

«به نام خدا»

استاد: دکتر میثم عبداللهی

نام: فاطمه زهرا بخشنده

شماره دانشجویی: 98522157

## گزارش تمرین چهارم:

فایل مربوط به هر سوال تمرین، در فایل زیپ قرار داده شده است.

### توضیحات هر سوال:

#### سوال اول:

در این سوال، مداری طراحی می کنیم که دنباله ای از بیت ها را جدا جدا دریافت کرده، و هر  $n$  بیت را در یک vector به نام `output_frame` خروجی می دهد.

```
entity SerialToParallel is
  generic (n: integer := 4);
  port
  (
    clk: in std_logic;
    input_bit: in std_logic;
    output_frame: out std_logic_vector(n - 1 downto 0);
    output_changing: out std_logic_vector(n - 1 downto 0)
  );
end SerialToParallel;
```

یک خروجی دیگر با عنوان `output_changing` قرار دادم که وضعیت `frame` را در هر لحظه نشان می دهد. اما خروجی اصلی که `output_frame` است تنها زمانی آپدیت می شود که  $n$  بار بیت جدید بگیریم، سپس در لبه بالا رونده کلاک بعدی، `output_frame` آپدیت شده و  $n$  بیت دریافتی اخیر را نشان می دهد.  $n$  یک پارامتر است که می توانیم به دلخواه مقدار آن را تغییر دهیم.

```

16
17 architecture Behavioral of SerialToParallel is
18
19     signal frame: std_logic_vector(n - 1 downto 0) := (others => '0');
20     shared variable count: integer := 0;
21     begin
22
23         process (clk, input_bit)
24         begin
25             if (clk'event and clk='1') then
26
27                 frame(n - 1 downto 1) <= frame(n - 2 downto 0);
28                 frame(0) <= input_bit;
29                 count := count + 1;
30
31                 if (count > n) then
32                     count := 1;
33                     output_frame <= frame;
34                 end if;
35
36             end if;
37
38         end process;
39
40         output_changing <= frame;
41         --output_frame <= frame;
42
43     end Behavioral;

```

عملکرد مدار به این صورت است که هر دفعه یک بیت به عنوان ورودی می گیرد، در صورت بالا بودن لبه کلاک، این بیت را به frame که یک vector با اندازه n است اضافه می کند.

اضافه شدن این بیت به frame به دلیل اینکه مقدار n هر مقداری می تواند باشد، با shift انجام می شود. یعنی هر دفعه که بیت جدید می آید یک شیفت به راست داریم و بیت جدید در خانه 0 frame قرار می گیرد.

یک count هم تعریف شده تا هر موقع n بیت پشت هم دریافت کردیم frame را در output\_frame قرار می دهد و نشان می دهد.

فایل testbench نیز برای این سوال موجود است. به دلیل استفاده از پارامتر n در سوال، در testbench یک CONSTANT parameter تعریف کرده و از GENERIC MAP استفاده می کنیم.

