به نام خدا

گزارش کار آزمایشگاه معماری کامپیوتر

آزمایش جلسه چهارم

عنوان آزمایش:

Advanced Digital Timer

نام استاد:

استاد علی جوادی

اعضای گروه:

غزل عربعلی – بهاره کاوسی نژاد

آزمایش: Advanced Digital Timer Implementation

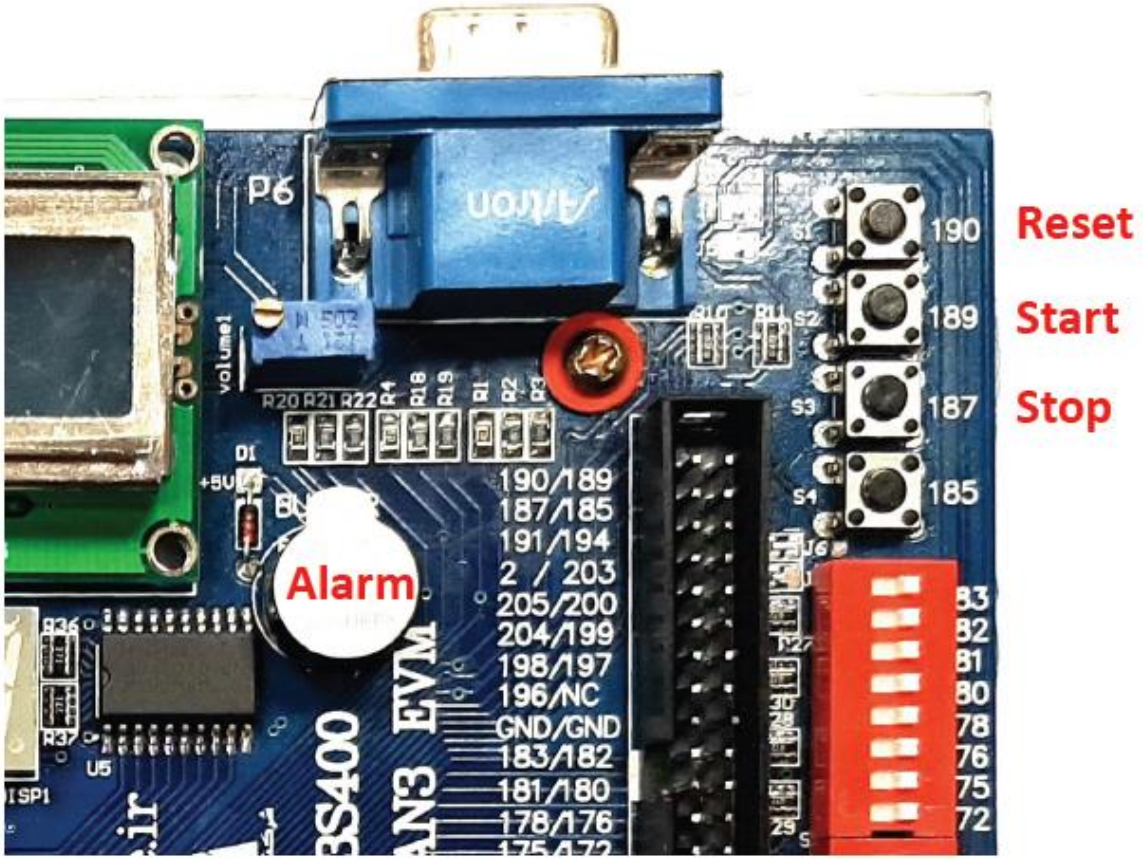
هدف آزمایش:

طراحی یک Digital Timer که دارای قابلیت های alarm، reset، start و stop است. همچنین ارقام را به صورت binary نمایش می دهد.

