"به نام یزدان پاک"

گزارش کار آزمایش دوم

اعضای گروه:

کیانا آقا کثیری 9831006

محمد چوپان 9831125

سارا تاجرنیا 9831016

نویسنده گزارش : سارا تاجرنیا

تاریخ آزمایش : 99/12/20

تاریخ تحویل گزارش :99/12/26

شرح آزمایش:

۱) یک فلیپ‌فلاپ از نوع D(DFF) با سیگنال Reset ناهمگام (asynchronous) در منطق منفی (active low) طراحی کنید.
۲) یک فلیپ‌فلاپ از نوع T(TFF) با سیگنال Reset ناهمگام طراحی کنید.
۳) یک Ripple Counter، ۴ بیتی که نمونه‌ای از شمارنده‌های ناهمگام می‌باشد را با استفاده از TFF ساخته شده در (۲) طراحی کنید.
۴) مدار یک Sequence detector برای رشته‌ی "۱۱۰۱" را ابتدا به صورت Mealy و سپس Moore طراحی کرده و آن را با استفاده از زبان VHDL طراحی کنید.
۵) مداری طراحی کنید که رخداد هر یک از دو رشته "۰۱۱۰" و "۰۱۰۱" را در ورودی تشخیص دهد.

شکل **1**:شرح آزمایش

خروجی‌های مورد انتظار آزمایش:

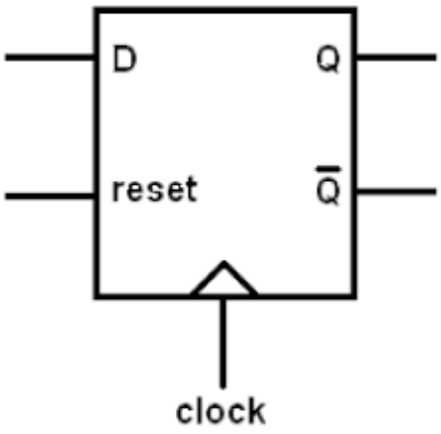
هریک از بخش‌های این آزمایش در قالب‌های زیر به ترتیب تحویل مدرس آزمایشگاه گردد:

- طراحی شماتیک طرح (محتوای توصیف) بر صورت فیزیکی
- پیاده‌سازی مدارها با استفاده از زبان‌های توصیف
- شبیه‌سازی مدارهای توصیف شده و تهیه Testbench برای آن و نشان دادن درستی عملکرد آن‌ها.
- سنتز و پیاده‌سازی مدارها بر روی برد FPGA و اثبات درستی عملکرد مدارها. (برای استفاده از کلاک برد و مشاهده نتایج نیاز به کاهش فرکانس برد می‌باشد، بدین جهت متناسب با شرایط کلاس این کاهش فرکانس توسط مدرس آزمایشگاه و یا دانشجویان انجام گیرد.)