بیت هارا نیز به این صورت به مدار می دهیم:

```
-- Stimulus process
stim_proc: process
begin
    input_bit<='0';

    wait for clk_period; -- 6ns
    input_bit<='1';

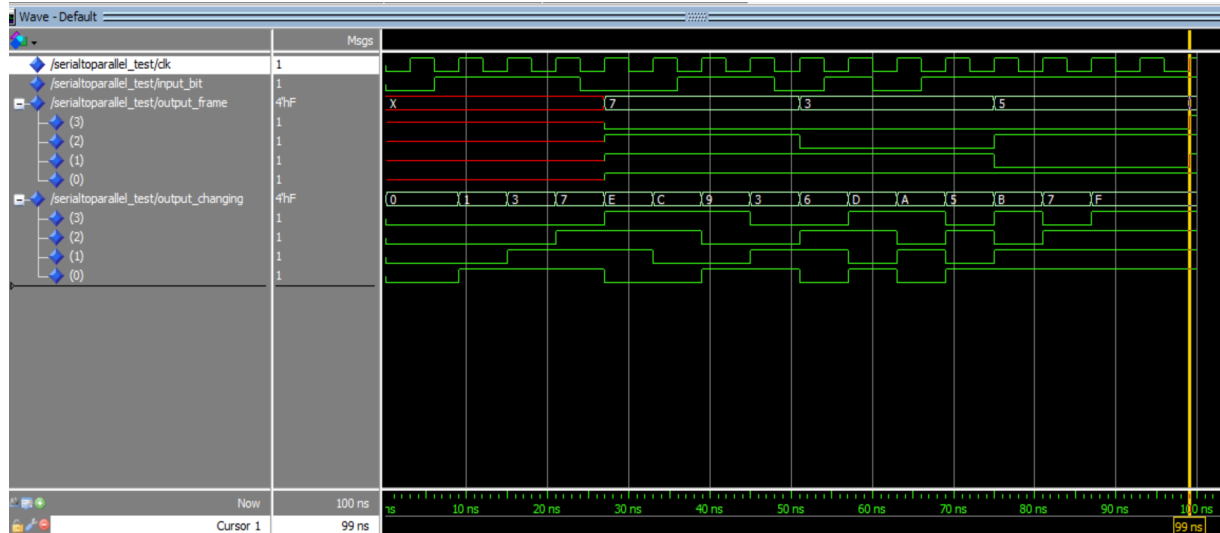
    wait for clk_period; -- 6ns
    input_bit<='1';

    wait for clk_period; -- 6ns
    input_bit<='1';

    wait for clk_period; -- 6ns
    input_bit<='0';

    wait for clk_period; -- 6ns
    input_bit<='0';
end
```

در نهایت نتیجه به صورت زیر است:



می بینیم که وکتور output\_frame پس از گرفتن هر 4 بیت آپدیت می شود و 4 بیت گرفته شده اخیر را نشان می دهد.

## سوال دوم:

در این سوال، یک وکتور 16 تایی را به عنوان ورودی می گیریم. که یکی از اعداد 0 تا 16 را به فرمت unary نمایش می دهد.

سپس در هنگام لبه بالا رونده کلاک، با استفاده از یک حلقه for، تعداد یک های ورودی را شمرده و در variable number می ریزیم. که این عدد همان عددی که برای خروجی می خواهیم است. برای نمایش آن به صورت vvector از std\_logic\_vector(number) استفاده می کنیم و آن را در output می ریزیم. مقدار output همان مقدار باینری ورودی ما است.

```
entity UnaryToBinary is
    port
    (
        clk: in std_logic;
        input: in std_logic_vector(15 downto 0);
        output: out std_logic_vector(4 downto 0)
    );
end UnaryToBinary;

architecture Behavioral of UnaryToBinary is

    SHARED variable number : unsigned(4 downto 0) := "00000";

begin
    process(clk, input)
    begin
        if (clk'event and clk='1') then

            number := "00000";
            for i in input'range loop    --check for all the bits.

                if(input(i) = '1') then --check if the bit is '1'
                    number := number + 1; --if its one, increment the number.
                end if;

            end loop;

            output <= std_logic_vector(number);

        end if;
    end process;
end architecture;
```

