"به نام یزدان پاک"

گزارش کار آزمایش دوم

اعضای گروه:

کیانا آقا کثیری 9831006

محمد چوپان 9831125

سارا تاجرنیا 9831016

نویسنده گزارش : کیانا آقا کثیری

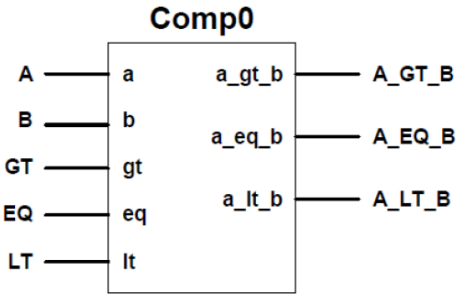
تاریخ آزمایش : 99/12/20

تاریخ تحویل گزارش :99/12/26

شرح آزمایش:

شرح آزمایش:

در این آزمایش، یک مالتی‌پلکسر، کدکننده، کدگشا و یک مقایسه‌گر ۴ بیتی براساس ماژول‌های تک‌بیتی آن‌ها طراحی و شبیه‌سازی گردد. طراحی انجام شده باید توسط زبان توصیف سخت‌افزار VHDL و در **سطح تجرید گیت** انجام شود. تهیه TestBench برای این جمع‌کننده نیز جزء الزام‌های آزمایش است. مدار تک بیتی مورد استفاده برای مقایسه‌گر مطابق با ساختار شکل ۱ پیاده‌سازی شود.



شکل ۱ ساختار ورودی/خروجی‌های مقایسه‌گر تک‌بیتی

شکل 1: شرح آزمایش

صورت سوال آزمایش ۲ قسمت مالتی پلکسر

یک مالتی پلکسر 4×1 (ورودی و خروجی تک بیتی) را یک بار با ساختار ارجاع شرطی و یکبار با ساختار ارجاع انتخابی به طور جداگانه پیاده‌سازی کنید.

سپس به وسیله یکی از ساختارهای پیاده‌سازی شده قسمت قبل یک مالتی پلکسر 16×1 (ورودی و خروجی تک بیتی) پیاده سازی کنید.

ساختار قسمت دو سوال همرا با اتصالات بین مالتی‌پلکسرها را رسم کنید(در سطح ماژول).

شکل 2: صورت سوال تغییر یافته آزمایش

خروجی مورد انتظار آزمایش:

خروجی‌های مورد انتظار آزمایش:

هر یک از موارد زیر باید تحویل داده شود:

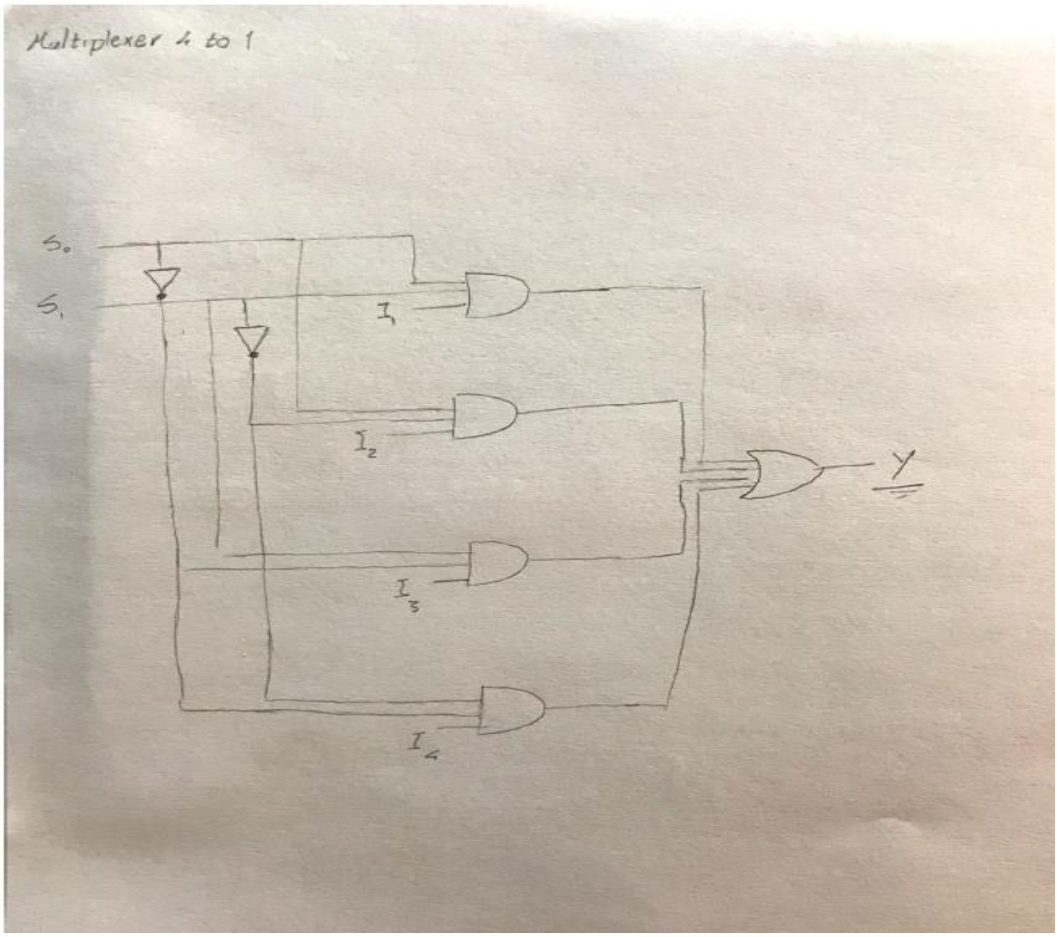
- طراحی شماتیک طرح (محتوای توصیف) به صورت فیزیکی (بر روی کاغذ)
- بررسی درستی سیگنال‌های خروجی مدارهای مورد نظر با توجه به ورودی‌ها در شبیه‌سازی

