۱- **طراحی و شبیه سازی واحد** ALU

ابتدا یک ماژول .F.A تک بیتی را به صورت زیر کد نویسی می کنیم :

```
1
2  library IEEE;
3  use IEEE.STD_LOGIC_1164.ALL;
4  use IEEE.NUMERIC_STD.ALL;
5
6  entity Full_Adder_1bit is
7      Port ( A : in  STD_LOGIC;
8             B : in  STD_LOGIC;
9             Sum : out  STD_LOGIC;
10            Cin : in  STD_LOGIC;
11            Cout : out  STD_LOGIC);
12 end Full_Adder_1bit;
13
14 architecture Behavioral of Full_Adder_1bit is
15
16 begin
17
18    Sum   <= A XOR B XOR Cin ;
19    Cout  <= (A AND B) OR (A AND Cin) OR (Cin AND B);
20
21 end Behavioral;
22
23
```

با استفاده از این تک ماژول .F.A ۸ بیتی را به صورت زیر کد نویسی می کنیم :

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

entity Full_Adder_8bit is
    Port ( A : in  STD_LOGIC_VECTOR (7 downto 0);
           B : in  STD_LOGIC_VECTOR (7 downto 0);
           Sum : out  STD_LOGIC_VECTOR (7 downto 0);
           Cout : out  STD_LOGIC;
           Cin : in  STD_LOGIC);
end Full_Adder_8bit;

architecture Behavioral of Full_Adder_8bit is

   COMPONENT Full_Adder_1bit
   PORT(
      A : IN std_logic;
      B : IN std_logic;
      Cin : IN std_logic;
      Sum : OUT std_logic;
      Cout : OUT std_logic
      );
   END COMPONENT;

   Signal Cary : std_logic_vector (6 downto 0) ;

begin
```

```vhdl
begin

   Inst_Full_Adder_1bit_0: Full_Adder_1bit PORT MAP(
      A =>A(0) ,
      B =>B(0) ,
      Sum =>Sum(0) ,
      Cin =>Cin ,
      Cout =>cary(0)
   );
      Inst_Full_Adder_1bit_1: Full_Adder_1bit PORT MAP(
      A =>A(1) ,
      B =>B(1) ,
      Sum =>Sum(1) ,
      Cin =>cary(0) ,
      Cout =>cary(1)
   );
      Inst_Full_Adder_1bit_2: Full_Adder_1bit PORT MAP(
      A =>A(2) ,
      B =>B(2) ,
      Sum =>Sum(2) ,
      Cin =>cary(1) ,
      Cout =>cary(2)
   );
      Inst_Full_Adder_1bit_3: Full_Adder_1bit PORT MAP(
      A =>A(3) ,
      B =>B(3) ,
      Sum =>Sum(3) ,
      Cin =>Cary(2) ,
      Cout =>cary(3)
   );
```

```
57        Inst_Full_Adder_1bit_4: Full_Adder_1bit PORT MAP(
58        A =>A(4) ,
59        B =>B(4) ,
60        Sum =>Sum(4) ,
61        Cin =>cary(3) ,
62        Cout =>cary(4)
63     );
64        Inst_Full_Adder_1bit_5: Full_Adder_1bit PORT MAP(
65        A =>A(5) ,
66        B =>B(5) ,
67        Sum =>Sum(5) ,
68        Cin =>cary(4) ,
69        Cout =>cary(5)
70     );
71        Inst_Full_Adder_1bit_6: Full_Adder_1bit PORT MAP(
72        A =>A(6) ,
73        B =>B(6) ,
74        Sum =>Sum(6) ,
75        Cin =>cary(5) ,
76        Cout =>cary(6)
77     );
78        Inst_Full_Adder_1bit_7: Full_Adder_1bit PORT MAP(
79        A =>A(7) ,
80        B =>B(7) ,
81        Sum =>Sum(7) ,
82        Cin =>cary(6) ,
83        Cout =>Cout
84     );
85
86
87  end Behavioral;
```

ساب ماژول شیفت به راست :

