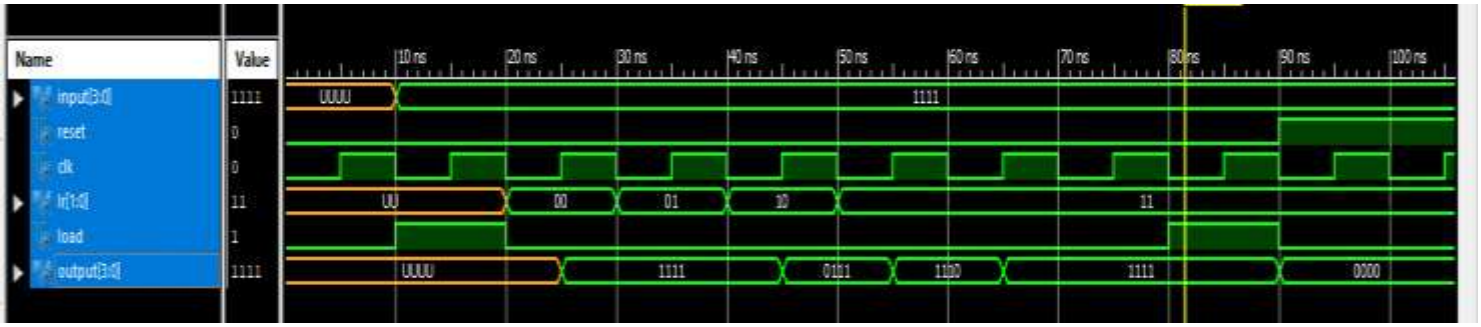
گزارش کارآزمایش ٦ معماری کامپیوتر

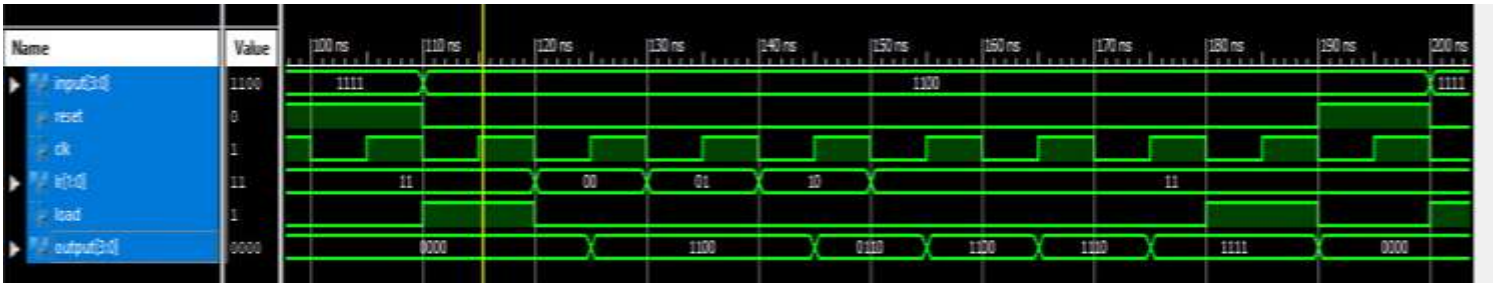
محمد مهدی نظری ۹۹۳۱۰۶۱ – آرمین ابراهیمی صبا ۹۹۳۱۰۸۶

در این آزمایش برای پیاده سازی شیفت رجیستر مربوطه با شرایط ذکر شده از دستورات شرطی if و else و بلوک process استفاده شده .

```
32  entity register_behavioral is
33  port (input    : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
34        output      : out STD_LOGIC_VECTOR(3 DOWNTO 0);
35        reset     : in std_logic ;
36        clk       : in std_logic ;
37        LR     : IN STD_LOGIC_VECTOR(1 DOWNTO 0);
38        load      : in std_logic
39        );
40  end register_behavioral;
41
42  architecture Behavioral of register_behavioral is
43  signal storage : STD_LOGIC_VECTOR(3 DOWNTO 0);
44  begin
45  process(clk, reset, LR)
46     begin
47     if reset = '1' then
48        output <= "0000";
49     elsif rising_edge(clk) then
50        if load = '1' then
51           storage <= input;
52        else
53     case LR is
54
55    when "00" => storage<= storage;
56    when "01" =>   storage <= '0' & storage( 3 downto 1);
57    when "10" =>  storage <= storage( 2 downto 0) & '0';
58    when "11" =>   storage <= storage(3) & storage( 3 downto 1);
59    when others => storage<= storage;
60     end case;
61        output <= storage;
62        end if;
63          end if;
64          end process;
65
66  end Behavioral;
```

تست بنچ :





به عنوان مثال با ورودی ۱۱۱۱ شروع کرده و LR را ۰۰ داده و میبینیم در دو لبه بالارونده ساعت خروجی تغییر نکرده .

