به نام خدا

گزارش کار آزمایشگاه معماری کامپیوتر

آزمایش جلسه پنجم

عنوان آزمایش:

Basic Calculations and Sending via UART

نام استاد:

استاد علی جوادی

اعضای گروه:

غزل عربعلی – بهاره کاوسی نژاد

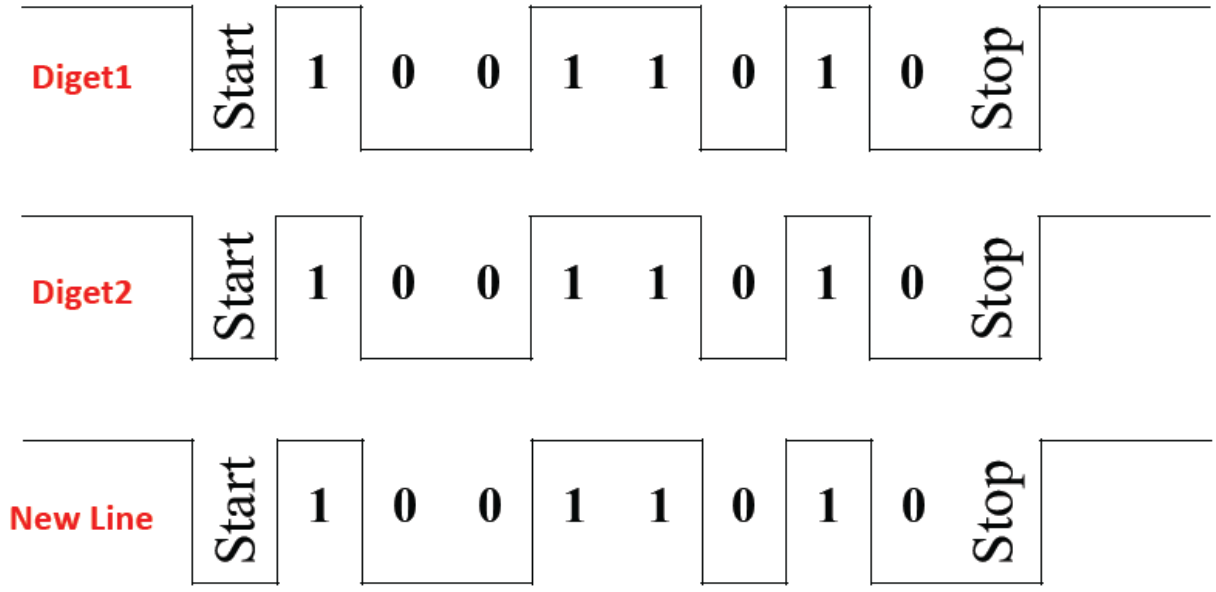
آزمایش: Basic Calculations and Sending via UART

هدف آزمایش: انجام عمل ضرب و نمایش بیت‌های نتیجه به صورت ASCII از طریق IDC Telecom Connector

تئوری آزمایش:

در این آزمایش می‌خواهیم دو عدد چهار بیتی که توسط Dip Switch ها مشخص می‌شوند را در هم ضرب کرده و سپس نتیجه را از طریق IDC Telecom Connector در برنامه Docklight به صورت ASCII نمایش دهیم.

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. در این آزمایش دو Diget و یک New Line برای خواناتر کردن نتایج داریم.



روش و چگونگی انجام آزمایش:

در ابتدا ورودی‌ها و خروجی را مشخص می‌کنیم:

- GCLK: ورودی
- TX: خروجی برای نمایش توسط IDC Telecom Connector

- Input1 و Input2: دو باس 4 بیتی ورودی که همان بیت‌های Dip Switchها هستند.

```
entity Multiplier is
    Port ( Input1 : in  STD_LOGIC_VECTOR (3 downto 0);
           Input2 : in  STD_LOGIC_VECTOR (3 downto 0);
           GCLK : in  STD_LOGIC;
           TX : out  STD_LOGIC);
end Multiplier;
```

shared variable ها را تعریف می‌کنیم:

```
architecture Behavioral of Multiplier is

signal CLOCK174 : STD_LOGIC := '0';
signal product : std_logic_vector(7 downto 0);
shared variable BCDOUTPUT : std_logic_vector(11 downto 0);
shared variable Digit1 : std_logic_vector(7 downto 0);
shared variable Digit2 : std_logic_vector(7 downto 0);
shared variable LineFeed : std_logic_vector(7 downto 0) := "00001010";
signal IntInput1 ,IntInput2 , IntResult : integer ;
shared variable binary_to_bcd :STD_LOGIC_VECTOR (7 downto 0);
shared variable LowNibble :std_logic_vector (7 downto 0);
shared variable HighNibble :std_logic_vector (7 downto 0);
shared variable firstdigit,seconddigit;

begin
```