```
1
2  library IEEE;
3  use IEEE.STD_LOGIC_1164.ALL;
4  use IEEE.NUMERIC_STD.ALL;
5
6  entity Shift_R_8bit is
7      Port ( A : in  STD_LOGIC_VECTOR (7 downto 0);
8             ShiftA : out  STD_LOGIC_VECTOR (7 downto 0) ;
9             Cout : out STD_LOGIC
10       );
11 end Shift_R_8bit;
12
13 architecture Behavioral of Shift_R_8bit is
14
15 begin
16    ShiftA(7)<='0' ;
17    ShiftA(6)<=A(7) ;
18    ShiftA(5)<=A(6) ;
19    ShiftA(4)<=A(5) ;
20    ShiftA(3)<=A(4) ;
21    ShiftA(2)<=A(3) ;
22    ShiftA(1)<=A(2) ;
23    ShiftA(0)<=A(1) ;
24    Cout<=A(0) ;
25
26 end Behavioral;
27
28
```

ساب ماژول شیفت به چپ :

```vhdl

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity Shift_L_8bit is
    Port ( A : in  STD_LOGIC_VECTOR (7 downto 0);
           ShiftA : out  STD_LOGIC_VECTOR (7 downto 0);
           Cout : out  STD_LOGIC);
end Shift_L_8bit;

architecture Behavioral of Shift_L_8bit is

begin

   ShiftA(0)<='0' ;
   ShiftA(1)<=A(0) ;
   ShiftA(2)<=A(1) ;
   ShiftA(3)<=A(2) ;
   ShiftA(4)<=A(3) ;
   ShiftA(5)<=A(4) ;
   ShiftA(6)<=A(5) ;
   ShiftA(7)<=A(6) ;
   Cout<=A(7) ;

end Behavioral;


```

در اینجا ما بیتی که به بیرون از ۸ بیت انتقال می یابد را در بیت کری قرار دادیم.

با اضافه کردن ساب ماژول های زیر واحد ALU را تکمیل می کنیم :

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

entity ALU is
    Port ( DR1 : in  STD_LOGIC_VECTOR (7 downto 0);
           DR2 : in  STD_LOGIC_VECTOR (7 downto 0);
           OPCODE : in  STD_LOGIC_VECTOR (3 downto 0);
           Cin : in  STD_LOGIC;
           AC : out  STD_LOGIC_VECTOR (7 downto 0);
           Cout : out  STD_LOGIC);
end ALU;

architecture Behavioral of ALU is

   COMPONENT Full_Adder_8bit
   PORT(
      A : IN std_logic_vector(7 downto 0);
      B : IN std_logic_vector(7 downto 0);
      Cin : IN std_logic;
      Sum : OUT std_logic_vector(7 downto 0);
      Cout : OUT std_logic
      );
   END COMPONENT;

   COMPONENT Shift_R_8bit
   PORT(
      A : IN std_logic_vector(7 downto 0);
      ShiftA : OUT std_logic_vector(7 downto 0);
      Cout : OUT std_logic
```

```
      );
   END COMPONENT;

   COMPONENT Shift_L_8bit
   PORT(
      A : IN std_logic_vector(7 downto 0);
      ShiftA : OUT std_logic_vector(7 downto 0);
      Cout : OUT std_logic
      );
   END COMPONENT;

   Signal ADD1 : std_logic_vector (7 downto 0 ) ;
   Signal temp : std_logic_vector (7 downto 0 ) ;
   Signal Cout1 : std_logic ;
   Signal ShiftR : std_logic_vector (7 downto 0 ) ;
   Signal ShiftL : std_logic_vector (7 downto 0 ) ;
   Signal CoutR : std_logic ;
   Signal CoutL : std_logic ;
   Signal cin1 : std_logic ;


begin

AC  <=ADD1 When OPCODE="0000" OR OPCODE="0001" OR OPCODE="0010" OR (OPCODE="0011
      DR1 AND DR2 When OPCODE="0100" ELSE
      DR1 OR DR2 When OPCODE="0101" ELSE
      DR1 XOR DR2 When OPCODE="0110" ELSE
      NOT DR1 When OPCODE="0111" ELSE
      ShiftR When (OPCODE and "1100")="1000" ELSE
      ShiftL When (OPCODE and "1100")="1100" ELSE
      "ZZZZZZZZ" ;

Cout  <=Cout1 When OPCODE="0000" OR OPCODE="0001" OR OPCODE="0010" OR OPCODE="00
      CoutR When (OPCODE and "1100")="1000"  ELSE
      CoutL When (OPCODE and "1100")="1100"  ELSE
      'Z' ;
Cin1 <= Cin when OPCODE="0000" OR OPCODE="0001" OR OPCODE="0010" else
      '0' ;
temp <="00000000" when OPCODE="0000" else
      DR2 when OPCODE="0001" else
      NOT DR2 when OPCODE="0010" else
      "11111111" when OPCODE="0011"else
      "00000000" ;


      Inst_Full_Adder_8bit_1: Full_Adder_8bit PORT MAP(
      A =>DR1 ,
      B =>temp ,
      Sum =>ADD1 ,
      Cout =>Cout1 ,
      Cin =>Cin1
   );
      Inst_Shift_R_8bit: Shift_R_8bit PORT MAP(
      A =>DR1 ,
      ShiftA =>ShiftR ,
      Cout =>CoutR
   );
      Inst_Shift_L_8bit: Shift_L_8bit PORT MAP(
      A =>DR1 ,
      ShiftA =>ShiftL ,
      Cout =>CoutL
   );
```

```
94
95
96
97
98  end Behavioral;
99
100
```