شکل 3: خروجی مورد انتظار آزمایش

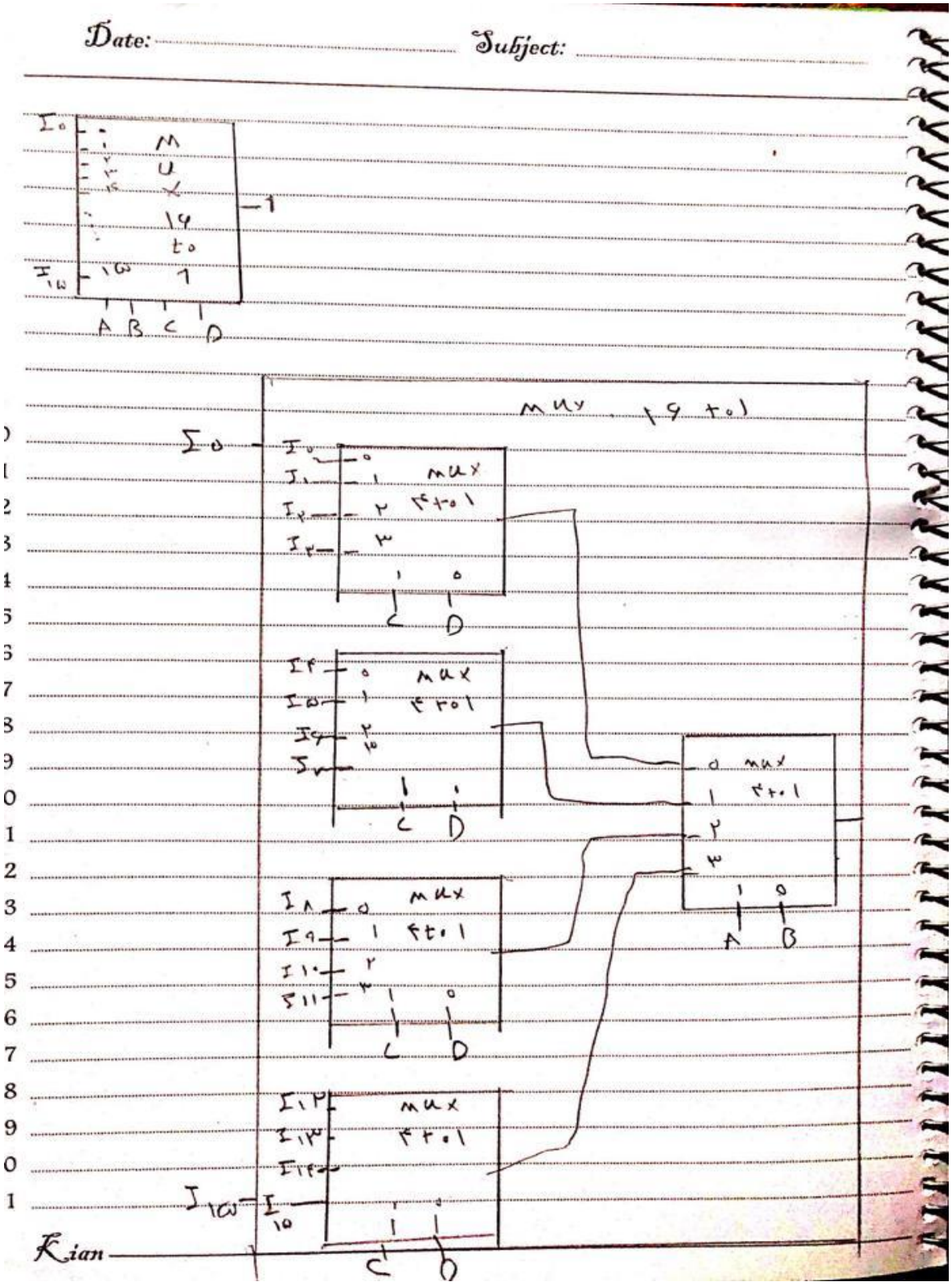
توضیح آزمایش :

مالتی پلکسر (MUX):

با استفاده از شکل 4 که در پیش گزارش آمده یک مالتی پلکسر 4 به 1 را با استفاده از شرط های - When else (ارجاع شرطی) و select (ارجاع انتخابی) طراحی کرده .و کد آن را با استفاده از زبان VHDL در ISE پیاده سازی میکنیم. در قسمت دوم سوال یک مالتی پلکسر 16 به یک را با استفاده از طراحی انجام شده در شکل 5 انجام می دهیم. برای پیاده سازی این ماژول در زبان VHDL ابتدا Component مالتی پلکسر 4 به 1 را تعریف میکنیم. سپس 5 نمونه از آن را ساخته و سپس ورودی و خروجی های ان را PORT MAP کرده و بهم وصل میکنیم.در نهایت برای تست کردن ماژول های خود Test Bench برای هر دو ماژول نوشته ورودی و خروجی های مرتبط را بهم وصل میکنیم و مقادیر دل خواهی را جهت تست به آن می دهیم.



شکل 4 :طراحی MUX در سطح گیت



شکل 5 : طراحی MUX 16 به 1 با استفاده از MUX 4 به 1

Behavior of 4 to 1 MUX:

(Conditional with-else)

```
entity MUX4X1_con is
    Port ( input  : in  STD_LOGIC_VECTOR (3 downto 0);
           select1 : in  STD_LOGIC_VECTOR (1 downto 0);
           output : out  STD_LOGIC);
end MUX4X1_con;

architecture Behavioral of MUX4X1_con is

begin
 output<=input(0) when select1="00" else
         input(1) when select1="01" else
         input(2) when select1="10" else
         input(3);


end Behavioral;
```

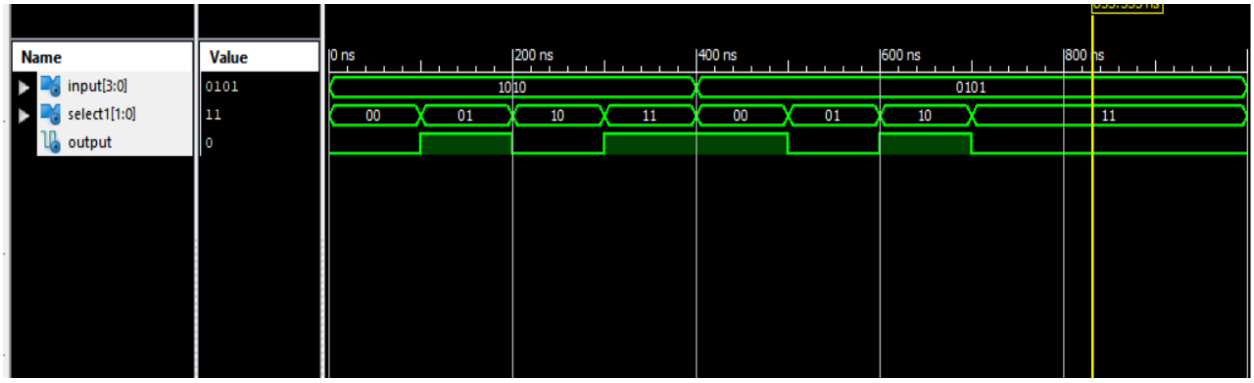
شکل 6 : توصیف مالتی پلکسر 4 به 1 با ساختار شرطی