در فایل تست بنچ نیز ورودی هارا به این صورت به مدار می دهیم:

```

-- Stimulus process
stim_proc: process
begin
    wait for clk_period;
    input<=('0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','1');

    wait for clk_period;
    input<=('0','0','0','0','0','0','0','0','0','0','0','0','0','0','1','1');

    wait for clk_period;
    input<=('0','0','0','0','0','0','0','0','0','0','0','0','1','1','1','1');

    wait for clk_period;
    input<=('0','0','0','0','0','0','0','0','0','0','1','1','1','1','1','1');

    wait for clk_period;
    input<=('0','0','0','0','0','0','0','0','1','1','1','1','1','1','1','1');

    wait for clk_period;
    input<=('0','0','0','0','1','1','1','1','1','1','1','1','1','1','1','1');

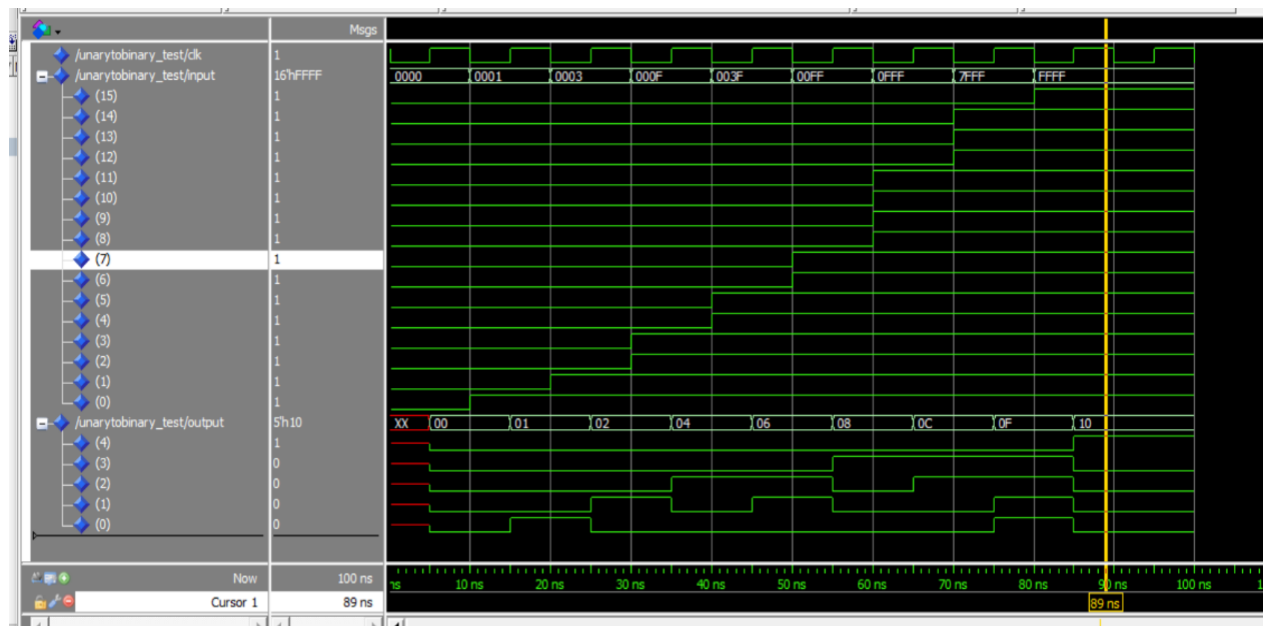
    wait for clk_period;
    input<=('0','1','1','1','1','1','1','1','1','1','1','1','1','1','1','1');

    wait for clk_period;
    input<=('1','1','1','1','1','1','1','1','1','1','1','1','1','1','1','1');

    wait;
end process;

```

در نهایت waveform به صورت زیر است:



## سوال سوم:

در این سوال، باید یک عدد سه رقمی را به صورت bcd در 12 بیت از ورودی بگیریم. و آن را به فرمت باینری نشان دهیم.

برای تبدیل bcd به باینری در هنگام لبه بالا رونده کلاک، به صورت زیر عمل می کنیم:

```
entity BCDToBinary is
  Port
  (
    clk: in std_logic;
    input: in std_logic_vector(11 downto 0); -- input BCD number
    output: out std_logic_vector(10 downto 0) -- output Binary number
  );
end BCDToBinary;

architecture Behavioral of BCDToBinary is
begin
  process (clk, input)
  begin
    if (clk'event and clk='1') then
      output <= (input(11 downto 8) * "1100100" -- digit_1 * 100
        + (input(7 downto 4) * "1010" -- + digit_2 * 10
        + input(3 downto 0); -- + digit_3 * 1
    end if;
  end process;
end Behavioral;
```

