بسمه تعالی

گزارش کار چهارم آزمایشگاه طراحی سیستم های دیجیتال

**توصیف رفتاری**

استاد: دکتر اجلالی

نویسندگان

مریم شیران 400109446

مهدی بهرامیان 401171593

مزدک تیموریان 401101495

دانشگاه صنعتی شریف  
تابستان 1403

فهرست مطالب

[1 آزمایش چهارم: توصیف رفتاری 1](#_Toc171802704)

[1-1 مقدمه 1](#_Toc171802705)

[2-1 طراحی ماژول 1](#_Toc171802706)

[3-1 کدنویسی ماژول اصلی در وریلاگ 3](#_Toc171802707)

[4-1 کدنویسی ماژول تست در وریلاگ 6](#_Toc171802708)

[5-1 شبیه سازی 10](#_Toc171802709)

[6-1 منابع و مراجع 11](#_Toc171802710)

# آزمایش چهارم: توصیف رفتاری

**Text

Description automatically generated with medium confidence**

## مقدمه

در این آزمایش، می‌خواهیم با استفاده از زبان وریلاگ یک پشته (استک) با ۸ خانه حافظه طراحی کنیم. هر خانه حافظه در این پشته ۴ بیتی است.

## طراحی ماژول

**ماژول طراحی شده لازم است ورودی و خروجی های زیر را دارا باشد:**

**ورودی‌ها:**

**Clk (سیگنال ساعت): این سیگنال زمانی برای هماهنگی عملکرد عناصر مدار در مدارهای همگام استفاده می‌شود.**

**RstN (سیگنال ریست): این سیگنال برای بازنشانی پشته به حالت اولیه‌اش استفاده می‌شود. معمولاً به صورت فعال پایین است، به این معنی که ریست زمانی اتفاق می‌افتد که این سیگنال '0' باشد.**

**Data\_In (داده ورودی 4 بیتی به پشته): این داده‌ای است که به پشته فشار داده می‌شود. عرض آن 4 بیت است.**

**Push (دستور فشار دادن): این سیگنال فرمان نشان می‌دهد که داده باید به پشته فشار داده شود.**

**Pop (دستور خارج کردن): این سیگنال فرمان نشان می‌دهد که داده باید از پشته خارج شود. هر گاه مقدار آن یک بشود و پشته خالی (empty) نباشد، مقدار موجود در data\_out به‌روزرسانی شده و مقدار موجود در خانه‌ای از حافظه استک که stack\_pointer منهای یک به آن اشاره می‌کند ؛ روی data\_out نوشته میشود.**

**خروجی‌ها:**

**Data\_Out (داده خروجی 4 بیتی از پشته): این داده‌ای است که از پشته خارج می‌شود. عرض آن 4 بیت است.**

**Full (پشته پر): هنگامی که این سیگنال '1' باشد، به این معنی است که پشته نمی‌تواند داده‌های بیشتری را قبول کند زیرا پر است.**

**Empty (پشته خالی): هنگامی که این سیگنال '0' باشد، به این معنی است که پشته خالی است و داده‌ای برای خارج کردن وجود ندارد.**

**مواردی که تا کنون گفته شد ورودی ها و خروجی ها بودند. برای طراحی به 2 متغیر دیگر نیز داریم.**

**اشاره‌گر پشته:**

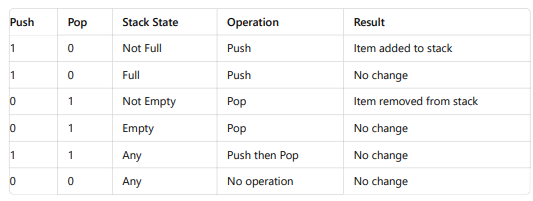
**این متغیر یک رجیستر با عرض ۴ بیت است. علی رغم اینکه پشته در کل ۸ خانه دارد و با ۳ بیت هم می‌توان تمام خانه‌ها را آدرس‌دهی کرد ما نیاز به 4 بیت داریم. چرا که این متغیر همیشه به اولین خانه خالی در حافظه پشته اشاره می‌کند، اگر پشته پر باشد، مقدار این متغیر باید ۸ باشد و در صورت استفاده از رجیستر ۳ بیتی، سرریز رخ داده و مقدار آن نامعتبر می‌شود.**

**حافظه پشته:**

**این متغیر یک آرایه از رجیسترهاست که عرض هر وکتور آن ۴ بیت و عمق آن ۸ درایه است. ذخیره‌سازی مقادیر ورودی به پشته در این آرایه انجام می‌شود.**

## کدنویسی ماژول اصلی در وریلاگ

با توجه به جدول زیر که رفتار مد نظر را توصیف میکند؛ کد مربوطه در وریلاگ به شرح زیر میشود.



سر تا سر کد، به منظور خوانایی و توضیح کامنت گذاری شده است.

module stack\_behavioural (

input clk, // Clock signal

input rstN, // Active-low reset signal

input [3:0] data\_in, // Data input for push operation

input push, // Push operation signal

input pop, // Pop operation signal

output reg [3:0] data\_out, // Data output for pop operation

output reg full, // Flag indicating if the stack is full

output reg empty // Flag indicating if the stack is empty

);

// Define an 8-depth stack memory with 4-bit wide data

reg [3:0] stack\_mem [7:0];

// Stack pointer to keep track of the stack position, points to the last free index in memory

reg [3:0] stack\_pointer;

// Integer variable for loop indexing

integer i;

// Always block triggered on the positive edge of the clock

always @(posedge clk) begin

if (!rstN) begin // If reset signal is active (low)

// Initialize the stack memory to zero

for (i = 0; i < 8; i = i + 1) begin

stack\_mem[i] <= 0;

end

// Reset the stack pointer to zero

stack\_pointer <= 0;

