S0 => S0,
      S1 => S1,
      A => A,
      B => B,
      G => G,
      Cout => Cout,
                  Vout => Vout
    );

  -- Stimulus process
stim_proc0: process
begin
            wait for 100 ns;
            Cin <= '0';
            S0 <= '0';
            S1 <= '0';
            A <= x"000A";
            B <= x"0000";
            --Should place 0xA000 into G

            wait for 100 ns;
            S0 <= '0';
            S1 <= '0';
            Cin <= '1';
            A <= x"000A";
            B <= x"0000";
            --Should place 0xA000 + 1 into G
```

```
wait for 100 ns;
S0 <= '0';
S1 <= '1';
Cin <= '0';
A <= x"000A";
B <= x"FFF1";
--Should place 0xA000 + ~(0xFFF1) into G

wait for 100 ns;
S0 <= '0';
S1 <= '1';
Cin <= '1';
A <= x"000A";
B <= x"FFF1";
--Should place 0xA000 + ~(0xFFF1) + 1 into G

wait for 100 ns;
S0 <= '1';
S1 <= '0';
Cin <= '0';
A <= x"000A";
B <= x"00A0";
--Should place 0xA000 + 0x00A0 into G

wait for 100 ns;
S0 <= '1';
S1 <= '0';
Cin <= '1';
A <= x"000A";
```

```
        B <= x"00A0";
        --Should place 0xA000 + 0x00A0 + 1 into G

        wait for 100 ns;
        S0 <= '1';
        S1 <= '1';
        Cin <= '0';
        A <= x"000A";
        B <= x"FFFF";
        --Should place 0xA000 - 1 into G

        wait for 100 ns;
        S0 <= '1';
        S1 <= '1';
        Cin <= '1';
        A <= x"000A";
        B <= x"FFFF";
        --Should place 0xA000 G

    end process;
END;
```

3.11 FUNCTIONAL UNIT

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY functional_unit_16bit_TestBench IS
END functional_unit_16bit_TestBench;

ARCHITECTURE behavior OF functional_unit_16bit_TestBench IS
```

```
-- Component Declaration for the Unit Under Test (UUT)

COMPONENT functional_unit_16bit
PORT(
    FSS : IN  std_logic_vector(4 downto 0);
    A : IN  std_logic_vector(15 downto 0);
    B : IN  std_logic_vector(15 downto 0);
    F : OUT  std_logic_vector(15 downto 0);
    V : OUT  std_logic;
    C : OUT  std_logic;
    N : OUT  std_logic;
    Z : OUT  std_logic
   );
END COMPONENT;


--Inputs
signal FSS : std_logic_vector(4 downto 0) := (others => '0');
signal A : std_logic_vector(15 downto 0) := (others => '0');
signal B : std_logic_vector(15 downto 0) := (others => '0');

  --Outputs
signal F : std_logic_vector(15 downto 0);
signal V : std_logic;
signal C : std_logic;
signal N : std_logic;
signal Z : std_logic;
-- No clocks detected in port list. Replace <clock> below with
```

```
  -- appropriate port name

BEGIN

    -- Instantiate the Unit Under Test (UUT)
  uut: functional_unit_16bit PORT MAP (
        FSS => FSS,
        A => A,
        B => B,
        F => F,
        V => V,
        C => C,
        N => N,
        Z => Z
      );


    --CYCLE THROUGH INPUTS TO TEST
    --   EACH OPERATION
  stim_proc1: process
  begin
    wait for 50 ns;
                FSS(0) <= not FSS(0);
  end process;

  stim_proc2: process
  begin
    wait for 100 ns;
                FSS(1) <= not FSS(1);
```

```
  end process;

  stim_proc3: process
  begin
    wait for 200 ns;
              FSS(2) <= not FSS(2);
  end process;

  stim_proc4: process
  begin
    wait for 400 ns;
              FSS(3) <= not FSS(3);
  end process;

  stim_proc5: process
  begin
    wait for 800 ns;
              FSS(4) <= not FSS(4);
  end process;

     stim_proc6: process
  begin
              A <= x"00FA";
              B <= x"002A";
    wait;
  end process;


END;
```

3.12 MEMORY

```
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 memory_512_TestBench IS
END memory_512_TestBench;

ARCHITECTURE behavior OF memory_512_TestBench IS

   -- Component Declaration for the Unit Under Test (UUT)

   COMPONENT memory_512
   PORT(
        address : IN  std_logic_vector(15 downto 0);
        write_data : IN  std_logic_vector(15 downto 0);
        MemWrite : IN  std_logic;
        read_data : OUT  std_logic_vector(15 downto 0)
       );
   END COMPONENT;


  --Inputs
  signal address : std_logic_vector(15 downto 0) := (others => '0');
  signal write_data : std_logic_vector(15 downto 0) := (others => '0');
  signal MemWrite : std_logic := '0';
```

```
        --Outputs
  signal read_data : std_logic_vector(15 downto 0);

BEGIN

        -- Instantiate the Unit Under Test (UUT)
  uut: memory_512 PORT MAP (
       address => address,
       write_data => write_data,
       MemWrite => MemWrite,
       read_data => read_data
      );

  -- Stimulus process
  stim_proc: process
  begin
    -- hold reset state for 100 ns.
    wait for 100 ns;
                write_data <= x"8003";
                address <= x"0002";
                MemWrite <= '1';
    wait for 100 ns;

    wait;
  end process;

END;
```

