به نام خدا

آزمایش چهارم

آزمایشگاه طراحی سیستم‌های دیجیتال

دانشکده مهندسی کامپیوتر

دانشگاه صنعتی شریف

نویسندگان:

رادین چراغی ۴۰۱۱۰۵۸۱۵

امیرمحمد محفوظی ۴۰۱۱۰۶۴۶۹

سیدعلی جعفری ۴۰۰۱۰۴۸۸۹

تاریخ ارائه تکلیف:

۱۴۰۳/۰۴/۲۵

در این آزمایش هدف ما طراحی یک پشته با عمق ۸ و پهنای ۴ بیت می‌باشد. ورودی و خروجی‌های این پشته در تصویر زیر مشخص شده است.

Inputs:	Clk	Clock signal
	RstN	Reset signal
	Data_In	4-bit data into the stack
	Push	Push Command
	Pop	Pop Command
Outputs:	Data_Out	4-bit output data from stack
	Full	Full=1 indicates that the stack is full
	Empty	Empty=0 indicates that the stack is empty

حال ماژول stack_behavioural را طراحی می‌کنیم. ابتدا با توجه به تصویر بالا ورودی‌ها و خروجی‌ها را تعریف می‌کنیم. سپس آرایه stack_mem را با عمق ۸ و پهنای ۴ تعریف می‌کنیم که همان حافظه‌ی استک می‌باشد. همچنین رجیستر ۴ بیتی stack_pointer را تعریف می‌کنیم و برابر با صفر می‌گذاریم. این رجیستر همواره به اولین خانه‌ی خالی بالای استک اشاره می‌کند. تصویر زیر این عملیات را نشان می‌دهد.

```
module stack_behavioural (
    input clk,
    input rstN,
    input [3:0] data_in,
    input push,
    input pop,
    output reg [3:0] data_out,
    output reg full,
    output reg empty
);

    reg [3:0] stack_mem [7:0];
    reg [3:0] stack_pointer = 0;

    integer i;
```

سپس در بلاک always به بخش ترتیبی مدار می‌پردازیم. لیست حساسیت این بلاک تشکیل شده از لبه‌ی بالارونده clk و لبه‌ی پایین رونده rstN می‌باشد. در صورتی که ریست غیر فعال باشد مدار ریست شده و تمامی حافظه صفر می‌شود همچنین سیگنال‌های دیگر نیز به حالت اولیه خود برمی‌گردند. در غیر این‌صورت دو حالت به وجود خواهد آمد. در صورتی که push فعال باشد و pop فعال نباشد عملیات push را انجام می‌دهیم و به stack_pointer یک واحد اضافه می‌کنیم. همچنین اگر pop فعال باشد و

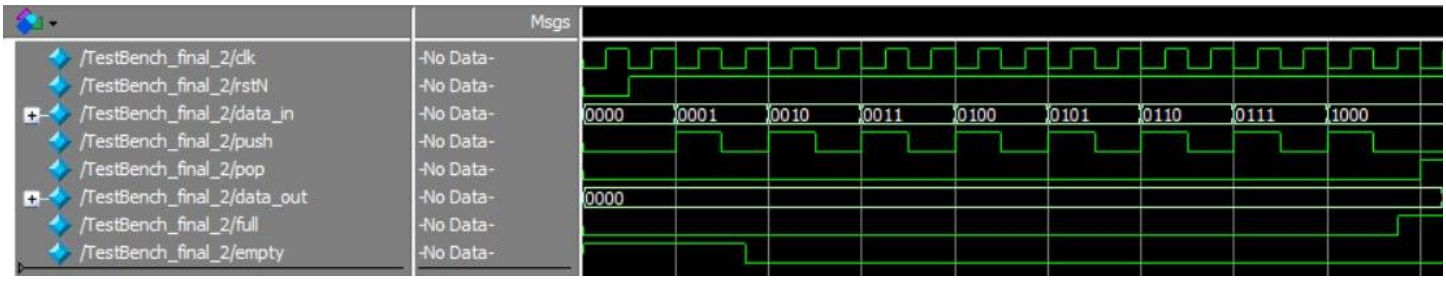
push فعال نباشد، از stack_pointer یک واحد کم می‌کنیم و عملیات pop را انجام می‌دهیم. در نهایت سیگنال‌های full و empty را بر اساس stack_pointer مقدار دهی می‌کنیم. تصویر زیر این عملیات را نشان می‌دهد.