توضیحات :

برای اجرای اعمال منطقی از عملگر های آن مانند گزارش قبل استفاده می کنیم . برای اعمال جمع و تفریق از واحد F.A.8bits استفاده می کنیم بدین صورت که مقدار ورودی اول DR1 می باشد و مقدار ورودی دوم را با متغییر temp با توجه به opcode انتخاب میکنیم . همچنین مقدار cin را در یکی از حالت ها باید برابر با صفر باشد را جدا کرده ایم. اعمال منطقی شیفت هم تنها به تاپ ماژول اضافه شده اند و با توجه به opcode اعمال می شوند.

شبیه سازی :

کد زیر را برای شبیه سازی می نویسیم :

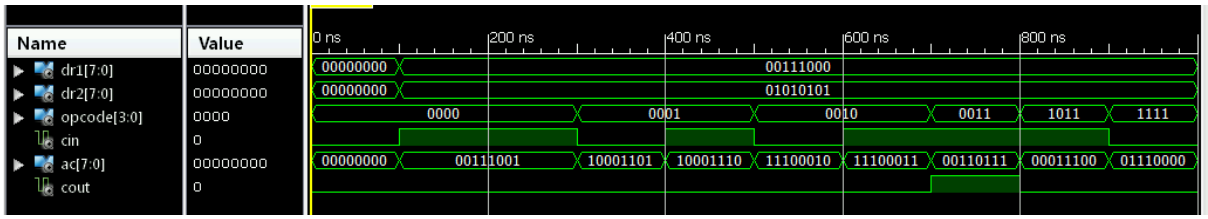
```
1   LIBRARY ieee;
2   USE ieee.std_logic_1164.ALL;
3   USE ieee.numeric_std.ALL;
4   use IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6   ENTITY ALU_test IS
7   END ALU_test;
8
9   ARCHITECTURE behavior OF ALU_test IS
10
11      -- Component Declaration for the Unit Under Test (UUT)
12
13      COMPONENT ALU
14      PORT(
15           DR1 : IN  std_logic_vector(7 downto 0);
16           DR2 : IN  std_logic_vector(7 downto 0);
17           OPCODE : IN  std_logic_vector(3 downto 0);
18           Cin : IN  std_logic;
19           AC : OUT  std_logic_vector(7 downto 0);
20           Cout : OUT  std_logic
21          );
22      END COMPONENT;
23
24
25     --Inputs
26     signal DR1 : std_logic_vector(7 downto 0) := (others => '0');
27     signal DR2 : std_logic_vector(7 downto 0) := (others => '0');
28     signal OPCODE : std_logic_vector(3 downto 0) := (others => '0');
29     signal Cin : std_logic := '0';
30
31     --Outputs
32     signal AC : std_logic_vector(7 downto 0);
```

```
33    signal Cout : std_logic;
34    -- No clocks detected in port list. Replace <clock> below with
35    -- appropriate port name
36
37
38  BEGIN
39
40     -- Instantiate the Unit Under Test (UUT)
41     uut: ALU PORT MAP (
42            DR1 => DR1,
43            DR2 => DR2,
44            OPCODE => OPCODE,
45            Cin => Cin,
46            AC => AC,
47            Cout => Cout
48          );
49
50
51     -- Stimulus process
52     stim_proc: process
53     begin
54        -- hold reset state for 100 ns.
55        wait for 100 ns;
56
57        -- insert stimulus here
58
59        OPCODE<="0000" ;
60        Cin<='1' ;
61        DR1<="00111000" ;
62        DR2<="01010101" ;
63        wait for 100 ns;
64
```

```
65        OPCODE<="0000" ;
66        Cin<='1' ;
67        DR1<="00111000" ;
68        DR2<="01010101" ;
69        wait for 100 ns;
70
71        OPCODE<="0001" ;
72        Cin<='0' ;
73        DR1<="00111000" ;
74        DR2<="01010101" ;
75        wait for 100 ns;
76
77        OPCODE<="0001" ;
78        Cin<='1' ;
79        DR1<="00111000" ;
80        DR2<="01010101" ;
81        wait for 100 ns;
82
83        OPCODE<="0010" ;
84        Cin<='0' ;
85        DR1<="00111000" ;
86        DR2<="01010101" ;
87        wait for 100 ns;
88
89        OPCODE<="0010" ;
90        Cin<='1' ;
91        DR1<="00111000" ;
92        DR2<="01010101" ;
93        wait for 100 ns;
94
95        OPCODE<="0011" ;
96        Cin<='1' ;
```

```
97        DR1<="00111000" ;
98        DR2<="01010101" ;
99        wait for 100 ns;
100
101       OPCODE<="1011" ;
102       Cin<='1' ;
103       DR1<="00111000" ;
104       DR2<="01010101" ;
105       wait for 100 ns;
106
107       OPCODE<="1111" ;
108       Cin<='0' ;
109       DR1<="00111000" ;
110       DR2<="01010101" ;
111       wait for 100 ns;
112
113       wait;
114    end process;
115
116 END;
117
```

