

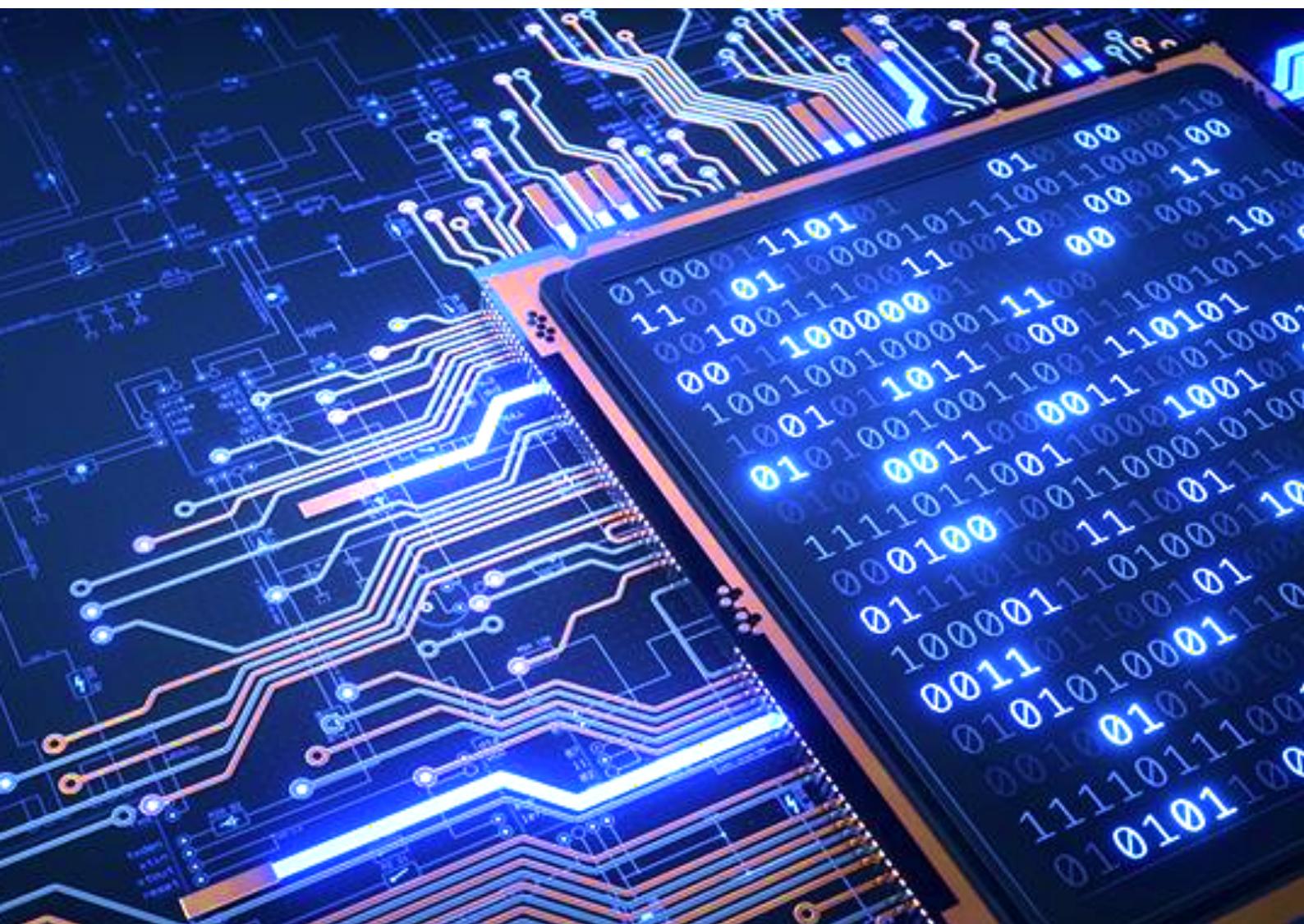


بهار ۱۴۰۳

تمرین کامپیوتری سوم

”مدارهای منطقی“

این تمرین با استفاده از ۱ روز گریس تحويل داده شد.