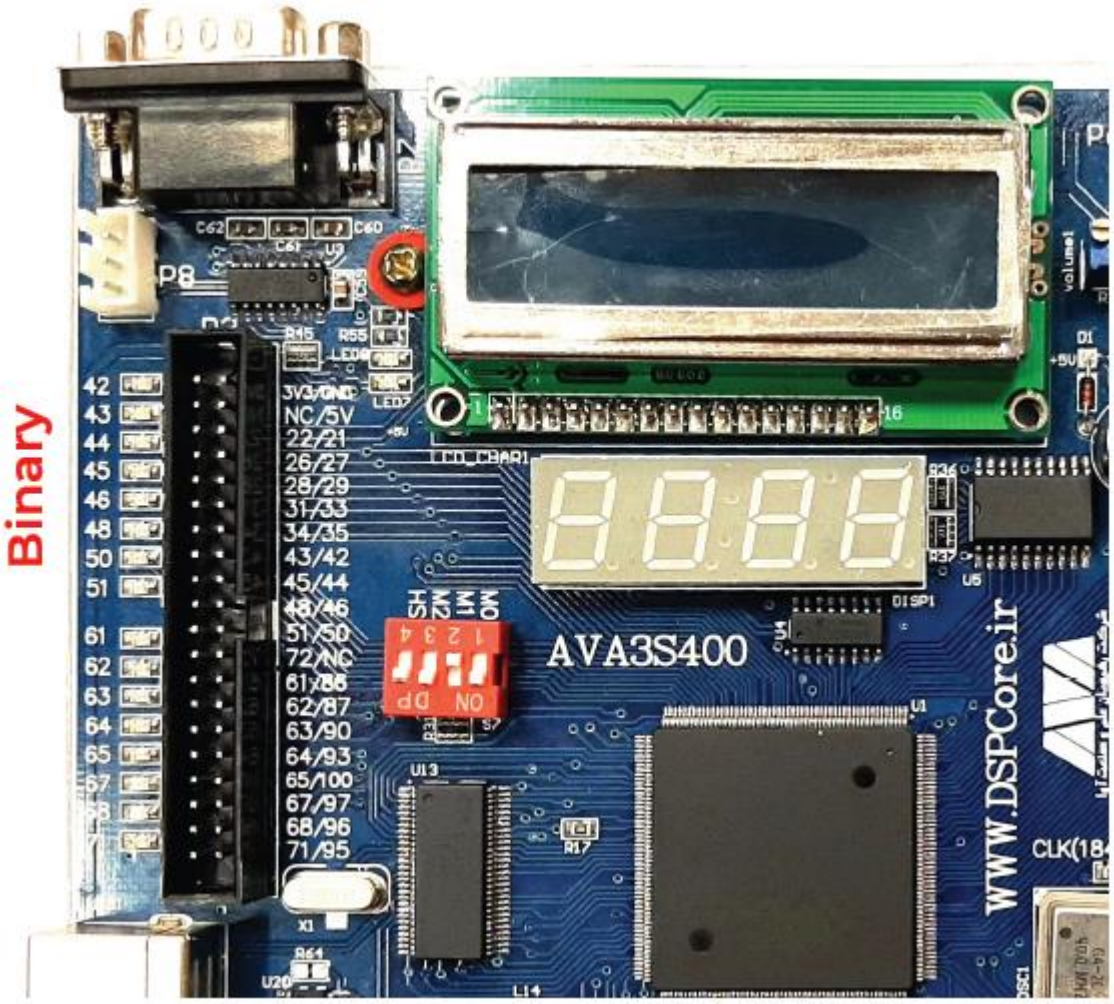
تئوری آزمایش:

در این آزمایش با استفاده از روش نمایش اعداد در آزمایش قبلی، یک تایمر دیجیتال می‌سازیم و آن را روی 7 segment نمایش می‌دهیم. این تایمر دارای ویژگی‌های زیر است:

- دکمه Reset ارقام تایمر را به 00:00 تغییر می‌دهد.
- دکمه Start شروع و ادامه کار تایمر را مشخص می‌کند.
- دکمه Stop تایمر را متوقف می‌کند.
- در زمان مشخص شده، Alarm به صدا در می‌آید.



این تایمر علاوه بر نمایش 7Segment، ارقام را به صورت Binary نیز نمایش می‌دهد.



روش و چگونگی انجام آزمایش:

ورودی و خروجی‌ها را مشخص می‌کنیم. همچنین سیگنال‌های مورد نیاز را تعیین می‌کنیم.

```
entity AdvancedSevenSegment is
    Port ( SEG_DATA : out  STD_LOGIC_VECTOR (7 downto 0);
           SEG_SEL : out   STD_LOGIC_VECTOR (4 downto 0);
           GCLK  : in  STD_LOGIC;
           START : in  STD_LOGIC;
           STOP  : in  STD_LOGIC;
           RESET : in  STD_LOGIC;
--         ALARM : out STD_LOGIC;
-- NET "ALARM" LOC = P13;
           OUTPUTBCD1 : out  STD_LOGIC_VECTOR (3 downto 0);
           OUTPUTBCD2 : out  STD_LOGIC_VECTOR (3 downto 0);
           OUTPUTBCD3 : out  STD_LOGIC_VECTOR (3 downto 0);
           OUTPUTBCD4 : out  STD_LOGIC_VECTOR (3 downto 0));
end AdvancedSevenSegment;

architecture Behavioral of AdvancedSevenSegment is
   shared variable STARTSTOPTOGGLE : STD_LOGIC ;
   signal CLOCK10MS,CLOCK1S : STD_LOGIC;
   signal SEGDATABCD1 , SEGDATABCD2,SEGDATABCD3 , SEGDATABCD4 , SEGDATABCD5 : STD_LOGIC_VECTOR ( 3 downto 0 ) ;
   signal SEGDATAREG1 , SEGDATAREG2 ,SEGDATAREG3,SEGDATAREG4 , SEGDATAREG5  : STD_LOGIC_VECTOR( 7 downto 0 ) ;
```