3.13 ZERO-FILL

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY zero_fill_TestBench IS
END zero_fill_TestBench;

ARCHITECTURE behavior OF zero_fill_TestBench IS

   -- Component Declaration for the Unit Under Test (UUT)

   COMPONENT zero_fill
   PORT(
        SB : IN  std_logic_vector(2 downto 0);
        constant_out : OUT  std_logic_vector(15 downto 0)
       );
   END COMPONENT;


  --Inputs
  signal SB : std_logic_vector(2 downto 0) := (others => '0');

          --Outputs
  signal constant_out : std_logic_vector(15 downto 0);

BEGIN

          -- Instantiate the Unit Under Test (UUT)
  uut: zero_fill PORT MAP (
```

```
        SB => SB,
        constant_out => constant_out
       );

  -- Stimulus process
  stim_proc: process
  begin
                --Test with the constant as 5
                SB <= "101";
    wait for 100 ns;

                --Test with the constant as 0
                SB <= "000";
    wait for 100 ns;

                --Test with the constant as 2
                SB <= "010";
                wait;
    end process;

END;
```

3.14 RIPPLE ADDER

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY ripple_adder_16bit_TestBench IS
END ripple_adder_16bit_TestBench;

ARCHITECTURE behavior OF ripple_adder_16bit_TestBench IS
```

```
 -- Component Declaration for the Unit Under Test (UUT)

 COMPONENT ripple_adder_16bit
 PORT(
     A : IN  std_logic_vector(15 downto 0);
     B : IN  std_logic_vector(15 downto 0);
     Cin : IN  std_logic;
     Cout : OUT  std_logic;
     Vout : OUT  std_logic;
     S : OUT  std_logic_vector(15 downto 0)
    );
 END COMPONENT;


 --Inputs
 signal A : std_logic_vector(15 downto 0) := (others => '0');
 signal B : std_logic_vector(15 downto 0) := (others => '0');
 signal Cin : std_logic := '0';

        --Outputs
 signal Cout : std_logic;
 signal Vout : std_logic;
 signal S : std_logic_vector(15 downto 0);
 -- No clocks detected in port list. Replace <clock> below with
 -- appropriate port name

BEGIN
```

```
        -- Instantiate the Unit Under Test (UUT)
uut: ripple_adder_16bit PORT MAP (
      A => A,
      B => B,
      Cin => Cin,
      Cout => Cout,
      Vout => Vout,
      S => S
    );

-- Stimulus process
stim_proc: process
begin
  -- hold reset state for 100 ns.
  wait for 100 ns;

                -- 0 + 0 = 0
                A <= x"0000";
                B <= x"0000";
                Cin <= '0';
  wait for 100 ns;

                -- 2 + 2 = 4
                A <= x"0002";
                B <= x"0002";
                Cin <= '0';
  wait for 100 ns;

                -- 0 + 0 + 1 = 1
```

```
                A <= x"0000";
                B <= x"0000";
                Cin <= '1';
    wait for 100 ns;

                -- 65535 + 65535 = 0 (+ carry)
                A <= x"FFFF";
                B <= x"FFFF";
                Cin <= '0';
    wait for 100 ns;

    end process;
END;
```

