Microcoded Instruction Set Processor

Laboratory 2

CS2022 Computer Architecture

Alexandru Sulea
D Stream
#12315152
8 April 2016

Contents

List of Tables

1 CODE

1.1 Microcoded Instruction Set Processor

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


entity microprogrammed_control is
    Port ( IR_mpc : in STD_LOGIC_VECTOR(15 downto 0);
           status_mpc : in STD_LOGIC_VECTOR(3 downto 0);
           reset_mpc : in STD_LOGIC;
           control_mpc : out STD_LOGIC_VECTOR(17 downto 0);
           PC_mpc : out STD_LOGIC_VECTOR(15 downto 0);
           MC_in_mpc : out STD_LOGIC;
           MC_TD_mpc : out STD_LOGIC;
           MC_TA_mpc : out STD_LOGIC;
           MC_TB_mpc : out STD_LOGIC);
end microprogrammed_control;
architecture Behavioral of microprogrammed_control is

   component mux8_1bit Port (
        in_00 : in STD_LOGIC;
          in_11 : in STD_LOGIC;
          in_C2 : in STD_LOGIC;
          in_V3 : in STD_LOGIC;
          in_Z4 : in STD_LOGIC;
          in_N5 : in STD_LOGIC;
          in_C6 : in STD_LOGIC;
          in_Z7 : in STD_LOGIC;
          in_MS : in STD_LOGIC_VECTOR(2 downto 0);
          MUXS_out : out STD_LOGIC);
   end component;

   component mux2_8bit Port (
        in_0 : in STD_LOGIC_VECTOR(7 downto 0);
          in_1 : in STD_LOGIC_VECTOR(7 downto 0);
          in_MC : in STD_LOGIC;
          MUXC_out : out STD_LOGIC_VECTOR(7 downto 0));
   end component;

   component CAR Port (
        car_in : in STD_LOGIC_VECTOR(7 downto 0);
          muxs_in : in STD_LOGIC;
          res : in STD_LOGIC;
          car_out : out STD_LOGIC_VECTOR(7 downto 0));
   end component;

   component extend Port (
           E_in : in STD_LOGIC_VECTOR(5 downto 0);
          E_out : out STD_LOGIC_VECTOR(15 downto 0));
   end component;
```

```vhdl
    component IR Port (
          IR_in : in STD_LOGIC_VECTOR(15 downto 0);
            IL : 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 PC Port (
          PC_in : in STD_LOGIC_VECTOR(15 downto 0);
            PL : in STD_LOGIC;
            PI : in STD_LOGIC;
            reset : in STD_LOGIC;
            PC_out : out STD_LOGIC_VECTOR(15 downto 0));
    end component;

    component control_memory Port (
          car_in : in STD_LOGIC_VECTOR(7 downto 0);
            NA : out STD_LOGIC_VECTOR(7 downto 0);
            MSout : out STD_LOGIC_VECTOR(2 downto 0);
            MC : out STD_LOGIC;
            IL : out STD_LOGIC;
            PI : out STD_LOGIC;
            PL : out STD_LOGIC;
            TD : out STD_LOGIC;
            TA : out STD_LOGIC;
            TB : out STD_LOGIC;
            MB : out STD_LOGIC;
            FSout : 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;

    signal src_mpc_in : STD_LOGIC_VECTOR(5 downto 0);
    signal src_mpc_out : STD_LOGIC_VECTOR(15 downto 0);
    signal src_control : STD_LOGIC_VECTOR(17 downto 0);
    signal src_opcode, src_car_in, src_car_out, src_na : STD_LOGIC_VECTOR(7 downto 0);
    signal src_ms, src_sa, src_sb, src_dr : STD_LOGIC_VECTOR(2 downto 0);
    signal src_mc, src_il, src_car_sig, src_pl, src_pi : STD_LOGIC;

begin
      reg_mux8_1bit: mux8_1bit PORT MAP (
         in_00 => '0',
            in_11 => '1',
            in_C2 => status_mpc(2),
            in_V3 => status_mpc(3),
            in_Z4 => status_mpc(0),
            in_N5 => status_mpc(1),
            in_C6 => not status_mpc(2),
            in_Z7 => not status_mpc(0),
            in_MS => src_ms,
            MUXS_out => src_car_sig);
```

```
reg_mux2_8bit :mux2_8bit PORT MAP (
      in_0 => src_na,
        in_1 => src_opcode,
        in_MC => src_mc,
        MUXC_out => src_car_out);

reg_CAR: CAR PORT MAP (
      car_in => src_car_out,
        muxs_in => src_car_sig,
        res => reset_mpc,
        car_out => src_car_in);

reg_extend : extend PORT MAP (
         E_in => src_mpc_in,
        E_out => src_mpc_out);

reg_IR : IR PORT MAP (
      IR_in => IR_mpc,
        IL => src_il,
        opcode => src_opcode(6 downto 0),
        DR => src_dr,
        SA => src_sa,
        SB => src_sb);

reg_PC : PC PORT MAP (
      PC_in => src_mpc_out,
        PL => src_pl,
        PI => src_pi,
        reset => reset_mpc,
        PC_out => PC_mpc);

reg_control_memory : control_memory PORT MAP (
      car_in => src_car_in,
        NA => src_na,
        MSout => src_ms,
        MC => src_mc,
        IL => src_il,
        PI => src_pi,
        PL => src_pl,
        TD => MC_TD_mpc,
        TA => MC_TA_mpc,
        TB => MC_TB_mpc,
        MB => src_control(8),
        FSout => src_control(7 downto 3),
        MD => src_control(2),
        RW => src_control(1),
        MM => src_control(0),
        MW => MC_in_mpc);

src_mpc_in(5 downto 3) <= src_dr;
src_mpc_in(2 downto 0) <= src_sb;

src_opcode(7) <= '0';
src_control(17 downto 15) <= src_dr;
src_control(14 downto 12) <= src_sa;
src_control(11 downto 9) <= src_sb;
```

```
   control_mpc <= src_control;

end Behavioral;
```

1.2 Memory M

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

entity memory_m is
Port ( data_in : in STD_LOGIC_VECTOR(15 downto 0);
      mw : in STD_LOGIC;
       address_in : in STD_LOGIC_VECTOR(15 downto 0);
          data_out : out STD_LOGIC_VECTOR(15 downto 0));
end memory_m;

architecture Behavioral of memory_m is
   type mem_array is array(0 to 511) of STD_LOGIC_VECTOR(15 downto 0);


begin
   mem_process : process(data_in,address_in, mw)
   variable data_mem : mem_array := (

   --0
      x"0000", --0
      x"0000", --1
      x"0241", --2
      x"0482", --3
      x"0964", --4
      x"0165", --5
      x"0982", --6
      x"A528", --7
      x"0000", --8
      x"0000", --9
      x"1AAA", --A
      x"11B7", --B
      x"1E54", --C
      x"1D85", --D
      x"1547", --E
      x"1528", --F
      --1
      x"4568", --0
      x"5622", --1
      x"EE52", --2
      x"ED56", --3
      x"AB85", --4
      x"0000", --5
      x"0000", --6
      x"0000", --7
      x"0000", --8
      x"0000", --9
      x"0000", --A
      x"0000", --B
      x"0000", --C
      x"0000", --D
      x"0000", --E
      x"0000", --F
```

```
--2
x"0000", --0
x"0000", --1
x"0000", --2
x"0000", --3
x"0000", --4
x"0000", --5
x"0000", --6
x"0000", --7
x"0000", --8
x"0000", --9
x"0000", --A
x"0000", --B
x"0000", --C
x"0000", --D
x"0000", --E
x"0000", --F
--3
x"0000", --0
x"0000", --1
x"0000", --2
x"0000", --3
x"0000", --4
x"0000", --5
x"0000", --6
x"0000", --7
x"0000", --8
x"0000", --9
x"0000", --A
x"0000", --B
x"0000", --C
x"0000", --D
x"0000", --E
x"0000", --F
--4
x"0000", --0
x"0000", --1
x"0000", --2
x"0000", --3
x"0000", --4
x"0000", --5
x"0000", --6
x"0000", --7
x"0000", --8
x"0000", --9
x"0000", --A
x"0000", --B
x"0000", --C
x"0000", --D
x"0000", --E
x"0000", --F
--5
     x"0000", --0
x"0000", --1
x"0000", --2
x"0000", --3
```

```
  x"0000", --4
  x"0000", --5
  x"0000", --6
  x"0000", --7
  x"0000", --8
  x"0000", --9
  x"0000", --A
  x"0000", --B
  x"0000", --C
  x"0000", --D
  x"0000", --E
  x"0000", --F
  --6
  x"0000", --0
  x"0000", --1
  x"0000", --2
  x"0000", --3
  x"0000", --4
  x"0000", --5
  x"0000", --6
  x"0000", --7
  x"0000", --8
  x"0000", --9
  x"0000", --A
  x"0000", --B
  x"0000", --C
  x"0000", --D
  x"0000", --E
  x"0000", --F

--7
        x"0000", --0
  x"0000", --1
  x"0000", --2
  x"0000", --3
  x"0000", --4
  x"0000", --5
  x"0000", --6
  x"0000", --7
  x"0000", --8
  x"0000", --9
  x"0000", --A
  x"0000", --B
  x"0000", --C
  x"0000", --D
  x"0000", --E
  x"0000", --F
  --8
  x"0000", --0
  x"0000", --1
  x"0000", --2
  x"0000", --3
  x"0000", --4
  x"0000", --5
  x"0000", --6
  x"0000", --7
```

```
x"0000", --8
x"0000", --9
x"0000", --A
x"0000", --B
x"0000", --C
x"0000", --D
x"0000", --E
x"0000", --F


   --9
x"0000", --0
x"0000", --1
x"0000", --2
x"0000", --3
x"0000", --4
x"0000", --5
x"0000", --6
x"0000", --7
x"0000", --8
x"0000", --9
x"0000", --A
x"0000", --B
x"0000", --C
x"0000", --D
x"0000", --E
x"0000", --F
--A
x"0000", --0
x"0000", --1
x"0000", --2
x"0000", --3
x"0000", --4
x"0000", --5
x"0000", --6
x"0000", --7
x"0000", --8
x"0000", --9
x"0000", --A
x"0000", --B
x"0000", --C
x"0000", --D
x"0000", --E
x"0000", --F
--B
x"0000", --0
x"0000", --1
x"0000", --2
x"0000", --3
x"0000", --4
x"0000", --5
x"0000", --6
x"0000", --7
x"0000", --8
x"0000", --9
x"0000", --A
```

```
    x"0000", --B
    x"0000", --C
    x"0000", --D
    x"0000", --E
    x"0000", --F
    --C
         x"0000", --0
    x"0000", --1
    x"0000", --2
    x"0000", --3
    x"0000", --4
    x"0000", --5
    x"0000", --6
    x"0000", --7
    x"0000", --8
    x"0000", --9
    x"0000", --A
    x"0000", --B
    x"0000", --C
    x"0000", --D
    x"0000", --E
    x"0000", --F
    --D
    x"0000", --0
    x"0000", --1
    x"0000", --2
    x"0000", --3
    x"0000", --4
    x"0000", --5
    x"0000", --6
    x"0000", --7
    x"0000", --8
    x"0000", --9
    x"0000", --A
    x"0000", --B
    x"0000", --C
    x"0000", --D
    x"0000", --E
    x"0000", --F

  --E
         x"0000", --0
    x"0000", --1
    x"0000", --2
    x"0000", --3
    x"0000", --4
    x"0000", --5
    x"0000", --6
    x"0000", --7
    x"0000", --8
    x"0000", --9
    x"0000", --A
    x"0000", --B
    x"0000", --C
    x"0000", --D
    x"0000", --E
```

```
x"0000", --F
--F
x"0000", --0
x"0000", --1
x"0000", --2
x"0000", --3
x"0000", --4
x"0000", --5
x"0000", --6
x"0000", --7
x"0000", --8
x"0000", --9
x"0000", --A
x"0000", --B
x"0000", --C
x"0000", --D
x"0000", --E
x"0000", --F



--0
x"0000", --0
x"0000", --1
x"0000", --2
x"0000", --3
x"0000", --4
x"0000", --5
x"0000", --6
x"0000", --7
x"0000", --8
x"0000", --9
x"0000", --A
x"0000", --B
x"0000", --C
x"0000", --D
x"0000", --E
x"0000", --F
--1
x"0000", --0
x"0000", --1
x"0000", --2
x"0000", --3
x"0000", --4
x"0000", --5
x"0000", --6
x"0000", --7
x"0000", --8
x"0000", --9
x"0000", --A
x"0000", --B
x"0000", --C
x"0000", --D
x"0000", --E
x"0000", --F
--2
```

```
x"0000", --0
x"0000", --1
x"0000", --2
x"0000", --3
x"0000", --4
x"0000", --5
x"0000", --6
x"0000", --7
x"0000", --8
x"0000", --9
x"0000", --A
x"0000", --B
x"0000", --C
x"0000", --D
x"0000", --E
x"0000", --F
--3
x"0000", --0
x"0000", --1
x"0000", --2
x"0000", --3
x"0000", --4
x"0000", --5
x"0000", --6
x"0000", --7
x"0000", --8
x"0000", --9
x"0000", --A
x"0000", --B
x"0000", --C
x"0000", --D
x"0000", --E
x"0000", --F
--4
x"0000", --0
x"0000", --1
x"0000", --2
x"0000", --3
x"0000", --4
x"0000", --5
x"0000", --6
x"0000", --7
x"0000", --8
x"0000", --9
x"0000", --A
x"0000", --B
x"0000", --C
x"0000", --D
x"0000", --E
x"0000", --F
--5
     x"0000", --0
x"0000", --1
x"0000", --2
x"0000", --3
x"0000", --4
```

```
    x"0000", --5
    x"0000", --6
    x"0000", --7
    x"0000", --8
    x"0000", --9
    x"0000", --A
    x"0000", --B
    x"0000", --C
    x"0000", --D
    x"0000", --E
    x"0000", --F
    --6
    x"0000", --0
    x"0000", --1
    x"0000", --2
    x"0000", --3
    x"0000", --4
    x"0000", --5
    x"0000", --6
    x"0000", --7
    x"0000", --8
    x"0000", --9
    x"0000", --A
    x"0000", --B
    x"0000", --C
    x"0000", --D
    x"0000", --E
    x"0000", --F

--7
          x"0000", --0
    x"0000", --1
    x"0000", --2
    x"0000", --3
    x"0000", --4
    x"0000", --5
    x"0000", --6
    x"0000", --7
    x"0000", --8
    x"0000", --9
    x"0000", --A
    x"0000", --B
    x"0000", --C
    x"0000", --D
    x"0000", --E
    x"0000", --F
    --8
    x"0000", --0
    x"0000", --1
    x"0000", --2
    x"0000", --3
    x"0000", --4
    x"0000", --5
    x"0000", --6
    x"0000", --7
    x"0000", --8
```

```
x"0000", --9
x"0000", --A
x"0000", --B
x"0000", --C
x"0000", --D
x"0000", --E
x"0000", --F


   --9
x"0000", --0
x"0000", --1
x"0000", --2
x"0000", --3
x"0000", --4
x"0000", --5
x"0000", --6
x"0000", --7
x"0000", --8
x"0000", --9
x"0000", --A
x"0000", --B
x"0000", --C
x"0000", --D
x"0000", --E
x"0000", --F
--A
x"0000", --0
x"0000", --1
x"0000", --2
x"0000", --3
x"0000", --4
x"0000", --5
x"0000", --6
x"0000", --7
x"0000", --8
x"0000", --9
x"0000", --A
x"0000", --B
x"0000", --C
x"0000", --D
x"0000", --E
x"0000", --F
--B
x"0000", --0
x"0000", --1
x"0000", --2
x"0000", --3
x"0000", --4
x"0000", --5
x"0000", --6
x"0000", --7
x"0000", --8
x"0000", --9
x"0000", --A
x"0000", --B
```

```
    x"0000", --C
    x"0000", --D
    x"0000", --E
    x"0000", --F
    --C
          x"0000", --0
    x"0000", --1
    x"0000", --2
    x"0000", --3
    x"0000", --4
    x"0000", --5
    x"0000", --6
    x"0000", --7
    x"0000", --8
    x"0000", --9
    x"0000", --A
    x"0000", --B
    x"0000", --C
    x"0000", --D
    x"0000", --E
    x"0000", --F
    --D
    x"0000", --0
    x"0000", --1
    x"0000", --2
    x"0000", --3
    x"0000", --4
    x"0000", --5
    x"0000", --6
    x"0000", --7
    x"0000", --8
    x"0000", --9
    x"0000", --A
    x"0000", --B
    x"0000", --C
    x"0000", --D
    x"0000", --E
    x"0000", --F

 --E
          x"0000", --0
    x"0000", --1
    x"0000", --2
    x"0000", --3
    x"0000", --4
    x"0000", --5
    x"0000", --6
    x"0000", --7
    x"0000", --8
    x"0000", --9
    x"0000", --A
    x"0000", --B
    x"0000", --C
    x"0000", --D
    x"0000", --E
    x"0000", --F
```

```
    --F
    x"0000", --0
    x"0000", --1
    x"0000", --2
    x"0000", --3
    x"0000", --4
    x"0000", --5
    x"0000", --6
    x"0000", --7
    x"0000", --8
    x"0000", --9
    x"0000", --A
    x"0000", --B
    x"0000", --C
    x"0000", --D
    x"0000", --E
    x"0000" );

  variable addr : integer range 0 to 511;
  variable address_out : STD_LOGIC_VECTOR(15 downto 0);
  begin
    addr := conv_integer(data_in(8 downto 0));
    address_out := data_mem(addr);
    if mw = '1' then
      data_mem(addr) := address_in;
    else
      data_out <= address_out;
    end if;
  end process;
end Behavioral;
```