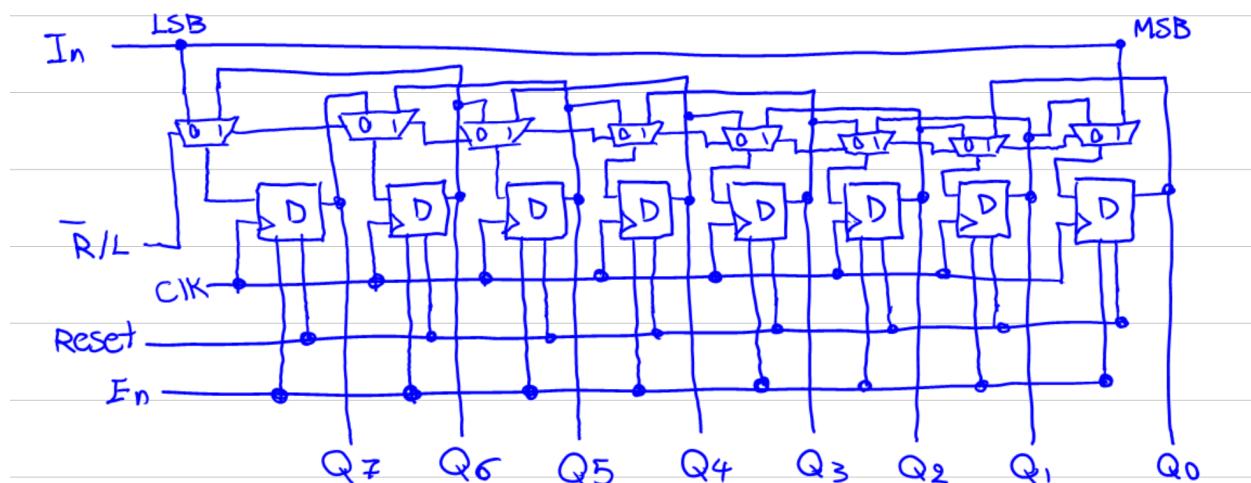


۸۱۰۱۰۱۴۵۹

مجید صادقی نژاد

بخش اول:

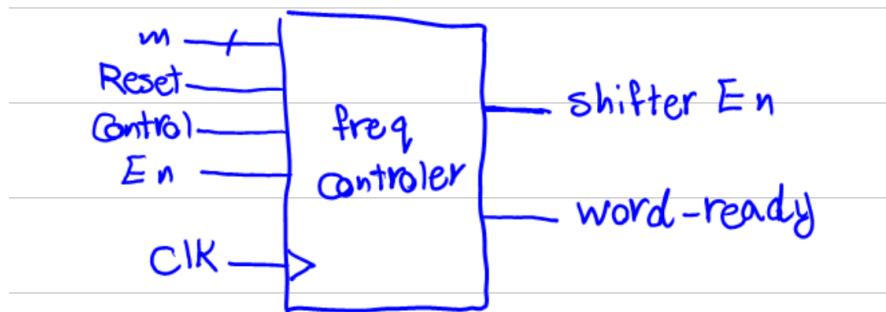
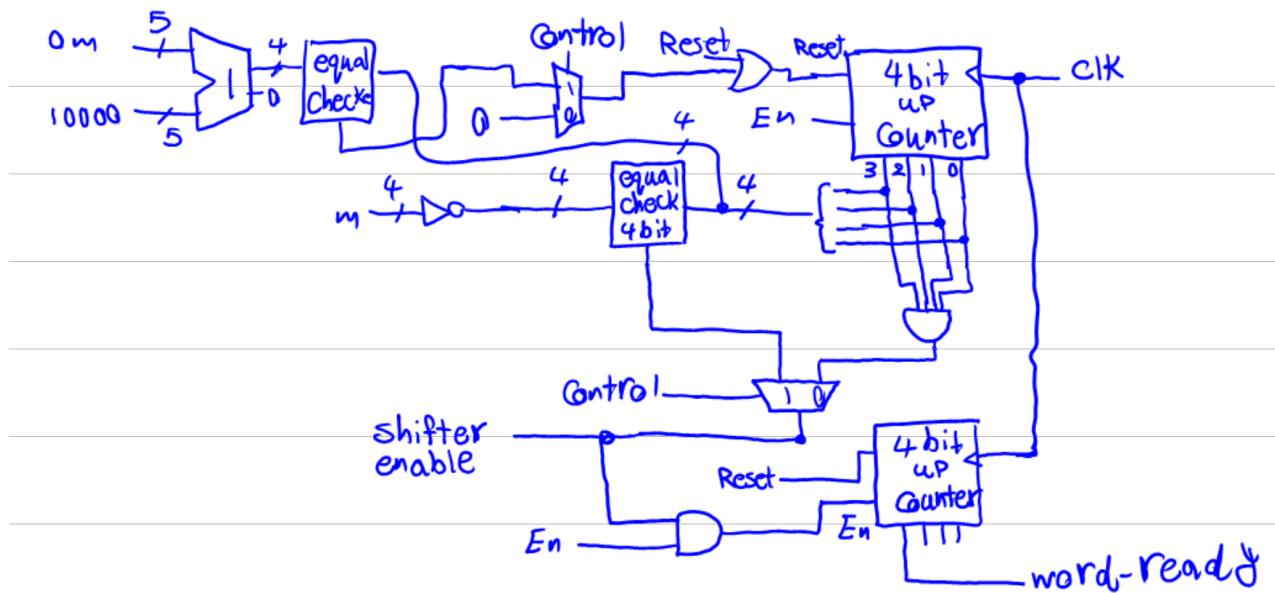
ابتدا با استفاده از فیلیپ فلاپ های نوع d و مالتی پلاکسر های دو به یک شیفت رجیستر طراحی می کنیم که قابل enable و شیفت به چپ یا راست داشته باشد ساختار آن در زیر آمده، که در واقع سیگنال R/L همان عکس بیت صفرم ورودی function است:



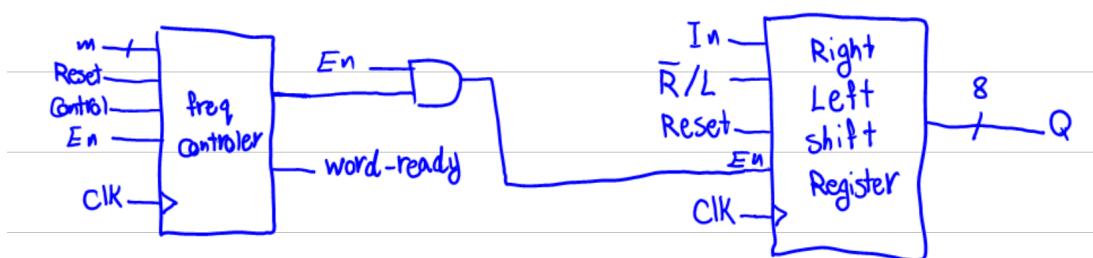
سیپس مداری برای کنترل تعداد سیکل های با استفاده از شمارنده ، جمع کننده ، مالتی پلکسر و بررسی کننده می برابری دو عدد به همراه چند گیت and , or می سازیم در زیر ابتدا مدار بررسی کننده می دو عدد و سیپس مدار کنترل گر تعداد سیکل ها آمده است لازم است ذکر است که منظور از سیگنال کنترل همان بیت اول function می باشد :



تمرین کامپیوتری سوم



در انتها با اتصال این دو المان و یک گیت and به یکدیگر مدار نهایی خواسته سوال را می سازیم که در آن خروجی Q برابر با همان word می باشد :



بخش دو:^۳

در این بخش هر دو حالت ساختاری و رفتاری پیدا سازی شده اند:

رفتاری:

در ابتداییک up counter به صورت رفتاری طراحی شد که با آمدن سیگنال clock ابتدا در صورتی که سیگنال فعال باشد عددش را صفر می کند و در غیر این صورت اگر enable فعال باشد یک عدد بالا می رود این شمارنده به صورت چهار بیتی طراحی شده است:

```
module nCounter(Reset,En,Clk,Q);
parameter n = 4 ;
output reg[n-1:0] Q;
input Reset,En,Clk ;

always @(posedge Reset or posedge Clk)
if (Reset)
Q <= 0;
else if (En)
Q <= Q + 1;

endmodule
```

سپس کد رفتاری شیفت رجستری نوشته شد که قابلیت شیفت به چپ و راست ، ریست و enable را با آمدن سیگنال clock دارد. در ابتدادر صورتی که ریست فعال باشد تمام بیت ها را صفر می کند در غیر این اگر سیگنال RL به سمت راست پاشد هر یک از بیت ها را به سمت راست شیفت داده و در غیر این صورت هر یک از بیت ها را به سمت چپ شیفت می دهد.

```
module LR_shifter(input In,RL,Reset,En,Clk, output reg[7:0] word);
integer k;

always @(posedge Reset or posedge Clk)
if (Reset)
word <= 0 ;
else if (En)
begin
if (!RL)
begin
for (k = 0; k < 7; k = k+1)
word[k] <= word[k+1];
word[7] <= In;
end
else
begin
for (k = 0; k < 7; k = k+1)
word[k+1] <= word[k];
word[0] <= In;
end
end
end

endmodule
```

تمرین کامپیوتری سوم

سپس مدار کنترل گر تعداد سیکل ها بر اساس دو شمارنده که در طراحی بخش اول آمده بود طراحی می گردد در این بخش تفريقي و عکس کردن اعداد با استفاده از مدار جمع کننده ی رفتاری انجام می شود و تمام مالتی پلکسرها و مدار های بررسی گر برابر دو عدد با شرط های if , else if پیاده شده اند بدین صورت که دو مالتی پلکسری که با سیگنال control کار می کنند در if اول مشخص شده اند بررسی کننده برابری عکس m و شمارنده با if دوم و بررسی کننده برابری تفريقي m و ۱۶ با if سوم مشخص شده است :

```
module controller(input control,Reset,En,Clk, input[3:0] m, output reg ready,shifterEn);

wire [3:0] Q1;
wire [3:0] Q2;
reg countReset ;

nCounter #(4) c1(Reset | countReset ,En, Clk, Q1);
nCounter #(4) c2(Reset, En & shifterEn, Clk, Q2);

assign ready = Q2[3];

always @(m,Q1,control)
begin
  if (!control)
    begin
      shifterEn = #Q1;
      countReset = 1'b0 ;
    end
  else
    begin
      if(Q1 == (4'd15 - m) )
        shifterEn = 1'bl;
      else
        shifterEn = 1'b0;
      if(Q1 == (5'd16 - m))
        countReset = 1'bl ;
      else
        countReset = 1'b0 ;
    end
  end
endmodule
```