3.15 REGISTER FILE

```
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 new_reg_file_9x16bit_TestBench IS
END new_reg_file_9x16bit_TestBench;

ARCHITECTURE behavior OF new_reg_file_9x16bit_TestBench IS

   -- Component Declaration for the Unit Under Test (UUT)

   COMPONENT new_reg_file_9x16bit
```

```
PORT(
    des_sel : IN  std_logic_vector(2 downto 0);
    a_sel : IN  std_logic_vector(2 downto 0);
    b_sel : IN  std_logic_vector(2 downto 0);
                    TD : IN std_logic;
                    TA : IN std_logic;
                    TB : IN std_logic;
    load_enable : IN  std_logic;
    reset : IN  std_logic;
    Clk : IN  std_logic;
    input_data : IN  std_logic_vector(15 downto 0);
    a_data : OUT  std_logic_vector(15 downto 0);
    b_data : OUT  std_logic_vector(15 downto 0)
   );
END COMPONENT;


--Inputs
signal des_sel : std_logic_vector(2 downto 0) := (others => '0');
signal a_sel : std_logic_vector(2 downto 0) := (others => '0');
signal b_sel : std_logic_vector(2 downto 0) := (others => '0');
signal load_enable : std_logic := '0';
signal reset : std_logic := '0';
signal TD : std_logic := '0';
signal TA : std_logic := '0';
signal TB : std_logic := '0';
signal Clk : std_logic := '0';
signal input_data : std_logic_vector(15 downto 0) := (others => '0');
```

```
        --Outputs
  signal a_data : std_logic_vector(15 downto 0);
  signal b_data : std_logic_vector(15 downto 0);

  -- Clock period definitions
  constant Clk_period : time := 10 ns;

BEGIN

        -- Instantiate the Unit Under Test (UUT)
  uut: new_reg_file_9x16bit PORT MAP (
       des_sel => des_sel,
       a_sel => a_sel,
       b_sel => b_sel,
                          TD => TD,
                          TA => TA,
                          TB => TB,
       load_enable => load_enable,
                          reset => reset,
       Clk => Clk,
       input_data => input_data,
       a_data => a_data,
       b_data => b_data
      );

  -- Clock process definitions
  Clk_process :process
  begin
                  Clk <= '0';
```

```
                wait for Clk_period/2;
                Clk <= '1';
                wait for Clk_period/2;
end process;



-- Stimulus process
stim_proc: process
begin
  -- hold reset state for 100 ns.
                reset <= '1';
  wait for 100 ns;
                load_enable <= '1';

                --Load 0x0 into reg0
                input_data <= x"0000";
                des_sel(0) <= '0';
                des_sel(1) <= '0';
                des_sel(2) <= '0';
                TD <= '0';
  wait for Clk_period*10;

                --Load 0x1 into reg1
                input_data <= x"0001";
                des_sel(0) <= '1';
                des_sel(1) <= '0';
                des_sel(2) <= '0';
                TD <= '0';
```

```
wait for Clk_period*10;

        --Load 0x2 into reg2
        input_data <= x"0002";
        des_sel(0) <= '0';
        des_sel(1) <= '1';
        des_sel(2) <= '0';
        TD <= '0';
wait for Clk_period*10;

        --Load 0x3 into reg3
        input_data <= x"0003";
        des_sel(0) <= '1';
        des_sel(1) <= '1';
        des_sel(2) <= '0';
        TD <= '0';
wait for Clk_period*10;

        --Load 0x4 into reg4
        input_data <= x"0004";
        des_sel(0) <= '0';
        des_sel(1) <= '0';
        des_sel(2) <= '1';
        TD <= '0';
wait for Clk_period*10;

        --Load 0x5 into reg5
        input_data <= x"0005";
        des_sel(0) <= '1';
```

```
                des_sel(1) <= '0';
                des_sel(2) <= '1';
                TD <= '0';
wait for Clk_period*10;

                --Load 0x6 into reg6
                input_data <= x"0006";
                des_sel(0) <= '0';
                des_sel(1) <= '1';
                des_sel(2) <= '1';
                TD <= '0';
wait for Clk_period*10;

                --Load 0x7 into reg7
                input_data <= x"0007";
                des_sel(0) <= '1';
                des_sel(1) <= '1';
                des_sel(2) <= '1';
                TD <= '0';
wait for Clk_period*10;

                --Load 0x8 into reg8
                input_data <= x"0008";
                des_sel(0) <= '0';
                des_sel(1) <= '0';
                des_sel(2) <= '0';
                TD <= '1';
wait for Clk_period*10;
```

```
                --Load reg3's value into a_data
                a_sel(0) <= '1';
                a_sel(1) <= '1';
                a_sel(2) <= '0';
                TA <= '0';


                --Load reg5's value into b_data
                b_sel(0) <= '1';
                b_sel(1) <= '0';
                b_sel(2) <= '1';
                TB <= '0';
                wait;
    end process;
END;
```

3.16 LOGIC CIRCUIT

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY logic_circuit_16bit_TestBench IS
END logic_circuit_16bit_TestBench;

ARCHITECTURE behavior OF logic_circuit_16bit_TestBench IS

   -- Component Declaration for the Unit Under Test (UUT)

   COMPONENT logic_circuit_16bit
   PORT(
      A : IN  std_logic_vector(15 downto 0);
```

```
        B : IN  std_logic_vector(15 downto 0);
        s0 : IN  std_logic;
        s1 : IN  std_logic;
        Y : OUT  std_logic_vector(15 downto 0)
       );
    END COMPONENT;


   --Inputs
   signal A : std_logic_vector(15 downto 0) := (others => '0');
   signal B : std_logic_vector(15 downto 0) := (others => '0');
   signal s0 : std_logic := '0';
   signal s1 : std_logic := '0';

          --Outputs
   signal Y : std_logic_vector(15 downto 0);
   -- No clocks detected in port list. Replace <clock> below with
   -- appropriate port name

BEGIN

          -- Instantiate the Unit Under Test (UUT)
   uut: logic_circuit_16bit PORT MAP (
        A => A,
        B => B,
        s0 => s0,
        s1 => s1,
        Y => Y
       );
```

```
  -- Stimulus process
  stim_proc0: process
  begin
    wait for 25 ns;
                S0 <= not S0;
  end process;

  stim_proc1: process
  begin
    wait for 50 ns;
                S1 <= not S1;
  end process;

        stim_proc2: process
  begin
                A <= x"00A0";
                B <= x"002A";
    wait for 100 ns;
  end process;
END;
```

3.17 DATAPATH

```
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 last_data_path_TestBench IS
END last_data_path_TestBench;

ARCHITECTURE behavior OF last_data_path_TestBench IS

   -- Component Declaration for the Unit Under Test (UUT)