1.3 Micro-Programmed Control

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


entity microprogrammed_control is
    Port ( IR_mpc : in STD_LOGIC_VECTOR(15 downto 0);
          status_mpc : in STD_LOGIC_VECTOR(3 downto 0);
          reset_mpc : in STD_LOGIC;
          control_mpc : out STD_LOGIC_VECTOR(17 downto 0);
          PC_mpc : out STD_LOGIC_VECTOR(15 downto 0);
          MC_in_mpc : out STD_LOGIC;
          MC_TD_mpc : out STD_LOGIC;
          MC_TA_mpc : out STD_LOGIC;
          MC_TB_mpc : out STD_LOGIC);
end microprogrammed_control;
architecture Behavioral of microprogrammed_control is

   component mux8_1bit Port (
        in_00 : in STD_LOGIC;
          in_11 : in STD_LOGIC;
          in_C2 : in STD_LOGIC;
          in_V3 : in STD_LOGIC;
          in_Z4 : in STD_LOGIC;
          in_N5 : in STD_LOGIC;
          in_C6 : in STD_LOGIC;
          in_Z7 : in STD_LOGIC;
          in_MS : in STD_LOGIC_VECTOR(2 downto 0);
          MUXS_out : out STD_LOGIC);
   end component;

   component mux2_8bit Port (
        in_0 : in STD_LOGIC_VECTOR(7 downto 0);
          in_1 : in STD_LOGIC_VECTOR(7 downto 0);
          in_MC : in STD_LOGIC;
          MUXC_out : out STD_LOGIC_VECTOR(7 downto 0));
   end component;

   component CAR Port (
        car_in : in STD_LOGIC_VECTOR(7 downto 0);
          muxs_in : in STD_LOGIC;
          res : in STD_LOGIC;
          car_out : out STD_LOGIC_VECTOR(7 downto 0));
   end component;

   component extend Port (
           E_in : in STD_LOGIC_VECTOR(5 downto 0);
          E_out : out STD_LOGIC_VECTOR(15 downto 0));
   end component;

   component IR Port (
        IR_in : in STD_LOGIC_VECTOR(15 downto 0);
```

```
        IL : 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 PC Port (
      PC_in : in STD_LOGIC_VECTOR(15 downto 0);
        PL : in STD_LOGIC;
        PI : in STD_LOGIC;
        reset : in STD_LOGIC;
        PC_out : out STD_LOGIC_VECTOR(15 downto 0));
end component;

component control_memory Port (
      car_in : in STD_LOGIC_VECTOR(7 downto 0);
        NA : out STD_LOGIC_VECTOR(7 downto 0);
        MSout : out STD_LOGIC_VECTOR(2 downto 0);
        MC : out STD_LOGIC;
        IL : out STD_LOGIC;
        PI : out STD_LOGIC;
        PL : out STD_LOGIC;
        TD : out STD_LOGIC;
        TA : out STD_LOGIC;
        TB : out STD_LOGIC;
        MB : out STD_LOGIC;
        FSout : 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;

signal src_mpc_in : STD_LOGIC_VECTOR(5 downto 0);
signal src_mpc_out : STD_LOGIC_VECTOR(15 downto 0);
signal src_control : STD_LOGIC_VECTOR(17 downto 0);
signal src_opcode, src_car_in, src_car_out, src_na : STD_LOGIC_VECTOR(7 downto 0);
signal src_ms, src_sa, src_sb, src_dr : STD_LOGIC_VECTOR(2 downto 0);
signal src_mc, src_il, src_car_sig, src_pl, src_pi : STD_LOGIC;

begin
    reg_mux8_1bit: mux8_1bit PORT MAP (
      in_00 => '0',
        in_11 => '1',
        in_C2 => status_mpc(2),
        in_V3 => status_mpc(3),
        in_Z4 => status_mpc(0),
        in_N5 => status_mpc(1),
        in_C6 => not status_mpc(2),
        in_Z7 => not status_mpc(0),
        in_MS => src_ms,
        MUXS_out => src_car_sig);

reg_mux2_8bit :mux2_8bit PORT MAP (
      in_0 => src_na,
```

```
       in_1 => src_opcode,
       in_MC => src_mc,
       MUXC_out => src_car_out);

reg_CAR: CAR PORT MAP (
     car_in => src_car_out,
       muxs_in => src_car_sig,
       res => reset_mpc,
       car_out => src_car_in);

reg_extend : extend PORT MAP (
        E_in => src_mpc_in,
       E_out => src_mpc_out);

reg_IR : IR PORT MAP (
     IR_in => IR_mpc,
       IL => src_il,
       opcode => src_opcode(6 downto 0),
       DR => src_dr,
       SA => src_sa,
       SB => src_sb);

reg_PC : PC PORT MAP (
     PC_in => src_mpc_out,
       PL => src_pl,
       PI => src_pi,
       reset => reset_mpc,
       PC_out => PC_mpc);

reg_control_memory : control_memory PORT MAP (
     car_in => src_car_in,
       NA => src_na,
       MSout => src_ms,
       MC => src_mc,
       IL => src_il,
       PI => src_pi,
       PL => src_pl,
       TD => MC_TD_mpc,
       TA => MC_TA_mpc,
       TB => MC_TB_mpc,
       MB => src_control(8),
       FSout => src_control(7 downto 3),
       MD => src_control(2),
       RW => src_control(1),
       MM => src_control(0),
       MW => MC_in_mpc);

src_mpc_in(5 downto 3) <= src_dr;
src_mpc_in(2 downto 0) <= src_sb;

src_opcode(7) <= '0';
src_control(17 downto 15) <= src_dr;
src_control(14 downto 12) <= src_sa;
src_control(11 downto 9) <= src_sb;
control_mpc <= src_control;
```

```
end Behavioral;
```

1.3.1 CAR

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

entity CAR is
    Port ( car_in : in STD_LOGIC_VECTOR(7 downto 0);
           muxs_in : in STD_LOGIC;
           res : in STD_LOGIC;
           car_out : out STD_LOGIC_VECTOR(7 downto 0));
end CAR;
architecture Behavioral of CAR is
begin
   process(res, car_in)
   variable car_now : STD_LOGIC_VECTOR(7 downto 0);
   variable car_temp : integer;
   variable car_next : std_logic_vector(7 downto 0);
   begin
      if(res = '1') then
         car_now := x"C0";
      elsif(muxs_in = '1') then
         car_now := car_in;
      elsif(muxs_in = '0') then
         car_temp := conv_integer(car_now);
         car_temp := car_temp + conv_integer(1);
         car_next := conv_std_logic_vector(car_temp, 8);
         car_now := car_next;
         end if;
         car_out <= car_now after 20ns;
   end process;
end Behavioral;
```

1.3.2 MUX8 TO 1BIT

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

entity mux8_1bit is
   Port ( in_00 : in STD_LOGIC;
          in_11 : in STD_LOGIC;
          in_C2 : in STD_LOGIC;
          in_V3 : in STD_LOGIC;
          in_Z4 : in STD_LOGIC;
          in_N5 : in STD_LOGIC;
          in_C6 : in STD_LOGIC;
          in_Z7 : in STD_LOGIC;
          in_MS : in STD_LOGIC_VECTOR(2 downto 0);
          MUXS_out : out STD_LOGIC);
end mux8_1bit;

architecture Behavioral of mux8_1bit is
begin
  MUXS_out <= in_00 after 1ns when in_MS = "000" else
              in_11 after 1ns when in_MS = "001" else
              in_C2 after 1ns when in_MS = "010" else
              in_V3 after 1ns when in_MS = "011" else
              in_Z4 after 1ns when in_MS = "100" else
              in_N5 after 1ns when in_MS = "101" else
              in_C6 after 1ns when in_MS = "110" else
              in_Z7 after 1ns when in_MS = "111";
end Behavioral;
```

1.3.3 MUX2 TO 8BIT

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

entity mux2_8bit is
    Port ( in_0 : in STD_LOGIC_VECTOR(7 downto 0);
           in_1 : in STD_LOGIC_VECTOR(7 downto 0);
           in_MC : in STD_LOGIC;
           MUXC_out : out STD_LOGIC_VECTOR(7 downto 0));
end mux2_8bit;
architecture Behavioral of mux2_8bit is
begin
   MUXC_out <= in_0 after 1ns when in_MC = '0' else
           in_1 after 1ns when in_MC = '1' else
           x"00" after 20ns;
end Behavioral;
```

1.3.4 extend

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

entity extend is
    Port ( E_in : in STD_LOGIC_VECTOR(5 downto 0);
           E_out : out STD_LOGIC_VECTOR(15 downto 0));
end extend;

architecture Behavioral of extend is
   signal e_src : STD_LOGIC_VECTOR(15 downto 0);
begin
   e_src(5 downto 0) <= E_in;
   e_src(15 downto 6) <= "0000000000" when E_in(5) = '0' else "1111111111";
   E_out <= e_src;
end Behavioral;
```

1.3.5 IR

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

entity IR is
    Port ( IR_in : in STD_LOGIC_VECTOR(15 downto 0);
          IL : 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
begin
   opcode <= IR_in(15 downto 9) after 1ns when IL ='1';
   DR <= IR_in(8 downto 6) after 1ns when IL = '1';
   SA <= IR_in(5 downto 3) after 1ns when IL = '1';
   SB <= IR_in(2 downto 0) after 1ns when IL = '1';
end Behavioral;
```

1.3.6 PC

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

entity PC is
   Port ( PC_in : in STD_LOGIC_VECTOR(15 downto 0);
          PL : in STD_LOGIC;
          PI : in STD_LOGIC;
          reset : in STD_LOGIC;
          PC_out : out STD_LOGIC_VECTOR(15 downto 0));
end PC;
architecture Behavioral of PC is
begin

   process(reset , PL , PI)
   variable PC_now : STD_LOGIC_VECTOR(15 downto 0);
   variable PC_this : integer;
   variable PC_next : STD_LOGIC_VECTOR(15 downto 0);
   begin
      if(reset = '1') then
         PC_now := x"0000";
      elsif(PL = '1') then
         PC_now := PC_now + PC_in;
      elsif(PI = '1') then
         PC_this := conv_integer(PC_now);
         PC_this := PC_this + conv_integer(1);
         PC_next := conv_std_logic_vector(PC_this, 16);
         PC_now := PC_next;
      end if;
      PC_out <= PC_now after 2ns;
   end process;
end Behavioral;
```

1.3.7 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 ( car_in : in STD_LOGIC_VECTOR(7 downto 0);
          NA : out STD_LOGIC_VECTOR(7 downto 0);
          MSout : out STD_LOGIC_VECTOR(2 downto 0);
          MC : out STD_LOGIC;
          IL : out STD_LOGIC;
          PI : out STD_LOGIC;
          PL : out STD_LOGIC;
          TD : out STD_LOGIC;
          TA : out STD_LOGIC;
          TB : out STD_LOGIC;
          MB : out STD_LOGIC;
          FSout : out STD_LOGIC_VECTOR(4 downto 0);
          MD : out STD_LOGIC;
          RW : out STD_LOGIC;
          MM : out STD_LOGIC;
          MW : out STD_LOGIC);
end control_memory;
architecture Behavioral of control_memory is

   type src_memory is array(0 to 255) of STD_LOGIC_VECTOR(27 downto 0);

begin
   memory : process(car_in)
   variable memory : src_memory := (

   x"F0E1D50",
   x"F0E1D51",
   x"F0E1D52",
   x"F0E1D53",
   x"F0E1D54",
   x"F0E1D55",
   x"F0E1D56",
   x"F0E1D57",
   x"F0E1D58",
   x"F0E1D59",
   x"F0E1D5A",
   x"F0E1D5B",
   x"F0E1D5C",
   x"F0E1D5D",
   x"F0E1D5E",
   x"F0E1D5F",

      x"F0E1D50",
   x"F0E1D51",
   x"F0E1D52",
   x"F0E1D53",
   x"F0E1D54",
   x"F0E1D55",
```

```
x"F0E1D56",
x"F0E1D57",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",

x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",

x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",


x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
```

```
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",


x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",

x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",


x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
```

```
x"0000000",
x"0000000",
x"0000000",


   x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",

x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",


x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
```

```
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",

x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",


x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",
x"0000000",

   x"0000000",
x"0000000",
```

```
   x"0000000",
   x"0000000",
   x"0000000",
   x"0000000",
   x"0000000",
   x"0000000",
   x"0000000",
   x"0000000",
   x"0000000",
   x"0000000",
   x"0000000",
   x"0000000",
   x"0000000",
   x"0000000",


   x"0000000",
   x"0000000",
   x"0000000",
   x"0000000",
   x"0000000",
   x"0000000",
   x"0000000",
   x"0000000",
   x"0000000",
   x"0000000",
   x"0000000",
   x"0000000",
   x"0000000",
   x"0000000",
   x"0000000",
   x"0000000");

   VARIABLE addr : integer;
   variable control_out : STD_LOGIC_VECTOR(27 downto 0);
   begin
      addr := conv_integer(car_in);
      control_out := memory(addr);
       NA <= control_out(27 downto 20);
          MSout <= control_out(19 downto 17);
          MC <= control_out(16);
          IL <= control_out(15);
          PI <= control_out(14);
          PL <= control_out(13);
          TD <= control_out(12);
          TA <= control_out(11);
          TB <= control_out(10);
          MB <= control_out(9);
          FSout <= control_out(8 downto 4);
          MD <= control_out(3);
          RW <= control_out(2);
          MM <= control_out(1);
          MW <= control_out(0);
   end process;
end Behavioral;
```

1.3.8 zero fill

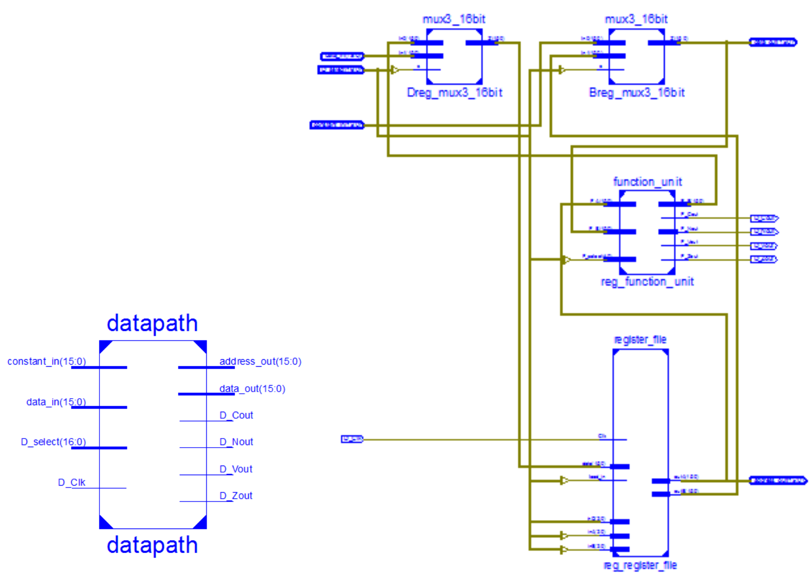
```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

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

architecture Behavioral of zero_fill is
   signal src_z : STD_LOGIC_VECTOR(15 downto 0);
begin
   src_z(2 downto 0) <= sb;
   src_z(15 downto 3) <= "0000000000000";
   z_out <= src_z;
end Behavioral;
```

1.4 bit

1.5 DATA PATH

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

entity datapath is
Port ( data_in : in STD_LOGIC_VECTOR(15 downto 0);
        PC_in : in STD_LOGIC_VECTOR(15 downto 0);
          control_in : in STD_LOGIC_VECTOR(17 downto 0);
          D_Clk : in STD_LOGIC;
          TD : in STD_LOGIC;
          TA : in STD_LOGIC;
          TB : in STD_LOGIC;
          address_out : out STD_LOGIC_VECTOR(15 downto 0);
          data_out : out STD_LOGIC_VECTOR(15 downto 0);
          path_out : out STD_LOGIC_VECTOR(3 downto 0));
end datapath;

architecture Behavioral of datapath is

component mux3_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;

Component function_unit
Port ( F_A : in STD_LOGIC_VECTOR(15 downto 0);
          F_B : in STD_LOGIC_VECTOR(15 downto 0);
          F_select : in STD_LOGIC_VECTOR(4 downto 0);
          F_Vout : out STD_LOGIC;
          F_Cout : out STD_LOGIC;
          F_Nout : out STD_LOGIC;
          F_Zout : out STD_LOGIC;
          F_F : out STD_LOGIC_VECTOR(15 downto 0));
end Component;

Component register_file
    Port ( inA : in STD_LOGIC_VECTOR(3 downto 0);
          inB : in STD_LOGIC_VECTOR(3 downto 0);
          inD : in STD_LOGIC_VECTOR(3 downto 0);
          Clk : in STD_LOGIC;
         load_in : in STD_LOGIC;
          data : in STD_LOGIC_VECTOR(15 downto 0);
           outA : out  STD_LOGIC_VECTOR(15 downto 0);
           outB : out  STD_LOGIC_VECTOR(15 downto 0));
end Component ;

Component zero_fill
    Port ( sb : in STD_LOGIC_VECTOR(2 downto 0);
          z_out : out STD_LOGIC_VECTOR(15 downto 0));
end Component;
```

```
signal src_muxM, src_muxD, src_muxB, src_regA, src_regB, src_out, src_z , src_pc:
    STD_LOGIC_VECTOR(15 downto 0);
signal src_outA, src_outB, src_outD, src_path : STD_LOGIC_VECTOR(3 downto 0);

begin
Dreg_mux3_16bit : mux3_16bit PORT MAP ( s => control_in(2),
          In0 => src_out,
          In1 => data_in,
          Z => src_muxD);

Breg_mux3_16bit : mux3_16bit PORT MAP ( s => control_in(8),
          In0 => src_regB,
          In1 => src_z,
          Z => src_muxB);

src_pc <= PC_in;

Mreg_mux3_16bit : mux3_16bit PORT MAP ( s => control_in(0),
          In0 => src_regA,
          In1 => src_pc,
          Z => src_muxM);

src_outD <= TD & control_in(17 downto 15);
src_outA <= TA & control_in(14 downto 12);
src_outB <= TB & control_in(11 downto 9);

reg_function_unit : function_unit PORT MAP (
           F_A => src_regA,
          F_B => src_muxB,
          F_select => control_in(7 downto 3),
          F_Vout => src_path(3),
          F_Cout => src_path(2),
          F_Nout => src_path(1),
          F_Zout => src_path(0),
          F_F => src_out);

reg_register_file : register_file PORT MAP(
           inA => src_outA,
          inB => src_outB,
          inD => src_outD,
          Clk => D_Clk,
         load_in => control_in(1),
          data => src_muxD,
           outA => src_regA,
           outB => src_regB);

reg_zero_fill : zero_fill PORT MAP(
           sb => control_in(11 downto 9),
          z_out => src_z);

data_out <= src_muxB;
address_out <= src_regA;
path_out <= src_path;

end Behavioral;
```


datapath
constant_in(15:0)
data_in(15:0)
D_select(16:0)
D_Clk
address_out(15:0)
data_out(15:0)
D_Cout
D_Nout
D_Vout
D_Zout
datapath
mux3_16bit
Dreg_mux3_16bit
mux3_16bit
Breg_mux3_16bit
function_unit
reg_function_unit
register_file
reg_register_file

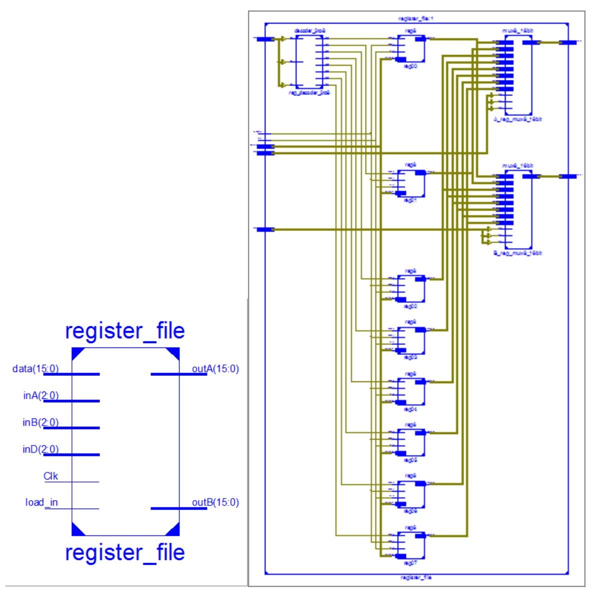
1.6 REGISTER FILE

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


entity register_file is
    Port (
          inA : in STD_LOGIC_VECTOR(3 downto 0);
          inB : in STD_LOGIC_VECTOR(3 downto 0);
          inD : in STD_LOGIC_VECTOR(3 downto 0);
          Clk : in STD_LOGIC;
         load_in : in STD_LOGIC;
          data : in STD_LOGIC_VECTOR(15 downto 0);
           outA : out  STD_LOGIC_VECTOR(15 downto 0);
           outB : out  STD_LOGIC_VECTOR(15 downto 0)
         );
end register_file;

architecture Behavioral of register_file is

   Component decoder_3to8
   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);
   End Component;

   Component mux3_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;

   Component reg8
   Port ( load0 : in STD_LOGIC;
           load1 : in STD_LOGIC;
         Clk : in STD_LOGIC;
         D : in STD_LOGIC_VECTOR(15 downto 0);
         Q : out STD_LOGIC_VECTOR(15 downto 0));
   End Component;
```

```
  Component mux8_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);
         Z : out STD_LOGIC_VECTOR(15 downto 0));
  End Component;

  signal load_reg0, load_reg1, load_reg2, load_reg3, load_reg4,
          load_reg5, load_reg6, load_reg7, load_reg8 : STD_LOGIC;
  signal reg0_q, reg1_q, reg2_q, reg3_q, reg4_q, reg5_q, reg6_q,
          reg7_q, reg8_q, d_mux, src_reg, src_A, src_B : STD_LOGIC_VECTOR(15 downto 0);

begin
  reg_decoder_3to8 : decoder_3to8 PORT MAP(
    A0 => inD(0),
    A1 => inD(1),
    A2 => inD(2),
    A3 => inD(3),
    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
    );

  A_reg_mux8_16bit : mux8_16bit PORT MAP(
    S0 => inA(0),
    S1 => inA(1),
    S2 => inA(2),
    S3 => inA(3),
    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,
    Z => src_A
    );
```