Test Bench :

```
ENTITY MUX4X1_con_tb IS
END MUX4X1_con_tb;
ARCHITECTURE behavior OF MUX4X1_con_tb IS
    COMPONENT MUX4X1_con
    PORT(
         input : IN  std_logic_vector(3 downto 0);
         select1 : IN  std_logic_vector(1 downto 0);
         output : OUT  std_logic
        );
    END COMPONENT;
   signal input : std_logic_vector(3 downto 0) := (others => '0');
   signal select1 : std_logic_vector(1 downto 0) := (others => '0');

   --Outputs
   signal output : std_logic;

BEGIN
   uut: MUX4X1_con PORT MAP (
          input => input,
          select1 => select1,
          output => output
        );
   input<="1010" ,"0101" after 400 ns;
   select1<="00","01" after 100 ns,"10" after 200 ns,"11" after 300 ns,"00" after 400 ns,"01" after 500 ns
   ,"10" after 600 ns ,"11" after 700 ns;
END;
```

شکل 7: تست بنچ مالتی پلکسر 4 به 1 با ساختار شرطی

Result of Simulation in isim :



شکل 7-1 : نتایج شبیه سازی

Behavior of 4 to 1 MUX:

(with select)

```
entity MUX4x1 is
    Port ( input : in  STD_LOGIC_VECTOR (3 DOWNTO 0);
           select1 : in  STD_LOGIC_VECTOR (1 DOWNTO 0);
           output : out  STD_LOGIC);
end MUX4x1;

architecture Behavioral of MUX4x1 is

begin
 with select1 select
  output<=input(0) when "00",
          input(1) when "01",
          input(2) when "10",
          input(3) when others;


end Behavioral;
```

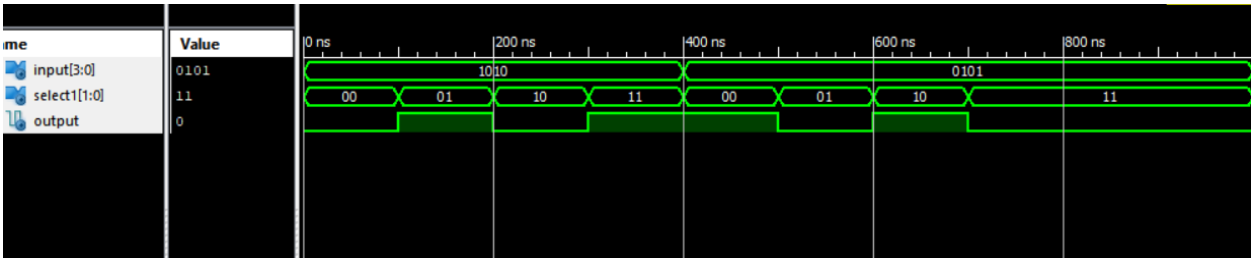
شکل 8 : توصیف مالتی پلکسر با استفاده از ساختار انتخابی

Test Bench:

```
--USE ieee.numeric_std.ALL;
ENTITY MUX4x1_tb IS
END MUX4x1_tb;
ARCHITECTURE behavior OF MUX4x1_tb IS
  -- Component Declaration for the Unit Under Test (UU
    COMPONENT MUX4x1
    PORT(
         input : IN  std_logic_vector(3 downto 0);
         select1 : IN  std_logic_vector(1 downto 0);
         output : OUT  std_logic
        );
    END COMPONENT;
   --Inputs
   signal input : std_logic_vector(3 downto 0) := (others => '0');
   signal select1 : std_logic_vector(1 downto 0) := (others => '0');

   --Outputs
   signal output : std_logic;
BEGIN
   -- Instantiate the Unit Under Test (UUT)
   uut: MUX4x1 PORT MAP (
          input => input,
          select1 => select1,
          output => output
        );
   input<="1010" ,"0101" after 400 ns;
   select1<="00","01" after 100 ns,"10" after 200 ns,"11" after 300 ns,"00" after 400 ns,"01" after 500 ns
   ,"10" after 600 ns ,"11" after 700 ns;
END;
```

شکل 9: تست بنچ

Result of simulation in isim:



شکل 10: نتایج شبیه سازی

حال با استفاده از 5 MUX 4 به 1 یک MUX 16 به 1 را به صورت زیر توصیف میکنیم.

Behavior of 16 to 1 MUX:

```
entity MUX16x1 is
    Port ( input : in  STD_LOGIC_VECTOR (15 DOWNTO 0);
           select1 : in  STD_LOGIC_VECTOR (3 DOWNTO 0);
           output : out  STD_LOGIC);
end MUX16x1;

architecture Behavioral of MUX16x1 is
   signal out_mux :STD_LOGIC_VECTOR (3 DOWNTO 0);
   component MUX4x1 is
    Port ( input : in  STD_LOGIC_VECTOR (3 DOWNTO 0);
           select1 : in  STD_LOGIC_VECTOR (1 DOWNTO 0);
           output : out  STD_LOGIC);
end component  MUX4x1;

begin
mux1: MUX4x1 port map (input=>input(3 downto 0),select1=>select1(1 downto 0),output=>out_mux(0));
mux2: MUX4x1 port map (input=>input(7 downto 4),select1=>select1(1 downto 0),output=>out_mux(1));
mux3: MUX4x1 port map (input=>input(11 downto 8),select1=>select1(1 downto 0),output=>out_mux(2));
mux4: MUX4x1 port map (input=>input(15 downto 12),select1=>select1(1 downto 0),output=>out_mux(3));
mux5: MUX4x1 port map (input=>out_mux(3 downto 0),select1=>select1(3 downto 2),output=>output);
end Behavioral;
```