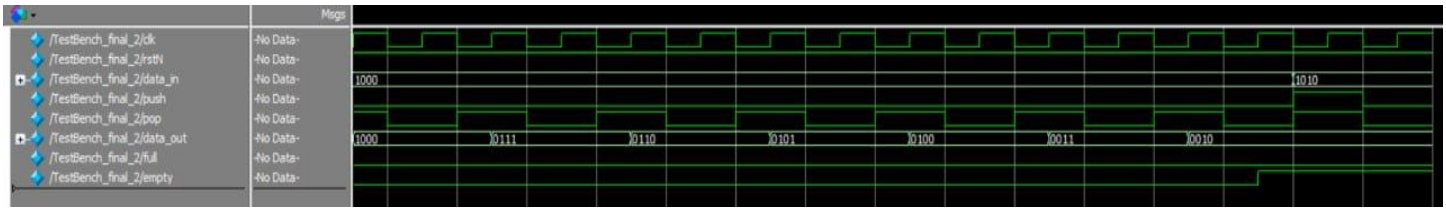
```
    always @(posedge clk or negedge rstN) begin
        if (!rstN) begin
            for (i = 0; i < 8; i = i + 1) begin
                stack_mem[i] <= 0;
            end
            stack_pointer <= 0;
            full <= 0;
            empty <= 1;
            data_out <= 0;
        end else begin
            if (push && !pop && !full) begin
                // Push operation
                stack_mem[stack_pointer] <= data_in;
                stack_pointer <= stack_pointer + 1;
            end else if (pop && !push && !empty) begin
                // Pop operation
                stack_pointer <= stack_pointer - 1;
                data_out <= stack_mem[stack_pointer];
            end

            // Update full and empty signals
            full <= (stack_pointer == 8);
            empty <= (stack_pointer == 0);
        end
    end

endmodule
```

حال tb را برای این مدار طراحی می‌کنیم. پس از تعریف کردن ورودی/ خروجی‌های مدار، سیگنال‌های لازم و نمونه گرفتن از استک تست‌های موجود در تصویر زیر را در این ماژول قرار می‌دهیم.

```
// Test scenarios
initial begin
    // Initialize signals
    rstN = 0;
    data_in = 0;
    push = 0;
    pop = 0;

    // Reset the stack
    #10 rstN = 1;

    // Test push operations
    $display("Testing Push Operations");
    #10 data_in = 4'b0001; push = 1; #10 push = 0;
    #10 data_in = 4'b0010; push = 1; #10 push = 0;
    #10 data_in = 4'b0011; push = 1; #10 push = 0;
    #10 data_in = 4'b0100; push = 1; #10 push = 0;
    #10 data_in = 4'b0101; push = 1; #10 push = 0;
    #10 data_in = 4'b0110; push = 1; #10 push = 0;
    #10 data_in = 4'b0111; push = 1; #10 push = 0;
    #10 data_in = 4'b1000; push = 1; #10 push = 0; // Stack should be full now

    // Test pop operations
    $display("Testing Pop Operations");
    #10 pop = 1; #10 pop = 0;
    #10 pop = 1; #10 pop = 0;
    #10 pop = 1; #10 pop = 0;
    #10 pop = 1; #10 pop = 0;
    #10 pop = 1; #10 pop = 0;
    #10 pop = 1; #10 pop = 0;
    #10 pop = 1; #10 pop = 0;
    #10 pop = 1; #10 pop = 0; // Stack should be empty now

    // Test push and pop simultaneously
    $display("Testing Push and Pop Simultaneously");
    #10 data_in = 4'b1010; push = 1; pop = 1; #10 push = 0; pop = 0;
```

تصاویر زیر خروجی waveform مدار را نشان می‌دهند.





خروجی flow summary مدار در تصویر زیر قابل مشاهده است.

Flow Summary	
Flow Status	Successful - Mon Jul 15 16:19:51 2024
Quartus II 64-Bit Version	13.1.0 Build 162 10/23/2013 SJ Web Edition
Revision Name	stack_behavioural
Top-level Entity Name	stack_behavioural
Family	Cyclone IV GX
Total logic elements	56 / 14,400 (< 1 %)
Total combinational functions	39 / 14,400 (< 1 %)
Dedicated logic registers	42 / 14,400 (< 1 %)
Total registers	42
Total pins	14 / 81 (17 %)
Total virtual pins	0
Total memory bits	0 / 552,960 (0 %)
Embedded Multiplier 9-bit elements	0
Total GXB Receiver Channel PCS	0 / 2 (0 %)
Total GXB Receiver Channel PMA	0 / 2 (0 %)
Total GXB Transmitter Channel PCS	0 / 2 (0 %)
Total GXB Transmitter Channel PMA	0 / 2 (0 %)
Total PLLs	0 / 3 (0 %)
Device	EP4CGX15BF14C6
Timing Models	Final

بخش از خروجی rtl_viewer مدار در تصویر زیر قابل مشاهده است. pdf کامل آن در پیوست آورده شده است.