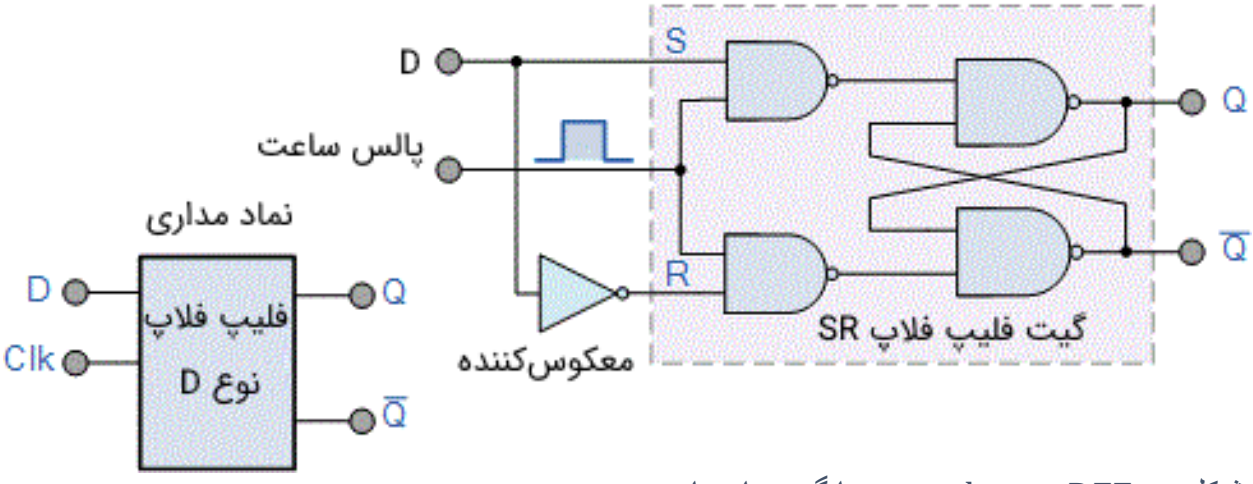
شکل **2**:خروجی مورد انتظار آزمایش

DFF

فلیپ فلاپD ، بدون شک یکی از مهمترین انواع فلیپ فلاپ‌های کلاکـدار است؛ زیرا این فلیپ فلاپ اطمینان حاصل می‌کند که ورودی های R و S به صورت همزمان با یکدیگر در سطح منطقی صفر نباشند. فلیپ فلاپD ، از یک فلیپ فلاپ RS ساخته شده است که یک معکوس‌کننده بین ورودی‌های R و S افزوده شده است و داده ورودی تکی D مطابق شکل زیر به فلیپ فلاپ وارد می‌شود. این داده ورودی تکی که با عنوان D مشخص می‌شود، بدون تاخیر به ورودی SET فلیپ فلاپ اعمال می‌شود. همچنین معکوس‌شده ورودی D را به ورودی RESET فلیپ فلاپ وارد می‌کنند. بنابراین از یک فلیپ فلاپ حساس به سطح RS، یک فلیپ فلاپ حساس به سطح D ایجاد می‌شود که در آن S=D و R = NOT D است.



شکل **3**: asynchrone DFF



شکل **4**: asynchrone DFF با گیت های پایه

```
entity DFF is
    Port ( input : in  STD_LOGIC;
           clk : in  STD_LOGIC;
           reset : in  STD_LOGIC;
           output : out  STD_LOGIC);
end DFF;

architecture Behavioral of DFF is


begin
REG:process(reset,clk)
begin
   if(reset='0') then
   --active low
      output<='1';

   else if(clk' event and clk='1') then
      output<= not input;
      end if;
   end if;

end process;
end Behavioral;
```

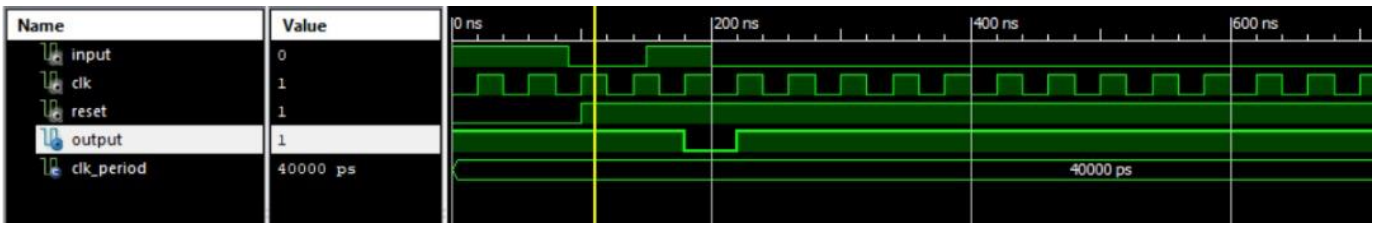
شکل **5**: ساختار DFF با سیگنال reset

Test Bench :

```
   --Inputs
   signal input : std_logic := '0';
   signal clk : std_logic := '0';
   signal reset : std_logic := '0';

   --Outputs
   signal output : std_logic;

   -- Clock period definitions
   constant clk_period : time := 40 ns;

BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: DFF PORT MAP (
          input => input,
          clk => clk,
          reset => reset,
          output => output
        );

   -- Clock process definitions
   clk_process :process
   begin
      clk <= '0';
      wait for clk_period/2;
      clk <= '1';
      wait for clk_period/2;
   end process;
   input<='1','0' after 90 ns,'1' after 150 ns,'0' after 200 ns;
   reset<='0','1' after 100 ns;

   -- Stimulus process
```