شکل 11 :توصیف مالتی پلکسر 16 به یک در سطح گیت

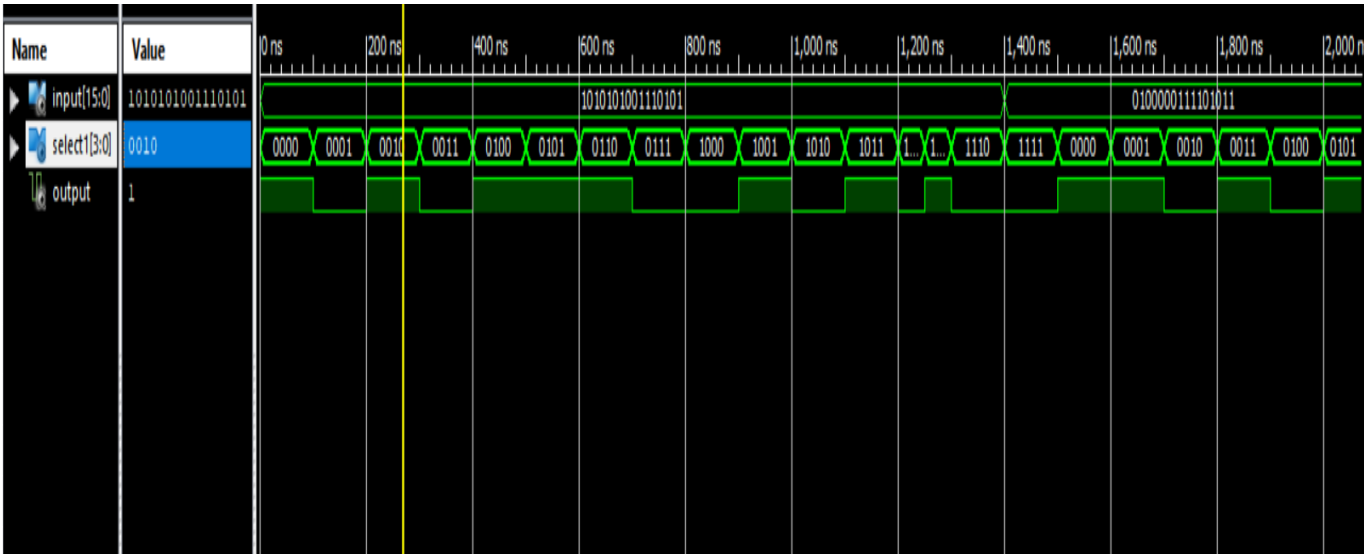
Test Bench:

```
   select1<="0000",
   "0001" after 100 ns,
   "0010" after 200 ns,
   "0011" after 300 ns,
   "0100" after 400 ns,
   "0101" after 500 ns,
   "0110" after 600 ns,
   "0111" after 700 ns,
   "1000" after 800 ns,
   "1001" after 900 ns,
   "1010" after 1000 ns,
   "1011" after 1100 ns,
   "1100" after 1200 ns,
   "1101" after 1250 ns,
   "1110" after 1300 ns,
   "1111" after 1400 ns,
   "0000" after 1500 ns,
   "0001" after 1600 ns,
   "0010" after 1700 ns,
   "0011" after 1800 ns,
   "0100" after 1900 ns,
   "0101" after 2000 ns,
   "0110" after 2100 ns,
   "0111" after 2200 ns,
   "1000" after 2300 ns,
   "1001" after 2400 ns,
   "1010" after 2500 ns,
   "1011" after 2600 ns,
   "1100" after 2700 ns,
   "1101" after 2800 ns,
   "1110" after 2900 ns,
   "1111" after 3000 ns;
   -- Clock process defini
```

```
ENTITY MUX16x1_tb IS
END MUX16x1_tb;

ARCHITECTURE behavior OF MUX16x1_tb IS

    COMPONENT MUX16x1
    PORT(
         input : IN  std_logic_vector(15 downto 0);
         select1 : IN  std_logic_vector(3 downto 0);
         output : OUT  std_logic
        );
    END COMPONENT;

   --Inputs
   signal input : std_logic_vector(15 downto 0) := (others => '0');
   signal select1 : std_logic_vector(3 downto 0) := (others => '0');

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

   -- Instantiate the Unit Under Test (UUT)
   uut: MUX16x1 PORT MAP (
          input => input,
          select1 => select1,
          output => output
        );
   input<="1010101001110101","0100000111101011" after 1400 ns;
   select1<="0000",
   "0001" after 100 ns,
   "0010" after 200 ns,
```

شکل 12 :تست بنچ 16 به 1

Result of simulation in isim:

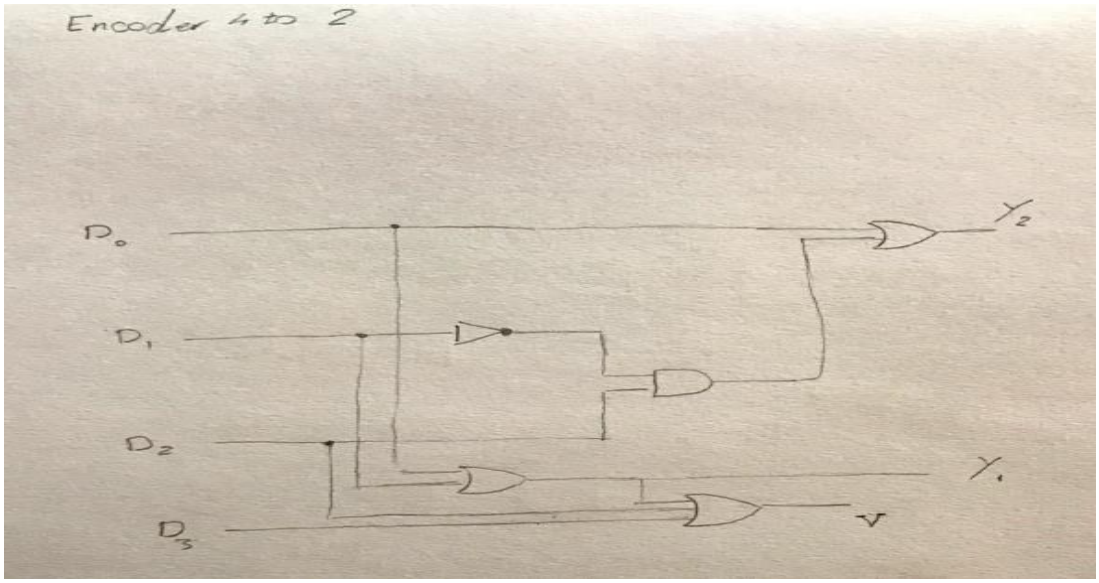


شکل 13: نتایج شبیه سازی

بخش دوم:

(Encoder) انکودر:

در این بخش هدف ما توصیف یک انکودر 4 به 2 در زبان VHDL در سطح گیت می باشد . این طراحی با استفاده از دستورات زبان و مدار طراحی شده در پیش گزارش انجام میدهیم. در شکل زیر یک انکودر الویت دار رسم شده است.