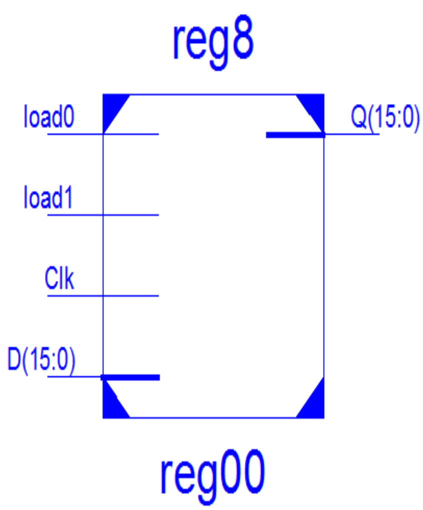
```vhdl
B_reg_mux8_16bit : mux8_16bit PORT MAP(
   S0 => inB(0),
   S1 => inB(1),
   S2 => inB(2),
   S3 => inB(3),
   In0 => reg0_q,
   In1 => reg1_q,
   In2 => reg2_q,
   In3 => reg3_q,
   In4 => reg4_q,
   In5 => reg5_q,
   In6 => reg6_q,
   In7 => reg7_q,
   In8 => reg8_q,
   Z => src_B
   );

reg00 : reg8 PORT MAP(
   load0 => load_reg0,
   load1 => load_in,
   Clk =>  Clk,
   D => data,
   Q => reg0_q);

reg01 : reg8 PORT MAP(
   load0 => load_reg1,
   load1 => load_in,
   Clk =>  Clk,
   D => data,
   Q => reg1_q);

reg02 : reg8 PORT MAP(
   load0 => load_reg2,
   load1 => load_in,
   Clk =>  Clk,
   D => data,
   Q => reg2_q);

reg03 : reg8 PORT MAP(
   load0 => load_reg3,
   load1 => load_in,
   Clk =>  Clk,
   D => data,
   Q => reg3_q);

reg04 : reg8 PORT MAP(
   load0 => load_reg4,
   load1 => load_in,
   Clk =>  Clk,
   D => data,
   Q => reg4_q);

reg05 : reg8 PORT MAP(
   load0 => load_reg5,
   load1 => load_in,
   Clk =>  Clk,
```

```
      D =>  data,
      Q => reg5_q);

   reg06 : reg8 PORT MAP(
      load0 => load_reg6,
      load1 => load_in,
      Clk =>  Clk,
      D =>  data,
      Q => reg6_q);

   reg07 : reg8 PORT MAP(
      load0 => load_reg7,
      load1 => load_in,
      Clk =>  Clk,
      D =>  data,
      Q => reg7_q);

   reg08 : reg8 PORT MAP(
      load0 => load_reg8,
      load1 => load_in,
      Clk =>  Clk,
      D =>  data,
      Q => reg8_q);

   outA <= src_A;
   outB <= src_B;


end Behavioral;
```



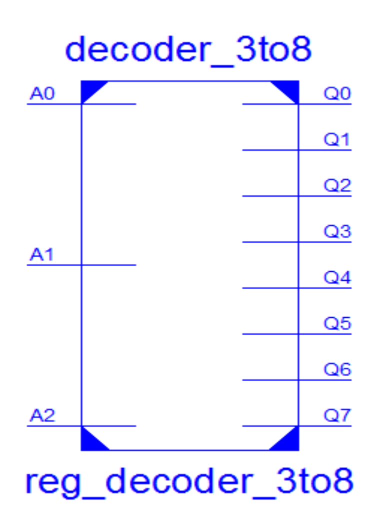
1.6.1 REG8

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity reg8 is
    Port ( load0 : in STD_LOGIC;
           load1 : in STD_LOGIC;
           Clk : in STD_LOGIC;
           D : in STD_LOGIC_VECTOR(15 downto 0);
           Q : out STD_LOGIC_VECTOR(15 downto 0));
end reg8;

architecture Behavioral of reg8 is

begin
process(Clk)
begin
      if(rising_edge(Clk)) then
         if(load0 ='1') and (load1 ='1') then
            Q<= D after 5ns;
         end if;
      end if;
   end process;
end Behavioral;
```



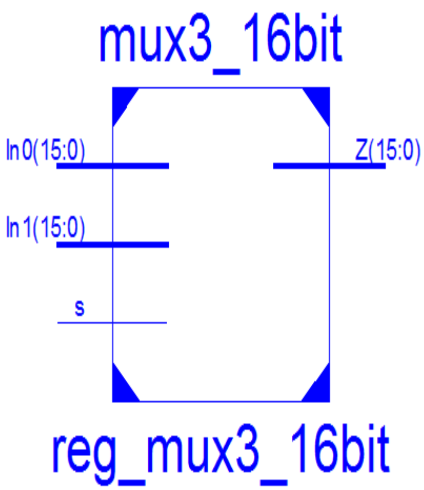
1.6.2 DECODER

```
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;
          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);
end decoder_3to8;

architecture Behavioral of decoder_3to8 is

begin
   Q0 <= (( NOT A0) AND (NOT A1) AND (NOT A2) AND (NOT A3)) AFTER 5ns;
   Q1 <= (( NOT A0) AND (NOT A1) AND A2 AND (NOT A2)) AFTER 5ns;
   Q2 <= (( NOT A0) AND A1 AND (NOT A2) AND (NOT A2)) AFTER 5ns;
   Q3 <= (( NOT A0) AND A1 AND A2 AND (NOT A2)) AFTER 5ns;
   Q4 <= (A0 AND (NOT A1) AND (NOT A2) AND (NOT A2)) AFTER 5ns;
   Q5 <= (A0 AND (NOT A1) AND A2 AND (NOT A2)) AFTER 5ns;
   Q6 <= (A0 AND A1 AND (NOT A2) AND (NOT A2)) AFTER 5ns;
   Q7 <= (A0 AND A1 AND A2 AND (NOT A2)) AFTER 5ns;
   Q8 <= A3 AFTER 5ns;
end Behavioral;
```


decoder_3to8
A0
A1
A2
Q0
Q1
Q2
Q3
Q4
Q5
Q6
Q7
reg_decoder_3to8

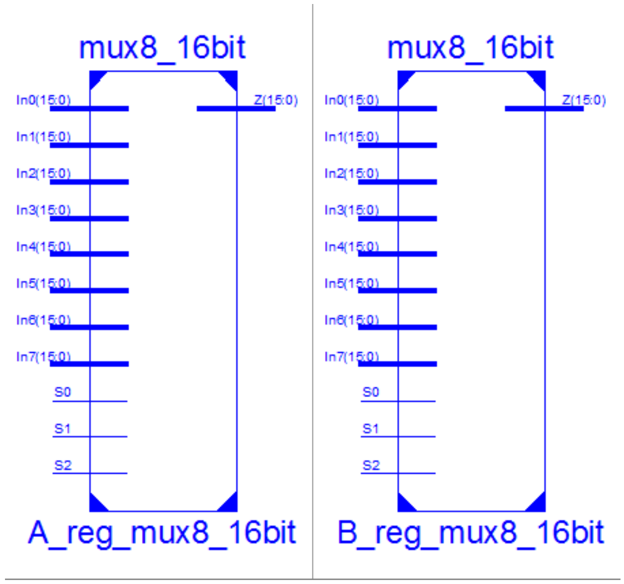
1.6.3 MUX3 TO 16BIT

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


entity mux3_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 mux3_16bit;

architecture Behavioral of mux3_16bit is

begin
   Z <= In0 after 5ns when s = '0' else
        In1 after 5ns when s = '1' else
        x"0000" after 5ns;

end Behavioral;
```



1.6.4 MUX8 TO 16BIT

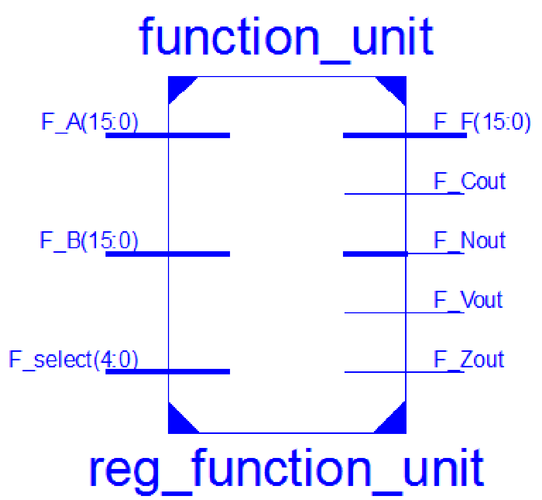
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;


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 after 5ns when S0 = '0' and S1 = '0' and S2 = '0' else
        In1 after 5ns when S0 = '0' and S1 = '0' and S2 = '1' else
        In2 after 5ns when S0 = '0' and S1 = '1' and S2 = '0' else
        In3 after 5ns when S0 = '0' and S1 = '1' and S2 = '1' else
        In4 after 5ns when S0 = '1' and S1 = '0' and S2 = '0' else
        In5 after 5ns when S0 = '1' and S1 = '0' and S2 = '1' else
        In6 after 5ns when S0 = '1' and S1 = '1' and S2 = '0' else
        In7 after 5ns when S0 = '1' and S1 = '1' and S2 = '1' else
        x"0000" after 5ns;
end Behavioral;
```

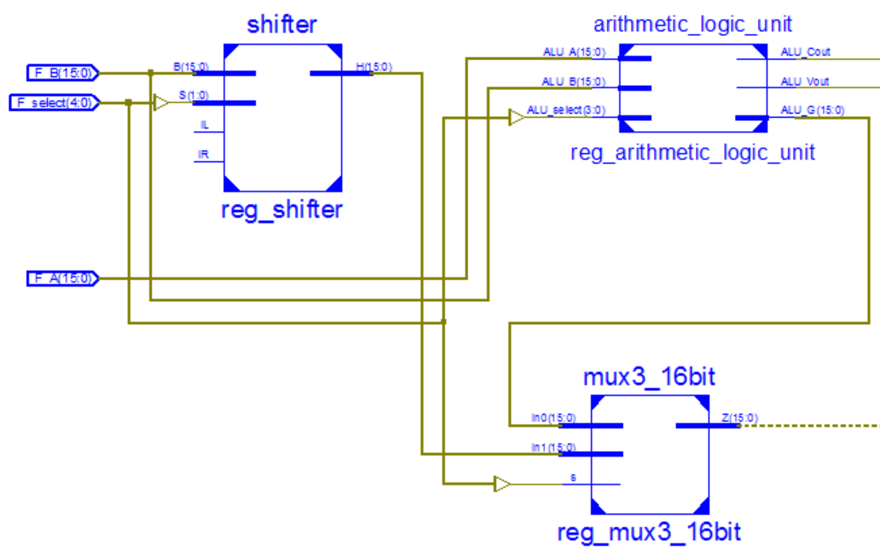

mux8_16bit
In0(15:0)
In1(15:0)
In2(15:0)
In3(15:0)
In4(15:0)
In5(15:0)
In6(15:0)
In7(15:0)
S0
S1
S2
Z(15:0)
A_reg_mux8_16bit
mux8_16bit
In0(15:0)
In1(15:0)
In2(15:0)
In3(15:0)
In4(15:0)
In5(15:0)
In6(15:0)
In7(15:0)
S0
S1
S2
Z(15:0)
B_reg_mux8_16bit

1.7 FUNCTION UNIT

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


entity function_unit is
Port ( F_A : in STD_LOGIC_VECTOR(15 downto 0);
         F_B : in STD_LOGIC_VECTOR(15 downto 0);
         F_select : in STD_LOGIC_VECTOR(4 downto 0);
         F_Vout : out STD_LOGIC;
         F_Cout : out STD_LOGIC;
         F_Nout : out STD_LOGIC;
         F_Zout : out STD_LOGIC;
         F_F : out STD_LOGIC_VECTOR(15 downto 0));
end function_unit;

architecture Behavioral of function_unit is

Component arithmetic_logic_unit
   Port ( ALU_A : in STD_LOGIC_VECTOR(15 downto 0);
         ALU_B : in STD_LOGIC_VECTOR(15 downto 0);
         ALU_select : in STD_LOGIC_VECTOR(3 downto 0);
         ALU_Cout : out STD_LOGIC;
         ALU_Vout : out STD_LOGIC;
         ALU_G : out STD_LOGIC_VECTOR(15 downto 0));
end Component;


Component shifter Port ( B : in STD_LOGIC_VECTOR(15 downto 0);
         S : in STD_LOGIC_VECTOR(1 downto 0);
         IR : in STD_LOGIC;
         IL : in STD_LOGIC;
         H : out STD_LOGIC_VECTOR(15 downto 0));
end Component;

Component mux3_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;

signal src_ALU, src_shift, src_mux : STD_LOGIC_VECTOR(15 downto 0);

begin

reg_arithmetic_logic_unit : arithmetic_logic_unit
   Port MAP ( ALU_A => F_A,
         ALU_B => F_B,
         ALU_select => F_select(3 downto 0),
         ALU_Cout => F_Cout,
         ALU_Vout => F_Vout,
         ALU_G => src_ALU);

reg_shifter : shifter Port MAP ( B => F_B,
         S => F_select(3 downto 2),
```

```
          IR => '0',
          IL => '0',
          H => src_shift);


reg_mux3_16bit : mux3_16bit Port MAP ( s => F_select(4),
          In0 => src_ALU,
          In1 => src_shift,
          Z  => src_mux);

F_F <= src_mux;
F_Nout <= src_mux(15);
F_Zout <= ((src_mux(15)) or (src_mux(14)) or (src_mux(13)) or
           (src_mux(12)) or (src_mux(11)) or (src_mux(10)) or
           (src_mux(9)) or (src_mux(8)) or (src_mux(7)) or
           (src_mux(6)) or (src_mux(5)) or (src_mux(4)) or
           (src_mux(3)) or (src_mux(2)) or
           (src_mux(1)) or (src_mux(0)));



end Behavioral;
```



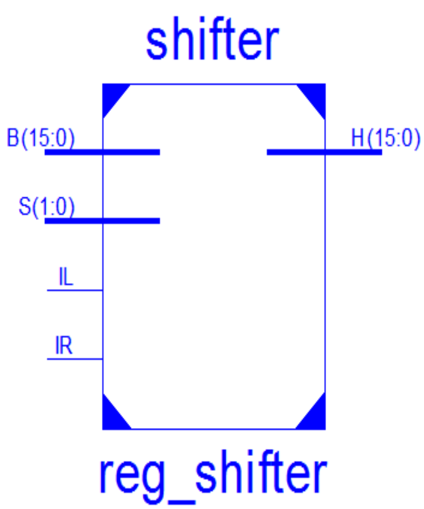


1.8 SHIFTER

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


entity shifter is
   Port ( B : in STD_LOGIC_VECTOR(15 downto 0);
          S : in STD_LOGIC_VECTOR(1 downto 0);
          IR : in STD_LOGIC;
          IL : in STD_LOGIC;
          H : out STD_LOGIC_VECTOR(15 downto 0));
end shifter;

architecture Behavioral of shifter is

component mux3_1bit Port ( In0 : in STD_LOGIC;
          In1 : in STD_LOGIC;
          In2 : in STD_LOGIC;
          S0 : in STD_LOGIC;
          S1 : in STD_LOGIC;
          Z : out STD_LOGIC);
end component;

begin



mux3_1bit00 : mux3_1bit PORT MAP ( In0 =>B(0) ,
          In1 => B(1),
          In2 => IL,
          S0 => S(0),
          S1 => S(1),
          Z  => H(0));

mux3_1bit01 : mux3_1bit PORT MAP ( In0 =>B(1) ,
          In1 => B(2),
          In2 => B(0),
          S0 => S(0),
          S1 => S(1),
          Z  => H(1));

mux3_1bit02 : mux3_1bit PORT MAP ( In0 =>B(2) ,
          In1 => B(3),
          In2 => B(1),
          S0 => S(0),
          S1 => S(1),
          Z  => H(2));

mux3_1bit03 : mux3_1bit PORT MAP ( In0 =>B(3) ,
          In1 => B(4),
          In2 => B(2),
          S0 => S(0),
          S1 => S(1),
          Z  => H(3));
```

```
mux3_1bit04 : mux3_1bit PORT MAP ( In0 =>B(4) ,
          In1 => B(5),
          In2 => B(3),
          S0 => S(0),
          S1 => S(1),
          Z  => H(4));

mux3_1bit05 : mux3_1bit PORT MAP ( In0 =>B(5) ,
          In1 => B(6),
          In2 => B(4),
          S0 => S(0),
          S1 => S(1),
          Z  => H(5));

mux3_1bit06 : mux3_1bit PORT MAP ( In0 =>B(6) ,
          In1 => B(7),
          In2 => B(5),
          S0 => S(0),
          S1 => S(1),
          Z  => H(6));

mux3_1bit07 : mux3_1bit PORT MAP ( In0 =>B(7) ,
          In1 => B(8),
          In2 => B(6),
          S0 => S(0),
          S1 => S(1),
          Z  => H(7));

mux3_1bit08 : mux3_1bit PORT MAP ( In0 =>B(8) ,
          In1 => B(9),
          In2 => B(7),
          S0 => S(0),
          S1 => S(1),
          Z  => H(8));

mux3_1bit09 : mux3_1bit PORT MAP ( In0 =>B(9) ,
          In1 => B(10),
          In2 => B(8),
          S0 => S(0),
          S1 => S(1),
          Z  => H(9));

mux3_1bit10 : mux3_1bit PORT MAP ( In0 =>B(10) ,
          In1 => B(11),
          In2 => B(9),
          S0 => S(0),
          S1 => S(1),
          Z  => H(10));

mux3_1bit11 : mux3_1bit PORT MAP ( In0 =>B(11) ,
          In1 => B(12),
          In2 => B(10),
          S0 => S(0),
          S1 => S(1),
          Z  => H(11));
```