خروجی مشاهده شده :



همانطور که مشاهده می کنیم (بررسی هر مورد جداگانه صورت گرفته است) کد به درستی عمل می کند.

۲- طراحی bus

در اینجا ما باس را به صورت هاروارد طراحی می کنیم . در اینجا ورودی سیگنال های کنترلی برای data با selectbusdata مشخص شده است و هر کدام از واحد ها را مقادیر داده آن ها را بر روی باس دیتا قرار می گیرد.

برای بخش ادرس هم که مختص رم ها می باشد نیز چون رم داده ادرس را می خواند و CPU ادرس را تولید می کند باس را به صورت خروجی در نظر گرفته ایم که به واحد های ادرس رم ها وارد می شود .

(البته این پیاده سازی خیلی کامل نشده است و نیاز به داده های بیشتر برای طراحی می باشد . مثلا ما در اینجا خروجی واحد ها را به صورت پورت تعریف کرده ایم در صورت که هر واحد یک بافر ورودی و یک بافر خروجی دارد و به صورت خودکار هنگام خواندن از بافر خروجی و هنگام نوشتن در بافر ورودی می نویسیم (در معماری های جدید))

کد نوشته شده :

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

entity BUS_8bit_Harvard is
    Port (   SelectBusData : in  STD_LOGIC_VECTOR (4 downto 0);
             SelectBusAddress : in  STD_LOGIC_VECTOR (3 downto 0);
             BusData : buffer STD_LOGIC_VECTOR (7 downto 0);
             BusADDRESS : in STD_LOGIC_VECTOR (7 downto 0);
             ALU_PORT : in STD_LOGIC_VECTOR (7 downto 0) ;
             AC_PORT  : in  STD_LOGIC_VECTOR (7 downto 0) ;
             DR_PORT  : in  STD_LOGIC_VECTOR (7 downto 0) ;
             AR_PORT  : in  STD_LOGIC_VECTOR (7 downto 0) ;
             PR_PORT  : in  STD_LOGIC_VECTOR (7 downto 0) ;
             IR_PORT  : in  STD_LOGIC_VECTOR (7 downto 0) ;
             RAM_00_PORT : in  STD_LOGIC_VECTOR (7 downto 0) ;
             RAM_01_PORT : in  STD_LOGIC_VECTOR (7 downto 0) ;
             RAM_02_PORT : in  STD_LOGIC_VECTOR (7 downto 0) ;
             RAM_03_PORT : in  STD_LOGIC_VECTOR (7 downto 0) ;
             RAM_04_PORT : in  STD_LOGIC_VECTOR (7 downto 0) ;
             RAM_05_PORT : in  STD_LOGIC_VECTOR (7 downto 0) ;
             RAM_06_PORT : in  STD_LOGIC_VECTOR (7 downto 0) ;
             RAM_07_PORT : in  STD_LOGIC_VECTOR (7 downto 0) ;
             ROM_00_PORT : in  STD_LOGIC_VECTOR (7 downto 0) ;
             ROM_01_PORT : in  STD_LOGIC_VECTOR (7 downto 0) ;
             ROM_02_PORT : in  STD_LOGIC_VECTOR (7 downto 0) ;
             ROM_03_PORT : in  STD_LOGIC_VECTOR (7 downto 0) ;
             RAM_00_PORT_Address : buffer STD_LOGIC_VECTOR (7 downto 0) ;
             RAM_01_PORT_Address : buffer STD_LOGIC_VECTOR (7 downto 0) ;
             RAM_02_PORT_Address : buffer STD_LOGIC_VECTOR (7 downto 0) ;
             RAM_03_PORT_Address : buffer STD_LOGIC_VECTOR (7 downto 0) ;
