# گزارش آزمایش هفتم آزمایشگاه
# طراحی سیستم دیجیتال


**مزدک تیموریان**

**۴۰۱۱۰۱۴۹۵**

**مریم شیران**

**۴۰۰۱۰۹۴۴۶**

**مهدی بهرامیان**

**۴۰۱۱۷۱۵۹۳**

**دانشکده مهندسی کامپیوتر**

**دانشگاه صنعتی شریف**

مرداد ۱۴۰۳

# فهرست مطالب

# ۱ مقدمه

هدف این آزمایش طراحی یک Universal Asynchronous Receiver Transmitter(UART) است. در قسمت ارسال کننده این دستگاه هربار یک کد ۷ بیتی ASCII بصورت سریال ارسال می گردد. در ابتدا یک بیت شروع (Start) سپس یك بیت توازن (Parity) و بعد ۷ بیت داده ارسال می شوند. در انتها نیز حداقل یك بیت خاتمه (Stop) ارسال می شود (در مجموع ۱۰ بیت).

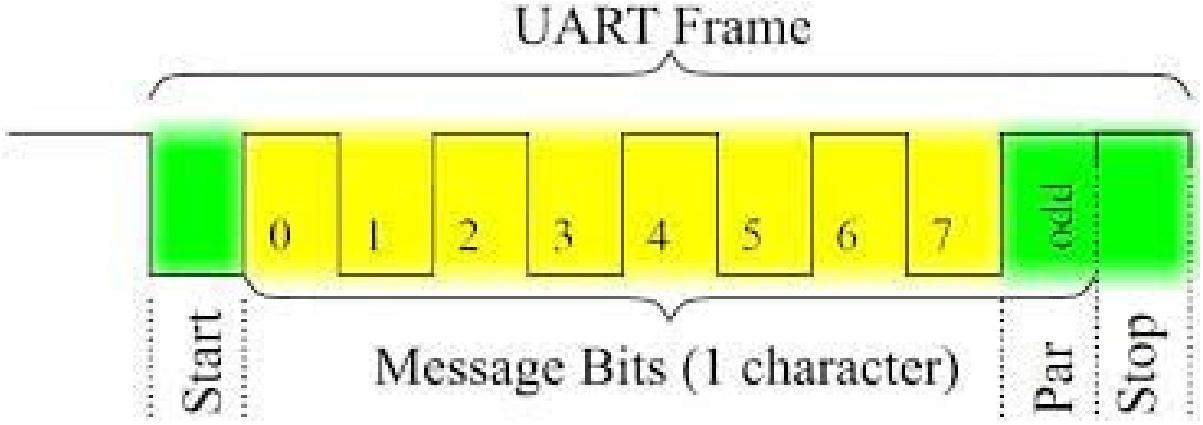
در قسمت گیرنده نیز پس از دریافت بیت شروع (Start) بیت مربوط به داده و توازن (Parity) بصورت سریال دریافت شده و در یک ثبات (Register) ۸ بیتی ذخیره میشود.

# ۲ شرح آزمایش

برنامه طراحی شده ۴ حالت زیر را دارد

- IDLE STOP
- START
- DATA
- PARITY

## ۱.۲ فرستنده




با توجه به شکل بالا در حالت Stop مقدار سیم باید ۱ باشد و با ۰ شدن آن انتقال داده شروع میشود. سپس با نرخ انتقال Baud Rate داده‌ها فرستاده میشوند.

در نهایت پس از انتقال تمامی ۷ بیت داده بیت Parity که حاصل XOR بیت‌های داده است فرستاده میشود.

