به نام خدا

آزمایشگاه طراحی سیستم‌های دیجیتال

آزمایش چهارم

طراحی پشته

دانشکده مهندسی کامپیوتر

دانشگاه صنعتی شریف

بهار ۱۴۰۱

استاد:

علیرضا اجلالی

دستیار آموزشی:

سحر رضاقلی

گروه:

هیربد بهنام

۹۹۱۷۱۳۳۳

علی نظری

۹۹۱۰۲۴۰۱

عرفان مجیبی

۹۹۱۰۵۷۰۷

فهرست

مقدمه

در این آزمایش قصد داریم یك پشته با عمق ۸ و پهناي ۴ بیت طراحی کنیم که دارای ورودی‌ها و خروجی‌های زیر باشد :

Inputs:
Clk: Clock signal
RstN: Reset signal
Data_In ۴-bit data into the stack
Push: Push Command
Pop: Pop Command Outputs:
Data_Out: ۴-bit output data from stack
Full: Full=۱ indicates that the stack is full
Empty: Empty=۰ indicates that the stack is empty

گزارش آزمایش

طراحی

عمق پشته برابر ۸ است پس یک آرایه هشت عضوه از رجیسترهای چهاربیتی لازم داریم. همچنین به یک متغیر برای شمارش تعداد اعضا نیاز داریم تا هم بتوانیم عضو جدید به آرایه اضافه کنیم و هم مقادیر Full ,Empty تعیین کنیم.
با ساختار پشته نیز آشنا هستیم و می‌دانیم اگر Push انجام شود، باید یک عضو به اعضا اضافه شود، و اگر Pop انجام شود، باید آخرین عضو استک از آن خارج شود. بر اساس این موارد می‌توانیم پشته را طراحی کنیم:

```
module stack (Clk, RstN, Data_In, Push, Pop, Data_Out, Full, Empty);
    input Clk, RstN, Push, Pop;
    input [3:0] Data_In;
    output reg Full, Empty;
    output reg [3:0] Data_Out;

    reg[3:0] memory[7:0];
    integer index = 0;
    integer i;
    always @(posedge Clk) begin
        if(RstN) begin
            for (i = 0; i < 8 ; i = i + 1) begin
                memory[i] = 0;
            end
            index = 0;
            Empty = 0;
            Full = 0;
        end
        else begin
            if(Push && Pop) begin
            //do nothing
            end
            else if(Push) begin
                if(index <= 7) begin
                    index = index + 1;
                    memory[index - 1] = Data_In;
                end
            end
            else if(Pop) begin
                if(index > 0) begin
                    Data_Out = memory[index - 1];
                    index = index - 1;
                end
            end
```

```
        end
    Full = (index == 8);
    Empty = !(index == 0);
    end
endmodule
```

توجه کنید مقادیر Full, Empty در هر کلاک بر اساس تعداد اعضا که همان index است، مشخص می‌شوند. با فشردن ریست، استک خالی می‌شود و مدار برای کار مجدد آماده می‌شود. همچنین اگر Push و Pop همزمان فشرده شوند، نباید اتفاقی بیفتد.

تست

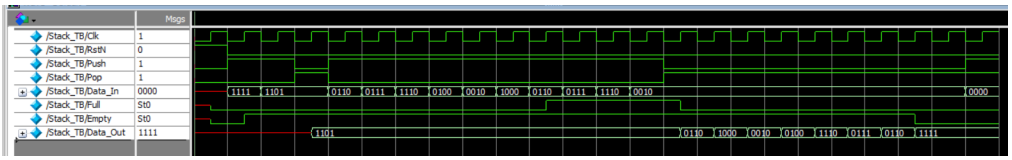
تست را به صورت زیر نوشته‌ایم که حالات مختلف، شامل پوش و پاپ، پر شدن استک و خالی کردن آن را بررسی می‌کند:

```
module Stack_TB;
    reg Clk, RstN, Push, Pop;
    reg [3:0] Data_In;
    wire Full, Empty;
    wire [3:0] Data_Out;
    stack stack0(Clk, RstN, Data_In, Push, Pop,  Data_Out, Full, Empty);

    //clk
    initial begin
        Clk = 1'b0;
    end
    always #5 Clk = ~Clk;

    initial
  begin
    RstN <= 1'b1;
    Push <= 1'b0;
    Pop <= 1'b0;
    #10
    RstN <= 1'b0;
    Data_In <= 4'b1111;
    Push <= 1'b1;
        //push 0110
    #10
    Data_In <= 4'b1101;
    Push <= 1'b1;
        //push 1101
    #10
    Push <= 1'b0;
    Pop <= 1'b1;
        //pop 1101
```

```
    #10
        Pop <= 1'b0;
    Push <= 1'b1;
    Data_In <= 4'b0110;
        //push 0110
    #10
    Push <= 1'b1;
    Data_In <= 4'b0111;
        //push 0111
    #10
    Push <= 1'b1;
    Data_In <= 4'b1110;
        //push 1110
    #10
    Push <= 1'b1;
    Data_In <= 4'b0100;
        //push 0100
    #10
    Push <= 1'b1;
    Data_In <= 4'b0010;
        //push 0010
    #10
    Push <= 1'b1;
    Data_In <= 4'b1000;
        //push 1000
    #10
    Push <= 1'b1;
    Data_In <= 4'b0110;
        //push 0110
    #10
    Push <= 1'b1;
    Data_In <= 4'b0111;
        //push 0111
    #10
    Push <= 1'b1;
    Data_In <= 4'b1110;
        //push 1110
    #10
    Push <= 1'b1;
    Data_In <= 4'b0010;
        //push 0010
    #10
        Push <= 1'b0;
    Pop <= 1'b1;
```

```
76      #10
77      Pop <= 1'b1;
78      #10
79      Pop <= 1'b1;
80      #10
81      Pop <= 1'b1;
82      #10
83      Pop <= 1'b1;
84      #10
85      Pop <= 1'b1;
86      #10
87      Pop <= 1'b1;
88      #10
89      Pop <= 1'b1;
90      #10
91      Pop <= 1'b1;
92      #10
93      Data_In <= 4'b0000;
94      Push <= 1'b1;
95      Pop <= 1'b1;
96              //nothing!
97      #10
98      $stop;
99    end
100 endmodule
```

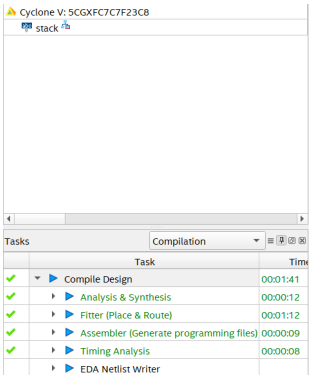
نتیجه اجرای تست به صورت زیر است:



شکل ۱: نتیجه اجرای تست

سنتز کد

این کار را در کوارتوس انجام می‌دهیم:



شکل ۲: سنتز کد

نتیجه‌گیری

در این آزمایش توانستیم با استفاده از وریلاگ، یک پشته به عمق ۸ برای اعداد ۴ بیتی طراحی کنیم.