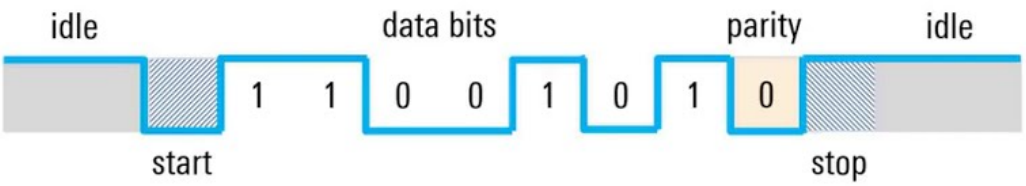
📝 گزارش آزمایش ۷

آزمایشگاه طراحی سیستم‌های دیجیتال

Alireza Habibzadeh `99109393`
May 31, 2022



مثالی از UART (متفاوت با سوال ما)

مقدمه

در این آزمایش قصد داریم یک UART طراحی کنیم که مقدار خط داده آن به طور پیشفرض 0 است، برای شروع یک بیت 1 ارسال می‌کند سپس بیت parity داده‌ای که قرار است بفرستد را می‌فرستد و سپس داده‌ها را بیت به بیت ارسال می‌کند. در انتها یک بیت stop ارسال می‌کند.

فرستنده (Transmitter)

عملکرد این قطعه به این صورت است که با بالا رفتن سیگنال `send` بلافاصله داده‌های موجود در ورودی کپی شده و پس از آن ابتدا بیت parity و سپس خود آن‌ها ارسال می‌شوند. در ابتدای و انتهای این فرایند نیز بیت‌های راهنمای Stop و Start ارسال می‌شوند.

برای محاسبه‌ی سیگنال parity از عملگر XOR (^) در توصیف رفتاری استفاده شده است. (خط ۲۴)

این قطعه دو پارامتر `DATA_WIDTH` (برابر با ۷ در سوال ما) و `bud` در نظر گرفته شده که باید با مقدار آن در گیرنده هماهنگ باشد.

```
1  module UART_transmitter(
2      input send,
3      input [DATA_WIDTH-1:0] data,
```

```
    output reg serial_out,
    output reg busy
    );

    parameter DATA_WIDTH = 7;
    parameter bud = 10;
    reg [DATA_WIDTH-1:0] data_reg;
    integer i;

    initial begin
        data_reg <= 0;
        serial_out <= 0;
        busy <= 0;
    end

    always @(posedge send) begin
         if (~busy) begin
              busy = 1'b1;
              data_reg = data;
            serial_out = 1'b1;
            #bud serial_out = ^data_reg;

            for (i = 0; i < DATA_WIDTH; i = i + 1) begin
                #bud serial_out = data_reg[i];
            end
              #bud serial_out = 1'b1; // stop signal
                #bud serial_out = 1'b0; // reset the dataline
                busy = 1'b0;
          end
    end
endmodule
```

گیرنده (Receiver)

نکته‌ی مهمی که در رابطه با گیرنده وجود دارد این است که پس از دریافت اولین سیگنال (سیگنال Start) بهتر است به میزان `#(bud/2)` تاخیر داشته باشیم تا سیگنال‌ها را هنگام شروع و وقتی ناپایدار هستند نخوانیم بلکه دقیقا وسط دو تغییر که سیگنال پایدار و ثابت شده آن را بخوانیم.

اینجا هم دو پارامتر `DATA_WIDTH` و `bud` در نظر گرفته شده که باید با مقدار آن در فرستنده هماهنگ باشد.