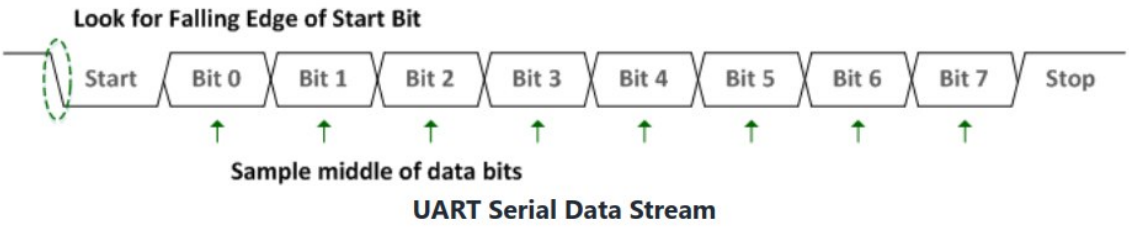
کد طراحی را در شکل زیر میتوانید مشاهده کنید.

```
module uart_trans #(
    parameter int CLOCK_RATE = 50_000_000,
    parameter int BAUD_RATE  = 115200
) (
    input clk,
    input rst,
    input [7:0] data,
    output reg tx
);
    localparam int BAUD_DELAY = (CLOCK_RATE / BAUD_RATE) - 1;
    localparam int BAUD_DELAY0 = (CLOCK_RATE / BAUD_RATE / 2) - 1;

    localparam int IDLE_STOP = 0;
    localparam int START = 1;
    localparam int DATA = 2;
    localparam int PARITY = 3;


    reg [ 1:0] state;
    reg [ 2:0] idx;
    reg [31:0] delay;

    always @(posedge clk, negedge rst) begin
        if (!rst) begin
            delay <= 0;
            state <= 0;
            idx   <= 0;
            tx    <= 1;
```

```
        end else begin
            if (|delay) begin
                delay <= delay - 1;
            end else begin
                case (state)
                    IDLE_STOP: begin
                        delay <= BAUD_DELAY;
                        state <= state + 1;
                        tx    <= 1;
                    end
                    START: begin
                        delay <= BAUD_DELAY;
                        state <= state + 1;
                        tx    <= 0;
                    end
                    DATA: begin
                        delay <= BAUD_DELAY;
                        tx    <= data[idx];
                        idx   <= idx + 1;
                        state <= state + &idx;
                    end
                    PARITY: begin
                        delay <= BAUD_DELAY;
                        tx    <= ^data;
                        state <= state + 1;
                    end
                endcase
            end
        end
    end

endmodule
```

## ۲.۲ گیرنده

ساختار قسمت گیرنده نیز بسیار مشابه فرستنده است با این تفاوت که اینجا اطلاعات از سیم دریافت و ذخیره میشوند. همچنین از آنجایی که میخواهیم در وسط سیگنال داده فرستاده شده داده را نمونه بگیریم ابتدا یک تأخیر به میزان نصف Baud Delay در نظر میگیریم.



UART Serial Data Stream

کد طراحی این قسمت نیز در شکل‌های زیر قابل مشاهده است.

```
module uart_rcv #(
    parameter int CLOCK_RATE = 50_000_000,
    parameter int BAUD_RATE  = 115200
) (
    input clk,
    input rst,
    output reg [7:0] data,
    output reg parity,
    input rx
);
    localparam int BAUD_DELAY = (CLOCK_RATE / BAUD_RATE) - 1;
    localparam int BAUD_DELAY0 = (CLOCK_RATE / BAUD_RATE / 2) - 1;

    localparam int IDLE_STOP = 0;
    localparam int START = 1;
    localparam int DATA = 2;
    localparam int PARITY = 3;


    reg [1:0] state;
    reg [2:0] idx;
    reg [31:0] delay;
    reg prx;

    always @(posedge clk, negedge rst) begin
        if (!rst) begin
            delay  <= 0;
            state  <= 0;
            idx    <= 0;
            data   <= 0;
            parity <= 0;
```

```
        end else begin
            prx <= rx;
            if (|delay) begin
                delay <= delay - 1;
            end else begin
                case (state)
                    IDLE_STOP: begin
                        if (!rx && prx) begin
                            delay <= BAUD_DELAY0;
                            state <= state + 1;
                        end
                    end
                    START: begin
                        delay <= BAUD_DELAY;
                        state <= state + 1;
                    end
                    DATA: begin
                        delay     <= BAUD_DELAY;
                        idx       <= idx + 1;
                        state     <= state + &idx;
                        data[idx] <= rx;
                    end
                    PARITY: begin
                        delay  <= BAUD_DELAY;
                        parity <= rx;
                        state  <= state + 1;
                    end
                endcase
            end
        end
    end

endmodule
```

همانطور که مشاهده میکنید در حالت IDLE STOP تأخیر BAUD_DELAY/۲ اضافه میشود که باعث میشود نمونه‌گیری‌های مان در وسط سیگنال باشند.