شکل :**6** تست بنچ DFF با سیگنال reset

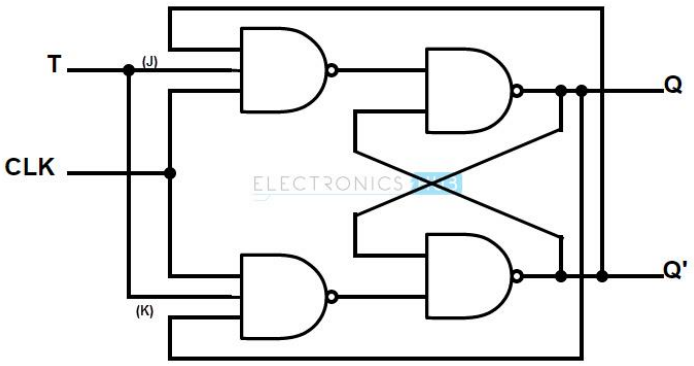
Result of Simulation in isim :



شکل :7 نتایج شبیه سازی

TFF

اشاره دارد. زمانی که کلید متصل به یک (Toggle) به حرف اول کلمه تاگل T در نام‌گذاری فلیپ فلاپ‌های نوع T حرف لامپ را تاگل کنید، در واقع حالت منطقی آن‌ها را از یک سطح منطقی به سطح منطقی دیگر تغییر می‌دهید. این دقیقا مشابه پدیده‌ای است که در آن، یک ورودی با سطح منطقی یک برای یک فلیپ فلاپ فراهم می‌کنید. در این شرایط اگر خروجی خود در سطح منطقی یک قرار داشته باشد، به سطح صفر منطقی تغییر می‌کند و اگر خود در سطح منطقی صفر باشد، به منطقی، باعث می‌شود که فلیپ فلاپ حالت خروجی فعلی LOW یک منطقی تغییر سطح می‌دهد. یک ورودی سطح صفر یا .در زیر آورده شده است Tخود را حفظ کند.



شکل :8 TFF

```
--use UNISIM.VComponents.all;

entity TFF is
    Port ( input : in  STD_LOGIC;
           clk : in  STD_LOGIC;
           reset : in  STD_LOGIC;
           output : out  STD_LOGIC);
end TFF;

architecture Behavioral of TFF is
   signal output1:std_logic;

begin
   reg:process(reset,clk)
   begin
   if(reset='0') then
      output1<='0';
   else  if (clk'event and clk='1') then
         if input='0' then
         -- it must be output1<=output1 but it's avtive low
            output1 <=  output1;
         elsif input='1' then
         -- it must be output1<=not(output1) but it's avtive low
         output1 <=  not(output1);
         end if;
         end if;
   end if;

end process;

   output<=output1;
end Behavioral;
```