```
mux3_1bit12 : mux3_1bit PORT MAP ( In0 =>B(12) ,
          In1 => B(13),
          In2 => B(11),
          S0 => S(0),
          S1 => S(1),
          Z  => H(12));

mux3_1bit13 : mux3_1bit PORT MAP ( In0 =>B(13) ,
          In1 => B(14),
          In2 => B(12),
          S0 => S(0),
          S1 => S(1),
          Z  => H(13));

mux3_1bit14 : mux3_1bit PORT MAP ( In0 =>B(14) ,
          In1 => B(15),
          In2 => B(13),
          S0 => S(0),
          S1 => S(1),
          Z  => H(14));

mux3_1bit15 : mux3_1bit PORT MAP ( In0 =>B(15) ,
          In1 => IR,
          In2 => B(14),
          S0 => S(0),
          S1 => S(1),
          Z  => H(15));
end Behavioral;
```



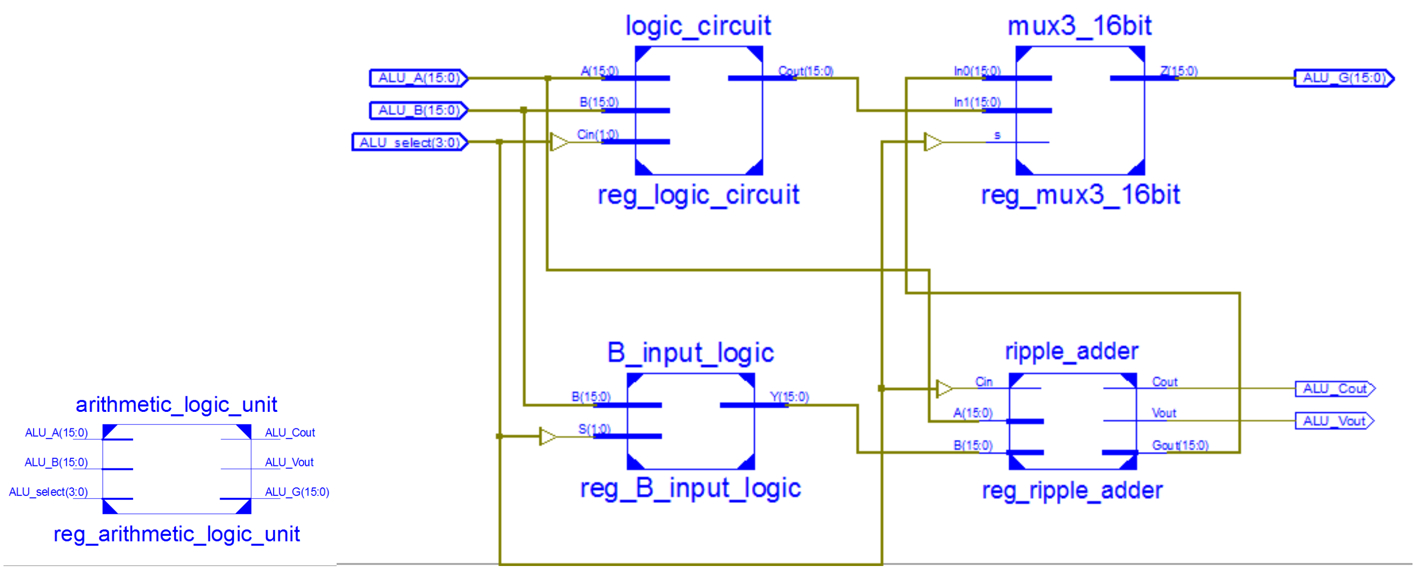
1.8.1 MUX3 TO 1BIT

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


entity mux3_1bit is
   Port ( In0 : in STD_LOGIC;
          In1 : in STD_LOGIC;
          In2 : in STD_LOGIC;
          S0 : in STD_LOGIC;
          S1 : in STD_LOGIC;
          Z : out STD_LOGIC);
end mux3_1bit;

architecture Behavioral of mux3_1bit is

begin
Z <= In0 after 1ns when S0 = '0' and S1='0' else
      In1 after 1ns when S0 ='0' and S1= '1' else
      In2 after 1ns when S0 ='0' and S1 = '0' else
      '0' after 1ns;

end Behavioral;
```

1.9 ALU

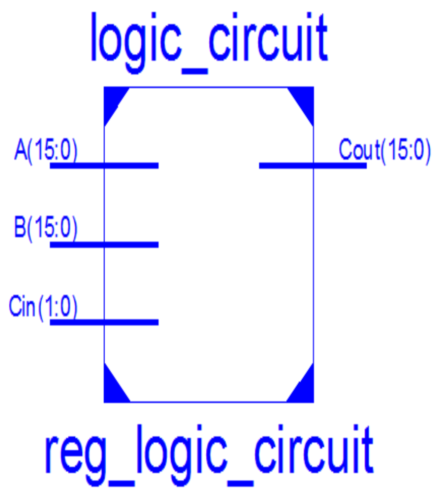
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;



entity arithmetic_logic_unit is
   Port ( ALU_A : in STD_LOGIC_VECTOR(15 downto 0);
          ALU_B : in STD_LOGIC_VECTOR(15 downto 0);
          ALU_select : in STD_LOGIC_VECTOR(3 downto 0);
          ALU_Cout : out STD_LOGIC;
          ALU_Vout : out STD_LOGIC;
          ALU_G : out STD_LOGIC_VECTOR(15 downto 0));
end arithmetic_logic_unit;

architecture Behavioral of arithmetic_logic_unit is

component ripple_adder 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;
          Gout : out STD_LOGIC_VECTOR(15 downto 0);
          Vout : out STD_LOGIC);
End Component;

component mux3_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;

component B_input_logic Port ( B : in STD_LOGIC_VECTOR(15 downto 0);
          S : in STD_LOGIC_VECTOR(1 downto 0);
          Y : out STD_LOGIC_VECTOR(15 downto 0));
End Component;

Component logic_circuit Port ( A : in STD_LOGIC_VECTOR(15 downto 0);
          B : in STD_LOGIC_VECTOR(15 downto 0);
          Cin : in STD_LOGIC_VECTOR(1 downto 0);
          Cout : out STD_LOGIC_VECTOR(15 downto 0));
End Component;

signal src_logic , src_B_input_logic, src_ripple : STD_LOGIC_VECTOR(15 downto 0);

begin


reg_ripple_adder : ripple_adder PORT MAP ( A => ALU_A,
          B => src_logic,
          Cin => ALU_select(0),
          Cout => ALU_Cout,
          Gout => src_ripple,
          Vout => ALU_Vout);
```

```
reg_mux3_16bit :mux3_16bit PORT MAP ( s => ALU_select(3),
          In0 => src_ripple,
          In1 => src_B_input_logic,
          Z =>ALU_G);


reg_B_input_logic : B_input_logic PORT MAP ( B => ALU_B,
          S => ALU_SELECT(2 downto 1),
          Y => src_logic);

reg_logic_circuit : logic_circuit PORT MAP( A => ALU_A ,
          B => ALU_B,
          Cin => ALU_select(2 downto 1),
          Cout => src_B_input_logic);



end Behavioral;
```



LOGIC

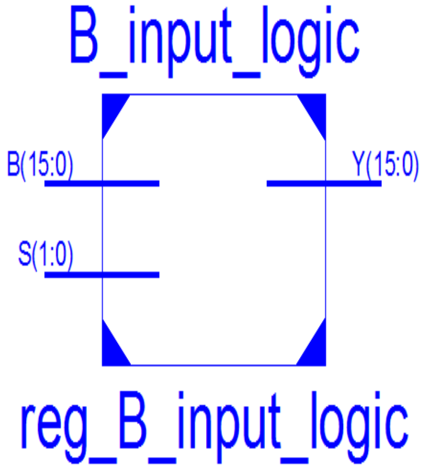
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;



entity logic_circuit is
   Port ( A : in STD_LOGIC_VECTOR(15 downto 0);
          B : in STD_LOGIC_VECTOR(15 downto 0);
          Cin : in STD_LOGIC_VECTOR(1 downto 0);
          Cout : out STD_LOGIC_VECTOR(15 downto 0));
end logic_circuit;

architecture Behavioral of logic_circuit is

begin
Cout <= (A and B) after 1ns when Cin = "00" else
        (A or B) after 1ns when Cin = "01" else
        (A xor B) after 1ns when Cin = "10" else
        (not (A)) after 1ns;
end Behavioral;
```



1.9.1 B LOGIC

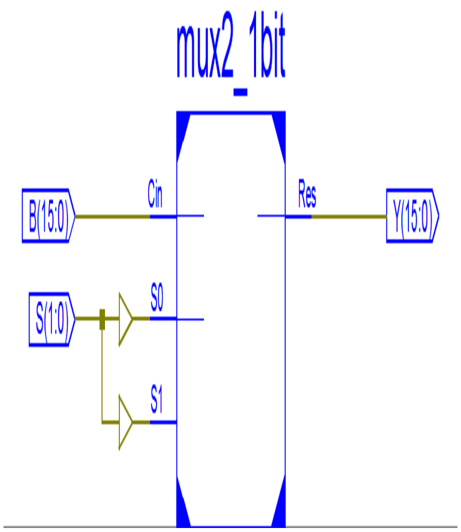
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;


entity B_input_logic is
   Port ( B : in STD_LOGIC_VECTOR(15 downto 0);
          S : in STD_LOGIC_VECTOR(1 downto 0);
          Y : out STD_LOGIC_VECTOR(15 downto 0));
end B_input_logic;

architecture Behavioral of B_input_logic is

component mux2_1bit Port ( S0 : in STD_LOGIC;
                           S1 : in STD_LOGIC;
                           Cin : in STD_LOGIC;
                           Res : out STD_LOGIC);
End Component;

begin
mux2_1bit00 : mux2_1bit PORT MAP( S0 => S(0),
                                  S1 => S(1),
                                  Cin =>B(0),
                                  Res => Y(0));

mux2_1bit01 : mux2_1bit PORT MAP( S0 => S(0),
                                  S1 => S(1),
                                  Cin =>B(1),
                                  Res => Y(1));

mux2_1bit02 : mux2_1bit PORT MAP( S0 => S(0),
                                  S1 => S(1),
                                  Cin =>B(2),
                                  Res => Y(2));

mux2_1bit03 : mux2_1bit PORT MAP( S0 => S(0),
                                  S1 => S(1),
                                  Cin =>B(3),
                                  Res => Y(3));

mux2_1bit04 : mux2_1bit PORT MAP( S0 => S(0),
                                  S1 => S(1),
                                  Cin =>B(4),
                                  Res => Y(4));

mux2_1bit05 : mux2_1bit PORT MAP( S0 => S(0),
                                  S1 => S(1),
                                  Cin =>B(5),
                                  Res => Y(5));

mux2_1bit06 : mux2_1bit PORT MAP( S0 => S(0),
                                  S1 => S(1),
                                  Cin =>B(6),
                                  Res => Y(6));
```

```
mux2_1bit07 : mux2_1bit PORT MAP( S0 => S(0),
                                 S1 => S(1),
                                 Cin =>B(7),
                                 Res => Y(7));

mux2_1bit08 : mux2_1bit PORT MAP( S0 => S(0),
                                 S1 => S(1),
                                 Cin =>B(8),
                                 Res => Y(8));

mux2_1bit09 : mux2_1bit PORT MAP( S0 => S(0),
                                 S1 => S(1),
                                 Cin =>B(9),
                                 Res => Y(9));

mux2_1bit10 : mux2_1bit PORT MAP( S0 => S(0),
                                 S1 => S(1),
                                 Cin =>B(10),
                                 Res => Y(10));

mux2_1bit11 : mux2_1bit PORT MAP( S0 => S(0),
                                 S1 => S(1),
                                 Cin =>B(11),
                                 Res => Y(11));

mux2_1bit12 : mux2_1bit PORT MAP( S0 => S(0),
                                 S1 => S(1),
                                 Cin =>B(12),
                                 Res => Y(12));

mux2_1bit13 : mux2_1bit PORT MAP( S0 => S(0),
                                 S1 => S(1),
                                 Cin =>B(13),
                                 Res => Y(13));

mux2_1bit14 : mux2_1bit PORT MAP( S0 => S(0),
                                 S1 => S(1),
                                 Cin =>B(14),
                                 Res => Y(14));

mux2_1bit15 : mux2_1bit PORT MAP( S0 => S(0),
                                 S1 => S(1),
                                 Cin =>B(15),
                                 Res => Y(15));

end Behavioral;
```


B_input_logic
B(15:0)
Y(15:0)
S(1:0)
reg_B_input_logic

1.9.2 MUX2 TO 1BIT

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


entity mux2_1bit is
   Port ( S0 : in STD_LOGIC;
          S1 : in STD_LOGIC;
          Cin : in STD_LOGIC;
          Res : out STD_LOGIC);
end mux2_1bit;

architecture Behavioral of mux2_1bit is

begin
Res <= S0 after 1ns when Cin = '1' else
        S1 after 1ns when Cin = '1' else
        '0' after 1ns;

end Behavioral;
```

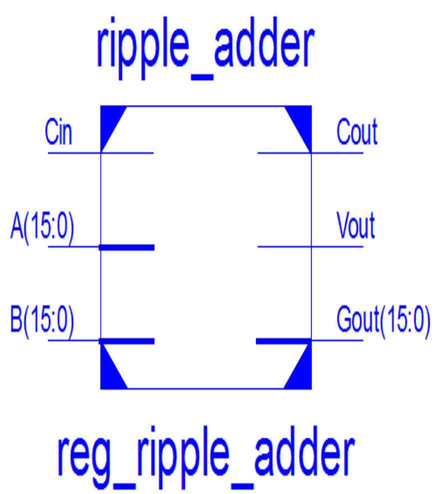


1.9.3 RIPPLE

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


entity ripple_adder 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;
           Gout : out STD_LOGIC_VECTOR(15 downto 0);
           Vout : out STD_LOGIC);
end ripple_adder;

architecture Behavioral of ripple_adder is

Component full_adder
PORT(X: in STD_LOGIC;
     Y: in STD_LOGIC;
     S: out STD_LOGIC;
     Cin: in STD_LOGIC;
     Cout: out STD_LOGIC);
End Component;

signal src_sig0, src_sig1, src_sig2, src_sig3, src_sig4,
 src_sig5, src_sig6, src_sig7, src_sig8, src_sig9, src_sig10,
 src_sig11, src_sig12, src_sig13, src_sig14, src_sig15,
 src_out: STD_LOGIC;
begin
full_adder00 : full_adder PORT MAP(X => A(0),
                                   Y => B(0),
                                   S => Gout(0),
                                   Cin => Cin,
                                   Cout =>src_sig0 );

full_adder01 : full_adder PORT MAP(X => A(1),
                                   Y => B(1),
                                   S => Gout(1),
                                   Cin => Cin,
                                   Cout =>src_sig1);

full_adder02 : full_adder PORT MAP(X => A(2),
                                   Y => B(2),
                                   S => Gout(2),
                                   Cin => Cin,
                                   Cout =>src_sig2);

full_adder03 : full_adder PORT MAP(X => A(3),
                                   Y => B(3),
                                   S => Gout(3),
                                   Cin => Cin,
                                   Cout =>src_sig3);
```

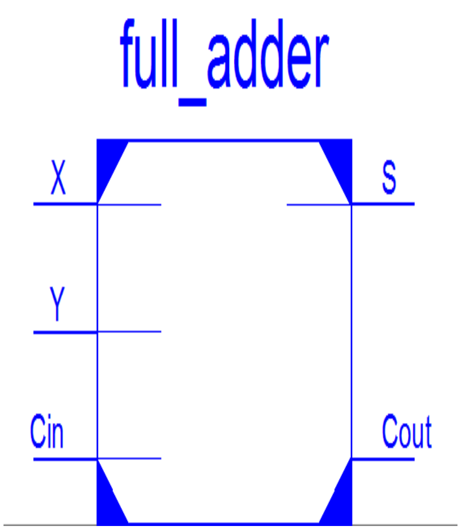
```
full_adder04 : full_adder PORT MAP(X => A(4),
                                Y => B(4),
                                S => Gout(4),
                                Cin => Cin,
                                Cout =>src_sig4);

full_adder05 : full_adder PORT MAP(X => A(5),
                                Y => B(5),
                                S => Gout(5),
                                Cin => Cin,
                                Cout =>src_sig5);

full_adder06 : full_adder PORT MAP(X => A(6),
                                Y => B(6),
                                S => Gout(6),
                                Cin => Cin,
                                Cout =>src_sig6);

full_adder07 : full_adder PORT MAP(X => A(7),
                                Y => B(7),
                                S => Gout(7),
                                Cin => Cin,
                                Cout =>src_sig7);

full_adder08 : full_adder PORT MAP(X => A(8),
                                Y => B(8),
                                S => Gout(8),
                                Cin => Cin,
                                Cout =>src_sig8);

full_adder09 : full_adder PORT MAP(X => A(9),
                                Y => B(9),
                                S => Gout(9),
                                Cin => Cin,
                                Cout =>src_sig9);

full_adder10 : full_adder PORT MAP(X => A(10),
                                Y => B(10),
                                S => Gout(10),
                                Cin => Cin,
                                Cout =>src_sig10);

full_adder11 : full_adder PORT MAP(X => A(11),
                                Y => B(11),
                                S => Gout(11),
                                Cin => Cin,
                                Cout =>src_sig11);

full_adder12 : full_adder PORT MAP(X => A(12),
                                Y => B(12),
                                S => Gout(12),
                                Cin => Cin,
                                Cout =>src_sig12);

full_adder13 : full_adder PORT MAP(X => A(13),
                                Y => B(13),
```

```
                                S => Gout(13),
                                Cin => Cin,
                                Cout =>src_sig13);

full_adder14 : full_adder PORT MAP(X => A(14),
                                Y => B(14),
                                S => Gout(14),
                                Cin => Cin,
                                Cout =>src_sig14);

full_adder15 : full_adder PORT MAP(X => A(15),
                                Y => B(15),
                                S => Gout(15),
                                Cin => Cin,
                                Cout =>src_sig15);



end Behavioral;
```



1.9.4 FULL ADDER

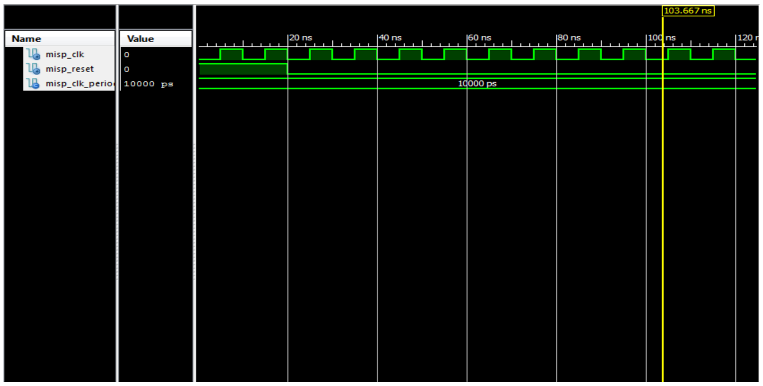
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;


entity full_adder is
PORT(X: in STD_LOGIC;
      Y: in STD_LOGIC;
      S: out STD_LOGIC;
      Cin: in STD_LOGIC;
      Cout: out STD_LOGIC);
end full_adder;

architecture Behavioral of full_adder is

   signal S1, S2, S3: STD_LOGIC;
begin
   S1 <= (X xor Y) after 1ns;
   S2 <= (Cin and S1) after 1ns;
   S3 <= (X and Y) after 1ns;
   S <= (S1 xor Cin) after 1ns;
   Cout <= (S2 or S3) after 1ns;

end Behavioral;
```



