به نام خدا

درس: طراحی سیستم های دیجیتال

نیم‌سال تحصیلی دوم 9900

مدرس: بهاروند

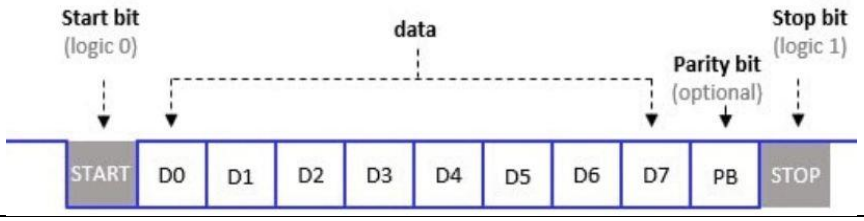
تمرین شماره: 5		موعد تحویل: 1400/3/10 ساعت ۲۳:55	
نام و نام خانوادگی:	سارا آذرنوش	شماره دانشجویی:	98170668

راه حل در همین فایل ارائه شود.

فایل به PDF تبدیل و در سایت بارگذاری شود.

عنوان تمرین

برای یک مدار **گیرنده** UART ابتدا ماشین حالت رسم نموده و سپس کد **رفتاری** متناظر آن را به صورت کامل طراحی کنید. در نظر داشته باشید داده‌های ورودی به UART دارای فرمت مطابق شکل زیر هستند. کلاک ورودی به مدار را uart_clock در نظر بگیرید و مدار را **پارامتری** طراحی کنید به نحوی که امکان کار با baudrate های مختلف در این گیرنده وجود داشته باشد. همچنین می‌خواهیم از پین ورودی این مدار گیرنده (با نام serial_data)، برای بخش فرستنده هم استفاده کنیم (یعنی خط ما bidirectional است و در هر لحظه فقط گیرنده یا فرستنده هستیم) و گذرگاه متصل به UART فقط یک سیم باشد؛ بدون نوشتن کد فرستنده، مکانیزمی در کد خود در نظر بگیرید که امکان این اتصال را فراهم کند. مدار بایستی دارای ری‌ست آسنکرون باشد. طراحی به گونه ای باشد که، بیت توازن (پاریتی) را هنگام دریافت بایتها چک کنید و در صورت اشتباه بودن، مدار کار خود را از نو شروع کند.



یک گیرنده uart طراحی میکنیم که سه ورودی کلاک (uart_clock)، ریست (rst) و یک ورودی برای گرفتن بیت ها به صورت تک بیتی است (با نام serial_data) دارد و خروجی عدد به صورت n بیتی و یک خروجی برای نشان دادن صحیح یا غلط بودن پریتی است.

Uart وسیله ای است که بیت ها را با هم ارسال و دریافت نمیکند بلکه به صورت تک بیتی ارسال و دریافت میکند . برای کمتر شدن خطا از پریتی استفاده میکنند تا تعداد یک ها را محاسبه کنند و در صورت اشتباه بودن مطلع کند و همچنین دو سیگنال شروع و پایان مقدار گرفتن و یک baudrate که در آن مقدار واحد زمانی ورودی بگیرد.

با گرفتن بیت 0 شروع به کار میکند و n بیت عدد را میگیرد و پس از آن یک بیت پریتی میگیرد که اگر این بیت با پریتی عدد برابر بود در خروجی par = 0 و اگر برابر نبود par = 1 می گذاریم (اکر تعداد یکها زوج بود پریتی عدد 0 و اگر تعداد فرد باشد 1 میشود که با xor کردن تمام بیت ها میتوان بدست آورد). پس از آن اگر بیت 1 بود می ایستد و اگر 0 بود مجددا شروع به کار میکند.

برای Baudrate نیز مقدار را به صورت پارامتری میگریم و میشماریم در صورتی که در مدار برابر با مقدار پارامتری بود بیت بعدی را دریافت میکنیم.