در نهايیت مدار نهايی با استفاده از دو قطعه ی اصلی دقيقا همانند دياگرام كشیده شده پيدا سازی می شود:

```
module serial_receiver (input clk, input reset, input [1:0] func, input data_in, input enable, input [3:0] m, output [7:0] word, output word_ready);

wire shifterEn ;

controller controller0 (func[1],reset, enable, clk, m, word_ready, shifterEn);
LR_shifter shifter0 (data_in,!func[0],reset, enable & shifterEn, clk, word);

endmodule
```

ساختاري:

بخش ساختاري دقيقا بر اساس بلوک دياگرام ساخته شده پيدا سازی شده است و کد دو المان اصلی به همراه مدار نهايی ساختاري به شكل زير است :

تمرین کامپیوتری سوم

```
module shifterRL(input In,RL,Clk,Reset,En, output [7:0] Q);
    wire [7:0] filpIn;

    mux2 m7(In,Q[6],RL,filpIn[7]);
    filpflop f7(filpIn[7],Clk,Reset,En,Q[7]);

    genvar i;
    generate
        for(i = 7; 1 < i; i=i-1) begin: pack_stack
            mux2 mm (Q[i],Q[i-2],RL,filpIn[i-1]);
            filpflop ff (filpIn[i-1], Clk, Reset,En, Q[i-1]) ;
        end
    endgenerate

    mux2 m0(Q[1],In,RL,filpIn[0]);
    filpflop f0(filpIn[0],Clk,Reset,En,Q[0]);

endmodule

module controllerS(input [3:0] m , input Reset,control,En,Clk, output shifterEn,word_ready);

    wire [3:0] Q1;
    wire [3:0] Q2;
    wire [1:0] countReset;
    wire [4:0] num;
    wire [1:0] pshifter ;

    reducer rl (5'd16,{0,m},num);
    equalChecker el (num[3:0],Q1,countReset[0]);
    mux2 m0 (0,countReset[0],control,countReset[1]);

    equalChecker e2 (~m,Q1,pshifter);
    mux2 m1 (&Q1 , pshifter,control,shifterEn);

    nCounter #(4) c1(Reset | countReset[1] ,En, Clk, Q1);
    nCounter #(4) c2(Reset, En & shifterEn, Clk, Q2);

    assign word_ready = Q2[3];
endmodule

module Sserial_receiver (input clk, input reset, input [1:0] func, input data_in, input enable, input [3:0] m, output [7:0] word, output word_ready);
    wire shifterEn ;

    controllerS co (m, reset, func[1], enable, clk, shifterEn, word_ready);
    shifterRL sh (data_in, ~func[0], clk, reset, enable & shifterEn, word);
endmodule
```

بخش سوم:

در تست طراحی شده هر سیکل clock برابر با ۱۰ نانوثانیه است. و همچنین اعداد داده شده به ورودی به ترتیب برابر از راست به چیزی که در زیر آورده شده است می باشد. همچنین کد نوشته شده برای تست به صورت زیر است که در هر مرحله بسته به اطلاعات ارائه داده شده مقادیر آن تغییر کرده است.