شکل 14 : مدار انکودر الویت دار در سطح گیت

Behavior of 4 to 2 Encoder:

```
entity EncoderD4x2 is
    Port ( input : in  STD_LOGIC_VECTOR (3 DOWNTO 0);
           output : out  STD_LOGIC_VECTOR (1 DOWNTO 0);
           v: out STD_LOGIC);
end EncoderD4x2;

architecture Behavioral of EncoderD4x2 is
   signal and1 : STD_LOGIC;

begin
  output(1)<=input(3) or input(2);
  and1<= (not input(2)) and input(1);
  output(0)<=and1 or input(3);
  v  <= input(1) or input(0) or input (2) or input (3);


end Behavioral;
```

شکل 15:توصیف انکودر در سطح گیت

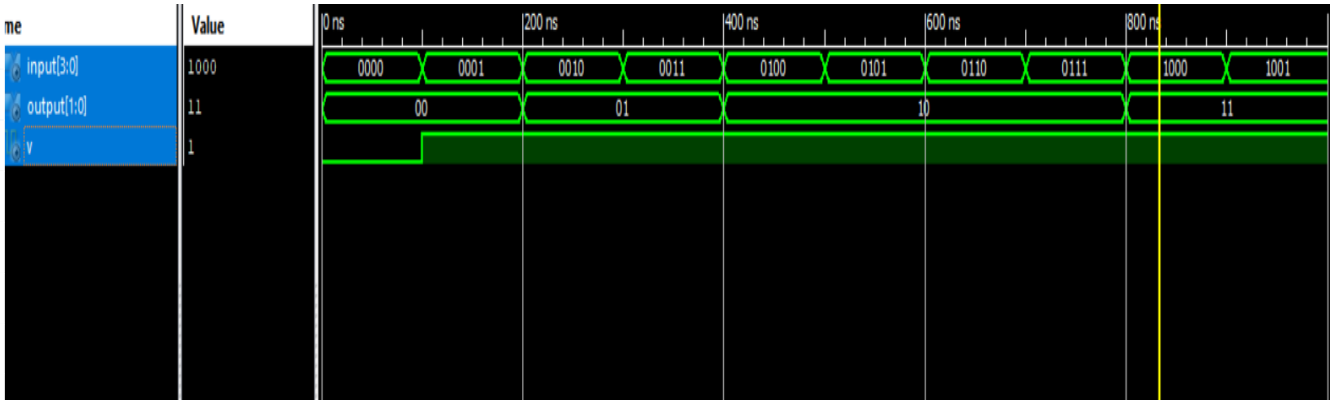
Test Bench:

```
      );
input<="0000",
"0001" after 100ns,
"0010" after 200ns,
"0011" after 300ns,
"0100" after 400ns,
"0101" after 500ns,
"0110" after 600ns,
"0111" after 700ns,
"1000" after 800ns,
"1001" after 900ns,
"1010" after 1000ns,
"1011" after 1100ns,
"1100" after 1200ns,
"1101" after 1300ns,
"1110" after 1400ns,
"1111" after 1500ns;
-- Clock process definitions
```

```
ENTITY Encoder_tb IS
END Encoder_tb;
ARCHITECTURE behavior OF Encoder_tb IS

    COMPONENT EncoderD4x2
    PORT(
         input : IN  std_logic_vector(3 downto 0);
         output : OUT  std_logic_vector(1 downto 0);
         v : OUT  std_logic
        );
    END COMPONENT;

   --Inputs
   signal input : std_logic_vector(3 downto 0) := (others => '0');

   --Outputs
   signal output : std_logic_vector(1 downto 0);
   signal v : std_logic;
   -- No clocks detected in port list. Replace <clock> below with
   -- appropriate port name BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: EncoderD4x2 PORT MAP (
          input => input,
          output => output,
          v => v
        );
```

شکل 16 : تست بنچ انکودر 4 به 2

Result of simulation in isim:

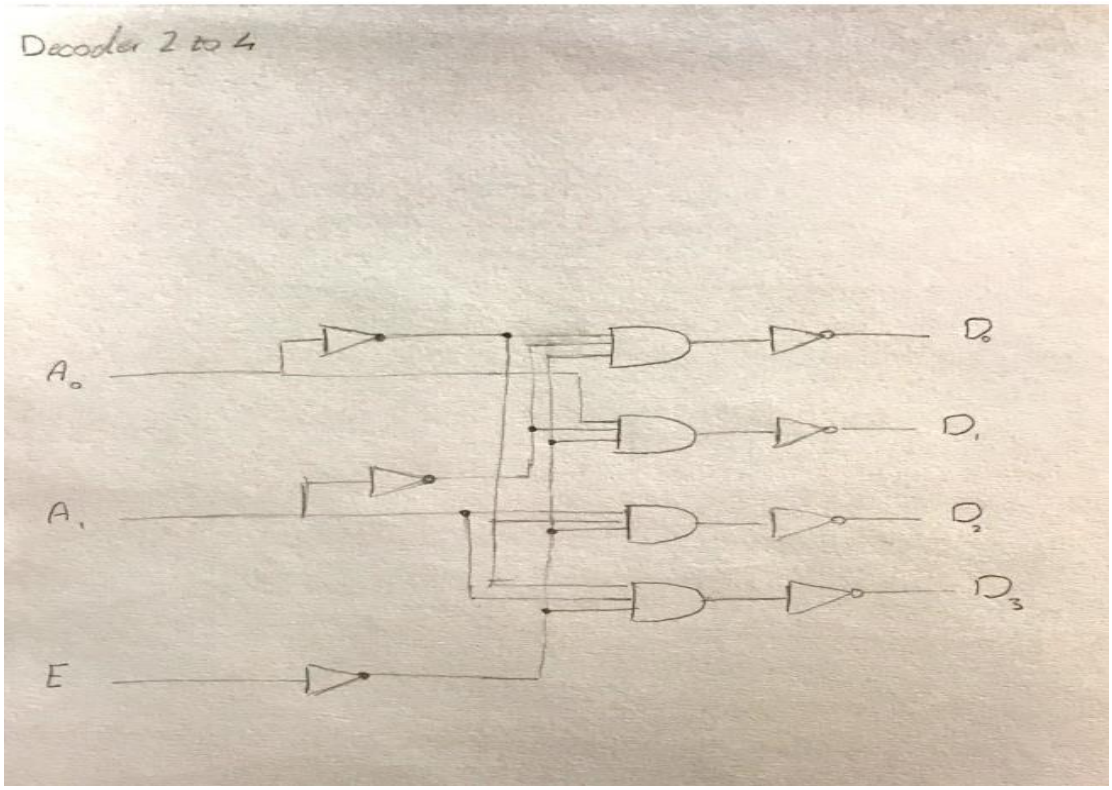


شکل 17: نتایج شبیه سازی

بخش سوم:

دیکودر (Decoder) :

در این بخش هدف ما توصیف یک دیکودر 2 به 4 در زبان VHDL در سطح گیت می باشد . این طراحی با استفاده از دستورات زبان و مدار طراحی شده در پیش گزارش انجام میدهیم.