             RAM_04_PORT_Address : buffer STD_LOGIC_VECTOR (7 downto 0) ;
             RAM_05_PORT_Address : buffer STD_LOGIC_VECTOR (7 downto 0) ;
             RAM_06_PORT_Address : buffer STD_LOGIC_VECTOR (7 downto 0) ;
             RAM_07_PORT_Address : buffer STD_LOGIC_VECTOR (7 downto 0) ;
             ROM_00_PORT_Address : buffer STD_LOGIC_VECTOR (7 downto 0) ;
             ROM_01_PORT_Address : buffer STD_LOGIC_VECTOR (7 downto 0) ;
             ROM_02_PORT_Address : buffer STD_LOGIC_VECTOR (7 downto 0) ;
             ROM_03_PORT_Address : buffer STD_LOGIC_VECTOR (7 downto 0)
           );
end BUS_8bit_Harvard;

architecture Behavioral of BUS_8bit_Harvard is


begin

   BusData <=RAM_00_PORT When SelectBusData = "00000" ELse
             RAM_01_PORT When SelectBusData = "00001" ELse
             RAM_02_PORT When SelectBusData = "00010" ELse
             RAM_03_PORT When SelectBusData = "00011" ELse
             RAM_04_PORT When SelectBusData = "00100" ELse
             RAM_05_PORT When SelectBusData = "00101" ELse
             RAM_06_PORT When SelectBusData = "00110" ELse
             RAM_07_PORT When SelectBusData = "00111" ELse
             ROM_00_PORT When SelectBusData = "01000" ELse
             ROM_01_PORT When SelectBusData = "01001" ELse
             ROM_02_PORT When SelectBusData = "01010" ELse
             ROM_03_PORT When SelectBusData = "01011" ELse
             ALU_PORT When SelectBusData = "01100" ELse
             AC_PORT When SelectBusData = "01101" ELse
             DR_PORT When SelectBusData = "01110" ELse
             AR_PORT When SelectBusData = "01111" ELse
             PR_PORT When SelectBusData = "10000" ELse
```

```
            IR_PORT When SelectBusData = "10001" ELse
            "ZZZZZZZZ" ;

        RAM_00_PORT_Address<=BusAddress When SelectBusAddress = "0000" ;
        RAM_01_PORT_Address<=BusAddress When SelectBusAddress = "0001" ;
        RAM_02_PORT_Address<=BusAddress When SelectBusAddress = "0010" ;
        RAM_03_PORT_Address<=BusAddress When SelectBusAddress = "0011" ;
        RAM_04_PORT_Address<=BusAddress When SelectBusAddress = "0100" ;
        RAM_05_PORT_Address<=BusAddress When SelectBusAddress = "0101" ;
        RAM_06_PORT_Address<=BusAddress When SelectBusAddress = "0110" ;
        RAM_07_PORT_Address<=BusAddress When SelectBusAddress = "0111" ;
        ROM_00_PORT_Address<=BusAddress When SelectBusAddress = "1000" ;
        ROM_01_PORT_Address<=BusAddress When SelectBusAddress = "1001" ;
        ROM_02_PORT_Address<=BusAddress When SelectBusAddress = "1010" ;
        ROM_03_PORT_Address<=BusAddress When SelectBusAddress = "1011" ;