اگر در حالت اولیه بود و مقدار 0 وارد شد شروع به کار میکند در غیر این صورت در حالت استاپ است ورودی نمیگیرد بعد از وارد شدن 0 مقادیر اولیه داده شده، عدد شیفت میخورد و بیت ها تک به تک وارد ایندکس 0 عدد میشوند تا بیت ها تمام شوند و وارد حالت گرفتن و محاسبه پریتی شود پس از آن نیز دوباره به حالت اولیه باز میگردد تا 0 وارد شود و مجددا شروع به کارکند.

```
`timescale 1ns / 1ps
`include "states.v"

module receive #(parameter n = 8 , parameter b =2)(
   input uart_clock,
   input serial_data,
        input rst,

   output reg [n-1:0] out,
        output reg par
);
//1 fard => par 1
//1 joz => par 0
reg [1:0] state;
reg [3:0] cnt;
reg [1:0] baud;
reg [n-1:0] data;
reg part;
integer i;

initial begin
   state = `IDLE;
        out = 0;
        data = 0;
end

always @(posedge uart_clock or posedge rst or negedge serial_data) begin
        if (rst) begin
                data = 0;
                state = `IDLE;
        end
        else if (uart_clock) begin
                if(state == `NEXT) begin
```

```
                                        data = data << 1;
                                        data[0] =serial_data;
                                        cnt = cnt + 1 ;
                                        baud = 0;
                                end
    end
    baud = baud + 1;
     if (cnt == n+5) begin
                                part = 0;
                                par =1;
                                for( i = 0 ; i < n ;i = i+1) begin
                                        part = part^data[i];
                                end
                                if (serial_data == part) begin
                                        par = 0;
                                end
                                state = `IDLE;
                                out = data;
    end
                end
        else if (!serial_data ) begin
                        if(state == `IDLE) begin
                                        cnt = 0;
                                        baud = 0;
                                        state = `NEXT;
                 end
  end
end

endmodule
```