در انتهای دریافت بیت‌ها دو چیز چک می‌شود، همخوانی بیت parity با داده‌های ارسالی و بیت Stop. در صورتی که هر کدام برقرار نبود یعنی خطایی رخ داده و سیگنال خروجی `fault` روشن می‌شود. (خط ۳۰)

```
1  module UART_receiver(
2      input serial_in,
3      output reg busy,
4      output reg [DATA_WIDTH-1:0] data_reg,
5      output reg fault
6      );
7
8      parameter DATA_WIDTH = 7;
9      parameter bud = 10;
10     reg parity;
11     reg stop_signal;
12     integer i;
13
14     initial begin
15         busy <= 1'b0;
16         data_reg <= 0;
17         fault <= 1'b0;
18     end
19
20     always @(posedge serial_in) begin
21         if (~busy) begin
22             busy = 1'b1;
23             #(bud/2);
```

```
            #bud parity = serial_in;

            for (i = 0; i < DATA_WIDTH; i = i + 1) begin
                #bud data_reg[i] = serial_in;
            end

            #bud fault = (parity != ^data_reg) & serial_in; // check parity and stop signal
            #bud busy = 1'b0;
        end
    end
endmodule
```

تست

برای بررسی صحت عملکرد مدار از test bench زیر استفاده شده است. قسمت‌هایی از آن به صورت خودکار توسط نرم‌افزار تولید می‌شود و کافی است داخل بلاک `initial` را تغییر دهیم. این بار یک بلاک `always` هم قرار دارد که وظیفه‌ی آن گزارش داده‌ی دریافت شده در کنار داده‌ی اصلی ارسالی هر وقت که داده کامل رسید (سیگنال `busy` گیرنده صفر شود) است.

(اسم این فایل به اشتباه `transmitter_tb.v` ذخیره شده در حالی که این مربوط به هر دو ماژول است)

کاراکترهای `"Salam!"` را به ترتیب در این تست ارسال کردیم. متاسفانه در وریلاگ راه ساده‌ای برای پیمودن یک استرینگ و جدا کردن کاراکترهای آن پیدا نکردم (بیشتر راه‌ها برای سیستم‌وریلاگ و... بود) بنابراین کاراکترها به صورت دستی و جداگانه ارسال شده‌اند (پس از کامپایل، نرم‌افزار به جای کاراکترهای ما اعداد ۷ بیتی قرار خواهد داد). موج ورودی و خروجی مدار در انتهای این pdf آمده است.

```
module transmitter_tb;
    // Inputs
    reg send;
    reg [6:0] data;
    // Outputs
    wire line;
    wire transmitter_busy;
    wire receiver_busy;
```

```
    wire receiver_fault;
    wire [6:0] received_data;

    // Instantiate the Unit Under Test (UUT)
    UART_transmitter transmitter (
        .send(send),
        .data(data),
        .serial_out(line),
        .busy(transmitter_busy)
    );

    UART_receiver receiver (
        .serial_in(line),
        .busy(receiver_busy),
        .data_reg(received_data),
        .fault(receiver_fault)
    );

    always @(negedge receiver_busy) begin
        $display("sent: %s, received: %s\nparity fault: %b\n", data, received_data, receiver_fault);
    end
    initial begin
        // Initialize Inputs
        send = 0;
        data = 0;
        // Wait 100 ns for global reset to finish
        #100;

        // Add stimulus here
        data = "S";
        #5 send = 1; #5 send = 0;
        #140;
```

```
41
42          data = "a";
43          #5 send = 1; #5 send = 0;
44          #140;
45
46          data = "l";
47          #5 send = 1; #5 send = 0;
48          #140;
49
50          data = "a";
51          #5 send = 1; #5 send = 0;
52          #140;
53
54          data = "m";
55          #5 send = 1; #5 send = 0;
56          #140;
57
58          data = "!";
59          #5 send = 1; #5 send = 0;
60          #140;
61          $finish;
62      end
63  endmodule
```