در فایل تست بنچ نیز ورودی هارا به این صورت به مدار می دهیم:

```
-- Stimulus process
stim_proc: process
begin
  wait for clk_period;
  input<= "011101100111";

  wait for clk_period;
  input<= "010101000011";

  wait for clk_period;
  input<= "011110000111";

  wait for clk_period;
  input<= "100110011001";

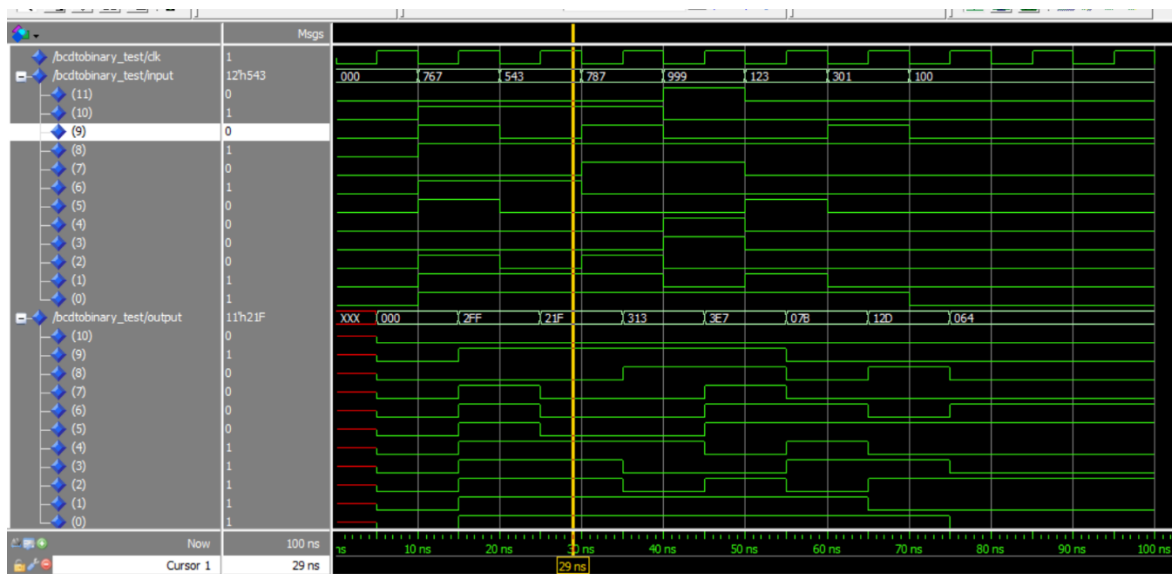
  wait for clk_period;
  input<= "000100100011";

  wait for clk_period;
  input<= "001100000001";

  wait for clk_period;
  input<= "000100000000";

  wait;
end process;
```

در نهایت خروجی waveform به صورت زیر است:



### سوال چهارم:

در این سوال، یک مدار Stopwatch با ورودی های کلاک و reset و start و سه خروجی (سگمنت ها) طراحی می کنیم.

```

4
5  entity Stopwatch is
6      generic (frequency: integer := 1);
7      PORT (
8          clk: in std_logic;
9          reset: in std_logic;
10         start: in std_logic;
11         segment_1: out std_logic_vector(7 downto 0);
12         segment_2: out std_logic_vector(7 downto 0);
13         segment_3: out std_logic_vector(7 downto 0)
14     );
15 end Stopwatch;
16

```

پارامتر generic فرکانس را برابر یک در نظر می گیریم. از یک شمارشگر count استفاده کرده و در هر کلاک آن را زیاد می کنیم.

اگر سیگنال reset فعال باشد، مقدار هر سه seconds\_1 و seconds\_2 و minute را صفر می کنیم.