یک کلاک 10 ms تعریف می‌کنیم:

```
--------- CLOCK 10MS --------------
process(GCLK)
   variable COUNTERDIVISION10MS : integer range 0 to 100000 := 0 ;
begin
   if rising_edge(GCLK) then
      if COUNTERDIVISION10MS < 80000 then
          COUNTERDIVISION10MS := COUNTERDIVISION10MS + 1;
      else
         COUNTERDIVISION10MS := 0;
         CLOCK10MS <= not CLOCK10MS;
      end if;
   end if;

end process;
```

به کمک کلاک 10 ms یک کلاک 1s تعریف می‌کنیم:

```
--------- CLOCK 1S WITH CLOCK10MS --------------
process(CLOCK10MS)
   variable COUNTERDIVISION1S : integer range 0 to 50 := 0 ;
begin
   if rising_edge(CLOCK10MS) then
      if COUNTERDIVISION1S < 50  then
         COUNTERDIVISION1S := COUNTERDIVISION1S + 1;
      else
         COUNTERDIVISION1S := 0;
         CLOCK1S <= not CLOCK1S;
      end if;
   end if;
end process;
```

همانند آزمایش قبل به map کردن اعداد BCD به Segment 7 می‌پردازیم. هر عدد 4 بیتی نشان دهنده یک رقم BCD و هر عدد 8 بیتی نشانگر Segment 7 است.

```
with SEGDATABCD1 select
   SEGDATAREG1 <= "00111111" when "0000" ,
                  "00000110" when "0001" ,
                  "01011011" when "0010" ,
                  "01001111" when "0011" ,
                  "01100110" when "0100" ,
                  "01101101" when "0101" ,
                  "01111101" when "0110",
                  "00000111" when "0111",
                  "01111111" when "1000",
                  "01101111" when "1001",
                  "00000000" when others ;
with SEGDATABCD2 select
   SEGDATAREG2 <= "00111111" when "0000" ,
                  "00000110" when "0001" ,
                  "01011011" when "0010" ,
                  "01001111" when "0011",
                  "01100110" when "0100",
                  "01101101" when "0101" ,
                  "01111101" when "0110",
                  "00000111" when "0111",
                  "01111111" when "1000",
                  "01101111" when "1001",
                  "00000000" when others ;
```

```
with SEGDATABCD3 select
   SEGDATAREG3 <= "00111111" when "0000" ,
                  "00000110" when "0001" ,
                  "01011011" when "0010" ,
                  "01001111" when "0011",
                  "01100110" when "0100",
                  "01101101" when "0101" ,
                  "01111101" when "0110",
                  "00000111" when "0111",
                  "01111111" when "1000",
                  "01101111" when "1001",
                  "00000000" when others ;
with SEGDATABCD4 select
   SEGDATAREG4 <= "00111111" when "0000" ,
                  "00000110" when "0001" ,
                  "01011011" when "0010" ,
                  "01001111" when "0011",
                  "01100110" when "0100",
                  "01101101" when "0101" ,
                  "01111101" when "0110",
                  "00000111" when "0111",
                  "01111111" when "1000",
                  "01101111" when "1001",
                  "00000000" when others ;

OUTPUTBCD1 <= SEGDATABCD1;
OUTPUTBCD2 <= SEGDATABCD2;
OUTPUTBCD3 <= SEGDATABCD3;
OUTPUTBCD4 <= SEGDATABCD4;
```

از سیگنال STARTSTOPTOGGLE برای تعیین کردن وضعیت start و stop استفاده می‌کنیم. اگر دکمه start فشرده شود، مقدار start به 0 تغییر می‌کند و مقدار STARTSTOPTOGGLE نیز 0 مشخص می‌شود.

```
process(CLOCK1S)
begin
if(rising_edge(CLOCK1S)) then
   if(START = '0') then
      STARTSTOPTOGGLE := '0';
   end if;
   if(STOP = '0') then
      STARTSTOPTOGGLE := '1';
   end if;
end if;
end process;
```

