📝 گزارش آزمایش ۴

آزمایشگاه طراحی سیستم‌های دیجیتال

> *Alireza Habibzadeh* `99109393`
> *April 19, 2022*

پیاده‌سازی پشته

برای نگه داشتن بلوک‌های پشته باید از یک رجیستر دوبعدی استفاده کنیم. یعنی ۱۶ عدد رجیستر که هر کدام ۸ بیت فضا دارند:

```
reg [7:0] storage [15:0];
```

همچنین از آنجایی که توصیف ما رفتاری است، کافی است در انتهای هر تغییر احتمالی متغییرها یکبار `Full` و `Empty` را به سادگی با همان تعریفشان آپدیت کنیم:

```
Full <= (counter == 16);
Empty <= (counter != 0);
```

از متغییر `count` برای نگه داشتن تعداد فعلی پشته (یا همان آدرس فعلی در پشته) استفاده شده است. در صورتی که برای این متغییر از ۴ بیت استفاده کنیم تنها تا ۱۵ می‌تواند نگه دارد و برای مقدار خاص ۱۶ باید کمی توصیفمان را پیچیده کنیم. (البته مدار در این حالت `Full` است و می‌توان با بررسی آن مانند بیت بعدی فهمید در این حالت هستیم) ولی با افزودن یک بیت به `counter` دیگر مشکل overflow نداریم و توصیف ساده‌تر می‌شود.

با این توضیحات کد نهایی می‌شود:

```
module stack(
    input Clk,
    input RstN,
    input [7:0] Data_In,
    input Push,
    input Pop,
    output reg [7:0] Data_Out,
```

```
    output reg Full,
    output reg Empty
    );

          reg [7:0] storage [15:0];
          reg [4:0] counter = 5'b0;

          always @(posedge Clk, negedge RstN) begin
                 if (~RstN)
                        begin
                        counter = 0;
                        Data_Out = 8'bz;
                        end
                 else if (Pop && Empty)
                        begin
                        counter = counter - 1;
                        Data_Out = storage[counter];
                        end
                 else if (Push && ~Full)
                        begin
                        storage[counter] = Data_In;
                        counter = counter + 1;
                        end

          Full <= (counter == 16);
          Empty <= (counter != 0);
          end
endmodule
```

تست

برای بررسی صحت عملکرد مدار از test bench زیر استفاده شده است. البته قسمت‌هایی از آن به صورت خودکار توسط نرم‌افزار تولید می‌شود و کافی است داخل بلاک `initial` را تغییر دهیم.

در این تست ابتدا اعداد ۱ و ۲ و ۳ و ۴ در چندین خانه‌ی پشته پر می‌شوند تا جایی که پشته به طور کامل پر می‌شود. در این حالت دیگر پشته ورودی جدیدی نمی‌گیرد و آن‌ها را نادیده می‌گیرد. همچنین می‌بینیم که با گرفتن اولین مقدار و همچنین با پر شدن پشته، سیگنال‌های `Full` و `Empty` به درستی کار می‌کنند.

همچنین باید ابتدای کار باید یک بار ماژول را ریست کنیم. و دقت کنیم که ریست آن active low است و در ابتدا در حالت ریست نیست و باید یک لبه‌ی پایین رونده‌ی ریست بزنیم.

```
`timescale 1ns / 1ps

module stack_tb;
        // Inputs
        reg Clk;
        reg RstN;
        reg [7:0] Data_In;
        reg Push;
        reg Pop;
        // Outputs
        wire [7:0] Data_Out;
        wire Full;
        wire Empty;
        // Instantiate the Unit Under Test (UUT)
        stack uut (
                .Clk(Clk),
                .RstN(RstN),
                .Data_In(Data_In),
                .Push(Push),
                .Pop(Pop),
                .Data_Out(Data_Out),
                .Full(Full),
                .Empty(Empty)
        );

```

```
26          always #5 Clk = ~Clk;
27
28          initial begin
29                  // Initialize Inputs
30                  Clk = 0;
31                  RstN = 1'b1;
32                  Data_In = 0;
33                  Push = 0;
34                  Pop = 0;
35                  // Wait 100 ns for global reset to finish
36                  #100;
37                  // Add stimulus here
38                  $monitor("Data in = %d, Data out = %d, Full =
%d, Empty = %d, Push = %d, Pop = %d", Data_In, Data_Out, Full,
Empty, Push, Pop);
39
40                  RstN = 1'b0; #10;
41                  RstN = 1'b1; #10;
42                  Data_In = 8'd1;
43                  Push = 1'b1;
44                  #40 Data_In = 8'd2;
45                  #40 Data_In = 8'd3;
46                  #40 Data_In = 8'd4;
47                  #40 Push = 1'b0;
48                  #30 Pop = 1'b1;
49                  #200;
50                  $stop;
51          end
52  endmodule
```