در غیر این صورت هنگام لبه بالا رونده کلاک، تغییرات لازم را روی این سه سیگنال، بسته به مقدار قبلی آن ها، اعمال می کنیم.

```
--reset signal

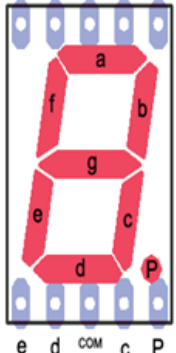
if (reset = '1') then
    seconds_1 := 0;
    seconds_2 := 0;
    minute := 0;

elsif (clk'event and clk = '1') then
    if (start = '1' and (seconds_1/=9 OR seconds_2/=5 OR minute/=9)) then
        seconds_1 := seconds_1 + 1;

        if (seconds_1 = 10) then --yekan
            seconds_1 := 0;
            seconds_2 := seconds_2 + 1;

            if (seconds_2 = 6) then --60 seconds
                seconds_2 := 0;
                minute := minute + 1;
            end if;
        end if;
    end if;
end if;
```

در process آخر، هر بار که مقدار این سه سیگنال تغییر کند، عددی که روی segment ها نشان می دهیم را طبق مقدار آن آپدیت می کنیم. بیت هایی که بر حسب هر عدد باید به یک segment داده شود با استفاده از case، طبق جدول زیر set می کنیم.

	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0	HEX value	
	dot	g	f	e	d	c	b	a		
DIGIT										
0	0	0	1	1	1	1	1	1	0x3F	
1	0	0	0	0	0	1	1	0	0x06	
2	0	1	0	1	1	1	1	1	0x5B	
3	0	1	0	0	1	1	1	1	0x4F	
4	0	1	1	0	0	1	1	0	0x66	
5	0	1	1	0	1	1	0	1	0x6D	
6	0	1	1	1	1	1	0	0	0x7C	
7	0	0	0	0	0	1	1	1	0x07	
8	0	1	1	1	1	1	1	1	0x7F	
9	0	1	1	0	1	1	1	1	0x6F	
	8	4	2	1	8	4	2	1		

[www.alselectro.com](http://www.alselectro.com)



در فایل تست بنچ نیز ورودی هارا به این صورت به مدار می دهیم:

```
-- Stimulus process

stim_proc: process
begin
    wait for clk_period; --10ns
    reset<='1';

    wait for clk_period; --10ns
    reset<='0';
    start<='0';

    wait for clk_period; --10ns
    start<='1';

    wait for clk_period*6; --60ns
    start<='0';
    reset<='1';

    wait;
end process;
```

در نهایت خروجی waveform به صورت زیر است:



## سوال پنجم:

در این سوال، ابتدا یک ماژول به عنوان پرسپترون با 3 جفت ورودی و 1 خروجی طراحی می کنیم. که خروجی را با فرمول ذکر شده محاسبه می کند:

```
entity Perceptron is
  Port
  (
    x0, w0: in integer;
    x1, w1: in integer;
    x2, w2: in integer;
    output: out integer
  );
end Perceptron;

architecture Behavioral of Perceptron is
begin
  output <= (x0 * w0) + (x1 * w1) + (x2 * w2);
end Behavioral;
```

در entity اصلی MLP، سه ورودی مدل به همراه 15 وزن را به عنوان input می گیریم. و با توجه به شکل سوال، دو خروجی داریم.

```
entity MLP is
  Port
  (
    x0, w00, w01, w02: in integer;
    x1, w10, w11, w12: in integer;
    x2, w20, w21, w22: in integer;
    w30, w31: in integer;
    w40, w41: in integer;
    w50, w51: in integer;
    out_1, out_2: out integer
  );
end MLP;

architecture Behavioral of MLP is
```