شکل 18 : مدار رسم شده دیکودر 2 به 4

Behavior of 2 to 4 Decoder:

```
entity Decoder2x4 is
    Port ( input : in  STD_LOGIC_VECTOR (1 DOWNTO 0);
           enable : in  STD_LOGIC;
           output : out  STD_LOGIC_VECTOR (3 DOWNTO 0));
end Decoder2x4;

architecture Behavioral of Decoder2x4 is
 signal ands:std_logic_vector (3 downto 0);
begin
   ands(3)<=enable and input(0) and input (1);
   ands(2)<=enable and (not input(0)) and input (1);
   ands(1)<=enable and input(0) and  not (input (1));
   ands(0)<=enable and (not input(0)) and (not input (1));
   output<=ands;

end Behavioral;
```

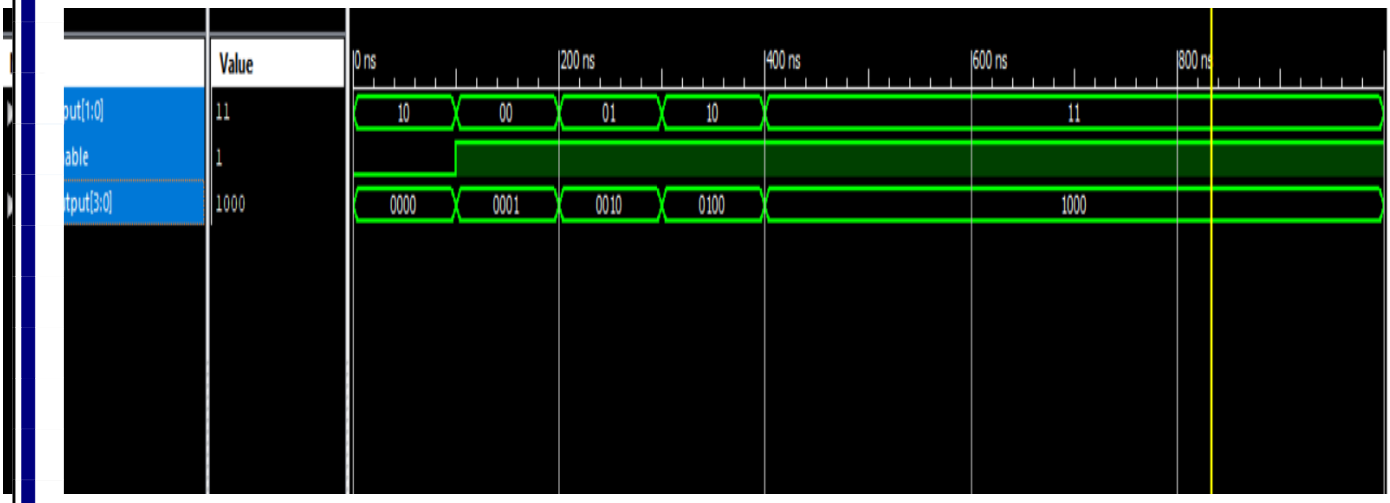
شکل 19 : توصیف دیکودر 2 به 4 در سطح گیت

Test Bench:

```
ENTITY Decoder_tb IS
END Decoder_tb;

ARCHITECTURE behavior OF Decoder_tb IS
    COMPONENT Decoder2x4
    PORT(
         input : IN  std_logic_vector(1 downto 0);
         enable : IN  std_logic;
         output : OUT  std_logic_vector(3 downto 0)
        );
    END COMPONENT;
   --Inputs
   signal input : std_logic_vector(1 downto 0) := (others => '0');
   signal enable : std_logic := '0';

   --Outputs
   signal output : std_logic_vector(3 downto 0);
   -- No clocks detected in port list. Replace <clock> below with
   -- appropriate port name
BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: Decoder2x4 PORT MAP (
          input => input,
          enable => enable,
          output => output
        );

   enable<='0','1' after 100 ns ;
   input<="10","00" after 100 ns ,"01" after 200 ns ,"10" after 300 ns,"11" after 400 ns;

END;
```

شکل 20: تست بنچ

Result of simulation in isim:

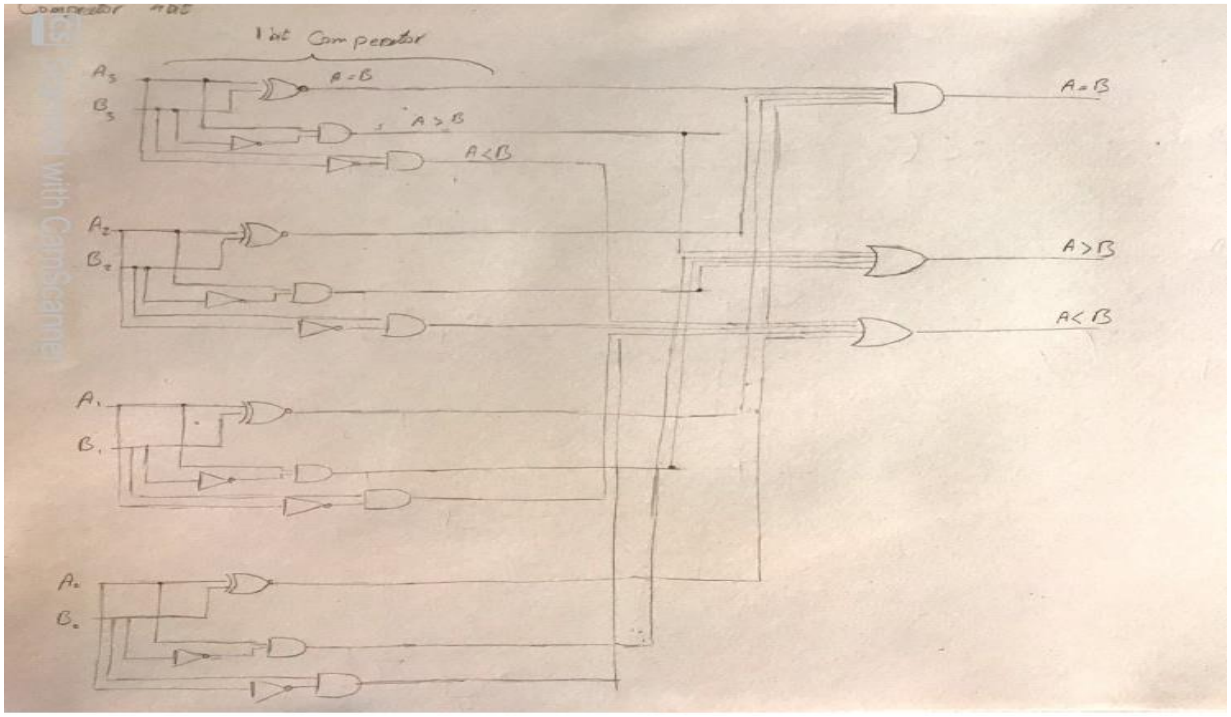


شکل 21 : نتایج شبیه سازی دیکودر

بخش چهارم:

