

آزمایشگاه طراحی سیستمهای دیجیتال

گزارش آزمایش ۴

۱۴۰۳ مرداد ۴



امیرحسین ملک محمدی ۰۰۱۱۰۶۵۷۷
متین محمدی ۰۰۱۱۱۰۳۲۹
صادق محمدیان ۰۰۱۱۰۸۴۷۷

۱ شرح آزمایش

در این آزمایش می‌خواهیم یک پشته را به صورت توصیف رفتاری طراحی کنیم. این پشته می‌تواند ۸ کلمه ۴ بیتی را در خود نگه دارد.

۱.۱ ورودی‌ها

ورودی‌های این پشته سیگنال‌های کلاک، ریست، داده ورودی، پوش، و پاپ اند. ورودی ریست یک سیگنال ناهمگام(asynchronous) است و وقتی که فعال شود پشته خالی می‌شود. زمانی که سیگنال پوش فعال شود و پشته پر نباشد باید داده ورودی را در پشته قرار دهیم و زمانی که سیگنال پاپ فعال شود و پشته خالی نباشد، داده آخر را خروجی می‌دهیم و آن را از پشته پاک می‌کنیم.

۲.۱ خروجی‌ها

خروجی‌های این پشته سیگنال‌های داده خروجی، پر بودن، و خالی بودن اند. زمانی که هیچ داده‌ای در پشته نباشد(یعنی نشانگر به خانه صفر اشاره کند) خروجی خالی بودن فعال می‌شود. زمانی که پشته پر باشد(یعنی نشانگر به خانه هشتم اشاره کند) خروجی پر بودن فعال می‌شود. اگر ورودی پاپ فعال باشد و پشته خالی نباشد آخرین داده را بر داده خروجی قرار می‌دهیم.

۲ کد وریلاغ و الگوریتم

در شکل ۱ کد وریلاغ پشته آمده.

در برنامه یک متغیر ۳ بیتی به نام index داریم که به آخرین خانه خالی پشته اشاره می‌کند. خروجی Empty با مقایسه index با ۰ بدست می‌آید و خروجی Full با مقایسه index با ۸ بدست می‌آید.

ورودی Rst به صورت ناهمگام کار می‌کند؛ هر وقت Rst از صفر یک شود متغیر index را برابر ۰ قرار می‌دهیم و دادهای خروجی نمی‌دهیم.

اگر در لبه بالارونده کلاک باشیم و Rst زده نشده باشد، سیگنال‌های Push و Pop را بررسی می‌کنیم. اگر Pop روشن شده باشد و پشته خالی نباشد اول index را یکی کم می‌کنیم سپس داده اخر پشته را خروجی می‌دهیم. وگرنه اگر Push فعال شده باشد و پشته پر نباشد داده ورودی را در آخرین خانه خالی پشته قرار می‌دهیم سپس index را یکی زیاد می‌کنیم. در غیر این دو صورت اتفاقی نمی‌افتد و داده خروجی برابر Z خواهد بود.

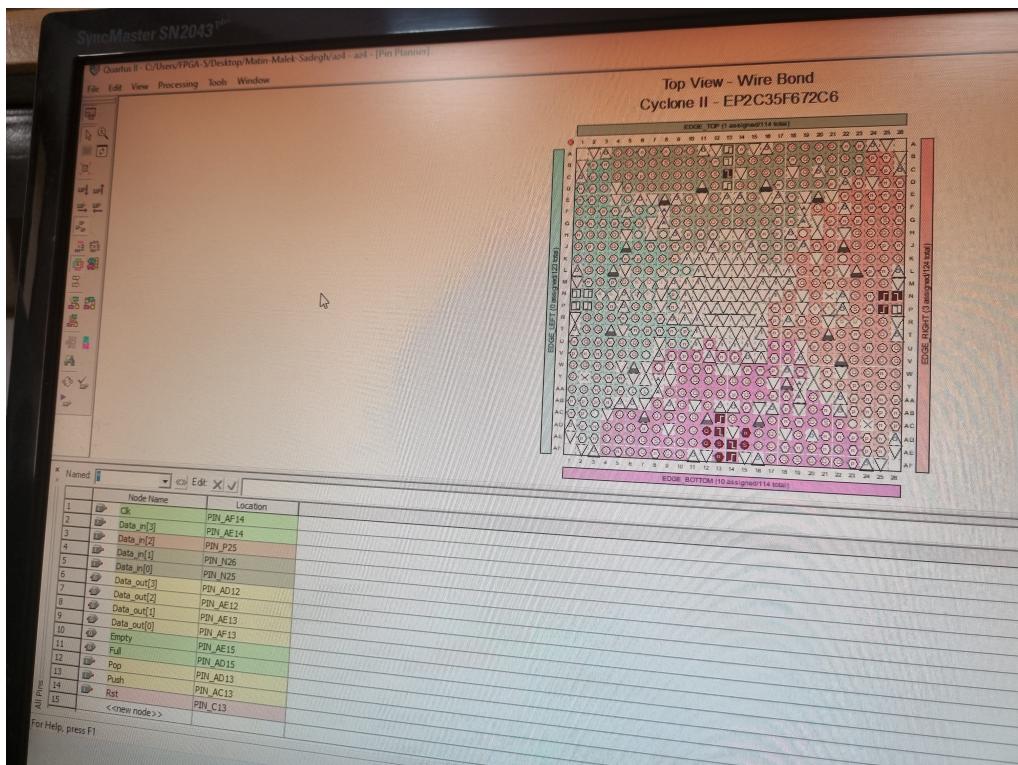
```

module STACK_Q (input Clk, input Rst, input Push, input Pop, input [3:0] Data_in,
                output Full, output Empty, output [3:0] Data_out);
    reg [3:0] memory [7:0];
    reg [3:0] index = 3'b0;
    assign Empty = (index == 0);
    assign Full = (index == 8);
    reg [3:0] data;
    assign Data_out = data;

    always @(*(posedge Clk, posedge Rst)) begin
        if (Rst) begin
            index = 3'b0;
            data = 4'bz;
        end
        else begin
            data = 4'bz;
            if (Pop && ~Empty ) begin
                index = index - 3'b1;
                data = memory[index];
            end
            else if (Push && ~Full ) begin
                memory[index] = Data_in;
                index = index + 3'b1;
            end
        end
    end
endmodule