## ۳ شبیه‌سازی و بررسی عملکرد UART

حال برای ماژول‌های بالا یک ماژول تست طراحی میکنیم.

```
module test ();
    reg clk0, clk1, rst0, rst1;
    reg  [7:0] tData;
    wire [7:0] rData;
    wire       rParity;


    wire       tx_rx;
    uart_trans #(
        .CLOCK_RATE(16),
        .BAUD_RATE (4)
    ) transmitter (
        .tx  (tx_rx),
        .clk (clk0),
        .rst (rst0),
        .data(tData)
    );

    uart_rcv #(
        .CLOCK_RATE(8),
        .BAUD_RATE (4)
    ) receiver (
        .rx(tx_rx),
        .clk(clk1),
        .rst(rst1),
        .data(rData),
        .parity(rParity)
    );

    // generating clk0!=clk1
    initial clk0 = 0;
    always #1 clk0 = !clk0;
    initial clk1 = 0;
    always #2 clk1 = !clk1;
```

```
  initial begin
    $monitor($time, " --  transmit : ", tData, " receive : ", rData, ",", rParity,
             "  r-error : ", ^rData ^ rParity);
    rst0  = 0;
    rst1  = 0;
    tData = 8'b0000_0000;
    #1;
    rst0 = 1;
    rst1 = 1;

    #8 wait (transmitter.state == 0 && transmitter.delay == 0) tData = 8'b1111_1111;
    #8 wait (transmitter.state == 0 && transmitter.delay == 0) tData = 8'b0000_0000;
    #8 wait (transmitter.state == 0 && transmitter.delay == 0) tData = 8'b0101_0101;
    #8 wait (transmitter.state == 0 && transmitter.delay == 0) tData = 8'b1100_1010;
    #8 wait (transmitter.state == 0 && transmitter.delay == 0) tData = 8'b0010_1100;
    #8 wait (transmitter.state == 0 && transmitter.delay == 0) tData = 8'b1000_0010;
    #8 wait (receiver.state == 0 && receiver.delay == 0);
    #8 wait (receiver.state == 0 && receiver.delay == 0);
    $finish(0);
  end
endmodule
```

خروجی این ماژول نیز در شکل زیر قابل مشاهده است.

```
  0 --  transmit :   0 receive :   0,0  r-error : 0
 87 --  transmit : 255 receive :   0,0  r-error : 0

110 --  transmit : 255 receive :   1,0  r-error : 1
118 --  transmit : 255 receive :   3,0  r-error : 0
126 --  transmit : 255 receive :   7,0  r-error : 1
134 --  transmit : 255 receive :  15,0  r-error : 0
142 --  transmit : 255 receive :  31,0  r-error : 1
150 --  transmit : 255 receive :  63,0  r-error : 0
158 --  transmit : 255 receive : 127,0  r-error : 1
166 --  transmit : 255 receive : 255,0  r-error : 0
175 --  transmit :   0 receive : 255,0  r-error : 0
198 --  transmit :   0 receive : 254,0  r-error : 1
206 --  transmit :   0 receive : 252,0  r-error : 0
214 --  transmit :   0 receive : 248,0  r-error : 1
222 --  transmit :   0 receive : 240,0  r-error : 0
230 --  transmit :   0 receive : 224,0  r-error : 1
238 --  transmit :   0 receive : 192,0  r-error : 0
246 --  transmit :   0 receive : 128,0  r-error : 1
254 --  transmit :   0 receive :   0,0  r-error : 0
263 --  transmit :  85 receive :   0,0  r-error : 0
286 --  transmit :  85 receive :   1,0  r-error : 1
302 --  transmit :  85 receive :   5,0  r-error : 0
318 --  transmit :  85 receive :  21,0  r-error : 1
334 --  transmit :  85 receive :  85,0  r-error : 0
351 --  transmit : 202 receive :  85,0  r-error : 0
374 --  transmit : 202 receive :  84,0  r-error : 1
382 --  transmit : 202 receive :  86,0  r-error : 0
390 --  transmit : 202 receive :  82,0  r-error : 1
398 --  transmit : 202 receive :  90,0  r-error : 0
406 --  transmit : 202 receive :  74,0  r-error : 1
430 --  transmit : 202 receive : 202,0  r-error : 0
439 --  transmit :  44 receive : 202,0  r-error : 0
470 --  transmit :  44 receive : 200,0  r-error : 1
478 --  transmit :  44 receive : 204,0  r-error : 0


502 --  transmit :  44 receive : 236,0  r-error : 1
510 --  transmit :  44 receive : 172,0  r-error : 0
518 --  transmit :  44 receive :  44,0  r-error : 1
526 --  transmit :  44 receive :  44,1  r-error : 0
527 --  transmit : 130 receive :  44,1  r-error : 0
558 --  transmit : 130 receive :  46,1  r-error : 1
566 --  transmit : 130 receive :  42,1  r-error : 0
574 --  transmit : 130 receive :  34,1  r-error : 1
590 --  transmit : 130 receive :   2,1  r-error : 0
606 --  transmit : 130 receive : 130,1  r-error : 1
614 --  transmit : 130 receive : 130,0  r-error : 0
```