"به نام یزدان پاک"

گزارش کار آزمایش ششم

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

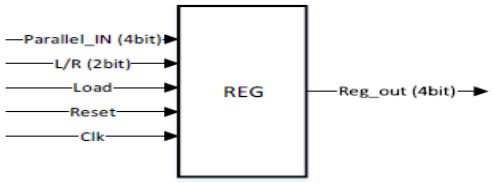
شرح آزمایش:

یکی از بحث‌های اصلی در مدارهای دیجیتال، بخش زمان‌بندی است. زمان‌بندی نامناسب مدارات می‌تواند مشکلات زیادی را در هنگام شبیه‌سازی و پیاده‌سازی به وجود آورد. در این آزمایش به چگونگی ایجاد تاخیر در مدارات دیجیتال پرداخته می‌شود. برای نیل به این منظور از ثبات‌های دارای قابلیت انتقال (منطقی و ریاضی) استفاده می‌کنیم.

الف) پیاده‌سازی ثبات با قابلیت انتقال (Shift):

یک ثبات چهار بیتی با قابلیت‌های زیر طراحی نمایید. ساختار بلوک ثبات در شکل ۱ نشان داده شده است. قابلیت‌های این بلوک به شرح زیر می‌باشد: (ثبات حساس به لبه بالارونده کلاک است.)

- انتقال منطقی (چپ/ راست)
- انتقال ریاضی (چپ / راست)



شکل ۱- ساختار ورودی/خروجی‌های بلوک ثبات

در بلوک ثبات فوق درگاه‌های ورودی خروجی به شرح زیر است:

- ورودی LR دوبیتی است که برای انتخاب بین حالات شیفت است. (LR="00" ثبات مقدار قبلی خود را حفظ می‌کند. (انتقال به چپ ریاضی و منطقی یک حالت فرض شوند.)

شکل 1: شرح آزمایش

- ورودی Load تک بیتی است که در صورت یک بودن، ورودی چهاربیتی Parallel_IN همزمان با لبه بالارونده کلاک در ثبات بارگذاری می‌شود. و در صورت صفر بودن ثبات مقدار قبلی خود را نگه می‌دارد.

ب) چگونگی ایجاد مقدار تاخیر مشخص برای نمایش خروجی روی برد:

برای ایجاد تاخیر و قابل رویت شدن تغییرات خروجی‌ها بر روی برد، از شمارنده برای تنظیم و کاهش فرکانس کلاک روی برد استفاده می‌شود. فرکانس کاری برد موجود در آزمایشگاه ۴۰ مگاهرتز می‌باشد، بنابراین برای دستیابی به تاخیر ۱ ثانیه، نیاز به شمارنده‌ای است که به میزان $۴\times۱۰^{۷}-۱$ بار بشمارد تا تاخیری به میزان ۱ ثانیه ایجاد گردد.

در این بخش از آزمایش شمارنده‌ای طراحی و به کد بخش الف اضافه کنید تا بتوان خروجی‌های شیفت رجیستر را با تاخیرهای مناسب بر روی برد مشاهده نمود.

ج) نمایش خروجی بر روی 7-Seg:

چگونگی اتصال پایه‌ها به منظور نمایش بر روی 7-seg به صورت زیر می‌باشد. (کلاک برد به پایه 184 متصل شده است.)

```
#PlanAhead Generated physical constraints

NET "SEG_DATA[0]" LOC = P10;
NET "SEG_DATA[1]" LOC = P7;
NET "SEG_DATA[2]" LOC = P11;
NET "SEG_DATA[3]" LOC = P5;
NET "SEG_DATA[4]" LOC = P4;
NET "SEG_DATA[5]" LOC = P12;
NET "SEG_DATA[6]" LOC = P9;
NET "SEG_DATA[7]" LOC = P3;

NET "SEG_SEL[0]" LOC = P15;
NET "SEG_SEL[1]" LOC = P20;
NET "SEG_SEL[2]" LOC = P19;
NET "SEG_SEL[3]" LOC = P18;
NET "SEG_SEL[4]" LOC = P16;
```