```

شکل ۱: کد وریلاگ پشتہ



شکل ۲: پین‌های ورودی و خروجی

۳ پیاده‌سازی روی بورد

ابتدا یک پروژه کوارتوس باز می‌کنیم و یک فایل وریلاگ به پروژه اضافه می‌کنیم و در آن کد پشته را قرار می‌دهیم.

سپس پروژه را کامپایل می‌کنیم.

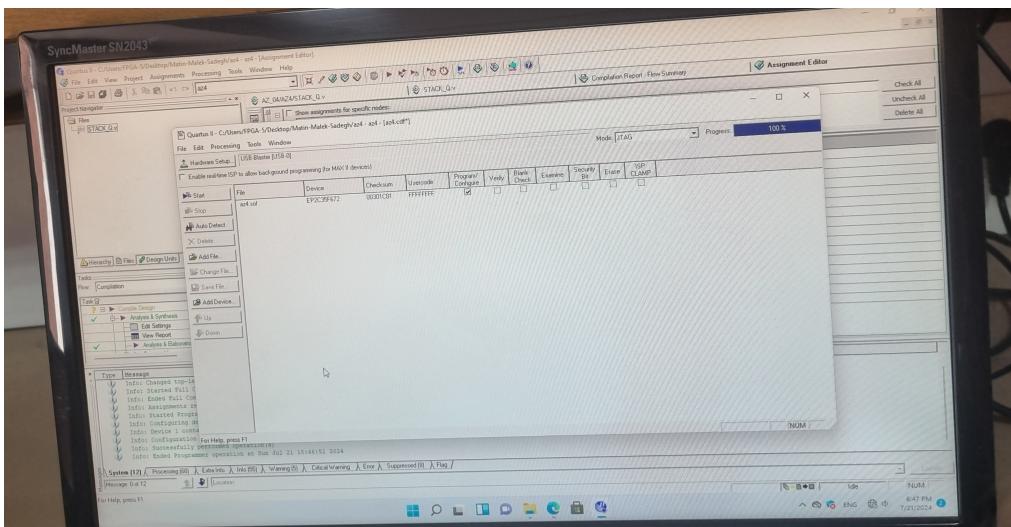
در مرحله بعد پین‌های ورودی و خروجی را معین می‌کنیم. در شکل ۲ نحوه مشخص کردن پین‌های ورودی و خروجی آمده. LED های قرمز شماره ۱۴ تا ۱۷ برای داده خروجی استفاده می‌شوند. شماره ۱۳ برای خروجی Empty و شماره ۱۲ برای خروجی Full است. کلیدهای شماره ۰ تا ۳ برای داده ورودی استفاده می‌شوند. کلید شماره ۴ ورودی کلاک است. کلیدهای ۵ و ۶ به ترتیب مربوط به ورودی‌های Push و Pop است. در نهایت کلید شماره ۵ برای ورودی ریست است.

پس از تخصیص پین‌ها بورد را روی حالت program می‌گذاریم و فایل scf را روی بورد آپلود می‌کنیم.

اتمام فرایند آپلود در شکل ۳ آمده است.

سپس بورد را روی حالت run قرار می‌دهیم. در این مرحله بورد آماده کار است.

دستیار آموزشی عملکرد صحیح بورد را بررسی کردن.



شکل ۳: فرایند آپلود برنامه

۴ نتیجه‌گیری

در این آزمایش ابتدا با زبان وریلاگ و به صورت توصیف رفتاری یک پیشه طراحی کردیم. پس از اطمینان از درستی کد وریلاگ، آن را کامپایل کردیم و پین‌های ورودی و خروجی را مشخص کردیم و روی FPGA آپلود کردیم. سپس بورد را بررسی کردیم و دیدیم که مطابق انتظار عمل می‌کند.