end Behavioral;

```

شبیه سازی :

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

ENTITY test IS
END test;

ARCHITECTURE behavior OF test IS

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

    COMPONENT BUS_8bit_Harvard
    PORT(
         SelectBusData : IN  std_logic_vector(4 downto 0);
         SelectBusAddress : IN  std_logic_vector(3 downto 0);
         BusData : Buffer  std_logic_vector(7 downto 0);
         BusADDRESS : IN  std_logic_vector(7 downto 0);
         ALU_PORT : IN  std_logic_vector(7 downto 0);
         AC_PORT : IN  std_logic_vector(7 downto 0);
         DR_PORT : IN  std_logic_vector(7 downto 0);
         AR_PORT : IN  std_logic_vector(7 downto 0);
         PR_PORT : IN  std_logic_vector(7 downto 0);
         IR_PORT : IN  std_logic_vector(7 downto 0);
         RAM_00_PORT : IN  std_logic_vector(7 downto 0);
         RAM_01_PORT : IN  std_logic_vector(7 downto 0);
         RAM_02_PORT : IN  std_logic_vector(7 downto 0);
         RAM_03_PORT : IN  std_logic_vector(7 downto 0);
         RAM_04_PORT : IN  std_logic_vector(7 downto 0);
         RAM_05_PORT : IN  std_logic_vector(7 downto 0);
         RAM_06_PORT : IN  std_logic_vector(7 downto 0);
         RAM_07_PORT : IN  std_logic_vector(7 downto 0);
```

```
33          ROM_00_PORT : IN  std_logic_vector(7 downto 0);
34          ROM_01_PORT : IN  std_logic_vector(7 downto 0);
35          ROM_02_PORT : IN  std_logic_vector(7 downto 0);
36          ROM_03_PORT : IN  std_logic_vector(7 downto 0);
37          RAM_00_PORT_Address : Buffer  std_logic_vector(7 downto 0);
38          RAM_01_PORT_Address : Buffer  std_logic_vector(7 downto 0);
39          RAM_02_PORT_Address : Buffer  std_logic_vector(7 downto 0);
40          RAM_03_PORT_Address : Buffer  std_logic_vector(7 downto 0);
41          RAM_04_PORT_Address : Buffer  std_logic_vector(7 downto 0);
42          RAM_05_PORT_Address : Buffer  std_logic_vector(7 downto 0);
43          RAM_06_PORT_Address : Buffer  std_logic_vector(7 downto 0);
44          RAM_07_PORT_Address : Buffer  std_logic_vector(7 downto 0);
45          ROM_00_PORT_Address : Buffer  std_logic_vector(7 downto 0);
46          ROM_01_PORT_Address : Buffer  std_logic_vector(7 downto 0);
47          ROM_02_PORT_Address : Buffer  std_logic_vector(7 downto 0);
48          ROM_03_PORT_Address : Buffer  std_logic_vector(7 downto 0)
49        );
50    END COMPONENT;
51
52
53    --Inputs
54    signal SelectBusData : std_logic_vector(4 downto 0) := (others => 'Z');
55    signal SelectBusAddress : std_logic_vector(3 downto 0) := (others => 'Z');
56    signal BusADDRESS : std_logic_vector(7 downto 0) := (others => 'Z');
57    signal ALU_PORT : std_logic_vector(7 downto 0) := (others => 'Z');
58    signal AC_PORT : std_logic_vector(7 downto 0) := (others => 'Z');
59    signal DR_PORT : std_logic_vector(7 downto 0) := (others => 'Z');
60    signal AR_PORT : std_logic_vector(7 downto 0) := (others => 'Z');
61    signal PR_PORT : std_logic_vector(7 downto 0) := (others => 'Z');
62    signal IR_PORT : std_logic_vector(7 downto 0) := (others => 'Z');
63    signal RAM_00_PORT : std_logic_vector(7 downto 0) := (others => 'Z');
64    signal RAM_01_PORT : std_logic_vector(7 downto 0) := (others => 'Z');
```

```
65    signal RAM_02_PORT : std_logic_vector(7 downto 0) := (others => 'Z');
66    signal RAM_03_PORT : std_logic_vector(7 downto 0) := (others => 'Z');
67    signal RAM_04_PORT : std_logic_vector(7 downto 0) := (others => 'Z');
68    signal RAM_05_PORT : std_logic_vector(7 downto 0) := (others => 'Z');
69    signal RAM_06_PORT : std_logic_vector(7 downto 0) := (others => 'Z');
70    signal RAM_07_PORT : std_logic_vector(7 downto 0) := (others => 'Z');
71    signal ROM_00_PORT : std_logic_vector(7 downto 0) := (others => 'Z');
72    signal ROM_01_PORT : std_logic_vector(7 downto 0) := (others => 'Z');
73    signal ROM_02_PORT : std_logic_vector(7 downto 0) := (others => 'Z');
74    signal ROM_03_PORT : std_logic_vector(7 downto 0) := (others => 'Z');
75
76    --Outputs
77    signal BusData : std_logic_vector(7 downto 0);
78    signal RAM_00_PORT_Address : std_logic_vector(7 downto 0);
79    signal RAM_01_PORT_Address : std_logic_vector(7 downto 0);
80    signal RAM_02_PORT_Address : std_logic_vector(7 downto 0);
81    signal RAM_03_PORT_Address : std_logic_vector(7 downto 0);
82    signal RAM_04_PORT_Address : std_logic_vector(7 downto 0);
83    signal RAM_05_PORT_Address : std_logic_vector(7 downto 0);
84    signal RAM_06_PORT_Address : std_logic_vector(7 downto 0);
85    signal RAM_07_PORT_Address : std_logic_vector(7 downto 0);
86    signal ROM_00_PORT_Address : std_logic_vector(7 downto 0);
87    signal ROM_01_PORT_Address : std_logic_vector(7 downto 0);
88    signal ROM_02_PORT_Address : std_logic_vector(7 downto 0);
89    signal ROM_03_PORT_Address : std_logic_vector(7 downto 0);
90    -- No clocks detected in port list. Replace <clock> below with
91    -- appropriate port name
92
93  BEGIN
94
95    -- Instantiate the Unit Under Test (UUT)
96    uut: BUS_8bit_Harvard PORT MAP (
```