   COMPONENT last_data_path
   PORT(
      PC : IN  std_logic_vector(15 downto 0);
      data_in : IN  std_logic_vector(15 downto 0);
      DR : IN  std_logic_vector(2 downto 0);
      SA : IN  std_logic_vector(2 downto 0);
      SB : IN  std_logic_vector(2 downto 0);
      TD : IN  std_logic;
      TA : IN  std_logic;
      TB : IN  std_logic;
      Clk : IN  std_logic;
                           reset : out std_logic;
      FS : IN  std_logic_vector(4 downto 0);
      RW : IN  std_logic;
      MB : IN  std_logic;
      MM : IN  std_logic;
      MD : IN  std_logic;
      data_out : OUT  std_logic_vector(15 downto 0);
      address_out : OUT  std_logic_vector(15 downto 0);
      V : OUT  std_logic;
      C : OUT  std_logic;
      N : OUT  std_logic;
```

```
     Z : OUT  std_logic
    );
 END COMPONENT;


--Inputs
signal PC : std_logic_vector(15 downto 0) := (others => '0');
signal data_in : std_logic_vector(15 downto 0) := (others => '0');
signal DR : std_logic_vector(2 downto 0) := (others => '0');
signal SA : std_logic_vector(2 downto 0) := (others => '0');
signal SB : std_logic_vector(2 downto 0) := (others => '0');
signal TD : std_logic := '0';
signal TA : std_logic := '0';
signal TB : std_logic := '0';
signal Clk : std_logic := '0';
signal reset : std_logic := '0';
signal FS : std_logic_vector(4 downto 0) := (others => '0');
signal RW : std_logic := '0';
signal MB : std_logic := '0';
signal MM : std_logic := '0';
signal MD : std_logic := '0';

        --Outputs
signal data_out : std_logic_vector(15 downto 0);
signal address_out : std_logic_vector(15 downto 0);
signal V : std_logic;
signal C : std_logic;
signal N : std_logic;
signal Z : std_logic;
```

```
  -- Clock period definitions
  constant Clk_period : time := 10 ns;

BEGIN

        -- Instantiate the Unit Under Test (UUT)
  uut: last_data_path PORT MAP (
       PC => PC,
       data_in => data_in,
       DR => DR,
       SA => SA,
       SB => SB,
       TD => TD,
       TA => TA,
       TB => TB,
       Clk => Clk,
       reset => reset,
       FS => FS,
       RW => RW,
       MB => MB,
       MM => MM,
       MD => MD,
       data_out => data_out,
       address_out => address_out,
       V => V,
       C => C,
       N => N,
       Z => Z
```

```
        );

    -- Clock process definitions
    Clk_process :process
    begin
                    Clk <= '0';
                    wait for Clk_period/2;
                    Clk <= '1';
                    wait for Clk_period/2;
    end process;


    -- Stimulus process
    stim_proc: process
    begin
      -- hold reset state for 100 ns.
      wait for 100 ns;

      wait for Clk_period*10;

      -- insert stimulus here

      wait;
    end process;

END;
```

3.18 DECODER3_8

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
```

```
ENTITY decoder_3to8_TestBench IS
END decoder_3to8_TestBench;

ARCHITECTURE behavior OF decoder_3to8_TestBench IS

   -- Component Declaration for the Unit Under Test (UUT)

   COMPONENT decoder_3to8
   PORT(
       A0 : IN  std_logic;
       A1 : IN  std_logic;
       A2 : IN  std_logic;
       Q0 : OUT  std_logic;
       Q1 : OUT  std_logic;
       Q2 : OUT  std_logic;
       Q3 : OUT  std_logic;
       Q4 : OUT  std_logic;
       Q5 : OUT  std_logic;
       Q6 : OUT  std_logic;
       Q7 : OUT  std_logic
      );
   END COMPONENT;

  --Inputs
  signal A0 : std_logic := '0';
  signal A1 : std_logic := '0';
  signal A2 : std_logic := '0';
```

```
        --Outputs
  signal Q0 : std_logic;
  signal Q1 : std_logic;
  signal Q2 : std_logic;
  signal Q3 : std_logic;
  signal Q4 : std_logic;
  signal Q5 : std_logic;
  signal Q6 : std_logic;
  signal Q7 : std_logic;

BEGIN

        -- Instantiate the Unit Under Test (UUT)
  uut: decoder_3to8 PORT MAP (
       A0 => A0,
       A1 => A1,
       A2 => A2,
       Q0 => Q0,
       Q1 => Q1,
       Q2 => Q2,
       Q3 => Q3,
       Q4 => Q4,
       Q5 => Q5,
       Q6 => Q6,
       Q7 => Q7
      );

  -- Stimulus process
  stim_proc0: process
```

```
  begin
    wait for 100 ns;
                A2 <= not A2;
  end process;

  stim_proc1: process
  begin
    wait for 50 ns;
                A1 <= not A1;
  end process;

  stim_proc2: process
  begin
    wait for 25 ns;
                A0 <= not A0;
  end process;
END;
```

3.19 DECODER4_16

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY decoder_4to16_TestBench IS
END decoder_4to16_TestBench;

ARCHITECTURE behavior OF decoder_4to16_TestBench IS

   -- Component Declaration for the Unit Under Test (UUT)