بعد آن را ۰۱ کرده و ورودی ۱ بیت به راست شیفت منطقی پیدا میکند .

سپس ۱۰ را به ورودی LR داده و خروجی قبلی (یعنی ۰۱۱۱) یک بیت به چپ (چه منطقی چه ریاضی) شیفت پیدا کرده و ۱۱۱۰ را تولید میکند .

در مرحله آخر هم ۱۱ را امتحان میکنیم و خروجی قبلی یک بیت به راست شیفت ریاضی پیدا کرده و بیت علامت خود را بدون تغییر حفظ میکند .

ورودی load در بین ۱۰ تا ۲۰ نانوثانیه و ورودی reset در ۹۰ تا ۱۱۰ نانوثانیه تست شده اند چون لود سنکرون است در پالس بالارونده ساعت بعدی اعمال میشود اما چون ریست آسنکرون است در لحظه خروجی را ۰ میکند .

بین ۱۱۰ تا ۲۰۰ نانوثتنیه هم یک ورودی دیگر تست شده .

۲ – شیفت رجیستر با استفاده از سطوح گیت

این شیفت رجیستر با استفاده از گیت های and و or و یک دیکودر ۲ به ۴ برای تعیین حالات ورودی LR و یک D-Flip Flop برای نگهداری مقادیر خروجی استفاده شده .

```
32  entity register_gate is
33  port (input    : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
34        output      : out STD_LOGIC_VECTOR(3 DOWNTO 0);
35        reset     : in std_logic ;
36        clk       : in std_logic ;
37        LR     : IN STD_LOGIC_VECTOR(1 DOWNTO 0);
38        load      : in std_logic
39        );
40  end register_gate;
41
42  architecture Behavioral of register_gate is
43  component dff is
44     port(
45        input : in std_logic ;
46        Q : out std_logic;
47        Clk :in std_logic;
48        reset: in std_logic;
49        D :in  std_logic ;
50        load : in std_logic
51     );
52  end component;
53
54  component decoder2to4 is
55   port(
56   a : in STD_LOGIC_VECTOR(1 downto 0);
57   b : out STD_LOGIC_VECTOR(3 downto 0)
58   );
59  end component;
60  signal dec_out : std_logic_vector(3 downto 0);
61  signal and_out : std_logic_vector(9 downto 0);
62  signal or_out : std_logic_VECTOR(1 DOWNTO 0);
63  signal output_or : std_logic_vector(3 downto 0);
64
```

```
65  begin
66  decode : decoder2to4 port map (
67  a => LR ,
68  b => dec_out
69  );
70
71  or_out(0) <= dec_out(2) or dec_out(3) ;
72  or_out(1) <= dec_out(0) or dec_out(3) ;
73
74  and_out(0) <= input(0) and dec_out(0) ;
75  and_out(1) <= input(0) and dec_out(1) ;
76  and_out(2) <= input(1) and or_out(0) ;
77  and_out(3) <= input(1) and dec_out(0) ;
78  and_out(4) <= input(1) and dec_out(1) ;
79  and_out(5) <= input(2) and or_out(0) ;
80  and_out(6) <= input(2) and dec_out(0) ;
81  and_out(7) <= input(2) and dec_out(1) ;
82  and_out(8) <= input(3) and or_out(0) ;
83  and_out(9) <= input(3) and or_out(1) ;
84
85  output_or(0) <= and_out(0) or and_out(2)  ;
86  output_or(1) <= and_out(1) or and_out(3) or and_out(5) ;
87  output_or(2) <= and_out(4) or and_out(6) or and_out(8) ;
88  output_or(3) <= and_out(7) or and_out(9)  ;
89
```

```
90   dff0 : dff port map (
91   input => input(0) ,
92   Q => output(0) ,
93   Clk => clk ,
94   reset => reset ,
95   D => output_or(0) ,
96   load => load );
97
98   dff1 : dff port map (
99   input => input(1) ,
100  Q => output(1) ,
101  Clk => clk ,
102  reset => reset ,
103  D => output_or(1) ,
104  load => load );
105
106  dff2 : dff port map (
107  input => input(2) ,
108  Q => output(2) ,
109  Clk => clk ,
110  reset => reset ,
111  D => output_or(2) ,
112  load => load );
113
114  dff3 : dff port map (
115  input => input(3) ,
116  Q => output(3) ,
117  Clk => clk ,
118  reset => reset ,
119  D => output_or(3) ,
120  load => load );
121
122  end Behavioral;
123
```

تست بنچ :

در این قسمت هم همان شرایط رجیستر با if و else چک شده .