در این قسمت ابتدا عملکرد دکمه reset را مشخص کرده و سپس اگر دکمه start فشرده شود، شمارش تایمر را آغاز می‌کنیم. تفاوت این بخش با آزمایش قبل در رقم دهگان ثانیه است که با رسیدن به 60 باید صفر شود.

```
process(CLOCK1S)
variable NUMBERCOUNTER : integer range 0 to 9999 := 0 ;
variable vsegdataBCD1 ,vsegdataBCD2 , vsegdataBCD3 , vsegdataBCD4 ,vsegdataBCD5 : std_logic_vector ( 3 downto 0 ) := "0000" ;
begin
   --STARTSTOPTOGGLE := '1';
   if(rising_edge(CLOCK1S)) then
      --SEGDATAREG5(6 downto 5) <= "11" xor SEGDATAREG5(6 downto 5);
      if(RESET = '0') then
         vsegdataBCD1 := "0000";
         vsegdataBCD2 := "0000";
         vsegdataBCD3 := "0000";
         vsegdataBCD4 := "0000";
         vsegdataBCD5 := "0000";
      else
         if(STARTSTOPTOGGLE = '0') then
            --if(NUMBERCOUNTER < 9999) then
               SEGDATAREG5 <= SEGDATAREG5 (7 downto 0) and "11011111" and "00110000";
               if(vsegdataBCD1 <"1001" ) then
                  vsegdataBCD1 :=(std_logic_vector(vsegdataBCD1) + 1);
                  --SEGDATAREG5 <= SEGDATAREG5(7 downto 2) & vsegdataBCD1(1) & vsegdataBCD1(0);
                  --SEGDATAREG5(7) <= vsegdataBCD1(0);
               else
                  vsegdataBCD1 := "0000";
                  if ((vsegdataBCD2 < "0101")) then
                     vsegdataBCD2 := (std_logic_vector(vsegdataBCD2) + 1);
                  else
                     vsegdataBCD2 := "0000";
                     if(vsegdataBCD3 < "1001") then
                        vsegdataBCD3 := (std_logic_vector(vsegdataBCD3) + 1);
                     else
                        vsegdataBCD3 := "0000";
                        if(vsegdataBCD4 < "1001") then
                           vsegdataBCD4 := (std_logic_vector(vsegdataBCD4) + 1);
                        else
                           vsegdataBCD4 := "0000" ;
                        end if ;

                     end if;
                  end if ;
               end if ;
            --end if ;
         else
         end if;
      end if;
            NUMBERCOUNTER := NUMBERCOUNTER +1 ;
            vsegdataBCD5 := not vsegdataBCD5 ;
      end if ;
         SEGDATABCD1 <= vsegdataBCD1;
         SEGDATABCD2 <= vsegdataBCD2;
         SEGDATABCD3 <= vsegdataBCD3;
         SEGDATABCD4 <= vsegdataBCD4;
         SEGDATABCD5 <= vsegdataBCD5 ;
end process;
```

در مرحله آخر نیز مانند آزمایش قبل روشن بودن هر کدام از ارقام نمایشگر 7segment را مشخص می‌کنیم.

```
process(CLOCK10MS)
variable RefreshSEG : integer range 0 to 4 :=0 ;
   begin
      if rising_edge(CLOCK10MS) then
         if RefreshSEG < 4 then
            RefreshSEG := RefreshSEG + 1;
         else RefreshSEG :=0;
      end if ;
      case RefreshSEG is
         when 0 =>
            SEG_SEL(4) <='0';
            SEG_SEL(0) <='1';
            SEG_DATA <= SEGDATAREG1;
         when 1 =>
            SEG_SEL(0) <='0';
            SEG_SEL(1) <='1';
            SEG_DATA <= SEGDATAREG2;
         when 2 =>
            SEG_SEL(1) <='0';
            SEG_SEL(2) <='1';
            SEG_DATA <= SEGDATAREG3;
         when 3 =>
            SEG_SEL(2) <='0';
            SEG_SEL(3) <='1';
            SEG_DATA <= SEGDATAREG4;
         when 4 =>
            SEG_SEL(3) <= '0';
            SEG_SEL(4) <= '1';
            SEG_DATA <= SEGDATAREG5;
      when others => null ;
   end case ;
   end if ;
end process;
end Behavioral;
```

بحث و نتیجه گیری:

در این آزمایش به ساخت یک تایمر دیجیتال پرداختیم که دارای قابلیت های start، stop، toggle و ... بود.