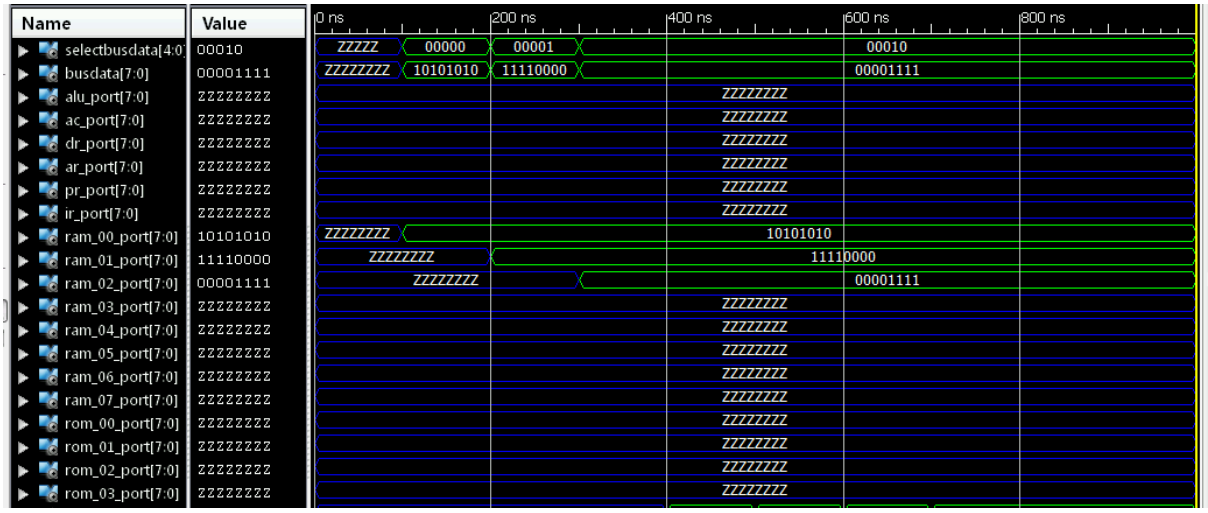
```
 97            SelectBusData => SelectBusData,
 98            SelectBusAddress => SelectBusAddress,
 99            BusData => BusData,
100            BusADDRESS => BusADDRESS,
101            ALU_PORT => ALU_PORT,
102            AC_PORT => AC_PORT,
103            DR_PORT => DR_PORT,
104            AR_PORT => AR_PORT,
105            PR_PORT => PR_PORT,
106            IR_PORT => IR_PORT,
107            RAM_00_PORT => RAM_00_PORT,
108            RAM_01_PORT => RAM_01_PORT,
109            RAM_02_PORT => RAM_02_PORT,
110            RAM_03_PORT => RAM_03_PORT,
111            RAM_04_PORT => RAM_04_PORT,
112            RAM_05_PORT => RAM_05_PORT,
113            RAM_06_PORT => RAM_06_PORT,
114            RAM_07_PORT => RAM_07_PORT,
115            ROM_00_PORT => ROM_00_PORT,
116            ROM_01_PORT => ROM_01_PORT,
117            ROM_02_PORT => ROM_02_PORT,
118            ROM_03_PORT => ROM_03_PORT,
119            RAM_00_PORT_Address => RAM_00_PORT_Address,
120            RAM_01_PORT_Address => RAM_01_PORT_Address,
121            RAM_02_PORT_Address => RAM_02_PORT_Address,
122            RAM_03_PORT_Address => RAM_03_PORT_Address,
123            RAM_04_PORT_Address => RAM_04_PORT_Address,
124            RAM_05_PORT_Address => RAM_05_PORT_Address,
125            RAM_06_PORT_Address => RAM_06_PORT_Address,
126            RAM_07_PORT_Address => RAM_07_PORT_Address,
127            ROM_00_PORT_Address => ROM_00_PORT_Address,
128            ROM_01_PORT_Address => ROM_01_PORT_Address,
```

```
129           ROM_02_PORT_Address => ROM_02_PORT_Address,
130           ROM_03_PORT_Address => ROM_03_PORT_Address
131         );
132
133
134    -- Stimulus process
135    stim_proc: process
136    begin
137       -- hold reset state for 100 ns.
138       wait for 100 ns;
139
140       SelectBusData<="00000" ;
141       RAM_00_PORT<="10101010" ;
142       wait for 100 ns;
143       SelectBusData<="00001" ;
144       RAM_01_PORT<="11110000" ;
145       wait for 100 ns;
146       SelectBusData<="00010" ;
147       RAM_02_PORT<="00001111" ;
148       wait for 100 ns;
149       SelectBusAddress<="0010" ;
150       BusAddress<="00001111" ;
151       wait for 100 ns;
152       SelectBusAddress<="0110" ;
153       BusAddress<="00111111" ;
154       wait for 100 ns;
155       SelectBusAddress<="0111" ;
156       BusAddress<="00100011" ;
157       wait for 100 ns;
158       SelectBusAddress<="1000" ;
159       BusAddress<="00111001" ;
160       wait for 100 ns;
```

```
161
162
163
164       -- insert stimulus here
165
166       wait;
167    end process;
168
169 END;
170
```