   COMPONENT decoder_4to16
```

```
 PORT(
      A0 : IN  std_logic;
      A1 : IN  std_logic;
      A2 : IN  std_logic;
      A3 : IN  std_logic;
      Q0 : OUT  std_logic;
      Q1 : OUT  std_logic;
      Q2 : OUT  std_logic;
      Q3 : OUT  std_logic;
      Q4 : OUT  std_logic;
      Q5 : OUT  std_logic;
      Q6 : OUT  std_logic;
      Q7 : OUT  std_logic;
      Q8 : OUT  std_logic;
      Q9 : OUT  std_logic;
      Q10 : OUT  std_logic;
      Q11 : OUT  std_logic;
      Q12 : OUT  std_logic;
      Q13 : OUT  std_logic;
      Q14 : OUT  std_logic;
      Q15 : OUT  std_logic
     );
 END COMPONENT;


--Inputs
signal A0 : std_logic := '0';
signal A1 : std_logic := '0';
signal A2 : std_logic := '0';
```

```
  signal A3 : std_logic := '0';

        --Outputs
  signal Q0 : std_logic;
  signal Q1 : std_logic;
  signal Q2 : std_logic;
  signal Q3 : std_logic;
  signal Q4 : std_logic;
  signal Q5 : std_logic;
  signal Q6 : std_logic;
  signal Q7 : std_logic;
  signal Q8 : std_logic;
  signal Q9 : std_logic;
  signal Q10 : std_logic;
  signal Q11 : std_logic;
  signal Q12 : std_logic;
  signal Q13 : std_logic;
  signal Q14 : std_logic;
  signal Q15 : std_logic;

BEGIN

        -- Instantiate the Unit Under Test (UUT)
  uut: decoder_4to16 PORT MAP (
       A0 => A0,
       A1 => A1,
       A2 => A2,
       A3 => A3,
       Q0 => Q0,
```

```
      Q1 => Q1,
      Q2 => Q2,
      Q3 => Q3,
      Q4 => Q4,
      Q5 => Q5,
      Q6 => Q6,
      Q7 => Q7,
      Q8 => Q8,
      Q9 => Q9,
      Q10 => Q10,
      Q11 => Q11,
      Q12 => Q12,
      Q13 => Q13,
      Q14 => Q14,
      Q15 => Q15
    );

-- Stimulus process

stim_proc3: process
begin
  wait for 200 ns;
                A3 <= not A3;
end process;

stim_proc0: process
begin
  wait for 100 ns;
                A2 <= not A2;
```

```
  end process;

  stim_proc1: process
  begin
    wait for 50 ns;
                A1 <= not A1;
  end process;

  stim_proc2: process
  begin
    wait for 25 ns;
                A0 <= not A0;
  end process;

END;
```

3.20 MUX16_16BIT

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY mux16_16bit_TestBench IS
END mux16_16bit_TestBench;

ARCHITECTURE behavior OF mux16_16bit_TestBench IS

   -- Component Declaration for the Unit Under Test (UUT)

   COMPONENT mux16_16bit
   PORT(
       S0 : IN  std_logic;
```

```
    S1 : IN  std_logic;
    S2 : IN  std_logic;
    S3 : IN  std_logic;
    In0 : IN  std_logic_vector(15 downto 0);
    In1 : IN  std_logic_vector(15 downto 0);
    In2 : IN  std_logic_vector(15 downto 0);
    In3 : IN  std_logic_vector(15 downto 0);
    In4 : IN  std_logic_vector(15 downto 0);
    In5 : IN  std_logic_vector(15 downto 0);
    In6 : IN  std_logic_vector(15 downto 0);
    In7 : IN  std_logic_vector(15 downto 0);
    In8 : IN  std_logic_vector(15 downto 0);
    In9 : IN  std_logic_vector(15 downto 0);
    In10 : IN  std_logic_vector(15 downto 0);
    In11 : IN  std_logic_vector(15 downto 0);
    In12 : IN  std_logic_vector(15 downto 0);
    In13 : IN  std_logic_vector(15 downto 0);
    In14 : IN  std_logic_vector(15 downto 0);
    In15 : IN  std_logic_vector(15 downto 0);
    Z : OUT  std_logic_vector(15 downto 0)
   );
 END COMPONENT;


--Inputs
signal S0 : std_logic := '0';
signal S1 : std_logic := '0';
signal S2 : std_logic := '0';
signal S3 : std_logic := '0';
```

```
  signal In0 : std_logic_vector(15 downto 0) := (others => '0');
  signal In1 : std_logic_vector(15 downto 0) := (others => '0');
  signal In2 : std_logic_vector(15 downto 0) := (others => '0');
  signal In3 : std_logic_vector(15 downto 0) := (others => '0');
  signal In4 : std_logic_vector(15 downto 0) := (others => '0');
  signal In5 : std_logic_vector(15 downto 0) := (others => '0');
  signal In6 : std_logic_vector(15 downto 0) := (others => '0');
  signal In7 : std_logic_vector(15 downto 0) := (others => '0');
  signal In8 : std_logic_vector(15 downto 0) := (others => '0');
  signal In9 : std_logic_vector(15 downto 0) := (others => '0');
  signal In10 : std_logic_vector(15 downto 0) := (others => '0');
  signal In11 : std_logic_vector(15 downto 0) := (others => '0');
  signal In12 : std_logic_vector(15 downto 0) := (others => '0');
  signal In13 : std_logic_vector(15 downto 0) := (others => '0');
  signal In14 : std_logic_vector(15 downto 0) := (others => '0');
  signal In15 : std_logic_vector(15 downto 0) := (others => '0');

        --Outputs
  signal Z : std_logic_vector(15 downto 0);

BEGIN

        -- Instantiate the Unit Under Test (UUT)
  uut: mux16_16bit PORT MAP (
       S0 => S0,
       S1 => S1,
       S2 => S2,
       S3 => S3,
       In0 => In0,
```

```
      In1 => In1,
      In2 => In2,
      In3 => In3,
      In4 => In4,
      In5 => In5,
      In6 => In6,
      In7 => In7,
      In8 => In8,
      In9 => In9,
      In10 => In10,
      In11 => In11,
      In12 => In12,
      In13 => In13,
      In14 => In14,
      In15 => In15,
      Z => Z
    );

  -- Stimulus process
stim_proc4: process
begin
   wait for 200 ns;
                    S3 <= not S3;
end process;

stim_proc0: process
begin
   wait for 100 ns;
                    S2 <= not S2;
```

```
end process;

stim_proc1: process
begin
  wait for 50 ns;
        S1 <= not S1;
end process;

stim_proc2: process
begin
  wait for 25 ns;
        S0 <= not S0;
end process;

    stim_proc3: process
    begin
        In0 <= x"0000";
        In1 <= x"0001";
        In2 <= x"0002";
        In3 <= x"0003";
        In4 <= x"0004";
        In5 <= x"0005";
        In6 <= x"0006";
        In7 <= x"0007";
        In8 <= x"0008";
        In9 <= x"0009";
        In10 <= x"000A";
        In11 <= x"000B";
        In12 <= x"000C";
```

```
        In13 <= x"000D";
        In14 <= x"000E";
        In15 <= x"000F";
        wait;
    end process;


END;
```