شبیه‌سازی

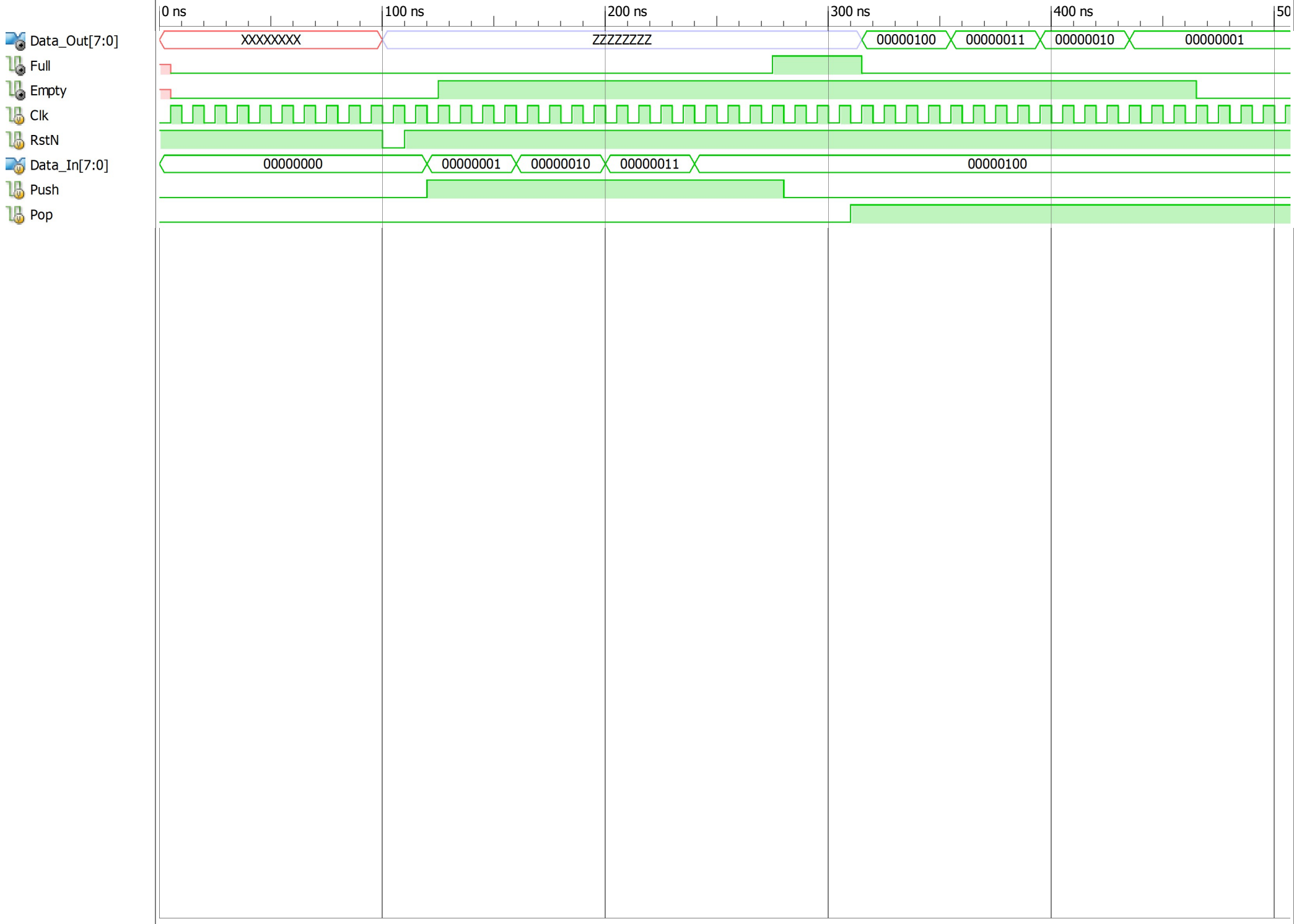
پس از شبیه‌سازی، دستور `$monitor` که در ابتدای بلاک `initial` قرار داده بودم خروجی زیر را در کنسول تولید می‌کند. البته نکته‌ای که وجود دارد این است که دستور `$monitor` تنها در صورت

تغییر متغییرهای داخل آن خروجی جدید را چاپ می‌کند. در تست ما تعدادی ورودی یکسان چند بار کلاک خورده‌اند که در اینجا مشخص نیستند اما در خروجی موج قابل بررسی هستند.

```
ISim P.20131013 (signature 0x7708f090)
This is a Full version of ISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
Data in =   0, Data out =   z, Full = 0, Empty = 0, Push = 0,
Pop = 0
Data in =   1, Data out =   z, Full = 0, Empty = 0, Push = 1,
Pop = 0
Data in =   1, Data out =   z, Full = 0, Empty = 1, Push = 1,
Pop = 0
Data in =   2, Data out =   z, Full = 0, Empty = 1, Push = 1,
Pop = 0
Data in =   3, Data out =   z, Full = 0, Empty = 1, Push = 1,
Pop = 0
Data in =   4, Data out =   z, Full = 0, Empty = 1, Push = 1,
Pop = 0
Data in =   4, Data out =   z, Full = 1, Empty = 1, Push = 1,
Pop = 0
Data in =   4, Data out =   z, Full = 1, Empty = 1, Push = 0,
Pop = 0
Data in =   4, Data out =   z, Full = 1, Empty = 1, Push = 0,
Pop = 1
Data in =   4, Data out =   4, Full = 0, Empty = 1, Push = 0,
Pop = 1
Data in =   4, Data out =   3, Full = 0, Empty = 1, Push = 0,
Pop = 1
Data in =   4, Data out =   2, Full = 0, Empty = 1, Push = 0,
Pop = 1
Data in =   4, Data out =   1, Full = 0, Empty = 1, Push = 0,
Pop = 1
Data in =   4, Data out =   1, Full = 0, Empty = 0, Push = 0,
Pop = 1
```

```
20 Stopped at time : 510 ns :  in File "C:/HW4-99109393-project/s
   tack_tb.v" Line 84
```

پس می‌توان نتیجه گرفت که مدار ما برای این تست درست کار می‌کند.
همچنین موج مدار برای این تست نیز در صفحه‌ی بعد آمده است.


0 ns
100 ns
200 ns
300 ns
400 ns
Data_Out[7:0]
XXXXXXXX
ZZZZZZZZ
00000100
00000011
00000010
00000001
Full
Empty
Clk
RstN
Data_In[7:0]
00000000
00000001
00000010
00000011
00000100
Push
Pop