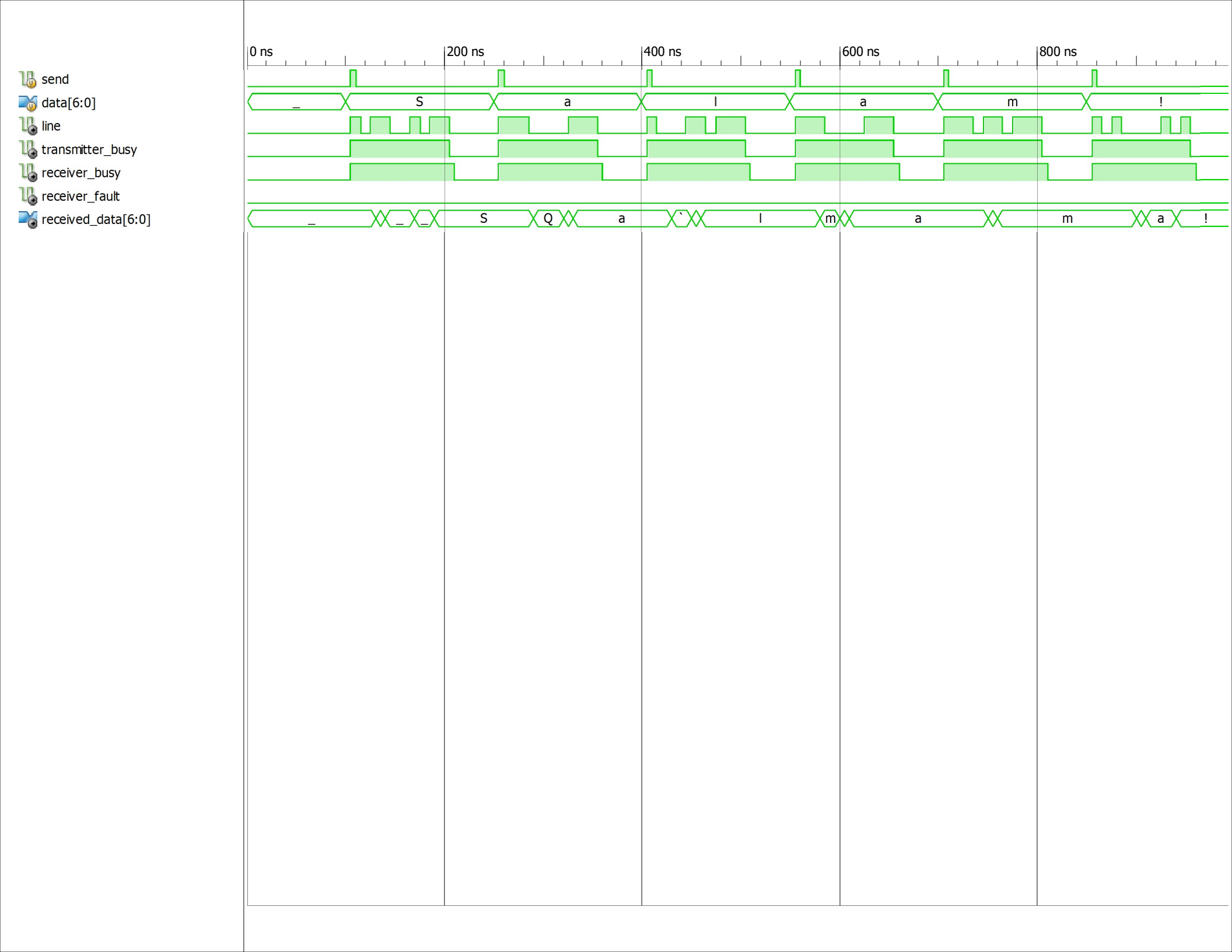
خروجی کنسول

```
1  ISim P.20131013 (signature 0x7708f090)
2  This is a Full version of ISim.
3  Time resolution is 1 ps
4  Simulator is doing circuit initialization process.
5  Finished circuit initialization process.
6  sent:   , received:
```

```
parity fault: 0

sent: S, received: S
parity fault: 0

sent: a, received: a
parity fault: 0

sent: l, received: l
parity fault: 0

sent: a, received: a
parity fault: 0

sent: m, received: m
parity fault: 0

sent: !, received: !
parity fault: 0

Stopped at time : 1 us : File "C:/hw7-99109393/transmitter_tb.v" Line 90
ISim>
```

سنتز

پس از سنتز موفقیت‌آمیز می‌توانیم گزارش سنتز را از نرم‌افزار دریافت کنیم. فایل کامل این گزارش در زیپ ارسالی پیوست شده است.


0 ns
200 ns
400 ns
600 ns
800 ns
send
data[6:0]
line
transmitter_busy
receiver_busy
receiver_fault
received_data[6:0]
_
S
a
l
a
m
!
_
_
_
S
Q
a
`
l
m
a
m
a
!