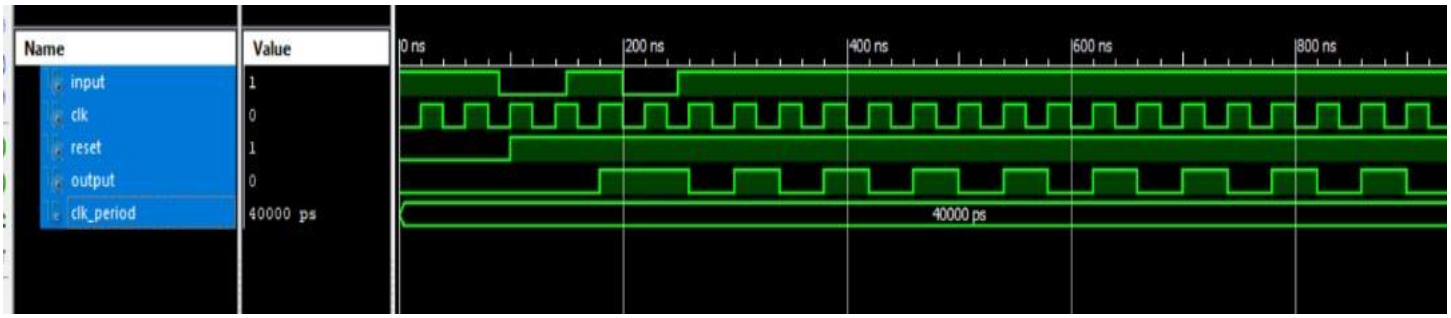
شکل :9 ساختار TFF با سیگنال reset

Test Bench :

```
BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: TFF PORT MAP (
          input => input,
          clk => clk,
          reset => reset,
          output => output
        );

   -- Clock process definitions
   clk_process :process
   begin
      clk <= '0';
      wait for clk_period/2;
      clk <= '1';
      wait for clk_period/2;
   end process;

   input<='1','0' after 90 ns,'1' after 150 ns,'0' after 200 ns,'1' after 250 ns,'1' after 300 ns;
   reset<='0','1' after 100 ns;


END;
```

شکل :**10** تست بنچ TFF با سیگنال reset

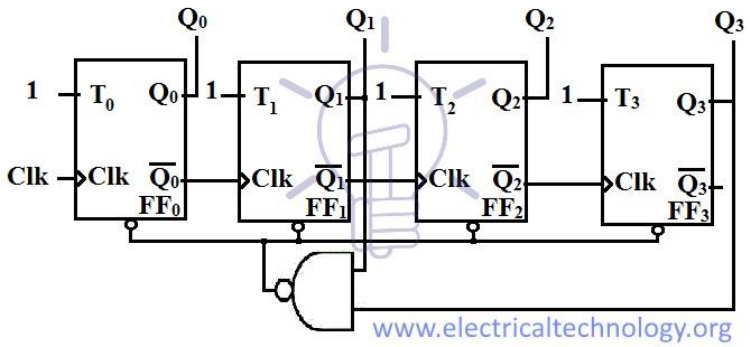
Result of Simulation in isim :



شکل :11 نتایج شبیه سازی

Ripple Counter

شمارنده ای که از فیلپ فلاپهای سری تشکیل می شود وقتی که وضعیت اولین فیلپ فلاپ تغییر می کند روی دومی اثر می گذارد ، دومی نیر به نوبه خود روی سومی اثر می گذراد و این روند به همین ترتیب تا تغییر وضعیت آخرین فلیپ فلاپ ادامه می یابد.



شکل **12**: شکل Ripple counter ۴ بیتی با استفاده از TFF

```
entity ripple is
    Port ( clk : in  STD_LOGIC;
           reset : in  STD_LOGIC;
           output : out  STD_LOGIC_vector (3 downto 0));
end ripple;

architecture Behavioral of ripple is
   component TFF is
   Port ( input : in  STD_LOGIC;
          clk : in  STD_LOGIC;
          reset : in  STD_LOGIC;
          output : out  STD_LOGIC);
   end component TFF;
   signal temp_out:std_logic_vector (3 downto 0);

begin

   TFF1: TFF port map(input=>'1',clk=>clk,reset=>reset,output=>temp_out(0));
   TFF2: TFF port map(input=>'1',clk=>temp_out(0),reset=>reset,output=>temp_out(1));
   TFF3: TFF port map(input=>'1',clk=>temp_out(1),reset=>reset,output=>temp_out(2));
   TFF4: TFF port map(input=>'1',clk=>temp_out(2),reset=>reset,output=>temp_out(3));
   output<=temp_out;
```

