آزمایشگاه سیستم های دیجیتال

گزارش آزمایش هفتم
UART

امیرحسین علمدار: 400105144

پیام تائبی: 400105867

علیرضا سلیمیان: 400105036

بهار 1402

شرح کلی

در این آزمایش سعی داشتیم تا با استفاده از پروتکل ارتباطی UART، مداری روی وریلاگ سنتز کنیم، تا به یک قطعه که در آزمایشگاه داده شد، داده هایی را بدهد و این قطعه آن هارا به یک سرور بفرستد، سپس درصورت پیاده سازی درست مدار، سرور اطلاعات را به درستی می خواند و با توجه به آن، چیزی مربوط به آیپی مورد نظر را در سایت تغییر می دهد، مثلا در این آزمایش، با ارسال داده، رنگ یک هدر در سایت تغییر می کرد، همچنین پیاده سازی باید قابلیت دریافت داده از سرور و نمایش آن برروی LED ها را داشته باشد، مثلا در این آزمایش یک عدد باینری را در سایت ورودی می دادیم و به سرور ریکوئست زده میشد، سپس سرور عدد را به قطعه ارتباطی که به pin های fpga وصل بود می فرستاد و آن قطعه داده را به fpga منتقل می کرد و به درستی روی آن نمایش داده میشد.

مشکلات و تغییرات

در پیش گزارش، فرضیات و روند کلی کار نشان داده شد و بنظر نحوه کار با آن فرضیات درست بود، اما در آزمایشگاه متوجه شدیم که داده ها به درستی ارسال نمی شود و مشکلی وجود دارد، بعد از مطالعه کد فهمیدیم و بررسی دوباره پروتکل، متوجه شدیم با یک شدن سیگنال start توسط کاربر، داده ها به درستی ارسال می شوند و دوباره به استیت اولیه باز می گردیم اما اگر وقتی به استیت اولیه بازگشتیم start همچنان 1 بود، دوباره تمام استیت ها طی خواهند شد و به اینصورت ممکن است هزاران بار داده تکراری بفرستیم که استفاده ای ندارد و درست خوانده نخواهد شد، زیرا در انتهای ارسال باید سیگنال hi شود تا مقصد بفهمد تمام شده است.

به توصیه آقای بحرینی، برای رهایی از این مشکل یک حالت Post_Start تعریف کردیم. به اینگونه که همواره در استیت Start هستیم و 1 شدن سیگنال start، به استیت Post_Start می رویم، ولی تا وقتی سیگنال شروع صفر نشود ادامه نمی دهیم، بعد از اینکه سیگنال شروع 0 شد، به استیت های بعد می رویم و در نهایت به حالت Start باز می گردیم. به اینصورت دیگر کاربر سیگنال شروع را روشن نمی گذارد تا هزاران بار داده ارسال شود، دقت کنید از آنجایی که در هر ثانیه 115200 بیت فرستاده می شد، حتی اگر سیگنال شروع مدت خیلی کوتاهی نیز یک میماند، داده هزاران بار ارسال می شد.(یکی از باگ هایی که بسیار زمان گرفت، این بود که استیت جدید تعریف کرده بودیم اما از آنجایی که متغیر state قبلا 4 حالت داشت، 2 بیتی بود و در حالت جدید باید 3 بیت برای آن در نظر گرفته میشد ولی از دیدمان پنهان مانده بود و ماژول فرستنده مدت زیادی کار نمی کرد تا این باگ پیدا شد)

به ترتیب کد ماژول فرستنده قبل و جدید را می گذاریم تا تفاوت دیده شود:

```
case (state)
        S_START: begin
            if (start) begin
                state <= S_POST_START;

            end
        end
        S_POST_START: begin
            if(!start) begin
                sent <= 0;
                data_index <= 0;
                data <= data_in;
                s_out <= START_SIG;
                state <= S_SEND;
            end
        end
        S_SEND: begin
            s_out <= data[data_index];
            if (data_index == 6)
                state <= S_PARITY;
            data_index <= data_index + 1;
        end
        S_PARITY: begin
            s_out <= parity_sig;
            state <= S_STOP;
        end
        S_STOP: begin
            s_out <= 1;
            sent <= 1;
            state <= S_START;

        end
        default: state <= S_START;
    endcase
```

```
case (state)
    S_START: begin
        if (start && !stop) begin
            s_out <= START_SIG;
            data_index <= 0;
            data <= data_in;
            state <= S_PARITY;
            sent <= 0;
        end else if(!start)
            stop <= 0;
    end
    S_PARITY: begin
        s_out <= parity_sig;
        state <= S_SEND;
    end
    S_SEND: begin
        s_out <= data[data_index];
        if (data_index == 6)
            state <= S_STOP;
        data_index <= data_index + 1;
    end
    S_STOP: begin
        s_out <= !START_SIG;
        state <= S_START;
        sent <= 1;
        stop <= 1;
    end
    default: state <= S_START;
endcase
```