2 TESTBENCHES

2.1 Microcoded Instruction Set Processor

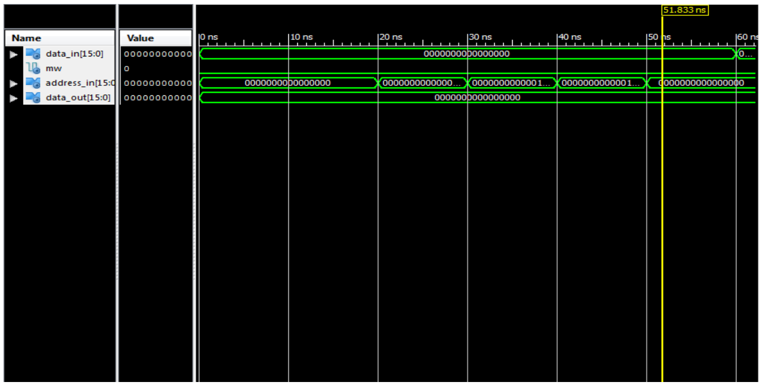
```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY test_misp IS
END test_misp;
ARCHITECTURE behavior OF test_misp IS

    COMPONENT microcoded_instruction_set_processor
    PORT(
         misp_clk : IN std_logic;
         misp_reset : IN std_logic
        );
    END COMPONENT;

   --Inputs
   signal misp_clk : std_logic := '0';
   signal misp_reset : std_logic := '0';

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

BEGIN
   -- Instantiate the Unit Under Test (UUT)
   uut: microcoded_instruction_set_processor PORT MAP (
          misp_clk => misp_clk,
          misp_reset => misp_reset
        );

   -- Clock process definitions
   misp_clk_process :process
   begin
      misp_clk <= '0';
      wait for misp_clk_period/2;
      misp_clk <= '1';
      wait for misp_clk_period/2;
   end process;

   -- Stimulus process
   stim_proc: process
   begin
      misp_reset <= '1';
      wait for 20 ns;
      misp_reset <= '0';
      wait;
   end process;
END;
```


103.667 ns
Name
Value
misp_clk
misp_reset
misp_clk_perio
0
0
10000 ps
20 ns
40 ns
60 ns
80 ns
100 ns
10000 ps

2.2 Memory M

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

ENTITY test_memory_m IS
END test_memory_m;

ARCHITECTURE behavior OF test_memory_m IS

    COMPONENT memory_m
    PORT(
         data_in : IN std_logic_vector(15 downto 0);
         mw : IN std_logic;
         address_in : IN std_logic_vector(15 downto 0);
         data_out : OUT std_logic_vector(15 downto 0)
        );
    END COMPONENT;


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

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

BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: memory_m PORT MAP (
          data_in => data_in,
          mw => mw,
          address_in => address_in,
          data_out => data_out);

   -- Stimulus process
   stim_proc: process
   begin
      wait for 10 ns;
      address_in <= x"0000";

      wait for 10 ns;
      address_in <= x"0001";

      wait for 10 ns;
      address_in <= x"0009";

      wait for 10 ns;
      address_in <= x"000A";

      wait for 10 ns;
      address_in <= x"0000";

      wait for 10 ns;
```

```
      data_in <= x"00B0";

      wait for 10 ns;
      data_in <= x"0000";

      wait;
   end process;

END;
```

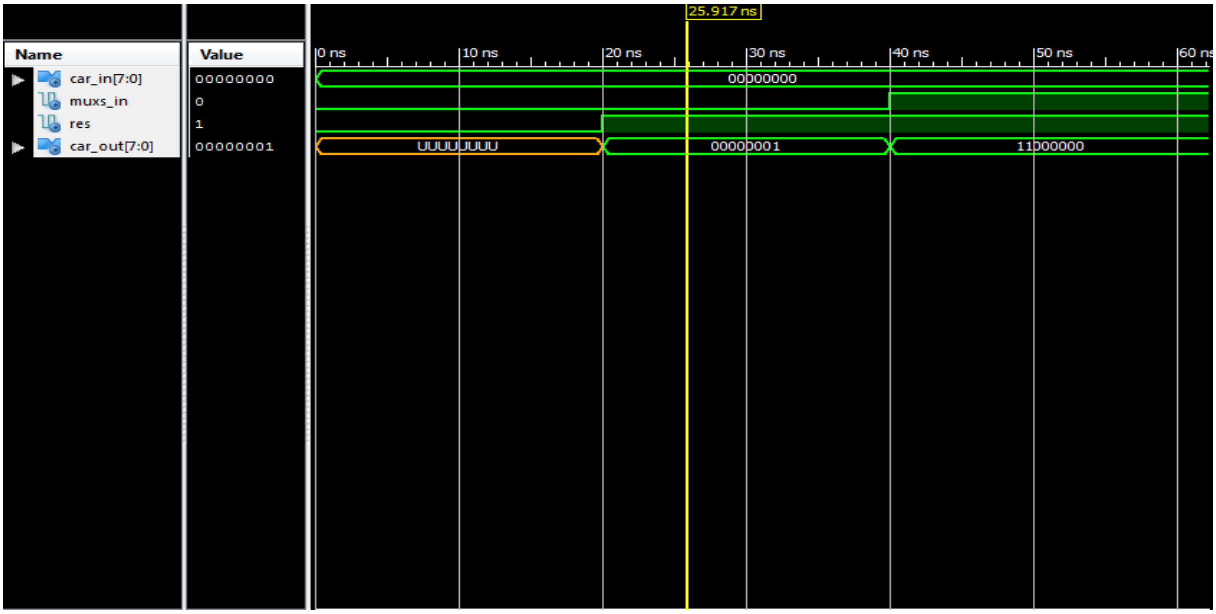


2.3 Micro Programmed Control

2.3.1 CAR

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

ENTITY test_CAR IS
END test_CAR;

ARCHITECTURE behavior OF test_CAR IS

    COMPONENT CAR
    PORT(
         car_in : IN std_logic_vector(7 downto 0);
         muxs_in : IN std_logic;
         res : IN std_logic;
         car_out : OUT std_logic_vector(7 downto 0)
        );
    END COMPONENT;


   --Inputs
   signal car_in : std_logic_vector(7 downto 0) := (others => '0');
   signal muxs_in : std_logic := '0';
   signal res : std_logic := '0';

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

BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: CAR PORT MAP (
          car_in => car_in,
          muxs_in => muxs_in,
          res => res,
          car_out => car_out
        );

   -- Stimulus process
   stim_proc: process
   begin

    wait for 10ns;
    res <='0';

    wait for 10ns;
    res <='1';

    wait for 10ns;
    muxs_in <='0';

    wait for 10ns;
    muxs_in <='1';

      wait;
   end process;
```

```
END;
```

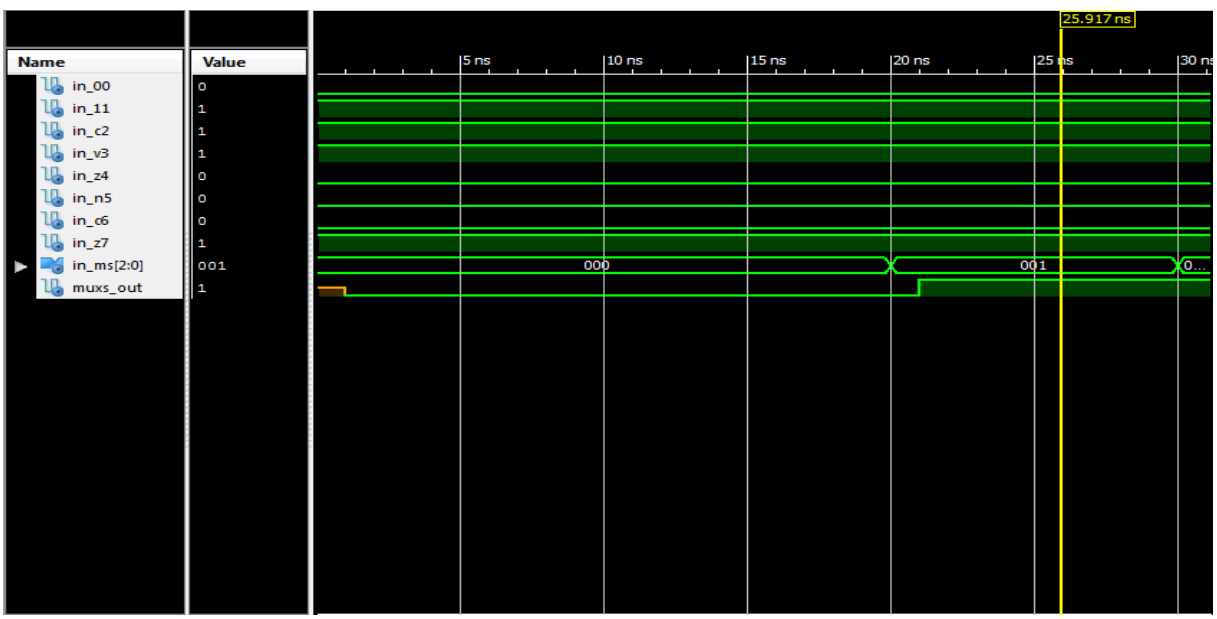


25.917 ns
Name
Value
car_in[7:0]
muxs_in
res
car_out[7:0]
00000000
0
1
00000001
0 ns
10 ns
20 ns
30 ns
40 ns
50 ns
UUUUUUUU
00000001
11000000

2.3.2 MUX8 TO 1BIT

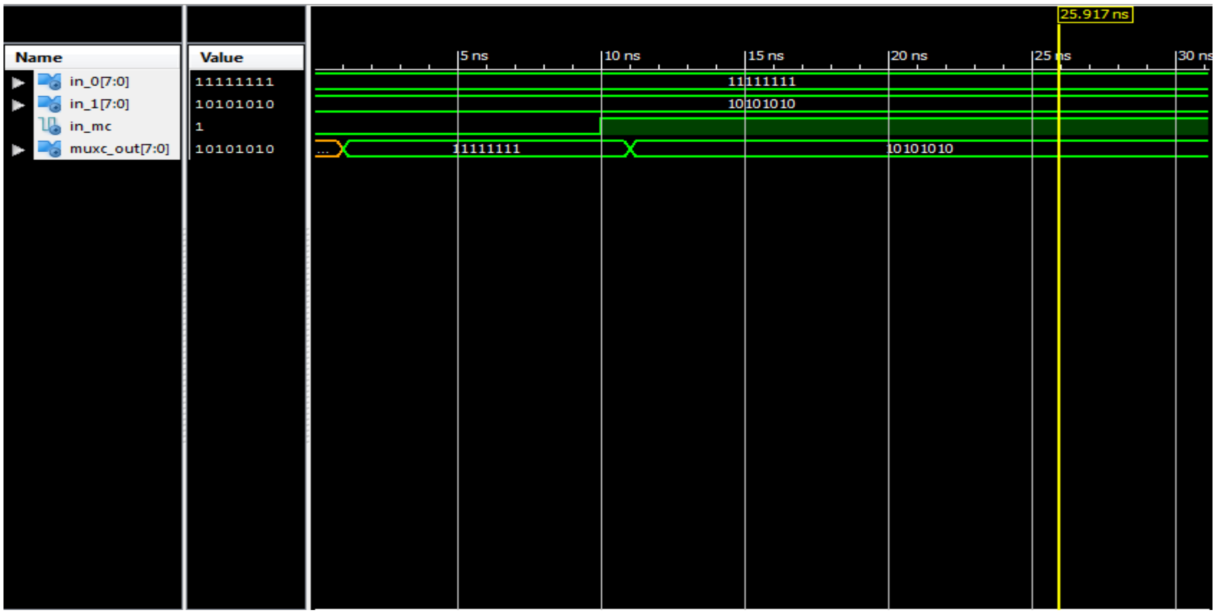
```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY test_mux8_1bit IS
END test_mux8_1bit;
ARCHITECTURE behavior OF test_mux8_1bit IS
   -- Component Declaration for the Unit Under Test (UUT)

   COMPONENT mux8_1bit
   PORT(
        in_00 : IN std_logic;
        in_11 : IN std_logic;
        in_C2 : IN std_logic;
        in_V3 : IN std_logic;
        in_Z4 : IN std_logic;
        in_N5 : IN std_logic;
        in_C6 : IN std_logic;
        in_Z7 : IN std_logic;
        in_MS : IN std_logic_vector(2 downto 0);
        MUXS_out : OUT std_logic
       );
   END COMPONENT;

  --Inputs
  signal in_00 : std_logic := '0';
  signal in_11 : std_logic := '0';
  signal in_C2 : std_logic := '0';
  signal in_V3 : std_logic := '0';
  signal in_Z4 : std_logic := '0';
  signal in_N5 : std_logic := '0';
  signal in_C6 : std_logic := '0';
  signal in_Z7 : std_logic := '0';
  signal in_MS : std_logic_vector(2 downto 0) := (others => '0');

  --Outputs
  signal MUXS_out : std_logic;
BEGIN
  -- Instantiate the Unit Under Test (UUT)
  uut: mux8_1bit PORT MAP (
         in_00 => in_00,
         in_11 => in_11,
         in_C2 => in_C2,
         in_V3 => in_V3,
         in_Z4 => in_Z4,
         in_N5 => in_N5,
         in_C6 => in_C6,
         in_Z7 => in_Z7,
         in_MS => in_MS,
         MUXS_out => MUXS_out
       );

  -- Stimulus process
  stim_proc: process
  begin
    in_11 <= '1';
```

```
      in_C2 <= '1';
      in_V3 <= '1';
      in_Z7 <= '1';

      wait for 10 ns;
       in_MS <= "000";

       wait for 10 ns;
       in_MS <= "001";

       wait for 10 ns;
       in_MS <= "010";

       wait for 10 ns;
       in_MS <= "011";

       wait for 10 ns;
       in_MS <= "100";

       wait for 10 ns;
       in_MS <= "001";

       wait for 10 ns;
       in_MS <= "010";

       wait for 10 ns;
       in_MS <= "111";
      wait;
   end process;
END;
```



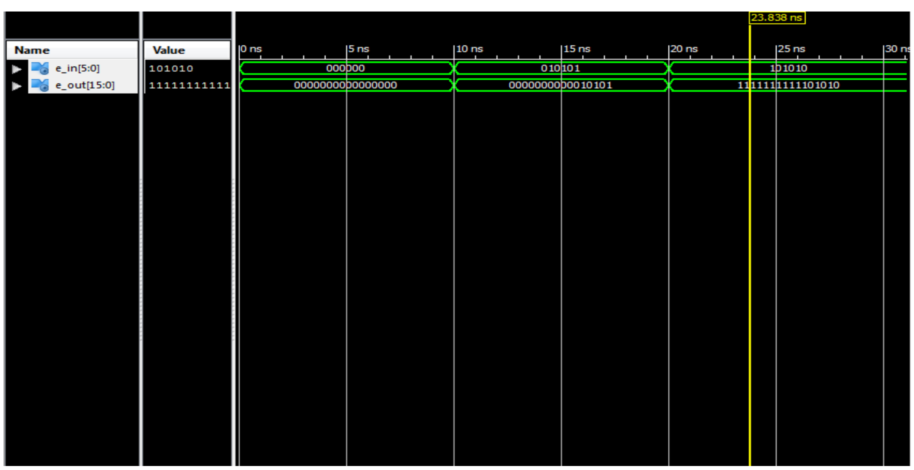
2.3.3 MUX2 TO 8BIT

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

ENTITY test_mux2_8bit IS
END test_mux2_8bit;

ARCHITECTURE behavior OF test_mux2_8bit IS

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

    COMPONENT mux2_8bit
    PORT(
         in_0 : IN std_logic_vector(7 downto 0);
         in_1 : IN std_logic_vector(7 downto 0);
         in_MC : IN std_logic;
         MUXC_out : OUT std_logic_vector(7 downto 0)
        );
    END COMPONENT;


   --Inputs
   signal in_0 : std_logic_vector(7 downto 0) := (others => '0');
   signal in_1 : std_logic_vector(7 downto 0) := (others => '0');
   signal in_MC : std_logic := '0';

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

BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: mux2_8bit PORT MAP (
          in_0 => in_0,
          in_1 => in_1,
          in_MC => in_MC,
          MUXC_out => MUXC_out
        );

   -- Stimulus process
   stim_proc: process
   begin
      in_0 <= x"FF";
      in_1 <= x"AA";

      wait for 10ns;
      in_MC <= '1';

      wait;
   end process;

END;
```


25.917 ns
Name
Value
in_0[7:0]
11111111
in_1[7:0]
10101010
in_mc
1
muxc_out[7:0]
10101010
5 ns
10 ns
15 ns
20 ns
25 ns
11111111
10101010
11111111
10101010

2.3.4 extend

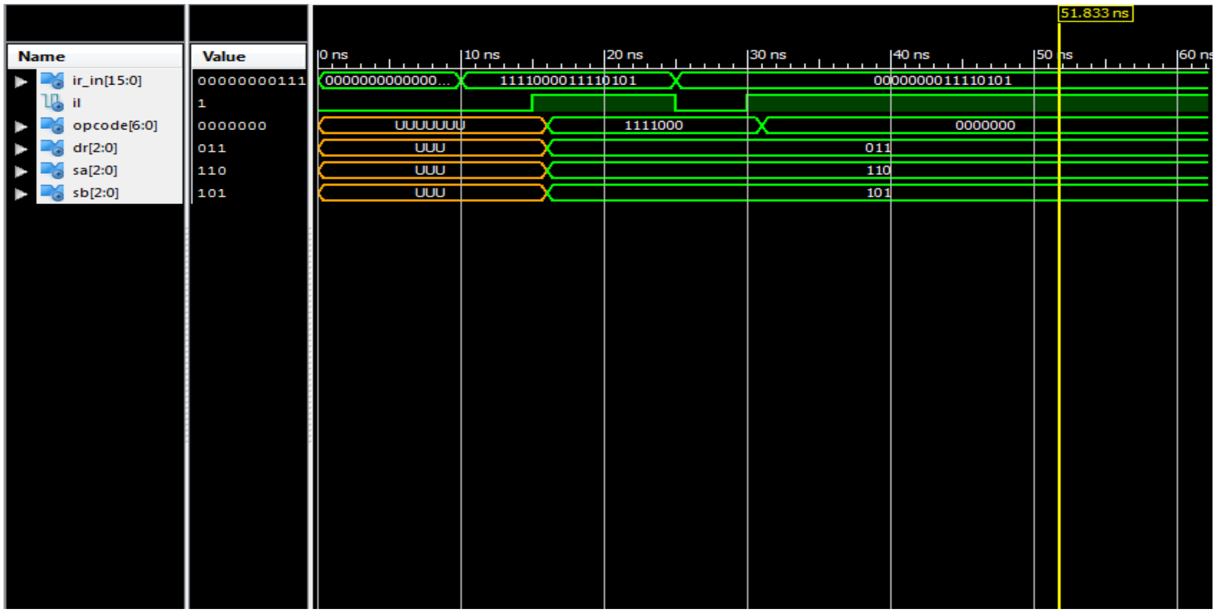
```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY test_extend IS
END test_extend;
ARCHITECTURE behavior OF test_extend IS
    -- Component Declaration for the Unit Under Test (UUT)
    COMPONENT extend
    PORT(
         E_in : IN std_logic_vector(5 downto 0);
         E_out : OUT std_logic_vector(15 downto 0)
        );
    END COMPONENT;