3.21 MUX16_1BIT

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY mux16_1bit_TestBench IS
END mux16_1bit_TestBench;

ARCHITECTURE behavior OF mux16_1bit_TestBench IS

  -- Component Declaration for the Unit Under Test (UUT)

  COMPONENT mux16_1bit
  PORT(
     S0 : IN  std_logic;
     S1 : IN  std_logic;
     S2 : IN  std_logic;
     S3 : IN  std_logic;
     A : IN  std_logic_vector(15 downto 0);
     F : OUT  std_logic
    );
  END COMPONENT;
```

```
  --Inputs
  signal S0 : std_logic := '0';
  signal S1 : std_logic := '0';
  signal S2 : std_logic := '0';
  signal S3 : std_logic := '0';
  signal A : std_logic_vector(15 downto 0) := (others => '0');

        --Outputs
  signal F : std_logic;
  -- No clocks detected in port list. Replace <clock> below with
  -- appropriate port name


BEGIN

        -- Instantiate the Unit Under Test (UUT)
  uut: mux16_1bit PORT MAP (
       S0 => S0,
       S1 => S1,
       S2 => S2,
       S3 => S3,
       A => A,
       F => F
      );

  -- Stimulus process
  stim_proc0: process
```

```
  begin
    wait for 25 ns;
                    S0 <= not S0;
  end process;

  stim_proc1: process
  begin
    wait for 50 ns;
                    S1 <= not S1;
  end process;

  stim_proc2: process
  begin
    wait for 100 ns;
                    S2 <= not S2;
  end process;
            stim_proc3: process
  begin
    wait for 200 ns;
                    S3 <= not S3;
  end process;
                    stim_proc4: process
  begin
                    A <= x"AAAA";
    wait for 400 ns;
  end process;

END;
```

3.22 MUX2_16BIT

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY mux_2_16bit_TestBench IS
END mux_2_16bit_TestBench;

ARCHITECTURE behavior OF mux_2_16bit_TestBench IS

   -- Component Declaration for the Unit Under Test (UUT)

   COMPONENT mux_2_16bit
   PORT(
        S : IN  std_logic;
        In0 : IN  std_logic_vector(15 downto 0);
        In1 : IN  std_logic_vector(15 downto 0);
        Z : OUT  std_logic_vector(15 downto 0)
       );
   END COMPONENT;

  --Inputs
  signal S : std_logic := '0';
  signal In0 : std_logic_vector(15 downto 0) := (others => '0');
  signal In1 : std_logic_vector(15 downto 0) := (others => '0');

          --Outputs
  signal Z : std_logic_vector(15 downto 0);

BEGIN
```

```
        -- Instantiate the Unit Under Test (UUT)
  uut: mux_2_16bit PORT MAP (
       S => S,
       In0 => In0,
       In1 => In1,
       Z => Z
      );

        -- Stimulus process
  stim_proc0: process
  begin
    In0 <= x"AAAA";
                In1 <= x"FFFF";
                wait for 300 ns;
  end process;

  -- Stimulus process
  stim_proc1: process
  begin
    wait for 100 ns;
                S <= not S;
  end process;
END;
```