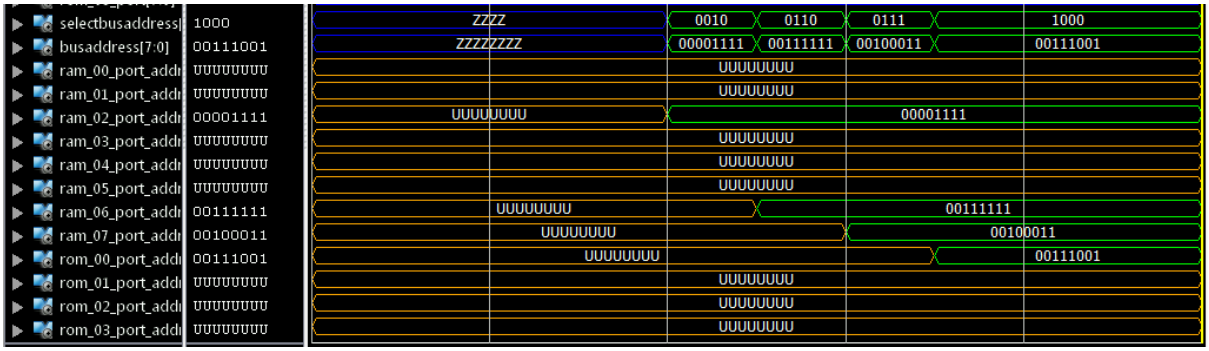
خروجی مشاهده شده :

برای باس داده :



در اینجا مشاهده می کنیم با تغییر select مقادیر موجود در پورت ها بر روی باس قرار می گیرد.

برای باس آدرس :



در اینجا مشاهده می کنیم با تغییر select مقادیر موجود در باس بر روی پورت ادرس رم ها قرار می گیرد.