برای استفاده از ماژول پرسپترون در MLP کامپوننت آن را به صورت زیر تعریف می کنیم. سپس مطابق شکل سوال و طبق وزن های داده شده در ورودی، 5 نود پرسپترون را به هم می بندیم.

```

architecture Behavioral of MLP is
1
2   COMPONENT Perceptron
3   Port
4   (
5       x0, w0: in integer;
6       x1, w1: in integer;
7       x2, w2: in integer;
8       output: out integer
9   );
10  end COMPONENT;
11
12  signal p1_out, p2_out, p3_out : integer;
13  signal output1, output2 : integer;
14
15  begin
16
17      p1 : Perceptron
18      port map(
19          x0 => x0, w0 => w00,
20          x1 => x1, w1 => w10,
21          x2 => x2, w2 => w20,
22          output => p1_out);
23
24      p2 : Perceptron
25      port map(
26          x0, w01,
27          x1, w11,
28          x2, w21,
29          p2_out);
30
31  end Behavioral;

```

در فایل تست بنچ ابتدا وزن هارا مشخص کرده، و هر دفعه، خروجی را با ورودی های X مختلف می سنجیم:

```

-- Stimulus process
stim_proc: process
begin
    w00<= 1; w01<= 2; w02<= 3;
    w10<= 4; w11<= 5; w12<= 6;
    w20<= 7; w21<= 8; w22<= 9;
    w30<= 10; w31<= 11;
    w40<= 12; w41<= 13;
    w50<= 14; w51<= 15;

    wait for 10 ns;
    x0<= 3;
    x1<= 5;
    x2<= 8;

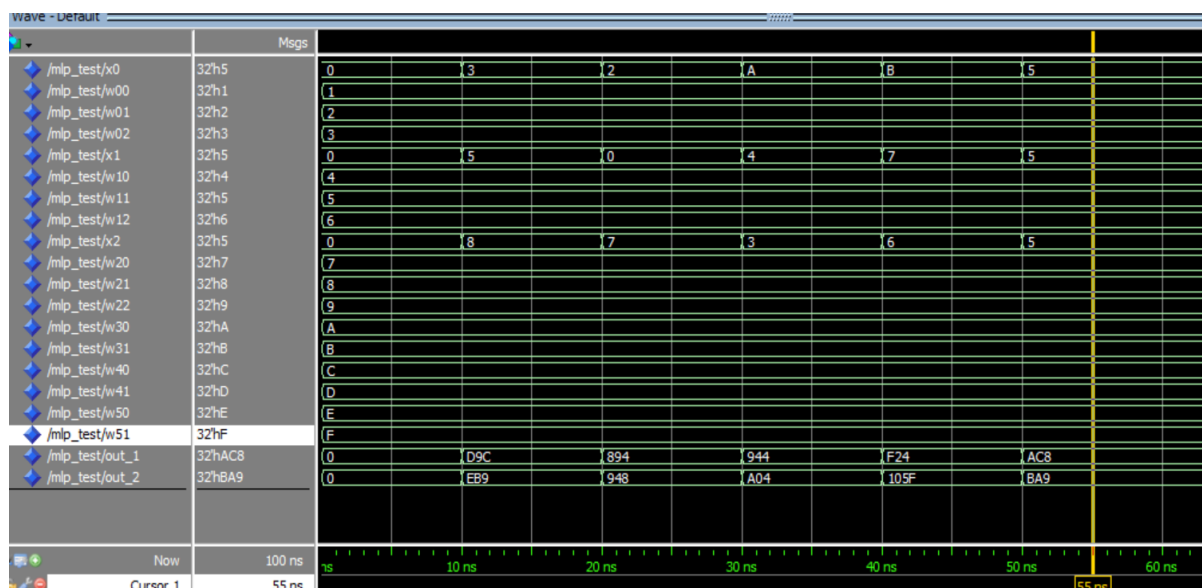
    wait for 10 ns;
    x0<= 2;
    x1<= 0;
    x2<= 7;

    wait for 10 ns;
    x0<= 10;
    x1<= 4;
    x2<= 3;

    wait for 10 ns;

```

در نهایت خروجی waveform به صورت زیر است:



می بینیم خروجی ها با وزن های داده شده، طبق هر ورودی به درستی محاسبه می شوند.