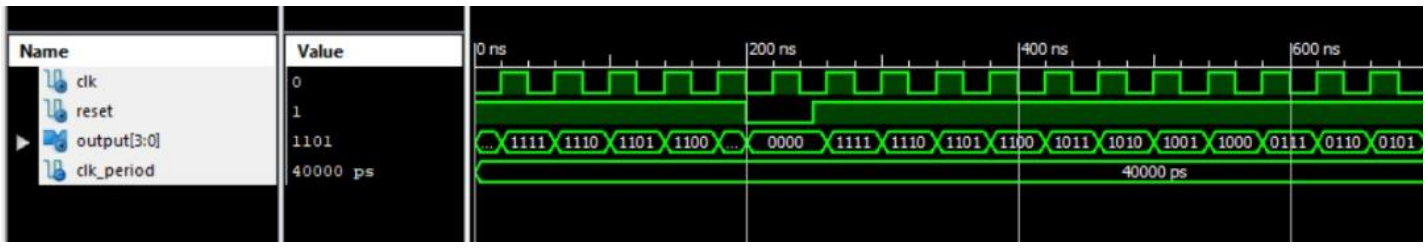
شکل **13**: ساختار Ripple counter ۴ بیتی با استفاده از TFF

Test Bench :

```
BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: ripple PORT MAP (
          clk => clk,
          reset => reset,
          output => output
        );

   -- Clock process definitions
   clk_process :process
   begin
      clk <= '0';
      wait for clk_period/2;
      clk <= '1';
      wait for clk_period/2;
   end process;
   reset<='1','0' after 200 ns,'1' after 250 ns;
```

شکل **14**: تست بنچ Ripple counter ۴ بیتی با استفاده از TFF

Result of Simulation in isim :



شکل 15: نتایج شبیه سازی

Detector 1101 mealy

```
entity SD_1101_mealy is
    Port ( input : in  STD_LOGIC;
           clk : in  STD_LOGIC;
           output : out  STD_LOGIC);
end SD_1101_mealy;

architecture Behavioral of SD_1101_mealy is
type state_t is (init , s0 , s1,s2);
   signal state : state_t := init;
   signal next_state : state_t := init;
   signal temp:std_logic;
begin
   CMB : process(state , input)
   begin
   temp<='0';
   case state is
         when init=>
               if(input = '1') then
                     next_state <= s0;
               else
                     next_state <= init ;
               end if;
         when s0=>
            if(input = '1') then
                     next_state <= s1;
               else
                     next_state <= init ;
               end if;

         when s1=>
                  if(input = '1') then
                     next_state <= s1;
                  else
                     next_state <= s2 ;
               end if;

         when s2=>
            if(input = '1') then
                     temp<='1';
                     next_state <= s0;

               else
                     next_state <= init ;
               end if;

   end case;
   end process;
   REG : process(clk)
   begin
      if(clk'event and clk = '1') then
               state <= next_state;
   end if;

      end process;
   output<=temp;
end Behavioral;
```

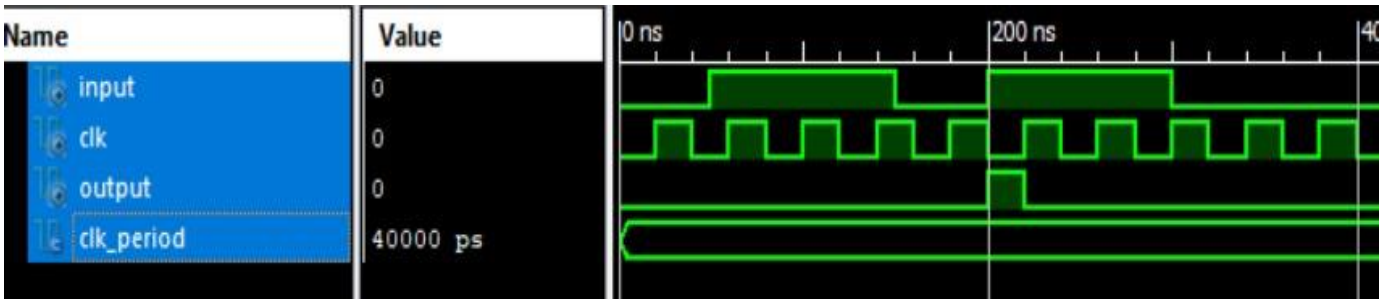
شکل **16**: ساختار sequence detector به صورت Mealy