   --Inputs
   signal E_in : std_logic_vector(5 downto 0) := (others => '0');

   --Outputs
   signal E_out : std_logic_vector(15 downto 0);
BEGIN
   -- Instantiate the Unit Under Test (UUT)
   uut: extend PORT MAP (
          E_in => E_in,
          E_out => E_out);
   -- Stimulus process
   stim_proc: process
   begin
      wait for 10 ns;
      E_in <="010101";
      wait for 10 ns;
      E_in <="101010";
      wait;
   end process;
END;
```



2.3.5 IR

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

ENTITY test_IR IS
END test_IR;

ARCHITECTURE behavior OF test_IR IS

    COMPONENT IR
    PORT(
         IR_in : IN std_logic_vector(15 downto 0);
         IL : 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 IR_in : std_logic_vector(15 downto 0) := (others => '0');
   signal IL : 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);

BEGIN

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

   -- Stimulus process
   stim_proc: process
   begin
      wait for 10ns;
      IR_in <= "1111000011110101";

      wait for 5ns;
      IL <= '1';

      wait for 10ns;
      IR_in <= "0000000011110101";
       IL <= '0';

      wait for 5ns;
```

```
      IL <= '1';
      wait;
   end process;

END;
```



2.3.6 PC

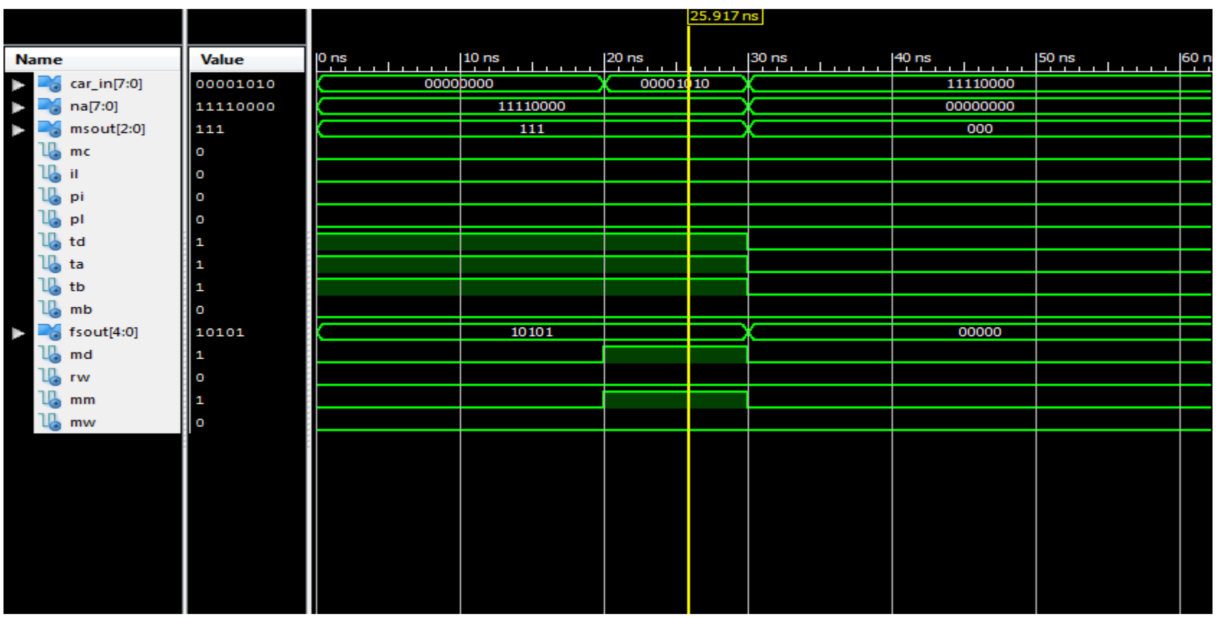
```vhdl
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY test_PC IS
END test_PC;

ARCHITECTURE behavior OF test_PC IS

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


   --Inputs
   signal PC_in : std_logic_vector(15 downto 0) := (others => '0');
   signal PL : std_logic := '0';
   signal PI : std_logic := '0';
   signal reset : std_logic := '0';

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

BEGIN

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

   -- Stimulus process
   stim_proc: process
   begin
    wait for 10ns;
    reset <= '1';
    PC_in <= x"0000";

     wait for 10ns;
    reset <= '0';

     wait for 10ns;
    PI <= '1';
    PC_in <= x"000A";

     wait for 10ns;
```

```
    reset <= '0';

    wait for 10ns;
    PI <= '0';
    PL <= '1';
    PC_in <= x"FFFF";

      wait;
   end process;

END;
```

2.3.7 Control Memory

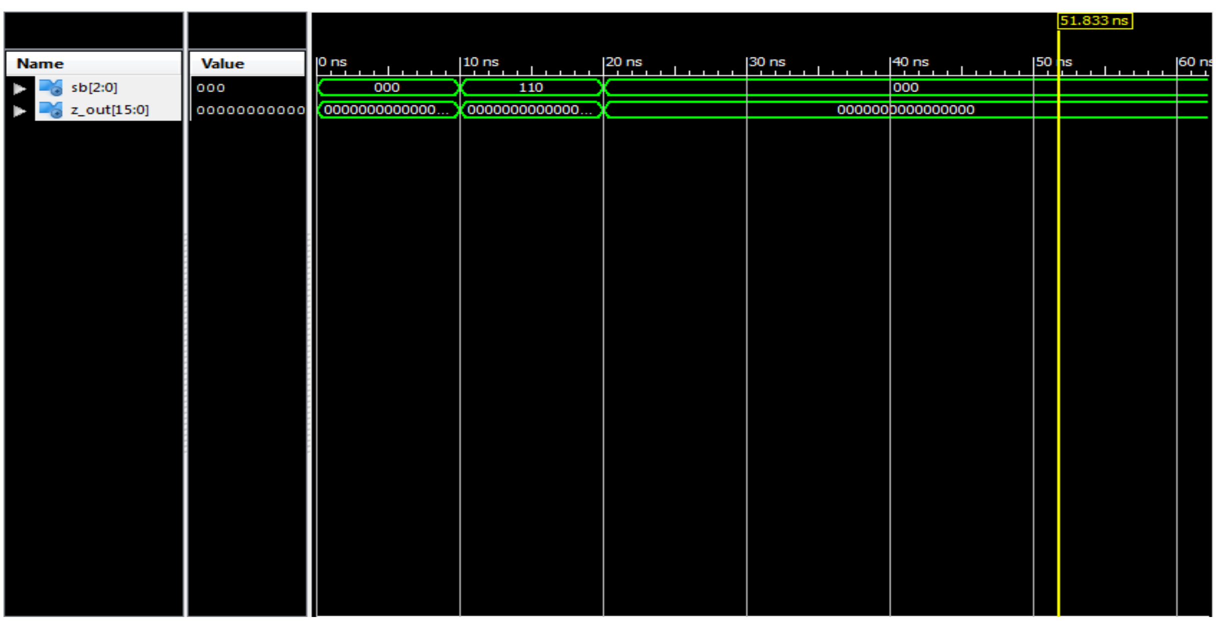
```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY test_control_memory IS
END test_control_memory;
ARCHITECTURE behavior OF test_control_memory IS
    -- Component Declaration for the Unit Under Test (UUT)
    COMPONENT control_memory
    PORT(
         car_in : IN std_logic_vector(7 downto 0);
         NA : OUT std_logic_vector(7 downto 0);
         MSout : OUT std_logic_vector(2 downto 0);
         MC : OUT std_logic;
         IL : OUT std_logic;
         PI : OUT std_logic;
         PL : OUT std_logic;
         TD : OUT std_logic;
         TA : OUT std_logic;
         TB : OUT std_logic;
         MB : OUT std_logic;
         FSout : 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 car_in : std_logic_vector(7 downto 0) := (others => '0');
   --Outputs
   signal NA : std_logic_vector(7 downto 0);
   signal MSout : std_logic_vector(2 downto 0);
   signal MC : std_logic;
   signal IL : std_logic;
   signal PI : std_logic;
   signal PL : std_logic;
   signal TD : std_logic;
   signal TA : std_logic;
   signal TB : std_logic;
   signal MB : std_logic;
   signal FSout : std_logic_vector(4 downto 0);
   signal MD : std_logic;
   signal RW : std_logic;
   signal MM : std_logic;
   signal MW : std_logic;
BEGIN
   -- Instantiate the Unit Under Test (UUT)
   uut: control_memory PORT MAP (
          car_in => car_in,
          NA => NA,
          MSout => MSout,
          MC => MC,
          IL => IL,
          PI => PI,
          PL => PL,
```

```
          TD => TD,
          TA => TA,
          TB => TB,
          MB => MB,
          FSout => FSout,
          MD => MD,
          RW => RW,
          MM => MM,
          MW => MW
        );
   -- Stimulus process
   stim_proc: process
   begin
      wait for 10ns;
      car_in <=x"00";

      wait for 10ns;
      car_in <=x"0A";

      wait for 10ns;
      car_in <=x"F0";
     wait;
   end process;
END;
```



2.3.8 zero fill

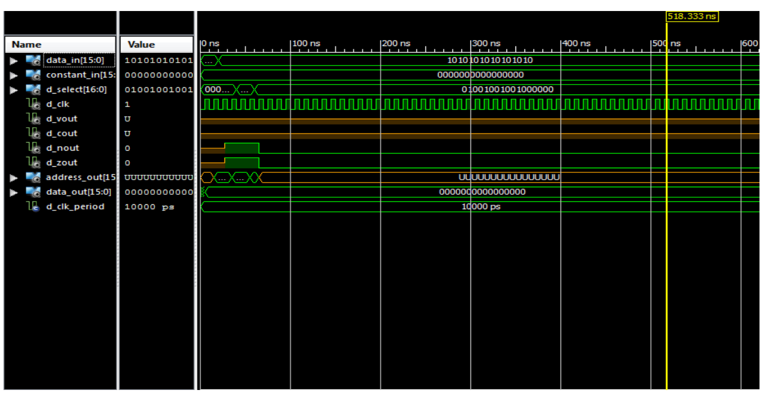
```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY test_zero_fill IS
END test_zero_fill;
ARCHITECTURE behavior OF test_zero_fill IS
    COMPONENT zero_fill
    PORT(
         sb : IN std_logic_vector(2 downto 0);
         z_out : OUT std_logic_vector(15 downto 0)
        );
    END COMPONENT;
   --Inputs
   signal sb : std_logic_vector(2 downto 0) := (others => '0');
   --Outputs
   signal z_out : std_logic_vector(15 downto 0);
BEGIN
   -- Instantiate the Unit Under Test (UUT)
   uut: zero_fill PORT MAP (
          sb => sb,
          z_out => z_out
        );
   -- Stimulus process
   stim_proc: process
   begin
      wait for 10ns;
      sb <= "110";
      wait for 10ns;
      sb <= "000";
      wait;
   end process;
END;
```



2.4 DATA PATH

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



ENTITY test_datapath IS
END test_datapath;

ARCHITECTURE behavior OF test_datapath IS

    COMPONENT datapath
    PORT(
         data_in : IN std_logic_vector(15 downto 0);
         constant_in : IN std_logic_vector(15 downto 0);
         D_select : IN std_logic_vector(16 downto 0);
         D_Clk : IN std_logic;
         D_Vout : OUT std_logic;
         D_Cout : OUT std_logic;
         D_Nout : OUT std_logic;
         D_Zout : OUT std_logic;
         address_out : OUT std_logic_vector(15 downto 0);
         data_out : OUT std_logic_vector(15 downto 0)
        );
    END COMPONENT;


   --Inputs
   signal data_in : std_logic_vector(15 downto 0) := (others => '0');
   signal constant_in : std_logic_vector(15 downto 0) := (others => '0');
   signal D_select : std_logic_vector(16 downto 0) := (others => '0');
   signal D_Clk : std_logic := '0';

   --Outputs
   signal D_Vout : std_logic;
   signal D_Cout : std_logic;
   signal D_Nout : std_logic;
   signal D_Zout : std_logic;
   signal address_out : std_logic_vector(15 downto 0);
   signal data_out : std_logic_vector(15 downto 0);

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

BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: datapath PORT MAP (
          data_in => data_in,
          constant_in => constant_in,
          D_select => D_select,
          D_Clk => D_Clk,
          D_Vout => D_Vout,
          D_Cout => D_Cout,
          D_Nout => D_Nout,
```

```vhdl
        D_Zout => D_Zout,
        address_out => address_out,
        data_out => data_out
      );

   -- Clock process definitions
   D_Clk_process :process
   begin
      D_Clk <= '0';
      wait for D_Clk_period/2;
      D_Clk <= '1';
      wait for D_Clk_period/2;
   end process;


   -- Stimulus process
   stim_proc: process
   begin
      data_in <= x"FFFF";
      constant_in <= x"0000";
      D_select <= "00000000100000011";

      wait for 20ns;
      data_in <= x"AAAA";
      D_select <= "00000000100000011";

      wait for 20ns;
      D_select <= "00000000100110001";

      wait for 20ns;
      D_select <= "01001001001000000";



      wait;
   end process;

END;
```

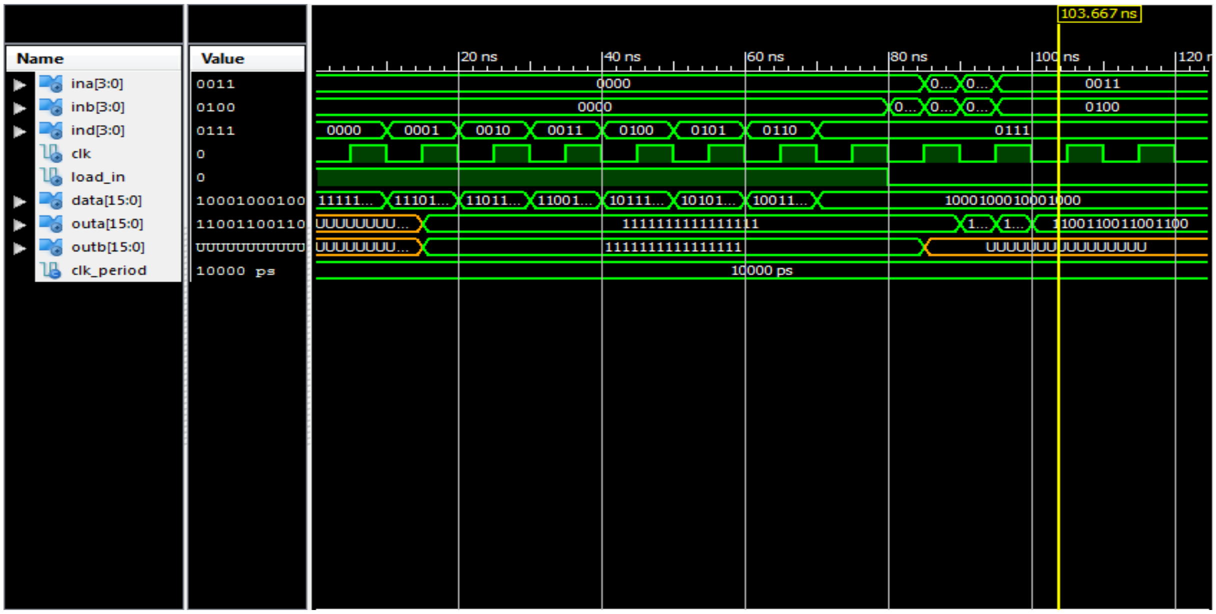


2.5 REGISTER FILE

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

ENTITY test_register_file IS
END test_register_file;

ARCHITECTURE behavior OF test_register_file IS

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

    COMPONENT register_file
    PORT(
         inA : IN std_logic_vector(3 downto 0);
         inB : IN std_logic_vector(3 downto 0);
         inD : IN std_logic_vector(3 downto 0);
         Clk : IN std_logic;
         load_in : IN std_logic;
         data : IN std_logic_vector(15 downto 0);
         outA : OUT std_logic_vector(15 downto 0);
         outB : OUT std_logic_vector(15 downto 0)
        );
    END COMPONENT;


   --Inputs
   signal inA : std_logic_vector(3 downto 0) := (others => '0');
   signal inB : std_logic_vector(3 downto 0) := (others => '0');
   signal inD : std_logic_vector(3 downto 0) := (others => '0');
   signal Clk : std_logic := '0';
   signal load_in : std_logic := '0';
   signal data : std_logic_vector(15 downto 0) := (others => '0');

   --Outputs
   signal outA : std_logic_vector(15 downto 0);
   signal outB : std_logic_vector(15 downto 0);

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

BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: register_file PORT MAP (
          inA => inA,
          inB => inB,
          inD => inD,
          Clk => Clk,
          load_in => load_in,
          data => data,
          outA => outA,
          outB => outB
        );

   -- Clock process definitions
```

```vhdl
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
 load_in <= '1';
 inD <= "0000";
 data <= x"FFFF";

 wait for 10ns;
 inD <= "0001";
 data <= x"EEEE";

 wait for 10ns;
 inD <= "0010";
 data <= x"DDDD";

 wait for 10ns;
 inD <= "0011";
 data <= x"CCCC";

 wait for 10ns;
 inD <= "0100";
 data <= x"BBBB";

    wait for 10ns;
 inD <= "0101";
 data <= x"AAAA";

 wait for 10ns;
 inD <= "0110";
 data <= x"9999";

 wait for 10ns;
 inD <= "0111";
 data <= x"8888";

 wait for 10ns;
 load_in <= '0';
 inA <= "0000";
 inB <= "0111";

 wait for 5ns;
 inA <= "0001";
 inB <= "0110";

 wait for 5ns;
 inA <= "0010";
 inB <= "0101";
```