3.23 MUX2_8BIT

```
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 mux_2_to_1_8bit_TestBench IS
END mux_2_to_1_8bit_TestBench;

ARCHITECTURE behavior OF mux_2_to_1_8bit_TestBench IS

   -- Component Declaration for the Unit Under Test (UUT)

   COMPONENT mux_2_to_1_8bit
   PORT(
        In0 : IN  std_logic_vector(7 downto 0);
        In1 : IN  std_logic_vector(7 downto 0);
        S : IN  std_logic;
        Z : OUT  std_logic_vector(7 downto 0)
       );
   END COMPONENT;


  --Inputs
  signal In0 : std_logic_vector(7 downto 0) := (others => '0');
  signal In1 : std_logic_vector(7 downto 0) := (others => '0');
  signal S : std_logic := '0';

          --Outputs
  signal Z : std_logic_vector(7 downto 0);

BEGIN
```

```
        -- Instantiate the Unit Under Test (UUT)
  uut: mux_2_to_1_8bit PORT MAP (
       In0 => In0,
       In1 => In1,
       S => S,
       Z => Z
      );

        -- Stimulus process
  stim_proc0: process
  begin
    In0 <= x"AA";
                In1 <= x"FF";
                wait for 300 ns;
  end process;

  -- Stimulus process
  stim_proc1: process
  begin
    wait for 100 ns;
                S <= not S;
  end process;
END;
```

3.24 MUX8_16BIT

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY mux8_16bit_TestBench IS
```

```
END mux8_16bit_TestBench;

ARCHITECTURE behavior OF mux8_16bit_TestBench IS

   -- Component Declaration for the Unit Under Test (UUT)

   COMPONENT mux8_16bit
   PORT(
       S0 : IN  std_logic;
       S1 : IN  std_logic;
       S2 : IN  std_logic;
       In0 : IN  std_logic_vector(15 downto 0);
       In1 : IN  std_logic_vector(15 downto 0);
       In2 : IN  std_logic_vector(15 downto 0);
       In3 : IN  std_logic_vector(15 downto 0);
       In4 : IN  std_logic_vector(15 downto 0);
       In5 : IN  std_logic_vector(15 downto 0);
       In6 : IN  std_logic_vector(15 downto 0);
       In7 : IN  std_logic_vector(15 downto 0);
       Z : OUT  std_logic_vector(15 downto 0)
      );
   END COMPONENT;

  --Inputs
  signal S0 : std_logic := '0';
  signal S1 : std_logic := '0';
  signal S2 : std_logic := '0';
  signal In0 : std_logic_vector(15 downto 0) := (others => '0');
  signal In1 : std_logic_vector(15 downto 0) := (others => '0');
```

```
  signal In2 : std_logic_vector(15 downto 0) := (others => '0');
  signal In3 : std_logic_vector(15 downto 0) := (others => '0');
  signal In4 : std_logic_vector(15 downto 0) := (others => '0');
  signal In5 : std_logic_vector(15 downto 0) := (others => '0');
  signal In6 : std_logic_vector(15 downto 0) := (others => '0');
  signal In7 : std_logic_vector(15 downto 0) := (others => '0');

        --Outputs
  signal Z : std_logic_vector(15 downto 0);

BEGIN

        -- Instantiate the Unit Under Test (UUT)
  uut: mux8_16bit PORT MAP (
       S0 => S0,
       S1 => S1,
       S2 => S2,
       In0 => In0,
       In1 => In1,
       In2 => In2,
       In3 => In3,
       In4 => In4,
       In5 => In5,
       In6 => In6,
       In7 => In7,
       Z => Z
      );

   -- Stimulus process
```

```
stim_proc0: process
begin
  wait for 100 ns;
                S2 <= not S2;
end process;

stim_proc1: process
begin
  wait for 50 ns;
                S1 <= not S1;
end process;

stim_proc2: process
begin
  wait for 25 ns;
                S0 <= not S0;
end process;

        stim_proc3: process
        begin
                In0 <= x"0000";
                In1 <= x"0001";
                In2 <= x"0002";
                In3 <= x"0003";
                In4 <= x"0004";
                In5 <= x"0005";
                In6 <= x"0006";
                In7 <= x"0007";
                wait for 200 ns;
```

```
        end process;