Test Bench:

```
BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: SD_1101_mealy PORT MAP (
          input => input,
          clk => clk,
          output => output
        );

   -- Clock process definitions
   clk_process :process
   begin
      clk <= '0';
      wait for clk_period/2;
      clk <= '1';
      wait for clk_period/2;
   end process;

   input<='0','1' after 50 ns,'1' after 100 ns,'0' after 150 ns,'1' after 200 ns,'1' after 250 ns,'0' after 300 ns;
END;
```

شکل **17**: تست بنچ sequence detector به صورت Mealy

Result of Simulation in isim :



شکل 18: نتایج شبیه سازی

Detector 1101 moore

```
entity SD_1101_moore is
    Port ( input : in  STD_LOGIC;
           clk : in  STD_LOGIC;
           output : out  STD_LOGIC);
end SD_1101_moore;

architecture Behavioral of SD_1101_moore is
   type state_t is (init , s0 , s1,s2,s3);
   signal state : state_t := init;
   signal next_state : state_t := init;
begin
   CMB : process(state , input)
   begin
   case state is
         when init=>
               if(input = '1') then
                     next_state <= s0;
               else
                     next_state <= init ;
               end if;
         when s0=>
            if(input = '1') then
                     next_state <= s1;
               else
                     next_state <= init ;
               end if;

         when s1=>
                  if(input = '1') then
                     next_state <= s1;
                  else
                     next_state <= s2 ;
               end if;

         when s2=>
            if(input = '1') then
                     next_state <= s3;
               else
                     next_state <= init ;
               end if;
         when s3=>
            if(input = '1') then
                     next_state <= s1;
               else
                     next_state <= init ;
               end if;


   end case;
   end process;
   REG : process(clk)
   begin
      if(clk'event and clk = '1') then
               state <= next_state;
   end if;

      end process;
   output<='1' when state=s3 else
           '0';
end Behavioral;
```

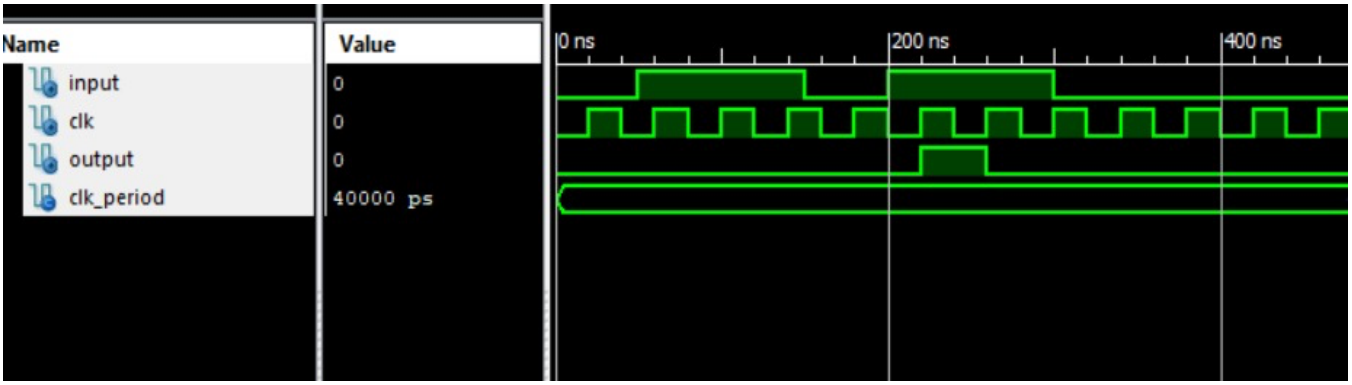
شکل **19**: ساختار sequence detector به صورت Moore