شکل 2: ادامه شرح آزمایش

خروجی مورد انتظار آزمایش:

خروجی‌های مورد انتظار آزمایش:
هریک از بخش‌های این آزمایش در قالب‌های زیر به ترتیب تحویل مدرس آزمایشگاه گردد:
- پیاده‌سازی مدارها با استفاده از زبان‌های توصیف سخت افزار VHDL
- شبیه‌سازی مدارهای توصیف شده و تهیه Testbench برای آن و نشان دادن درستی عملکرد آن‌ها.
- سنتز و پیاده‌سازی مدارها بر روی بورد FPGA و اثبات درستی عملکرد مدارها.

شکل 3: خروجی مورد انتظار آزمایش

توضیح آزمایش :

ثبات با قابلیت انتقال (shift_register):

با استفاده از ساختار ثبات و ساختار process و استفاده از شکل داده شده در صورت آزمایش و پیش گزارش یک ثبات را طراحی می کنیم.

Behavior Shift Register :

```
entity shift_reg is
 port(
      parallel_in      : in std_logic_vector(3 downto 0);
      LR               : in std_logic_vector(1 downto 0);
      CLK, reset, load : in std_logic;
      reg_out          : out std_logic_vector(3 downto 0) );
end shift_reg;

architecture behavioral of shift_reg is

begin

 process (CLK, reset)
 begin
 if reset = '1' then
   reg_out <= "0000";
 elsif (CLK'event and CLK='1') then
   if load = '1' then
      case LR is
         when "00" =>
            reg_out <= parallel_in;
         when "01" => -- (both) left shift
            reg_out(3 downto 1) <= parallel_in(2 downto 0);
            reg_out(0) <= '0';
         when "10" => -- arithmetic right shift
            reg_out(2 downto 0) <= parallel_in(3 downto 1);
            reg_out(3) <= parallel_in(3); -- if parallel_in(3) is 0, it's just a logical right shift!!
         when "11" => -- logical right shift
            reg_out(2 downto 0) <= parallel_in(3 downto 1);
            reg_out(3) <= '0';
         when others =>
      end case;
   end if;
 end if;
```

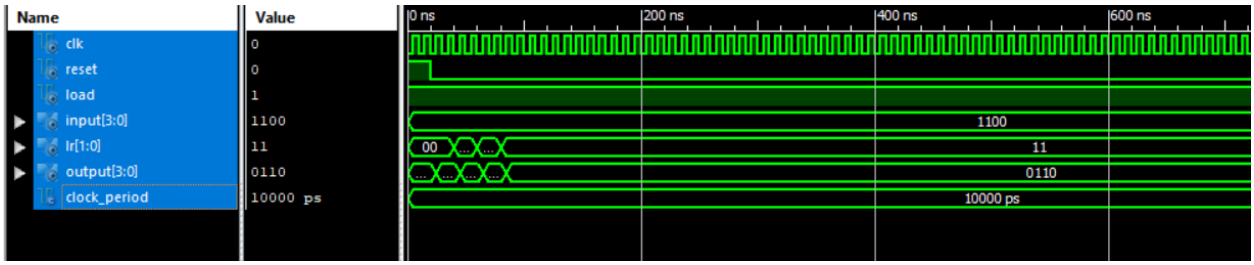
شکل4 : توصیف ثبات انتقال دهنده

Test Bench :

```
stim_proc: process
     begin
      input <= "1100";
      load <= '1';
      reset <= '1';
      LR <= "00";

      wait for 20 ns;

      input <= "1100";
      load <= '1';
      reset <= '0';
      LR <= "00";

      wait for 20 ns;
      input <= "1100";
      load <= '1';
      reset <= '0';
      LR <= "01";

      wait for 20 ns;
      input <= "1100";
      load <= '1';
      reset <= '0';
      LR <= "10";

      wait for 20 ns;

      input <= "1100";
      load <= '1';
      reset <= '0';
      LR <= "11";
```