END;
```

3.25 MUX8_1BIT

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY mux_8_to_1_1bit_TestBench IS
END mux_8_to_1_1bit_TestBench;

ARCHITECTURE behavior OF mux_8_to_1_1bit_TestBench IS

   -- Component Declaration for the Unit Under Test (UUT)

   COMPONENT mux_8_to_1_1bit
   PORT(
        A : IN  std_logic_vector(7 downto 0);
        S0 : IN  std_logic;
        S1 : IN  std_logic;
        S2 : IN  std_logic;
        line_out : OUT  std_logic
       );
   END COMPONENT;


  --Inputs
  signal A : std_logic_vector(7 downto 0) := (others => '0');
  signal S0 : std_logic := '0';
  signal S1 : std_logic := '0';
```

```
  signal S2 : std_logic := '0';

        --Outputs
  signal line_out : std_logic;
  -- No clocks detected in port list. Replace <clock> below with
  -- appropriate port name


BEGIN

        -- Instantiate the Unit Under Test (UUT)
  uut: mux_8_to_1_1bit PORT MAP (
       A => A,
       S0 => S0,
       S1 => S1,
       S2 => S2,
       line_out => line_out
      );

   stim_proc0: process
  begin
    wait for 25 ns;
                S0 <= not S0;
  end process;

  stim_proc1: process
  begin
    wait for 50 ns;
                S1 <= not S1;
```

```
  end process;

  stim_proc2: process
  begin
    wait for 100 ns;
                S2 <= not S2;
  end process;

        stim_proc4: process
  begin
                A <= x"AA";
    wait for 400 ns;
  end process;

END;
```

3.26 FULL ADDER

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY full_adder_TestBench IS
END full_adder_TestBench;

ARCHITECTURE behavior OF full_adder_TestBench IS

   -- Component Declaration for the Unit Under Test (UUT)

   COMPONENT full_adder
   PORT(
       A : IN  std_logic;
```

```
        B : IN  std_logic;
        Cin : IN  std_logic;
        S : OUT  std_logic;
        Cout : OUT  std_logic
       );
    END COMPONENT;


   --Inputs
   signal A : std_logic := '0';
   signal B : std_logic := '0';
   signal Cin : std_logic := '0';

           --Outputs
   signal S : std_logic;
   signal Cout : std_logic;

BEGIN

           -- Instantiate the Unit Under Test (UUT)
   uut: full_adder PORT MAP (
         A => A,
         B => B,
         Cin => Cin,
         S => S,
         Cout => Cout
        );

   -- Stimulus process
```

```
  stim_proc0: process
  begin
                    wait for 25 ns;
    Cin <= not Cin;
  end process;

  -- Stimulus process
  stim_proc1: process
  begin
    wait for 50 ns;
    B <= not B;
  end process;

  -- Stimulus process
  stim_proc2: process
  begin
    wait for 100 ns;
    A <= not A;
  end process;

END;
```

3.27 MUXB_Y_16BIT

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY b_to_y_16bit_TestBench IS
END b_to_y_16bit_TestBench;

ARCHITECTURE behavior OF b_to_y_16bit_TestBench IS
```

```
   -- Component Declaration for the Unit Under Test (UUT)

   COMPONENT b_to_y_16bit
   PORT(
        B : IN  std_logic_vector(15 downto 0);
        Y : OUT  std_logic_vector(15 downto 0);
        S0 : IN  std_logic;
        S1 : IN  std_logic
       );
   END COMPONENT;


  --Inputs
  signal B : std_logic_vector(15 downto 0) := (others => '0');
  signal S0 : std_logic := '0';
  signal S1 : std_logic := '0';

            --Outputs
  signal Y : std_logic_vector(15 downto 0);
  -- No clocks detected in port list. Replace <clock> below with
  -- appropriate port name

BEGIN

            -- Instantiate the Unit Under Test (UUT)
  uut: b_to_y_16bit PORT MAP (
          B => B,
          Y => Y,
```

```
        S0 => S0,
        S1 => S1
       );
    -- Stimulus process

  stim_proc0: process
  begin
                wait for 100 ns;
                B <= x"AAAA";
  end process;

  stim_proc1: process
  begin
    wait for 50 ns;
                S1 <= not S1;
  end process;

  stim_proc2: process
  begin
    wait for 25 ns;
                S0 <= not S0;
  end process;

END;
```

3.28 REGISTER

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY reg16_TestBench IS
```

```
END reg16_TestBench;

ARCHITECTURE behavior OF reg16_TestBench IS

   -- Component Declaration for the Unit Under Test (UUT)

   COMPONENT reg16
   PORT(
        D : IN  std_logic_vector(15 downto 0);
        load : IN  std_logic;
        Clk : IN  std_logic;
        Q : OUT  std_logic_vector(15 downto 0)
       );
   END COMPONENT;

  --Inputs
  signal D : std_logic_vector(15 downto 0) := (others => '0');
  signal load : std_logic := '0';
  signal Clk : std_logic := '0';

            --Outputs
  signal Q : std_logic_vector(15 downto 0);

  -- Clock period definitions
  constant Clk_period : time := 10 ns;

BEGIN

            -- Instantiate the Unit Under Test (UUT)
  uut: reg16 PORT MAP (
         D => D,
```

```
      load => load,
      Clk => Clk,
      Q => Q
    );

-- Clock process definitions
Clk_process :process
begin
                Clk <= '0';
                wait for Clk_period/2;
                Clk <= '1';
                wait for Clk_period/2;
end process;

  -- Stimulus process
stim_proc0: process
begin
                wait for Clk_period*2;
                load <= '1';
                D <= x"AAAA";
                --Should load 0xAAAA into D

                wait for Clk_period*2;
                load <= '0';
                D <= x"1111";
                --Should not load 0x1111 into Q

                wait for Clk_period*2;
                load <= '1';
                D <= x"1111";
                --Should load 0x1111 into Q
```

```
   end process;
END;
```