به نام خدا

گزارش کار آزمایشگاه معماری کامپیوتر

آزمایش جلسه نهم

عنوان آزمایش:

LED Light Control with Serial - PWM

نام استاد:

استاد علی جوادی

اعضای گروه:

غزل عربعلی – بهاره کاوسی نژاد

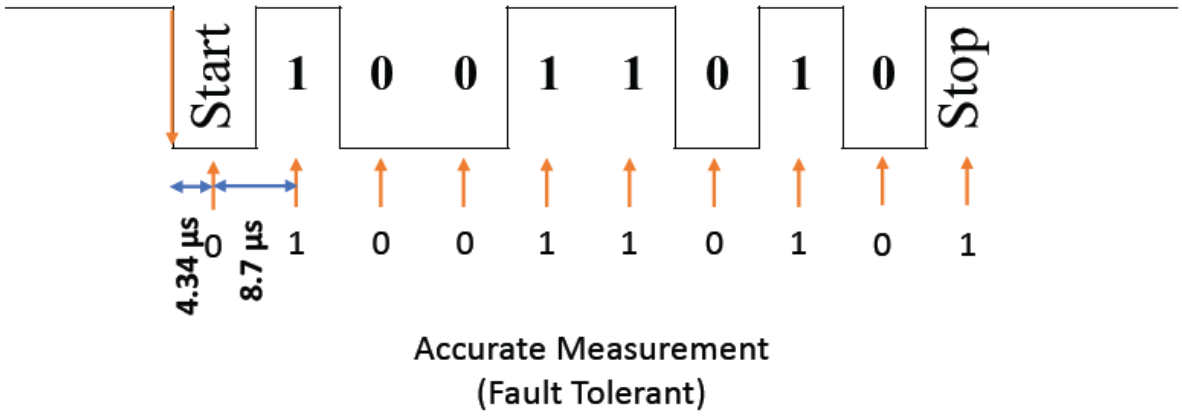
آزمایش: LED Light Control With Serial - PWM

هدف آزمایش: دریافت دیتا از پورت سریال و کنترل نور LED بر اساس دیتای ورودی

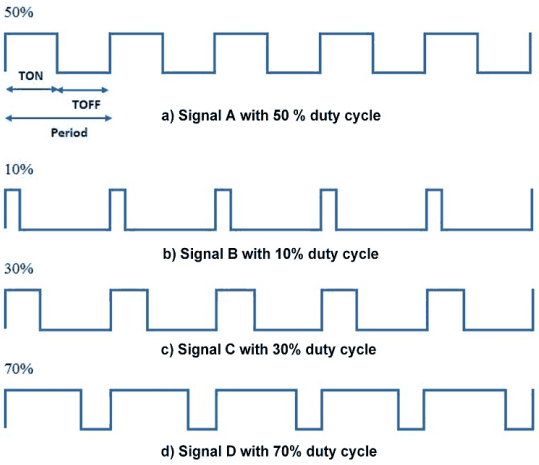
تئوری آزمایش:

در این آزمایش می‌خواهیم یک عدد هشت بیتی را در برنامه Docklight به پورت سریال بدهیم و روی LEDها نمایش دهیم؛ همچنین مقدار نور یک LED دیگر را بر اساس مقدار ورودی کم و زیاد کنیم.

روش کار پروتکل مانند آزمایش قبل به این صورت است که برای مشخص ساختن بیت اول ابتدا یک بیت 0 و برای مشخص ساختن پایان نیز 01 ارسال می‌کند. همچنین برای اطمینان از دریافت دیتای درست، پس از مشاهده اولین 0، نیم کلاک جلو می‌رویم.



برای تنظیم نور LED، مقدار duty cycle را تغییر می‌دهیم تا زمان روشن بودن LED کم و زیاد شود و این تغییرات توسط چشم انسان به شکل کم نور و پر نور شدن LED دیده می‌شوند.