مقایسه گر (Comparator)

در این بخش ابتدا یک مقایسه کننده تک بیتی را در سطح گیت در زبان VHDL با استفاده از مدار های طراحی شده در پیش گزارش توصیف میکنیم. سپس با استفاده از 4 مقایسه گر تک بیتی یک مقایسه گر 4 بیتی را طراحی میکنیم. که ابتدا از با ارزش ترین بیت شروع به مقایسه کرده و در هر مرحله در صورت برابر بودن آن دو بیت به بیت کم ارزش تر مراجعه میکنیم. در زبان VHDL ابتدا Component مقایسه گر تک بیتی را برای مقایسه گر 4 بیتی تعریف میکنیم. سپس 4 نمونه از آن را ساخته و ورودی و خروجی ها را با استفاده از Port MAP به هم وصل می کنیم و در نهایت نیز با نوشتن حالت های مختلف در Test Bench این دو مقایسه گر خود را آزمایش میکنیم و با استفاده شبیه سازی نتیجه آن را مشاهده میکنیم.



شکل 22: توصیف مقایسه گر 4 بیتی با استفاده از تک بیتی

Behavior of 1 bit comparator :

```
entity bit_comp is
     Port ( A : in  STD_LOGIC;
            B : in  STD_LOGIC;
            GT1 : in  STD_LOGIC;
            EQ : in  STD_LOGIC;
            LT : in  STD_LOGIC;
            A_GT_B : out  STD_LOGIC;
            A_EQ_B : out  STD_LOGIC;
            A_LT_B : out  STD_LOGIC);
end bit_comp;

architecture Behavioral of bit_comp is
    signal equal: std_logic;
    signal GT111:std_logic;
    signal LT1:std_logic;
    signal great:std_logic;
    signal less:std_logic;
begin
 equal<= A xnor B;
 GT111<= A and (not B) ;
 LT1<= (not A) and B ;
 great<=equal and     GT1;
 less<= equal and LT;
 A_EQ_B<= EQ and equal ;
 A_GT_B<= gt111 or great;
 A_LT_B<= LT1 or less;
end Behavioral;
```

شکل 23: توصیف مقایسه گر تک بیتی

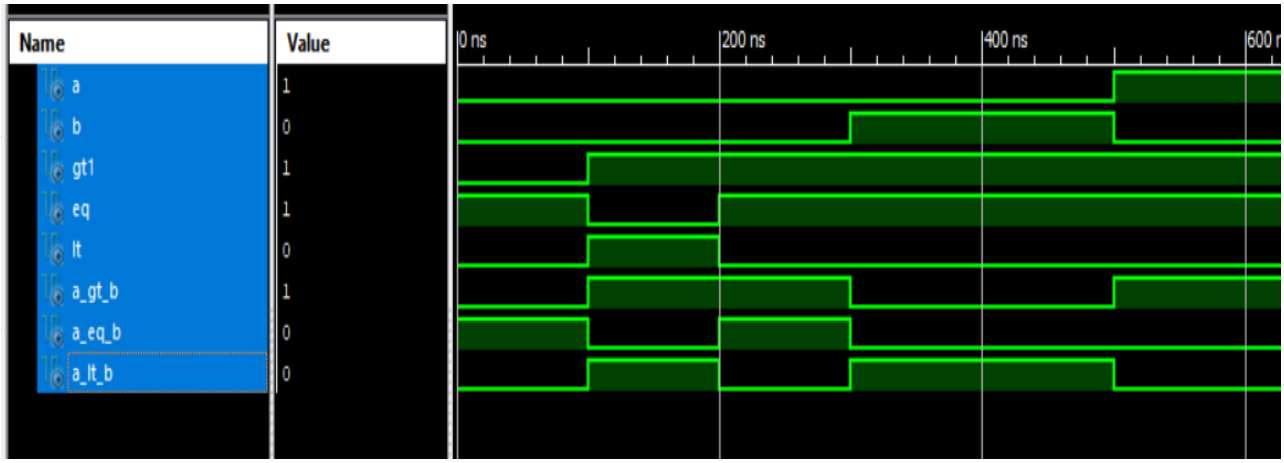
Test Bench:

```
BEGIN
   -- Instantiate the Unit Under Test (UUT)
   uut: bit_comp PORT MAP (
          A => A,
          B => B,
          GT1 => GT1,
          EQ => EQ,
          LT => LT,
          A_GT_B => A_GT_B,
          A_EQ_B => A_EQ_B,
          A_LT_B => A_LT_B
        );

  A<='0' ,'1' after 500 ns;
  B<='0','1' after 300 ns,'0' after 500 ns;
  GT1<='0','1' after 100 ns;
   EQ<='1','0' after 100 ns,'1' after 200 ns;
   LT<='0','1' after 100 ns,'0' after 200 ns;

END;
```

```
ENTITY comprator_1bit IS
END comprator_1bit;
ARCHITECTURE behavior OF comprator_1bit IS
    -- Component Declaration for the Unit Un
    COMPONENT bit_comp
    PORT(
         A : IN  std_logic;
         B : IN  std_logic;
         GT1 : IN  std_logic;
         EQ : IN  std_logic;
         LT : IN  std_logic;
         A_GT_B : OUT  std_logic;
         A_EQ_B : OUT  std_logic;
         A_LT_B : OUT  std_logic
        );
    END COMPONENT;
   --Inputs
   signal A : std_logic := '0';
   signal B : std_logic := '0';
   signal GT1 : std_logic := '0';
   signal EQ : std_logic := '0';
   signal LT : std_logic := '0';

   --Outputs
   signal A_GT_B : std_logic;
   signal A_EQ_B : std_logic;
   signal A_LT_B : std_logic;
   -- No clocks detected in port list. Repla
   -- appropriate port name
BEGIN
```

شکل 24: تست بنچ

Result of simulation in isim:



شکل 25 : نتایج شبیه سازی

حال با استفاده از 4 مقایسه گر تک بیتی یک مقایسه گر 4 بیتی زا توصیف میکنیم.