در مرحله بعد CLOCK174 را می‌سازیم:

```
architecture Behavioral of RS232SerialCommunication is
   signal CLOCK174 : STD_LOGIC := '0';
begin

process(GCLK)
   variable CounterClock174 : integer range 0 to 100000 := 0;
   begin
      if(rising_edge(GCLK)) then
         if(CounterClock174 < 174) then
            CounterClock174 := CounterClock174  + 1;
         else
            CounterClock174  := 0;
            CLOCK174 <= not CLOCK174;
         end if;
      end if;
end process;
```

انجام عملیات ضرب:

```
process(Input1,Input2)
   begin
   IntInput1 <= to_integer(unsigned(Input1));
   IntInput2 <= to_integer(unsigned(Input2));
   IntResult <= IntInput1 * IntInput2;
   product <= STD_LOGIC_VECTOR(to_unsigned(IntResult,8));
end process;
```

تبدیل باینری به BCD:

```
--Binary to BCD converter
process(product)
begin
   BCDOUTPUT := (others => '0');
   binary_to_bcd := product(7 downto 0);
   for i in binary_to_bcd' range loop
      if BCDOUTPUT(3 downto 0) > "0100" then
         BCDOUTPUT(3 downto 0) := std_logic_vector(unsigned(BCDOUTPUT(3 downto 0))+ "0011");
      end if;
      if BCDOUTPUT(7 downto 4) > "0100" then
         BCDOUTPUT(7 downto 4):= std_logic_vector(unsigned(BCDOUTPUT(3 downto 0))+ "0011");
      end if;
      BCDOUTPUT := BCDOUTPUT(10 downto 0) & binary_to_bcd(7);
      binary_to_bcd := binary_to_bcd(6 downto 0) & '0';
   end loop;
        LowNibble(3 downto 0) := BCDOUTPUT(3 downto 0);
        HighNibble (3 downto 0) := BCDOUTPUT(7 downto 4);

        firstdigit := to_integer(to_unsigned(LowNibble)) + 48;
        seconddigit := to_integer(to_unsigned(LowNibble)) + 48;
        -- first 4 bit equal to zero :0
        Digit1 := std_logic_vector(to_integer(unsigned(LowNibble+48),8);
        Digit2 := std_logic_vector(to_integer(unsigned(HighNibble+48),8);
end process;
```

ارسال بیت ها:

```
process(CLOCK174)
variable ThreeDigitBits : integer range 0 to 7 := 0;
-- 1 start(=0) + 8 bit + 2 stop(=01) => 11 bits
variable BitCounter : integer range 0 to 32 := 1;
-- variable readyFlag ;
variable ConcatedDigits :std_logic_vector (23 downto 0);
-- For each 11 bits reading
variable ClockCycle : integer range 0 to 80000 := 0;
begin

   ConcatedDigits(7 downto 0) := Digit1;
   ConcatedDigits(15 downto 8) := Digit2;
   ConcatedDigits(23 downto 16) := LineFeed;

   if (rising_edge(CLOCK174)) then
    if (ClockCycle = 80000) then
        -- Resetting BitCounter and ThreeDigitBits after sending three serial numbers :/
        if (BitCounter = 32) then
            BitCounter := 0;
            ThreeDigitBits:= 0;
            ClockCycle := 0;
        else
            case BitCounter is
                when 0 | 11 | 22 =>
                    -- first bit = start
                    TX <= '0';
                when 9 | 20 | 31 =>
                    -- One before the last bit
                    TX <= '0';
                when 10 | 21 | 32 =>
                    -- Last bit
                    TX <= '1';
                when others =>
                    -- Data
                    TX <= ConcatedDigits(ThreeDigitBits);
                    -- Increment the data index
                            if (ThreeDigitBits < 24) then
                                ThreeDigitBits := ThreeDigitBits + 1;
                            -- Data finished
                            else
                                ThreeDigitBits := 0;
                            end if;
                 end case;

                 if (BitCounter < 33) then
                     BitCounter := BitCounter + 1;
                 else
                     BitCounter := 0;
                 end if;
            end if;
        else
            ClockCycle := ClockCycle + 1;
        end if;
    end if;
    end process;


    end Behavioral;
```

بحث و نتیجه گیری:

در انتها با استفاده از Docklight، نتیجه را مشاهده می‌کنیم.