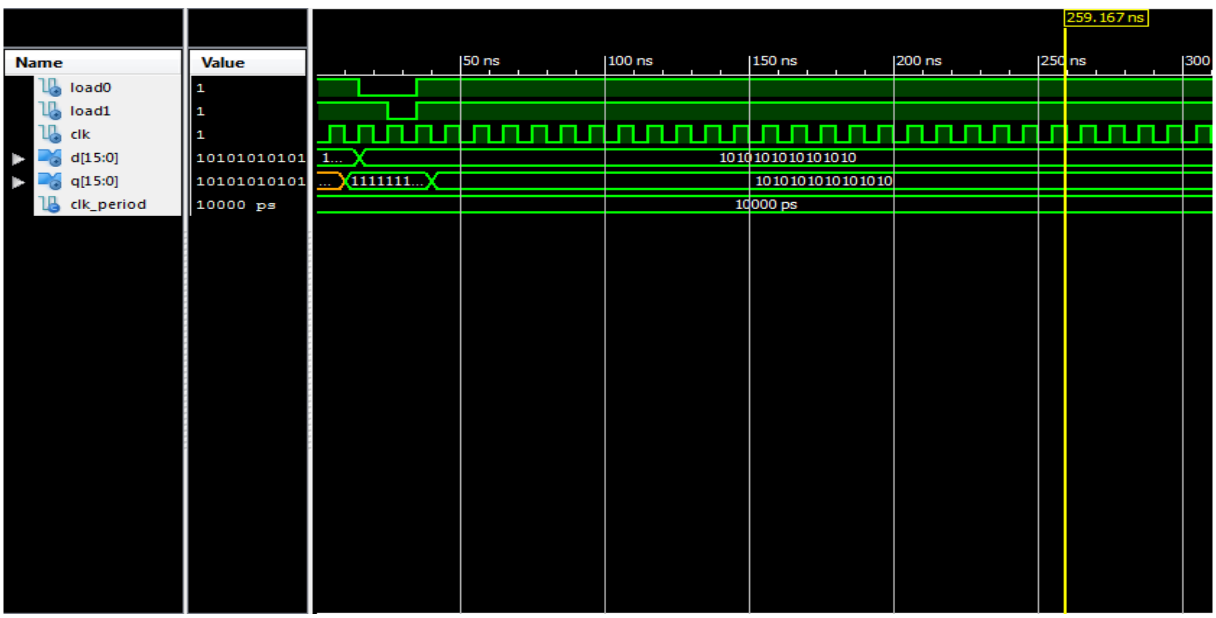
```
    wait for 5ns;
    inA <= "0011";
    inB <= "0100";

      wait;
   end process;

END;
```



2.5.1 REG8

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


ENTITY test_reg8 IS
END test_reg8;

ARCHITECTURE behavior OF test_reg8 IS

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

    COMPONENT reg8
    PORT(
         load0 : IN std_logic;
         load1 : IN std_logic;
         Clk : IN std_logic;
         D : IN std_logic_vector(15 downto 0);
         Q : OUT std_logic_vector(15 downto 0)
        );
    END COMPONENT;


   --Inputs
   signal load0 : std_logic := '0';
   signal load1 : std_logic := '0';
   signal Clk : std_logic := '0';
   signal D : std_logic_vector(15 downto 0) := (others => '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: reg8 PORT MAP (
          load0 => load0,
          load1 => load1,
          Clk => Clk,
          D => D,
          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;
```

```vhdl
   -- Stimulus process
   stim_proc: process
   begin
      D <= x"FFFF";
      load0 <= '1';
      load1 <= '1';

      wait for 15ns;
      D <= x"AAAA";
      load0 <= '0';

      wait for 10ns;
      load1 <= '0';

      wait for 10ns;
      load0 <= '1';
      load1 <= '1';

      wait;
   end process;

END;
```

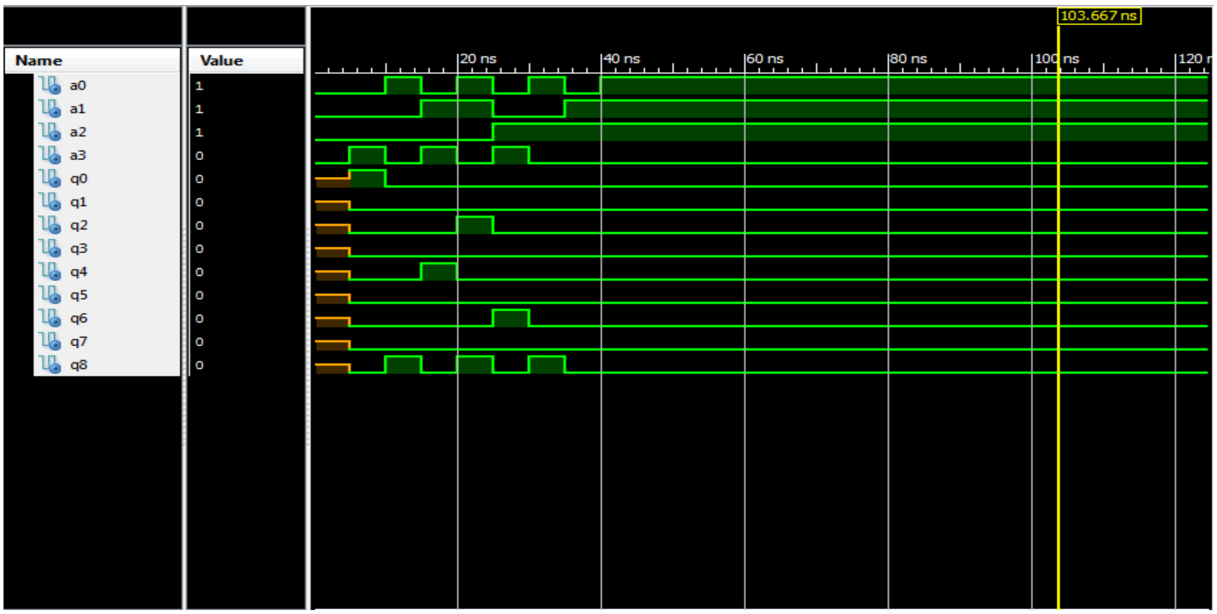


2.5.2 DECODER

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


ENTITY test_decoder_3to8 IS
END test_decoder_3to8;

ARCHITECTURE behavior OF test_decoder_3to8 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;
         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
        );
    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;


BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: decoder_3to8 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
    );


stim_proc: process
begin

   wait for 5ns;
      A0 <= '0';
      A1 <= '0';
      A2 <= '0';
      A3 <= '1';

   wait for 5ns;
      A0 <= '1';
      A1 <= '0';
      A2 <= '0';
       A3 <= '0';

   wait for 5ns;
      A0 <= '0';
      A1 <= '1';
      A2 <= '0';
       A3 <= '1';

   wait for 5ns;
      A0 <= '1';
      A1 <= '1';
      A2 <= '0';
       A3 <= '0';

   wait for 5ns;
      A0 <= '0';
      A1 <= '0';
      A2 <= '1';
       A3 <= '1';

   wait for 5ns;
      A0 <= '1';
      A1 <= '0';
      A2 <= '1';
       A3 <= '0';

   wait for 5ns;
      A0 <= '0';
      A1 <= '1';
```

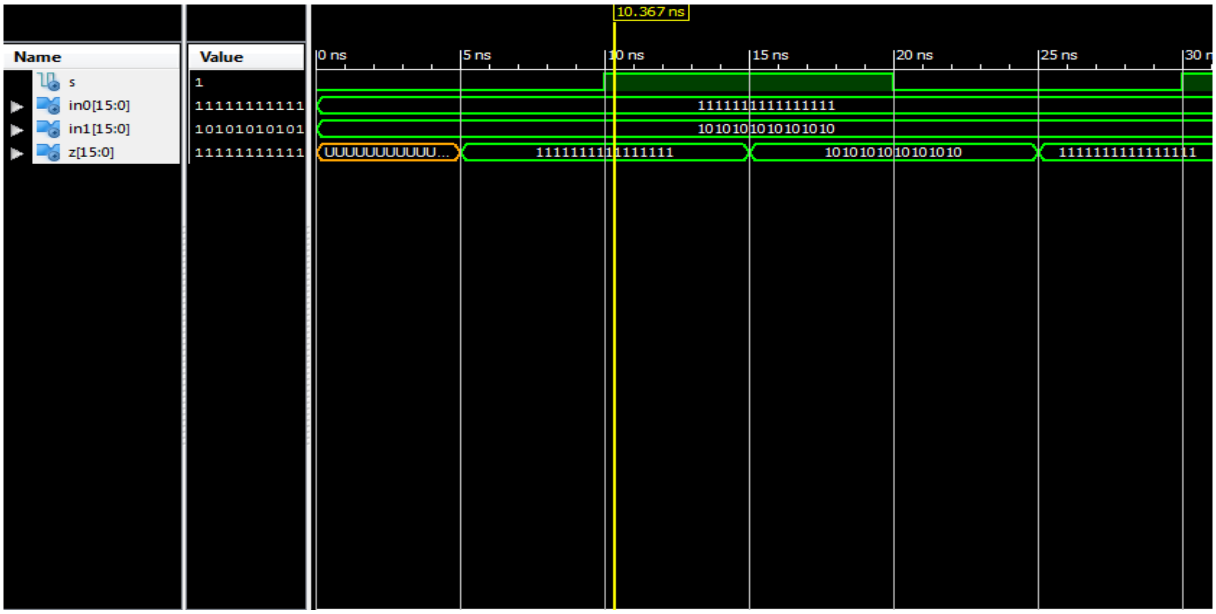
```
        A2 <= '1';
         A3 <= '0';

      wait for 5ns;
        A0 <= '1';
        A1 <= '1';
        A2 <= '1';
         A3 <= '0';

      wait;
   end process;

END;
```



2.5.3 MUX3 TO 16BIT

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



ENTITY test_mux3_16bit IS
END test_mux3_16bit;

ARCHITECTURE behavior OF test_mux3_16bit IS

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

    COMPONENT mux3_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);
   -- No clocks detected in port list. Replace <clock> below with
   -- appropriate port name

BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: mux3_16bit PORT MAP (
          s => s,
          In0 => In0,
          In1 => In1,
          Z => Z
        );

   -- Stimulus process
   stim_proc: process
   begin
          In0 <= x"FFFF";
          In1 <= x"AAAA";

          wait for 10ns;
          s <= '1';
          wait for 10ns;
          s <= '0';
          wait for 10ns;
```

```
        s <= '1';
    wait;
  end process;

END;
```

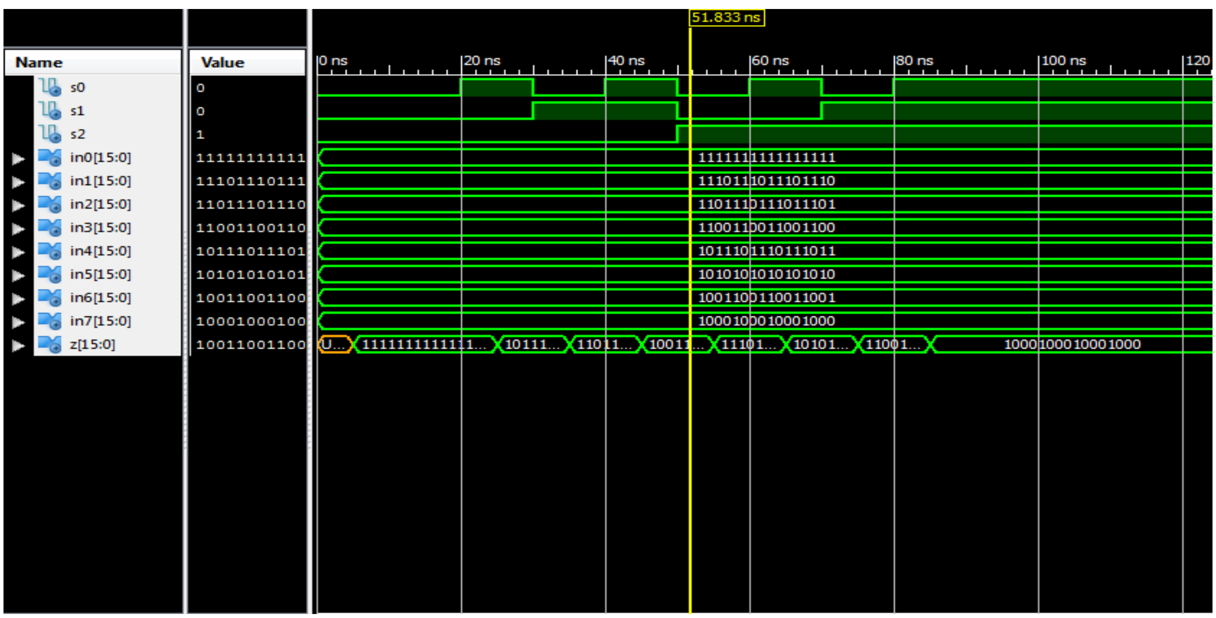


2.5.4 MUX8 TO 16BIT

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


ENTITY test_mux8_16bit IS
END test_mux8_16bit;

ARCHITECTURE behavior OF test_mux8_16bit 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;
         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);
         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');

   --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,
       S3 => S3,
      In0 => In0,
      In1 => In1,
      In2 => In2,
      In3 => In3,
      In4 => In4,
      In5 => In5,
      In6 => In6,
      In7 => In7,
      In8 => In8,
      Z => Z
    );

stim_proc: process
begin
   In0 <= x"FFFF";
   In1 <= x"EEEE";
   In2 <= x"DDDD";
   In3 <= x"CCCC";
   In4 <= x"BBBB";
   In5 <= x"AAAA";
   In6 <= x"9999";
   In7 <= x"8888";
   In8 <= x"7777";

   wait for 10ns;
    S0 <='0';
    S1 <='0';
    S2 <='0';
    S3 <='0';

   wait for 10ns;
    S0 <='1';
    S1 <='0';
    S2 <='0';
    S3 <='1';

   wait for 10ns;
    S0 <='0';
    S1 <='1';
    S2 <='0';
    S3 <='0';

   wait for 10ns;
    S0 <='1';
    S1 <='1';
    S2 <='0';
    S3 <='1';

   wait for 10ns;
    S0 <='0';
    S1 <='0';
    S2 <='1';
    S3 <='0';
```

```
      wait for 10ns;
       S0 <='1';
       S1 <='0';
       S2 <='1';
       S3 <='0';

      wait for 10ns;
       S0 <='0';
       S1 <='1';
       S2 <='1';
       S3 <='0';

      wait for 10ns;
       S0 <='1';
       S1 <='1';
       S2 <='1';
      S3 <='0';

      wait;
   end process;

END;
```



2.6 FUNCTION UNIT

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



ENTITY test_function_unit IS
END test_function_unit;

ARCHITECTURE behavior OF test_function_unit IS

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

    COMPONENT function_unit
    PORT(
         F_A : IN std_logic_vector(15 downto 0);
         F_B : IN std_logic_vector(15 downto 0);
         F_select : IN std_logic_vector(4 downto 0);
         F_Vout : OUT std_logic;
         F_Cout : OUT std_logic;
         F_Nout : OUT std_logic;
         F_Zout : OUT std_logic;
         F_F : OUT std_logic_vector(15 downto 0)
        );
    END COMPONENT;


   --Inputs
   signal F_A : std_logic_vector(15 downto 0) := (others => '0');
   signal F_B : std_logic_vector(15 downto 0) := (others => '0');
   signal F_select : std_logic_vector(4 downto 0) := (others => '0');

   --Outputs
   signal F_Vout : std_logic;
   signal F_Cout : std_logic;
   signal F_Nout : std_logic;
   signal F_Zout : std_logic;
   signal F_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: function_unit PORT MAP (
          F_A => F_A,
          F_B => F_B,
          F_select => F_select,
          F_Vout => F_Vout,
          F_Cout => F_Cout,
          F_Nout => F_Nout,
          F_Zout => F_Zout,
          F_F => F_F
        );
```

```vhdl
-- Stimulus process
stim_proc: process
begin

   F_A <= x"AAAA";
   F_B <= x"BBBB";

   wait for 20ns;
   F_select <= "00000";

   wait for 10ns;
   F_select <= "00001";
   wait for 10ns;
   F_select <= "00010";
   wait for 10ns;
   F_select <= "00011";
   wait for 20ns;
   F_select <= "00100";

   wait for 20ns;
   F_select <= "00101";
   wait for 20ns;
   F_select <= "00110";

   wait for 10ns;
   F_select <= "00111";
   wait for 10ns;
   F_select <= "01000";

   wait for 10ns;
   F_select <= "01001";
   wait for 10ns;
   F_select <= "01010";

   wait for 10ns;
   F_select <= "01011";
   wait for 10ns;
   F_select <= "01100";


   wait for 10ns;
   F_select <= "01101";
   wait for 10ns;
   F_select <= "01110";

   wait for 10ns;
   F_select <= "01111";
   wait for 10ns;
   F_select <= "10000";

   wait for 10ns;
   F_select <= "10001";
   wait for 10ns;
   F_select <= "10010";
```

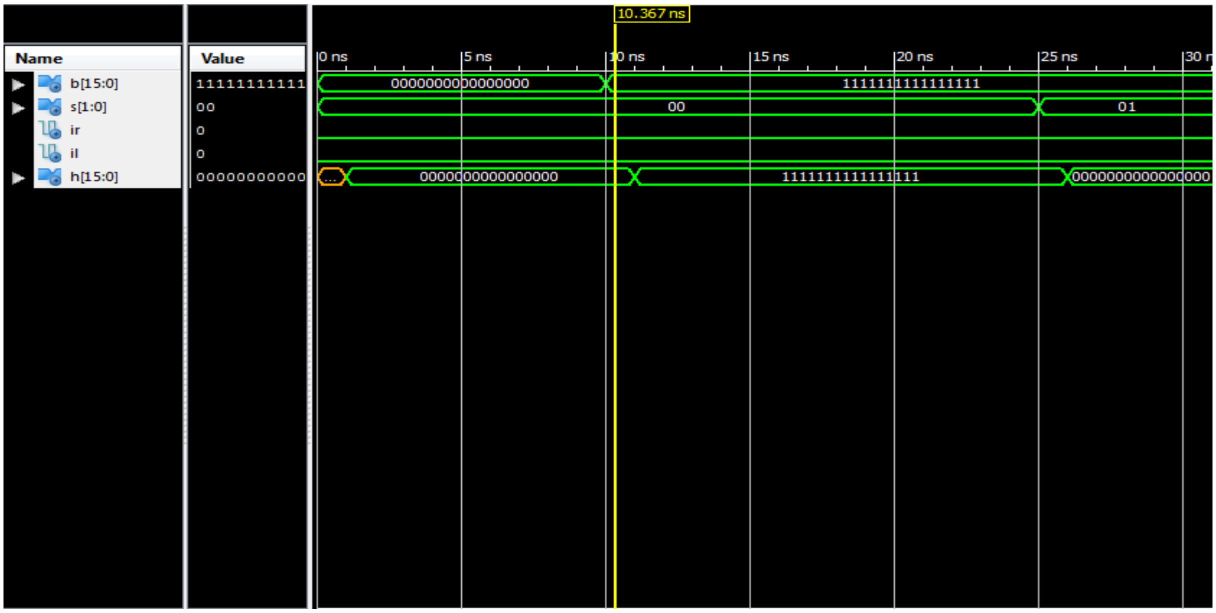
```
    wait for 10ns;
    F_select <= "10011";
    wait for 10ns;
    F_select <= "10100";

    wait for 10ns;
    F_select <= "10101";
    wait for 10ns;
    F_select <= "10110";

    wait for 10ns;
    F_select <= "10111";
    wait for 10ns;
    F_select <= "11000";


    wait for 10ns;
    F_select <= "11001";
    wait for 10ns;
    F_select <= "11010";

    wait for 10ns;
    F_select <= "11011";
    wait for 10ns;
    F_select <= "11100";

    wait for 10ns;
    F_select <= "11101";
    wait for 10ns;
    F_select <= "11110";
    wait for 10ns;
    F_select <= "11111";

    wait;
  end process;