Test Bench:

```
BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: SD_1101_moore PORT MAP (
          input => input,
          clk => clk,
          output => output
        );

   -- Clock process definitions
   clk_process :process
   begin
      clk <= '0';
      wait for clk_period/2;
      clk <= '1';
      wait for clk_period/2;
   end process;


   input<='0','1' after 50 ns,'1' after 100 ns,'0' after 150 ns,'1' after 200 ns,'1' after 250 ns,'0' after 300 ns;
END;
```

شکل **20**: تست بنچ sequence detector به صورت Moore

Result of Simulation in isim :



شکل 21: نتایج شبیه سازی

```
entity SD_both is
    Port ( input : in  STD_LOGIC;
           clk : in  STD_LOGIC;
           output : out  STD_LOGIC);
end SD_both;

architecture Behavioral of SD_both is
   component SD_1101_moore is
    Port ( input : in  STD_LOGIC;
           clk : in  STD_LOGIC;
           output : out  STD_LOGIC);
end component SD_1101_moore;
component SD_0101_mealy is
    Port ( input : in  STD_LOGIC;
           clk : in  STD_LOGIC;
           output : out  STD_LOGIC);
end  component SD_0101_mealy;
   signal temp:std_logic_vector (1 downto 0);

begin
   SD1: SD_1101_moore port map (input=> input,clk=>clk,output=>temp(0));
   SD2: SD_0101_mealy port map (input=> input,clk=>clk,output=>temp(1));

   output<=(temp(0)) or (temp(1));

end Behavioral;
```

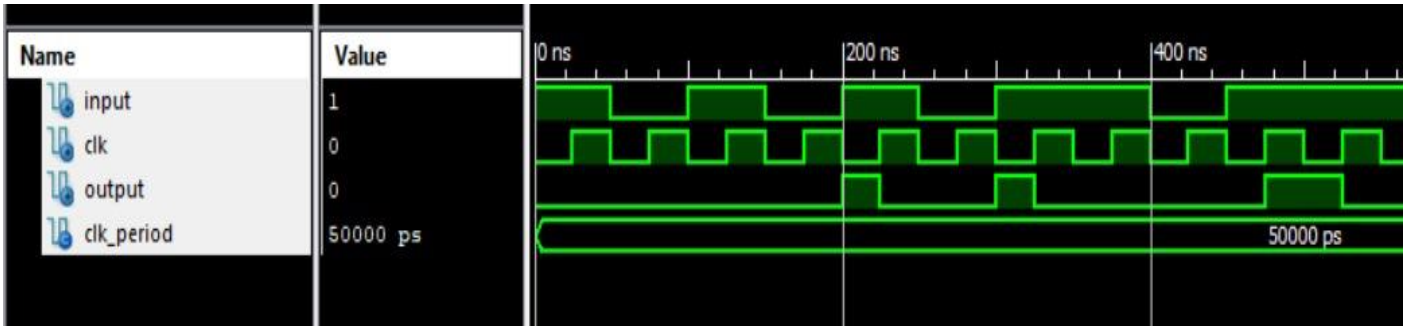
شکل**22**: ساختار Both Detector برای رشته های "0110" و "0101"

Test Bench:

```
BEGIN

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

   -- Clock process definitions
   clk_process :process
   begin
      clk <= '0';
      wait for clk_period/2;
      clk <= '1';
      wait for clk_period/2;
   end process;
 input<='1','0' after 50 ns,'1' after 100 ns,'0' after 150 ns,'1' after 200 ns,'0' after 250 ns,'1' after 300 ns,'0' after 400 ns,'1' after 450


   -- Stimulus process

END;
```

شکل**23**: تست بنچ Both Detector

Result of Simulation in isim :



شکل :24 نتایج شبیه سازی

نتیجه گیری:

در این آزمایش کار با مدار های ترتیبی و ترکیبی و DFF و TFF را مورد بررسی قرار دادیم که هر کدام را به طور کامل شرح دادیم.