شکل 5: تست بنچ ثبات انتقال دهنده

Result of Simulation in isim :



شکل6 : نتایج شبیه سازی

بخش دوم:

(counter) شمارنده برای ایجاد تاخیر:

در این بخش هدف ما توصیف یک شمارنده در زبان VHDL در سطح گیت می باشد . این طراحی با استفاده از دستورات زبان و مدار گفته شده در شرح آزمایش طراحی میکنیم.که این شمارنده از 1 تا 40 MHZ می شمارد تا تاخیری به این اندازه در هر شمارش ایجاد کند.

Behavior of Counter:

```
entity counter is
port (
      clk,reset: in std_logic;
      clock_out: out std_logic);
end counter;

architecture Behavioral of counter is

signal count: integer:=1;
signal tmp : std_logic := '0';

begin

process(clk,reset)
begin
if(reset='1') then
   count <= 1;
   tmp <= '0';
elsif(clk'event and clk='1') then
   count <= count+1;
if (count = 40000000) then
   tmp <= NOT tmp;
   count <= 1;
end if;
end if;
clock_out <= tmp;
end process;

end bhv;
```

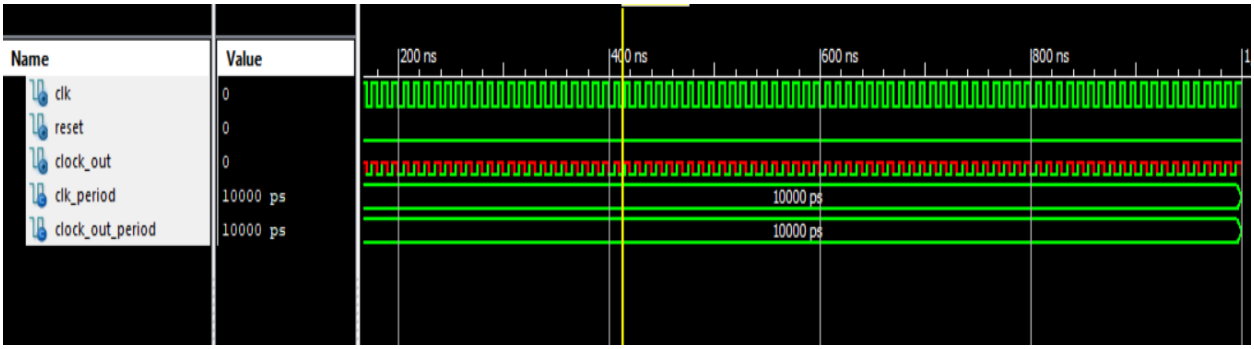
شکل 7:توصیف شمارنده تاخیر

Test Bench:

```
BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: counter PORT MAP (
          clk => clk,
          reset => reset,
          clock_out => clock_out
        );
   reset<='1','0' after 100 ns;
   -- Clock process definitions
   clk_process :process
   begin
      clk <= '0';
      wait for clk_period/2;
      clk <= '1';
      wait for clk_period/2;
   end process;

   clock_out_process :process
   begin
      clock_out <= '0';
      wait for clock_out_period/2;
      clock_out <= '1';
      wait for clock_out_period/2;
   end process;
```

شکل 8: تست شمارنده تاخیر

Result of simulation in isim:



شکل 9 نتایج شبیه سازی

بخش سوم:

: (seven segment)

در این بخش هدف ما توصیف seven segment در زبان VHDL در سطح گیت می باشد . این طراحی با استفاده از دستورات زبان و مدار طراحی شده در پیش گزارش انجام میدهیم.