تمرین کامپیوتری سوم

```
'timescale 1ns/1ps
module receiver_TB();
    logic clk, reset, enable;
    logic data_in;
    logic [1:0] func;
    logic [3:0];
    wire [7:0] word;
    wire word_ready;
    integer K;

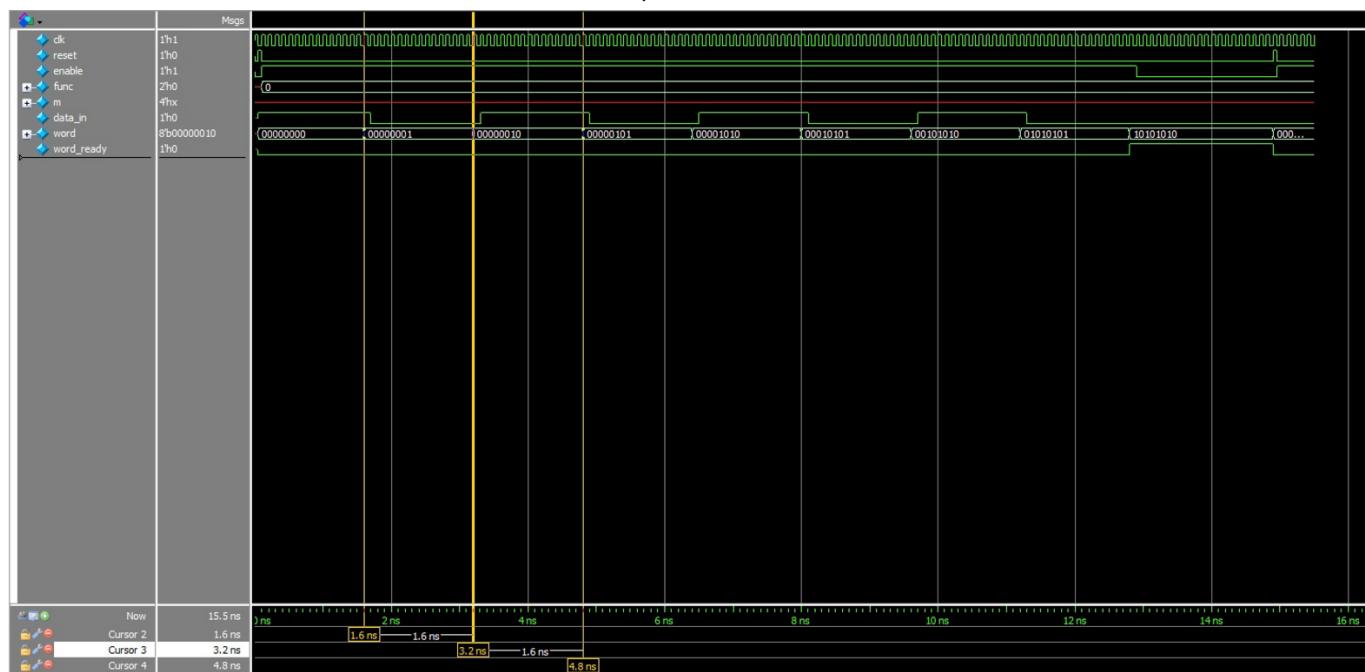
    serial_receiver UUT0 (clk, reset, func, data_in, enable, m, word, word_ready);

    initial begin
        $time = 1'b1;
        for (k = 0; k < 310; k = k+1)
            #0.05 clk = ~clk;
    end

    initial begin
        reset = 1'b0;
        enable = 1'b0;
        #0.05
        reset = 1'b1;
        data_in = 1'b1;
        #0.05
        reset = 1'b0;
        enable = 1'b1;
        func = 2'b11;
        m = 4'b0100;
        #1.2
        #1.2 data_in = 1'b0;
        #1.2 data_in = 1'b1;
        #1.2 data_in = 1'b0;
        #1.2 data_in = 1'b1;
        #1.2 data_in = 1'b0;
        #1.2 data_in = 1'b1;
        #1.2 data_in = 1'b0;
        #1.2
        enable = 1'b0;
        #2 reset = 1'b1;
        #0.05
        reset = 1'b0;
        enable = 1'b1;
    end
endmodule
```

تست اول function = ۰۰

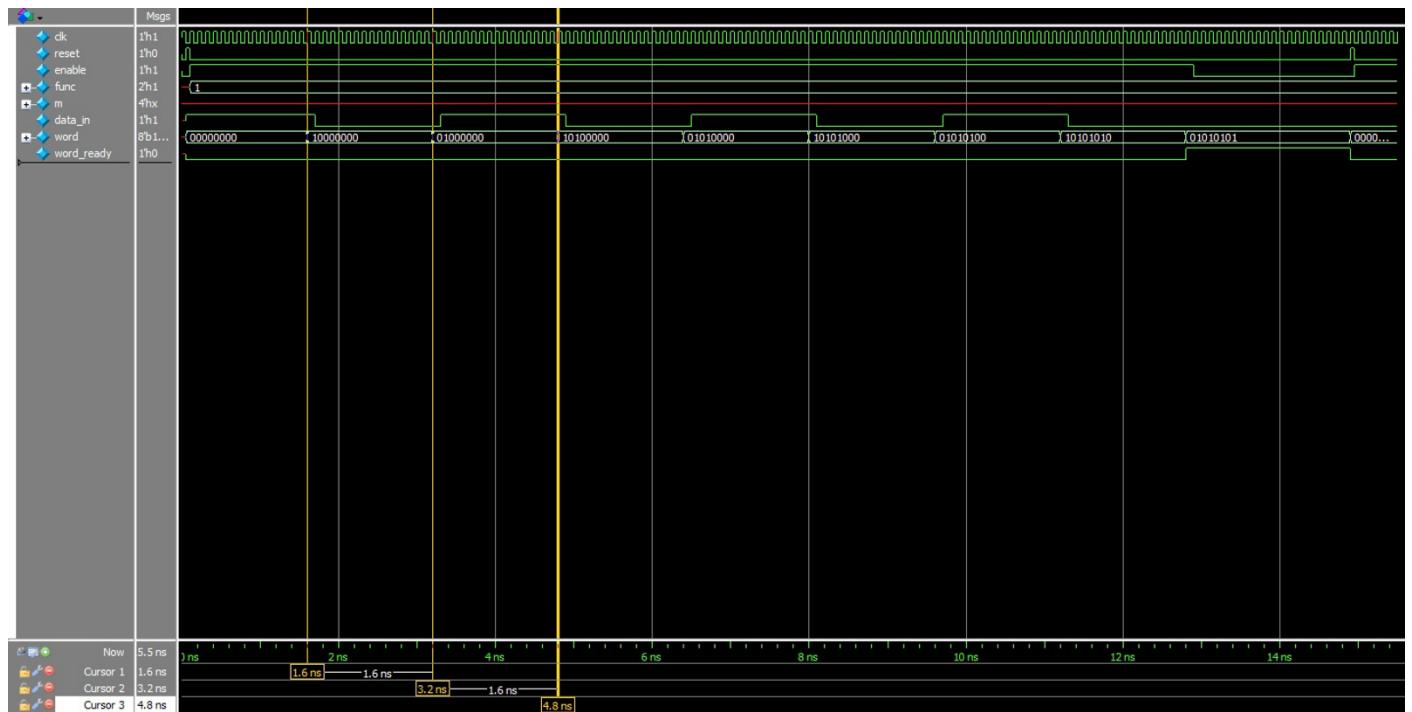
انتظار داریم بین هر دو دریافت ۱.۶ نانوثانیه فاصله باشد و پس از هشت دریافت، خروجی ما برابر ۰۰۰۰۰۰۰۰ باشد:



تمرین کامپیوتری سوم

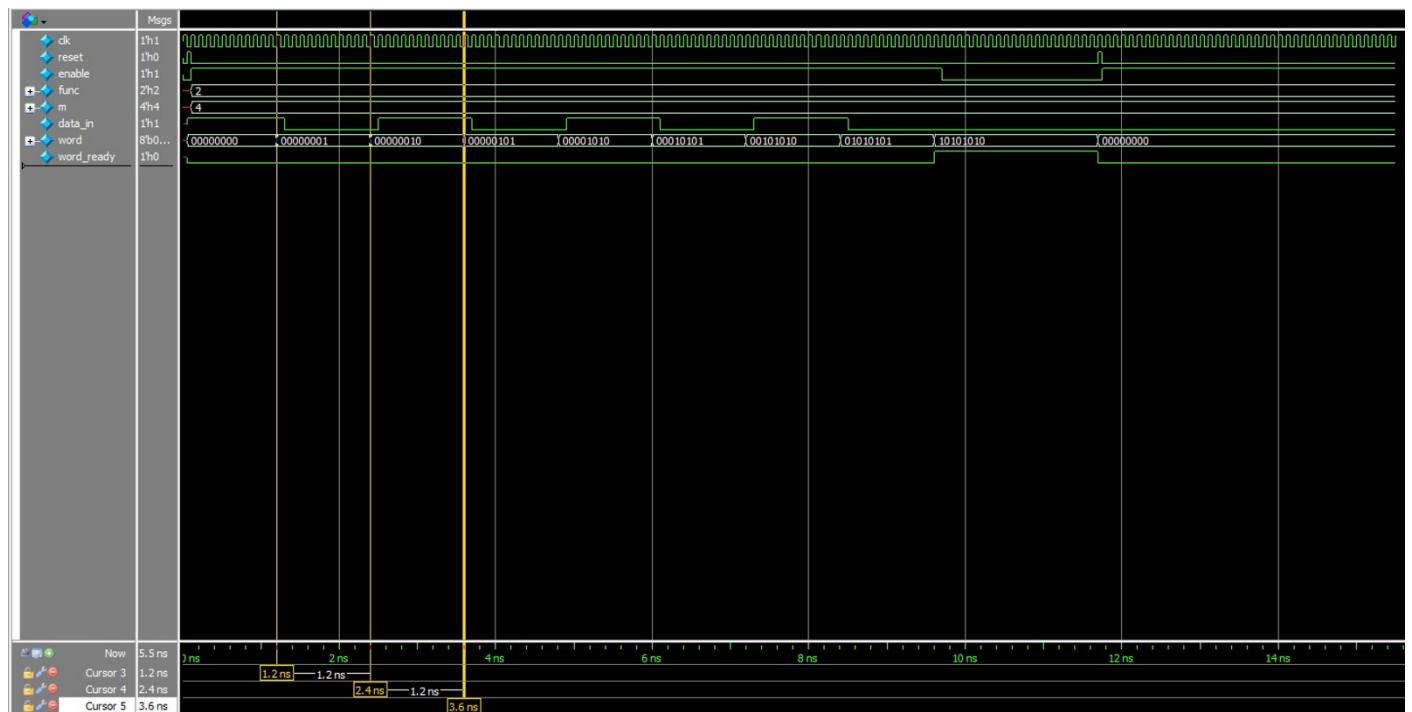
تست دوم = ۰۱

انتظار داریم بین هر دو دریافت ۱.۶ نانوثانیه فاصله باشد و پس از هشت دریافت، خروجی ما برابر ۱۰۱۰۱۰۱ باشد:



تست سوم = ۱۰ , m = ۴

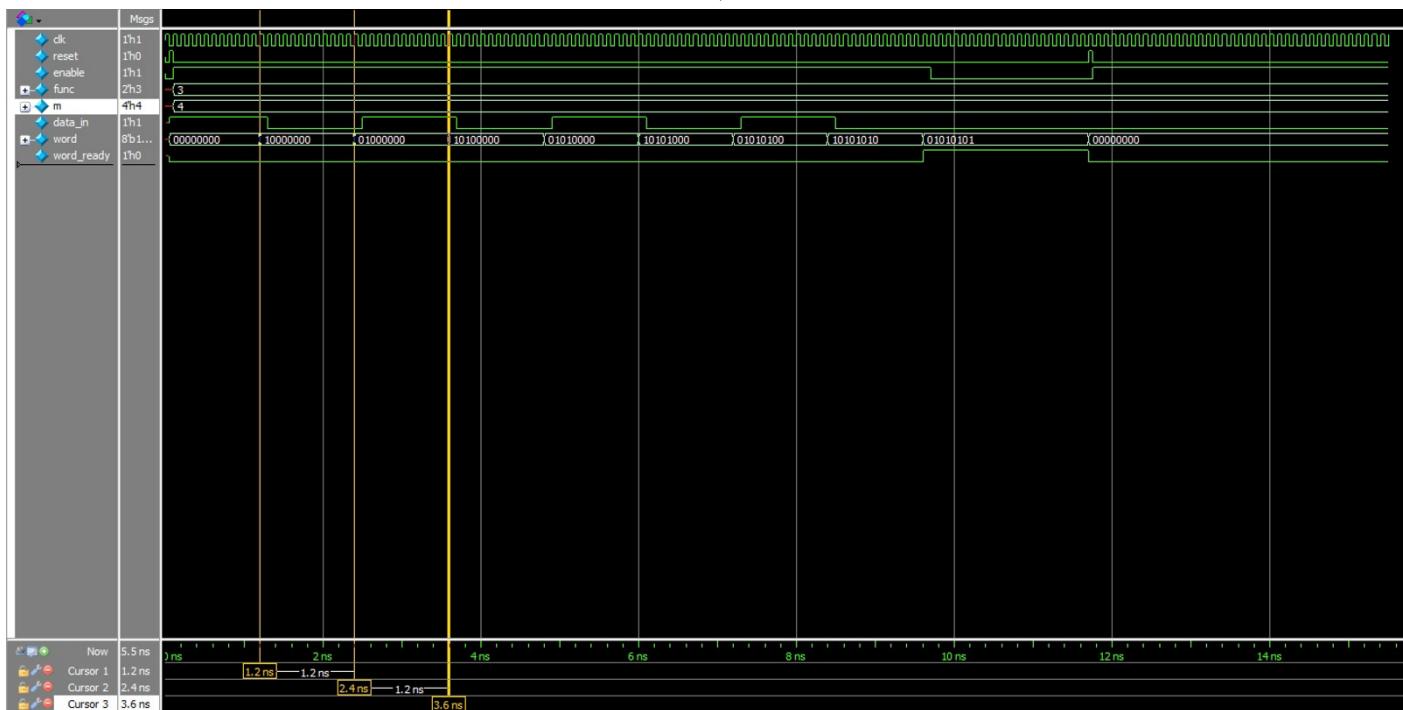
انتظار داریم بین هر دو دریافت ۱.۲ نانوثانیه فاصله باشد و پس از هشت دریافت، خروجی ما برابر ۱۰۱۰۱۰۰ باشد:



تمرین کامپیوتری سوم

تست چهارم $f = 11, m = 4$

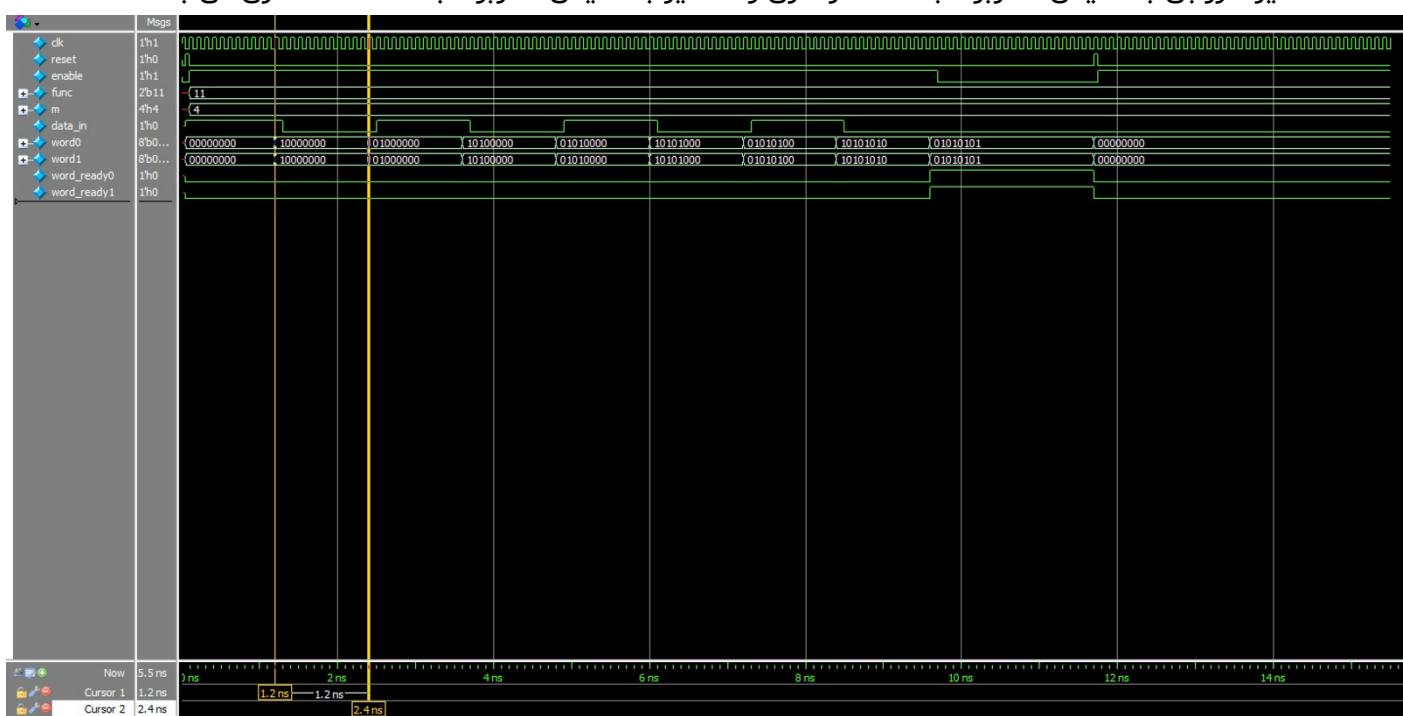
انتظار داریم بین هر دو دریافت ۱.۲ نانوثانیه فاصله باشد و پس از هشت دریافت، خروجی ما برابر ۱۱۰۱۱۰۱۰ باشد:



تست برابر حالت ساختاری و رفتاری $f = 11, m = 4$

انتظار داریم بین هر دو دریافت ۱.۲ نانوثانیه فاصله باشد و پس از هشت دریافت، خروجی ما برابر ۱۱۰۱۱۰۱۰ باشد و در

تست مقادیر خروجی با اندیس ۰ مربوط به حالت رفتاری و مقادیر با اندیس ۱ مربوط به حالت ساختاری می باشند:



همچین کد تست حالت ساختاری و رفتاری به شکل زیر است:

تمرین کامپیوتری سوم

```
`timescale 1ns/1ps
module receiver_TB();
    logic clk, reset, enable;
    logic data_in;
    logic [1:0] func;
    logic [3:0] m ;
    wire [7:0] word0;
    wire word_ready0;
    wire [7:0] word1;
    wire word_ready1;
    integer k;

    serial_receiver UUT0 (clk, reset, func, data_in, enable, m, word0, word_ready0);
    Sserial_receiver UUT1 (clk, reset, func, data_in, enable, m, word1, word_ready1);

initial begin
    clk = 1'b1;
    for (k = 0; k < 310; k = k+1)
        #0.05 clk = ~clk ;
end

initial begin
    reset = 1'b0 ;
    enable = 1'b0;
    #0.05
    reset = 1'b1 ;
    data_in = 1'b1 ;
    #0.05
    reset = 1'b0 ;
    enable = 1'b1 ;

    func = 2'b11;
    m = 4'b0100 ;
    #1.2 data_in = 1'b0;
    #1.2 data_in = 1'b1;
    #1.2 data_in = 1'b0;
    #1.2 data_in = 1'b1;
    #1.2 data_in = 1'b0;
    #1.2 data_in = 1'b1;
    #1.2 data_in = 1'b0;
    #1.2
    enable = 1'b0 ;
    #2 reset = 1'b1;
    #0.05
    reset = 1'b0;
    enable = 1'b1;

end
endmodule
```