Behavior of 4 bit comparator:

```
entity comprator_4bit is
    Port ( A : in  STD_LOGIC_VECTOR (3 DOWNTO 0);
           B : in  STD_LOGIC_VECTOR (3 DOWNTO 0);
           GT1 : in  STD_LOGIC;
           EQ : in  STD_LOGIC;
           LT : in  STD_LOGIC;
           A_GT_B : out  STD_LOGIC;
           A_EQ_B : out  STD_LOGIC;
           A_LT_B : out  STD_LOGIC);
end comprator_4bit;

architecture Behavioral of comprator_4bit is
signal GT111 :STD_LOGIC_VECTOR (2 DOWNTO 0);
signal EQ1 :STD_LOGIC_VECTOR (2 DOWNTO 0);
signal LT1 :STD_LOGIC_VECTOR (2 DOWNTO 0);
     component bit_comp is
    Port ( A : in  STD_LOGIC;
           B : in  STD_LOGIC;
           GT1 : in  STD_LOGIC;
           EQ : in  STD_LOGIC;
           LT : in  STD_LOGIC;
           A_GT_B : out  STD_LOGIC;
           A_EQ_B : out  STD_LOGIC;
           A_LT_B : out  STD_LOGIC);
end  component bit_comp;
begin
   comp1:bit_comp port map (A=>A(0),B=>B(0),GT1=>GT1,EQ=>EQ,LT=>LT,A_GT_b=>GT111(0),A_EQ_B=>EQ1(0),A_LT_B=>LT1(0));
   comp2:bit_comp port map (A=>A(1),B=>B(1),GT1=>GT111(0),EQ=>EQ1(0),LT=>LT1(0),A_GT_B=>GT111(1),A_EQ_B=>EQ1(1),A_LT_B=>LT1(1));
   comp3:bit_comp port map (A=>A(2),B=>B(2),GT1=>GT111(1),EQ=>EQ1(1),LT=>LT1(1),A_GT_B=>GT111(2),A_EQ_B=>EQ1(2),A_LT_B=>LT1(2));
   comp4:bit_comp port map (A=>A(3),B=>B(3),GT1=>GT111(2),EQ=>EQ1(2),LT=>LT1(2),A_GT_B=>A_GT_B,A_EQ_B=>A_EQ_B,A_LT_B=>A_LT_B);
end Behavioral;
```

شکل 26: توصیف مقایسه گر 4 بیتی در زبان

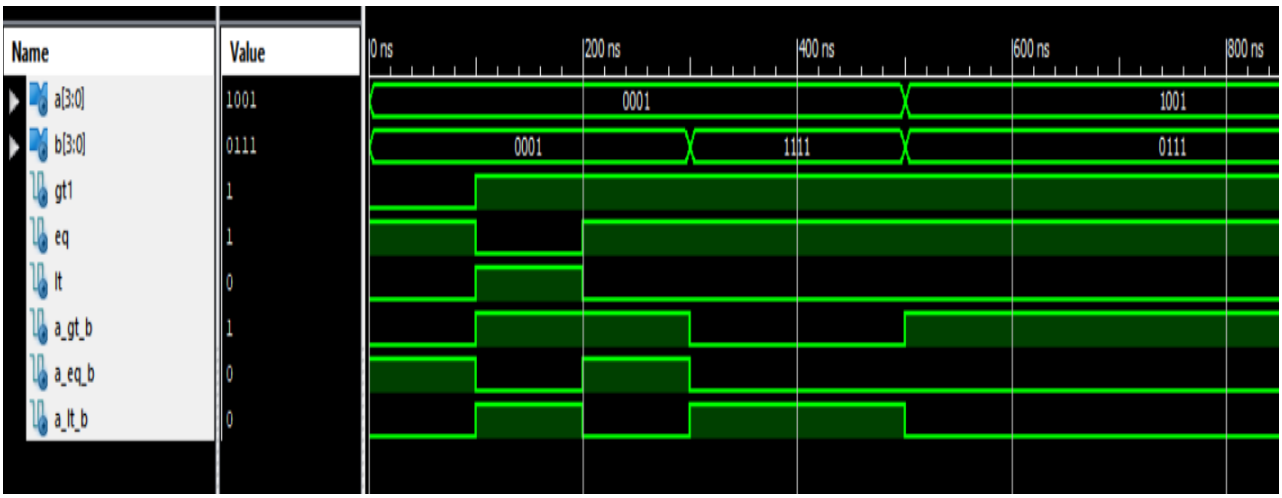
Test Bench:

```
uut: comprator_4bit PORT MAP (
        A => A,
        B => B,
        GT1 => GT1,
        EQ => EQ,
        LT => LT,
        A_GT_B => A_GT_B,
        A_EQ_B => A_EQ_B,
        A_LT_B => A_LT_B
      );

   -- Clock process definitions
    A<="0001" ,"1001" after 500 ns;
  B<="0001","1111" after 300 ns,"0111" after 500 ns;
  GT1<='0','1' after 100 ns;
   EQ<='1','0' after 100 ns,'1' after 200 ns;
   LT<='0','1' after 100 ns,'0' after 200 ns;

END;
```

```
ENTITY comprator4_tb IS
END comprator4_tb;
ARCHITECTURE behavior OF comprator4_tb IS
-- Component Declaration for the Unit Under Test (UUT
    COMPONENT comprator_4bit
    PORT(
         A : IN  std_logic_vector(3 downto 0);
         B : IN  std_logic_vector(3 downto 0);
         GT1 : IN  std_logic;
         EQ : IN  std_logic;
         LT : IN  std_logic;
         A_GT_B : OUT  std_logic;
         A_EQ_B : OUT  std_logic;
         A_LT_B : OUT  std_logic
        );
    END COMPONENT;
   --Inputs
   signal A : std_logic_vector(3 downto 0) := (others => '0');
   signal B : std_logic_vector(3 downto 0) := (others => '0');
   signal GT1 : std_logic := '0';
   signal EQ : std_logic := '0';
   signal LT : std_logic := '0';    --Outputs
   signal A_GT_B : std_logic;
   signal A_EQ_B : std_logic;
   signal A_LT_B : std_logic;
   -- No clocks detected in port list. Replace <clock> below with
   -- appropriate port name
BEGIN
```

شکل 27: تست بنچ

Result of simulation in isim:



نتیجه گیری:

به طور کلی و در این آزمایش هدف آشنایی با نحوه عملکرد و پیاده سازی هر یک از مدارهای پایه در سطح تجرید گیت است.و یادگیری بهتر زبان VHDL و کار با ساختار ارجاع شرطی و ارجاع انتخابی که با پیاده سازی ماژول های مالتی پلکسر این کار را انجام دادیم . سپس یاد آوری درس مدار منطقی و آموزش کار با زبان VHDL که با استفاده از ماژول های دیکودر انکودر و مقایسه گر ها این کار به درستی انجام شد.