states

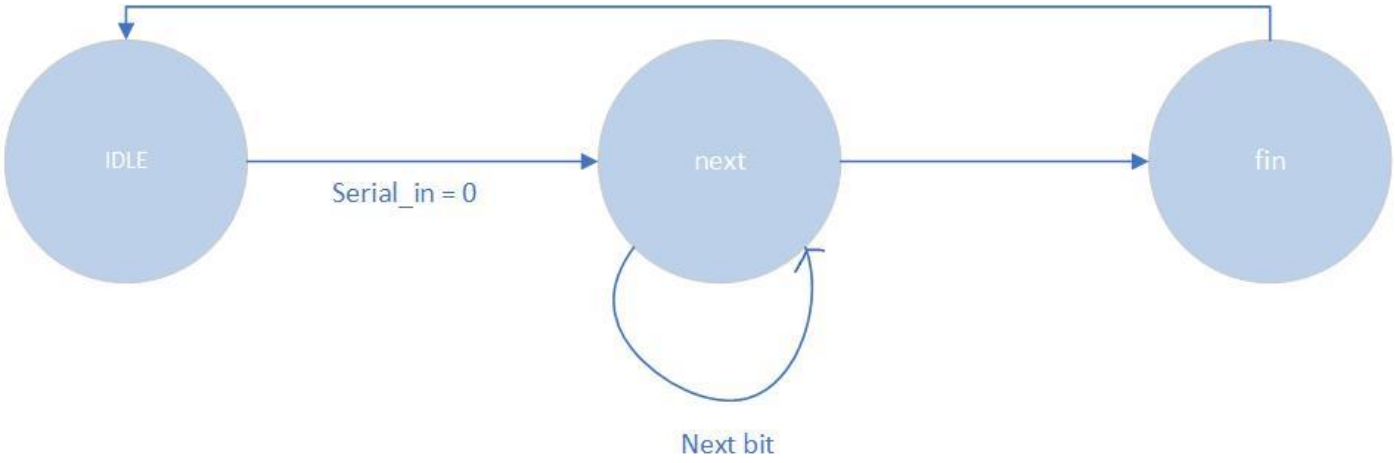
`define IDLE 2'b00

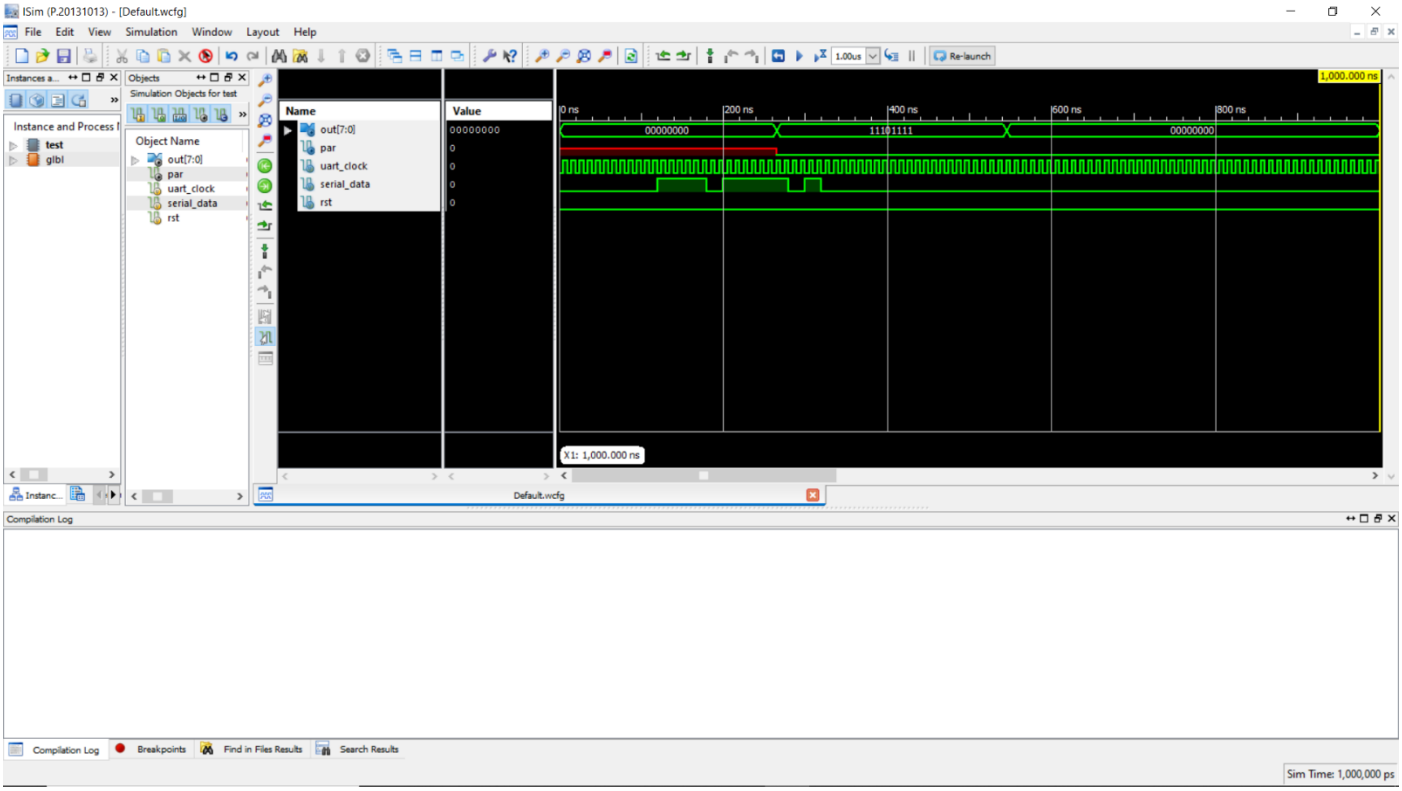
`define NEXT 2'b01

`define FIN 2'b10

test

```
timescale 1ns / 1ps

module test;

	// Inputs reg uart_clock;

	reg serial_data;

	reg rst;

	// Outputs wire [7:0] out;

	wire par;

	// Instantiate the Unit Under Test (UUT)

	receive uut (

		.uart_clock(uart_clock),

		.serial_data(serial_data),

		.rst(rst),

		.out(out),

		.par(par)

	);

	initial begin

		uart_clock = 0;

		serial_data = 0;

		rst = 0;

		#100;

			serial_data = 0; //start

		#20	serial_data = 1;

		#20	serial_data = 1;

		#20	serial_data = 1;

		#20	serial_data = 0;

		#20	serial_data = 1;

		#20	serial_data = 1;

		#20	serial_data = 1;

		#20	serial_data = 1;

		#20	serial_data = 0; //parity

		#20	serial_data = 1; //stop

		#20	serial_data = 0; //start and next num


	end

    always uart_clock = #5 ~uart_clock;

endmodule
```





با 0 شروع به کار میکند و عدد 11101111 با پریتی 1 وارد میشود و چون تعداد 1 ها فرد است پریتی 1 است و با پریتی وارد شده برابر است بنابراین 0 خروجی میدهد و پس از آن 1 می آید و می ایستد تا 0 وارد شود و مقدار میگرد و چون تماما 0 است به کار خود ادامه میدهد و تعداد 1 ها زوج است و 0 میشود که با پریتی 0 وارد شده برابر است و 0 میدهد.