روش و چگونگی انجام آزمایش:

در ابتدا ورودی‌ها و خروجی را مشخص می‌کنیم:

- GCLK: ورودی
- RX: ورودی برای دریافت دیتا توسط IDC Telecom Connector
- LED: خروجی

```
entity PWM_RXSerial is
    Port ( GCLK : in  STD_LOGIC;
           RX : in  STD_LOGIC;
           LED : out  STD_LOGIC
           );
           --OUTPUT : out  STD_LOGIC_VECTOR (7 downto 0));
end PWM_RXSerial;
```

چند سیگنال تعریف می‌کنیم:

- second_process_clock : کلاک process دوم که برابر با نصف کلاک آزمایش قبل است
- is_started: برای چک کردن شروع شدن دریافت دیتای 8 بیتی
- rx_temp: دیتای 8 بیتی

```
architecture Behavioral of PWM_RXSerial is
signal second_process_clock : std_logic := '0'; -- second process should start after setting this
signal is_started : std_logic := '0'; -- 8 bit digit should be read after start
signal rx_temp : std_logic_vector(7 downto 0); -- 8 digit to read
begin
```

تعریف کلاک جدید برای دریافت ورودی‌ها:

```
process(GCLK)
variable counter_clock : integer range 0 to 88 := 0;-- half a clock
variable duty_cycle : std_logic_vector(7 downto 0) := (others => '0');
begin
   if (rising_edge(GCLK)) then
      if (RX = '0') then -- should start reading 8 digit
         is_started <= '1';
      end if;
      --if (duty_cycle = "11111111") then -- each 255 times
      if (duty_cycle /= "11111111") then
         duty_cycle := duty_cycle + 1;
         if (duty_cycle >= rx_temp) then
            LED <= '0';
         else
            LED <= '1';
         end if;
      else
         duty_cycle := (others => '0');
      end if;
      -- half clock needed duty_cycle = "11111111"

      if (counter_clock >= 87) then
         counter_clock := 0;
         second_process_clock <= not second_process_clock;
      else
         counter_clock := counter_clock + 1;
      end if;

   end if;
end process;
```

دریافت دیتا و تنظیم نور LED:

```
-- second clock set
process(second_process_clock)
variable rx_digit_counter : integer range 0 to 10 := 0; -- 11 digit read (8 bit data - 1 bit start - 2 bit stop)
variable cycle : integer range 0 to 8 := 0;
variable is_finished : std_logic := '0';
begin
   if (rising_edge(second_process_clock)) then
      if (is_started = '1') then
         if (cycle = 1 or cycle = 3 or cycle = 5 or cycle = 7) then -- falling edge
            if(is_finished = '0') then
               if (rx_digit_counter = 0) then
                  rx_digit_counter := rx_digit_counter + 1;
               elsif (rx_digit_counter < 9) then
                  rx_temp(rx_digit_counter - 1) <= RX;
                  --OUTPUT(rx_digit_counter - 1) <= RX; -- other leds should show the hole number
                  rx_digit_counter := rx_digit_counter + 1;
               elsif (rx_digit_counter = 9) then
                  rx_digit_counter := 0;
                  is_finished := '1';
               end if;
            end if;
         end if;
         cycle := cycle + 1;
      end if;
   end if;

   if (is_finished = '1') then
      if ( RX = '0' ) then -- should be started again
         is_finished := '0';
      end if;
   end if;
end process;

end Behavioral;
```

در این قسمت سه variable تعریف می‌کنیم:

- rx_digit_counter: برای شمارش 11 بیت ورودی (8 بیت دیتا + 1 بیت شروع + 2 بیت پایان)
- cycle: شمارش 8 بیت دیتا
- is_finished: بیت پایانی دریافت شده است یا خیر

بحث و نتیجه گیری:

در انتها با استفاده از Docklight، ورودی ها را به مدار داده و نتیجه را روی LED مشاهده می‌کنیم.