END;
```


51.833 ns
Name
Value
f_a[15:0]
f_b[15:0]
f_select[4:0]
f_vout
f_cout
f_nout
f_zout
f_f[15:0]
10101010101
10111011101
00011
U
U
0
1
01010101010
0 ns
50 ns
100 ns
150 ns
200 ns
250 ns
300
1010101010101010
1011101110111011
00000
00011
00100
00101

2.7 SHIFTER

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

ENTITY test_shifter IS
END test_shifter;

ARCHITECTURE behavior OF test_shifter IS

    COMPONENT shifter
    PORT(
         B : IN std_logic_vector(15 downto 0);
         S : IN std_logic_vector(1 downto 0);
         IR : IN std_logic;
         IL : IN std_logic;
         H : OUT std_logic_vector(15 downto 0)
        );
    END COMPONENT;


   --Inputs
   signal B : std_logic_vector(15 downto 0) := (others => '0');
   signal S : std_logic_vector(1 downto 0) := (others => '0');
   signal IR : std_logic := '0';
   signal IL : std_logic := '0';

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

BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: shifter PORT MAP (
          B => B,
          S => S,
          IR => IR,
          IL => IL,
          H => H
        );

   stim_proc: process
   begin

    wait for 10ns;
    B <= x"FFFF";
    S <= "00";

    wait for 15ns;
    s <= "01";

    wait for 15ns;
    B <= H;

    wait for 15ns;
```

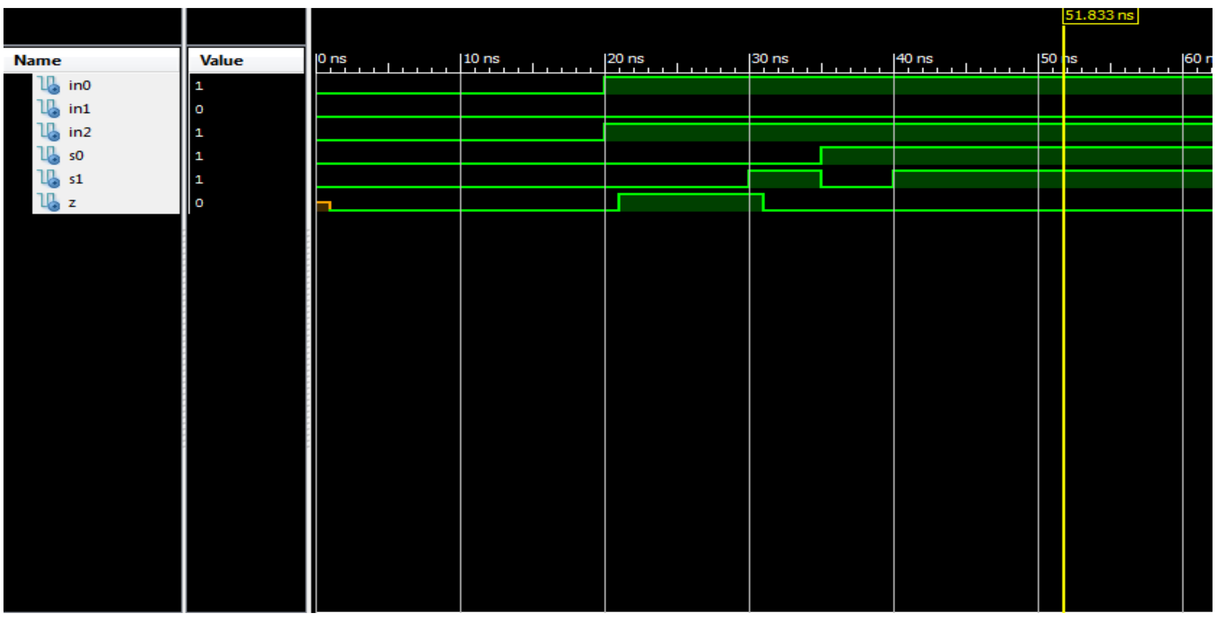
```
    B <= H;
    S <= "10";

      wait;
   end process;

END;
```



2.7.1 MUX3 TO 1BIT

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

ENTITY test_mux3_1bit IS
END test_mux3_1bit;

ARCHITECTURE behavior OF test_mux3_1bit IS

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

    COMPONENT mux3_1bit
    PORT(
         In0 : IN std_logic;
         In1 : IN std_logic;
         In2 : IN std_logic;
         S0 : IN std_logic;
         S1 : IN std_logic;
         Z : OUT std_logic
        );
    END COMPONENT;


   --Inputs
   signal In0 : std_logic := '0';
   signal In1 : std_logic := '0';
   signal In2 : std_logic := '0';
   signal S0 : std_logic := '0';
   signal S1 : std_logic := '0';

   --Outputs
   signal Z : std_logic;


BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: mux3_1bit PORT MAP (
          In0 => In0,
          In1 => In1,
          In2 => In2,
          S0 => S0,
          S1 => S1,
          Z => Z
        );

   stim_proc: process
   begin
      wait for 10ns;
      In0 <= '0';
      In1 <= '0';
      In2 <= '0';

      wait for 10ns;
```

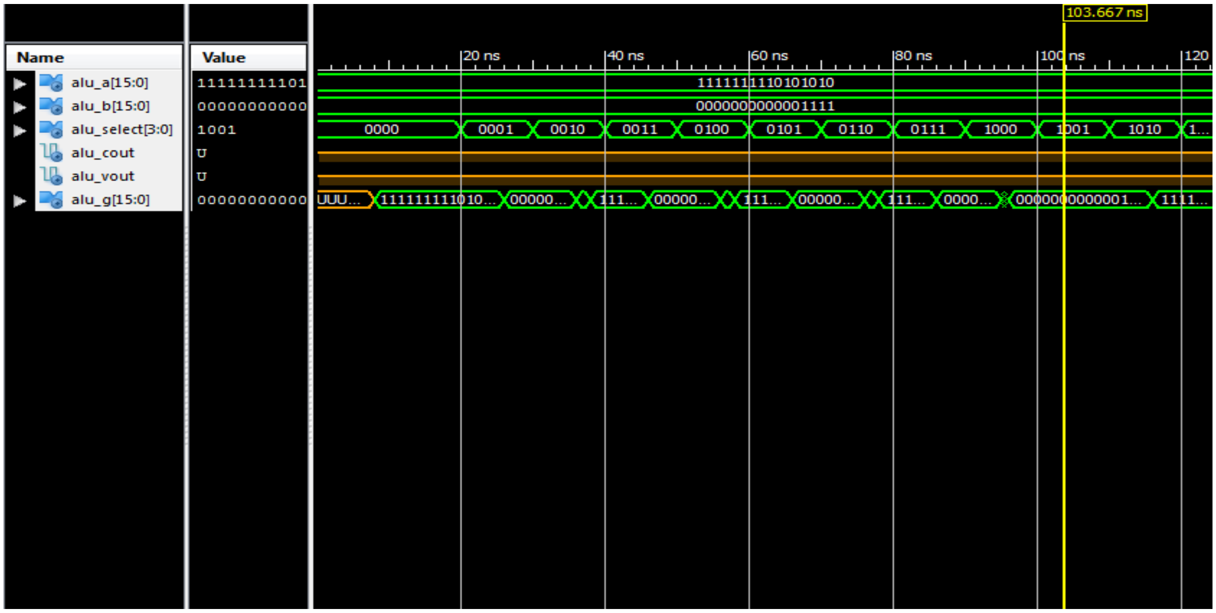
```
      In0 <= '1';
      In1 <= '0';
      In2 <= '1';

      wait for 5ns;
      S0 <= '0';
      S1 <= '0';

      wait for 5ns;
      S0 <= '0';
      S1 <= '1';

      wait for 5ns;
      S0 <= '1';
      S1 <= '0';

      wait for 5ns;
      S0 <= '1';
      S1 <= '1';
      wait;
   end process;

END;
```



2.8 ALU

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

ENTITY test_alu IS
END test_alu;

ARCHITECTURE behavior OF test_alu IS

    COMPONENT arithmetic_logic_unit
    PORT(
         ALU_A : IN std_logic_vector(15 downto 0);
         ALU_B : IN std_logic_vector(15 downto 0);
         ALU_select : IN std_logic_vector(3 downto 0);
         ALU_Cout : OUT std_logic;
         ALU_Vout : OUT std_logic;
         ALU_G : OUT std_logic_vector(15 downto 0)
        );
    END COMPONENT;


   --Inputs
   signal ALU_A : std_logic_vector(15 downto 0) := (others => '0');
   signal ALU_B : std_logic_vector(15 downto 0) := (others => '0');
   signal ALU_select : std_logic_vector(3 downto 0) := (others => '0');

   --Outputs
   signal ALU_Cout : std_logic;
   signal ALU_Vout : std_logic;
   signal ALU_G : std_logic_vector(15 downto 0);

BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: arithmetic_logic_unit PORT MAP (
          ALU_A => ALU_A,
          ALU_B => ALU_B,
          ALU_select => ALU_select,
          ALU_Cout => ALU_Cout,
          ALU_Vout => ALU_Vout,
          ALU_G => ALU_G
        );

   stim_proc: process
   begin
    ALU_A <= x"FFAA";
    ALU_B <= x"000F";
    ALU_select <= "0000";

    wait for 10ns;
    ALU_select <= "0000";

    wait for 10ns;
    ALU_select <= "0001";
```

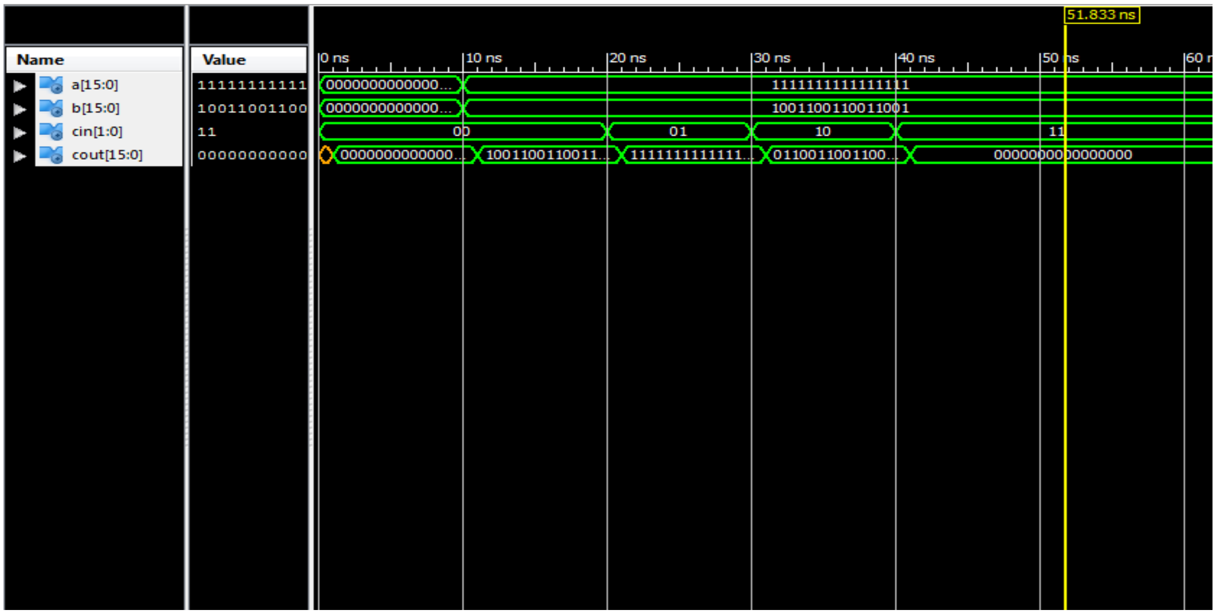
```
    wait for 10ns;
    ALU_select <= "0010";

    wait for 10ns;
    ALU_select <= "0011";

    wait for 10ns;
    ALU_select <= "0100";

    wait for 10ns;
    ALU_select <= "0101";

    wait for 10ns;
    ALU_select <= "0110";

    wait for 10ns;
    ALU_select <= "0111";

    wait for 10ns;
    ALU_select <= "1000";

    wait for 10ns;
    ALU_select <= "1001";

    wait for 10ns;
    ALU_select <= "1010";

    wait for 10ns;
    ALU_select <= "1011";

    wait for 10ns;
    ALU_select <= "1100";

    wait for 10ns;
    ALU_select <= "1101";

    wait for 10ns;
    ALU_select <= "1110";

    wait for 10ns;
    ALU_select <= "1111";

      wait;
   end process;

END;
```


103.667 ns
Name
Value
alu_a[15:0]
alu_b[15:0]
alu_select[3:0]
alu_cout
alu_vout
alu_g[15:0]
20 ns
40 ns
60 ns
80 ns
100 ns
0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010

2.8.1 LOGIC

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

ENTITY test_logic IS
END test_logic;

ARCHITECTURE behavior OF test_logic IS

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

    COMPONENT logic_circuit
    PORT(
         A : IN std_logic_vector(15 downto 0);
         B : IN std_logic_vector(15 downto 0);
         Cin : IN std_logic_vector(1 downto 0);
         Cout : 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_vector(1 downto 0) := (others => '0');

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

BEGIN

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

   stim_proc: process
   begin
      wait for 10ns;
       A <= x"FFFF";
       B <= x"9999";
       Cin <= "00";

       wait for 10ns;
       Cin <= "01";

       wait for 10ns;
       Cin <= "10";

       wait for 10ns;
       Cin <= "11";
```

```vhdl
      wait;
   end process;

END;
```



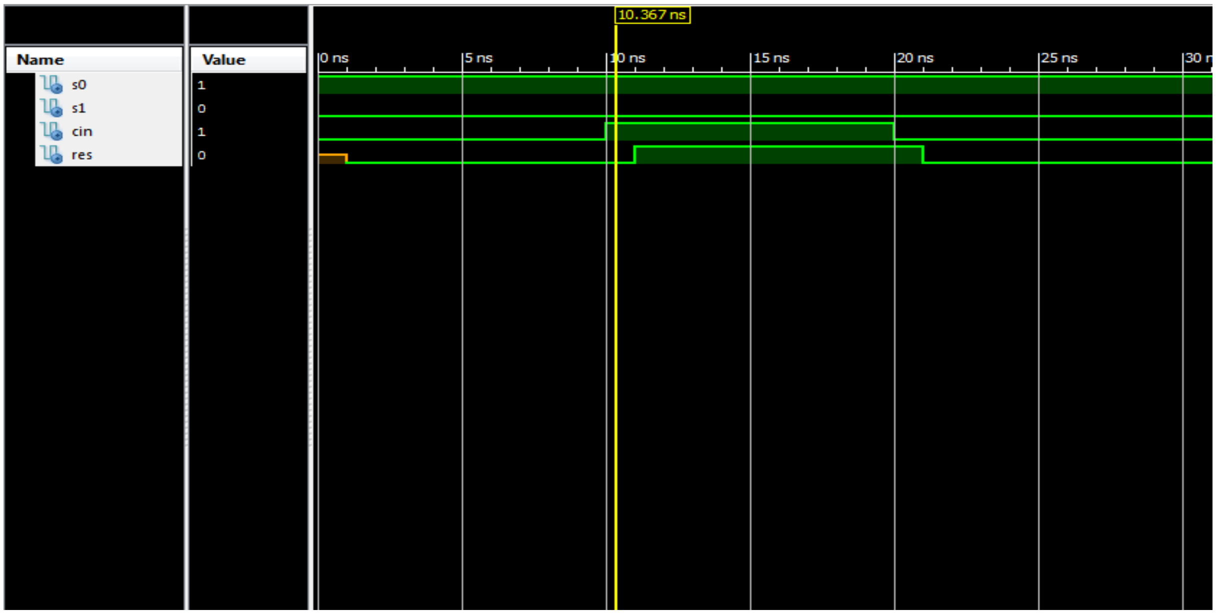
2.8.2 B LOGIC

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

ENTITY test_Blogic IS
END test_Blogic;

ARCHITECTURE behavior OF test_Blogic IS

    COMPONENT B_input_logic
    PORT(
         B : IN std_logic_vector(15 downto 0);
         S : IN std_logic_vector(1 downto 0);
         Y : OUT std_logic_vector(15 downto 0)
        );
    END COMPONENT;


   --Inputs
   signal B : std_logic_vector(15 downto 0) := (others => '0');
   signal S : std_logic_vector(1 downto 0) := (others => '0');

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

BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: B_input_logic PORT MAP (
          B => B,
          S => S,
          Y => Y
        );

   stim_proc: process
   begin
     B <=x"AAAA";
     S <= "00";

      wait for 5ns;
      S <= "00";
      wait for 5ns;
      S <= "01";
      wait for 5ns;
      S <= "10";
      wait for 5ns;
      S <= "11";

      wait;
   end process;

END;
```


0.005 ns
Name
Value
b[15:0]
s[1:0]
y[15:0]
10101010101
00
UUUUUUUUUUU
0 ns
5 ns
10 ns
15 ns
20 ns
25 ns
1010101010101010
00
01
10
11
0000000000000000
10101010101...
0000000000000...
1010101010101010

2.8.3 MUX2 TO 1BIT

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

ENTITY test_mux2_1bit IS
END test_mux2_1bit;

ARCHITECTURE behavior OF test_mux2_1bit IS

    COMPONENT mux2_1bit
    PORT(
         S0 : IN std_logic;
         S1 : IN std_logic;
         Cin : IN std_logic;
         Res : OUT std_logic
        );
    END COMPONENT;

   --Inputs
   signal S0 : std_logic := '0';
   signal S1 : std_logic := '0';
   signal Cin : std_logic := '0';

   --Outputs
   signal Res : std_logic;

BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: mux2_1bit PORT MAP (
          S0 => S0,
          S1 => S1,
          Cin => Cin,
          Res => Res
        );

   stim_proc: process
   begin
       S0 <= '1';
       S1 <= '0';

       wait for 10ns;
       Cin <= '1';

       wait for 10ns;
       Cin <= '0';

      wait;
   end process;

END;
```


10.367 ns
Name
Value
s0
1
s1
0
cin
1
res
0
0 ns
5 ns
10 ns
15 ns
20 ns
25 ns

2.8.4 RIPPLE

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

ENTITY test_ripple IS
END test_ripple;

ARCHITECTURE behavior OF test_ripple IS

    COMPONENT ripple_adder
    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;
         Gout : OUT std_logic_vector(15 downto 0);
         Vout : OUT std_logic
        );
    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 Gout : std_logic_vector(15 downto 0);
   signal Vout : std_logic;

BEGIN

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

   stim_proc: process
   begin

    A <=x"AAAA";
    B <=x"FBAA";
    Cin <= '1';

     wait for 10ns;
     A <=x"FFFF";
    B <=x"0000";
    Cin <= '1';
```

```
      wait;
   end process;

END;
```

2.8.5 FULL ADDER

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

ENTITY test_full_adder IS
END test_full_adder;

ARCHITECTURE behavior OF test_full_adder IS

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

    COMPONENT full_adder
    PORT(
         X : IN std_logic;
         Y : IN std_logic;
         S : OUT std_logic;
         Cin : IN std_logic;
         Cout : OUT std_logic
        );
    END COMPONENT;


   --Inputs
   signal X : std_logic := '0';
   signal Y : 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 (
          X => X,
          Y => Y,
          S => S,
          Cin => Cin,
          Cout => Cout
        );

   stim_proc: process
   begin

      wait for 10ns;
       X <= '0';
       Y <= '0';

      wait for 10ns;
       X <= '0';
       Y <= '1';

      wait for 10ns;
```

```
      X <= '1';
      Y <= '0';

     wait for 10ns;
      X <= '1';
      Y <= '1';

     wait for 10ns;
      Cin <= '1';


     wait;
   end process;

END;
```