Behavior of seven segment :

```
entity seven_segment is
port(
   input : in std_logic_vector(3 downto 0);
   a,b,c,d,e,f,g : out std_logic := '0'
);
end seven_segment;

architecture Behavioral of seven_segment is
begin

process (input)
begin

if(input < "1010") then

   a <= input(3) or input(1) or (input(2)and input(0)) or (not(input(2)) and (not(input(0))));
   b <= not(input(2)) or (not(input(1)) and not(input(0))) or (input(1)and input(0));
   c <= input(2) or not(input(1)) or input(0);
   d <= (not(input(2)) and (not(input(0)))) or (input(1) and not(input(0))) or (input(2) and not(input(1)) and input(0));
   e <= (not(input(2)) and (not(input(0)))) or (input(1) and not(input(0)));
   f <= input(3) or (not(input(1)) and not(input(0))) or (input(2) and not(input(1))) or (input(2) and (not(input(0))));
   g <= input(3) or (input(2) and not(input(1))) or (not(input(2)) and input(1)) or (input(1) and not(input(0)));


end if;
end process;
```

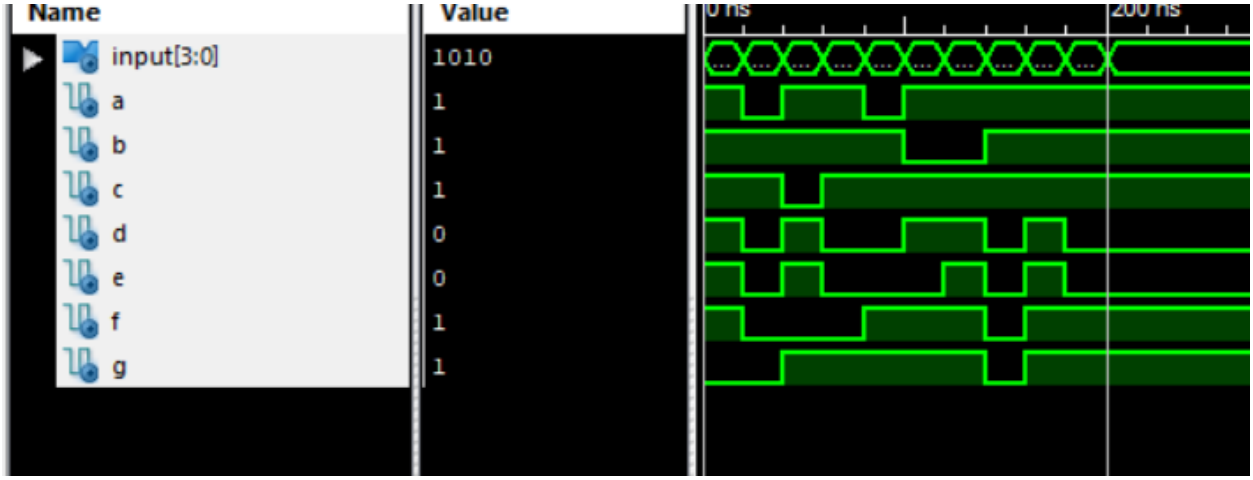
شکل 10: توصیف seven segment

Test Bench:

```
begin

    input <= "0000";
    wait for 20 ns;

    input <= "0001";
    wait for 20 ns;

    input <= "0010";
    wait for 20 ns;

    input <= "0011";
    wait for 20 ns;

    input <= "0100";
    wait for 20 ns;

    input <= "0101";
    wait for 20 ns;

    input <= "0110";
    wait for 20 ns;

    input <= "0111";
    wait for 20 ns;

    input <= "1000";
    wait for 20 ns;

    input <= "1001";
    wait for 20 ns;

    input <= "1010";
    wait for 20 ns;
```

شکل 11: تست بنچ

Result of simulation in isim:



شکل 12: نتایج شبیه سازی ***seven segment***

نتیجه گیری:

به طور کلی و در این آزمایش هدف آشنایی با نحوه عملکرد و پیاده سازی ثبات انتقال دهنده و و .و یادگیری بهتر زبان VHDL و کار با ساختار process & forو که با پیاده سازی ماژول های شیفت ریجستر این کار را انجام دادیم . و با ماژول seven segment نحوه نمایش ان را آموختیم.