// Set the full flag to 0 (stack is not full)

full <= 0;

// Set the empty flag to 1 (stack is empty)

empty <= 1;

end else begin

// If push signal is active, pop signal is not active, and stack is not full

if (push && !pop && !full) begin

// Push the data into the stack memory at the current stack pointer position

stack\_mem[stack\_pointer] <= data\_in;

// Increment the stack pointer

stack\_pointer <= stack\_pointer + 1;

// Update the full flag: stack is full if stack pointer is 8

full <= (stack\_pointer == 8);

// Update the empty flag: stack is empty if stack pointer is 0

empty <= 0;

end

// If pop signal is active, push signal is not active, and stack is not empty

else if (pop && !push && !empty) begin

// Decrement the stack pointer

stack\_pointer <= stack\_pointer - 1;

// Pop the data from the stack memory at the new stack pointer position

data\_out <= stack\_mem[stack\_pointer];

// Update the full flag: stack is full if stack pointer is 8

full <= 0;

// Update the empty flag: stack is empty if stack pointer is 0

empty <= (stack\_pointer == 0);

end

end

end

endmodule

## کدنویسی ماژول تست در وریلاگ

هم اکنون ماژول تست زیر را نوشته و در ادامه آن را اجرا خواهیم کرد تا از صحت کد خود مطمین شویم.

سر تا سر کد، به منظور خوانایی و توضیح کامنت گذاری شده است.

module testbench;

// Declare testbench signals

reg clk; // Clock signal

reg rstN; // Reset signal (active low)

reg [3:0] data\_in; // 4-bit data input

reg push; // Push control signal

reg pop; // Pop control signal

// Output signals from the stack module

wire [3:0] data\_out; // 4-bit data output

wire full; // Full flag

wire empty; // Empty flag

// Clock period constant

localparam CLK\_PERIOD = 10;

// Generate clock signal with a period of CLK\_PERIOD

always #(CLK\_PERIOD/2) clk = ~clk;

// Instantiate the stack\_behavioural module

stack\_behavioural sb (

.clk(clk),

.rstN(rstN),

.data\_in(data\_in),

.push(push),

.pop(pop),

.data\_out(data\_out),

.full(full),

.empty(empty)

);

// Initial block for setup

initial begin

clk = 0; // Initialize clock signal

push = 0; // Initialize push signal

pop = 0; // Initialize pop signal

rstN = 0; // Initialize reset signal (active low)

data\_in = 4'b0000; // Initialize data input to 0

$dumpfile("testbench.vcd"); // Specify VCD file for waveform dump

$dumpvars(0, testbench); // Dump all variables in the testbench module

end

// Initial block for test sequence

initial begin

rstN = 0; // Assert reset signal to reset the stack

#(CLK\_PERIOD\*2); // Wait for 2 clock cycles

rstN = 1; // Deassert reset signal to start normal operation

pop = 1; // Attempt to pop from empty stack

#(CLK\_PERIOD\*3); // Wait for 3 clock cycles

pop = 0; // Deassert pop signal

push = 1; // Assert push signal

data\_in = 4'b1101; // Load 13 into data input

#(CLK\_PERIOD\*1); // Wait for 1 clock cycle

data\_in = 4'b1111; // Load 15 into data input

#(CLK\_PERIOD\*1); // Wait for 1 clock cycle

data\_in = 4'b0010; // Load 2 into data input

#(CLK\_PERIOD\*1); // Wait for 1 clock cycle

data\_in = 4'b1001; // Load 9 into data input

#(CLK\_PERIOD\*1); // Wait for 1 clock cycle

push = 0; // Deassert push signal

pop = 1; // Assert pop signal

#(CLK\_PERIOD\*1); // Wait for 1 clock cycle

push = 1; // Assert push signal

pop = 0; // Deassert pop signal

data\_in = 4'b0011; // Load 3 into data input

#(CLK\_PERIOD\*1); // Wait for 1 clock cycle

data\_in = 4'b1001; // Load 9 into data input

#(CLK\_PERIOD\*1); // Wait for 1 clock cycle

push = 0; // Deassert push signal

pop = 1; // Assert pop signal

#(CLK\_PERIOD\*8.5); // Wait for 8.5 clock cycles

$finish; // End simulation

end

endmodule

تست بنچ ارائه شده برای ماژول پشته‌ی Verilog به نام stack\_behavioural طراحی شده است تا رفتار پشته را تحت شرایط مختلف شبیه‌سازی کند. در اینجا توضیحی از آنچه تست بنچ انجام می‌دهد به طور دقیق آورده شده است:

ابتدای تست بنچ

تولید سیگنال کلاک:

یک سیگنال کلاک (clk) با دوره‌ی 10 واحد تولید می‌شود که هر 5 واحد (نیم دوره) تغییر می‌کند.

تعریف سیگنال‌ها:

سیگنال‌های مختلف (clk، rstN، data\_in، push، pop، data\_out، full، empty) تعریف می‌شوند.

ماژول پشته stack\_behavioural با استفاده از سیگنال‌های تعریف‌شده نمونه‌سازی می‌شود.

ذخیره‌سازی داده‌های شبیه‌سازی:

داده‌های شبیه‌سازی در فایلی به نام "testbench.vcd" ذخیره می‌شوند تا بعداً با مشاهده‌گرهای موجی مشاهده شوند.

## شبیه سازی

مرحله‌ی بازنشانی:

پشته با تنظیم rstN به 0 بازنشانی می‌شود.

این حالت به مدت دو دوره کلاک نگه داشته می‌شود (#(CLK\_PERIOD\*2)).

شروع شبیه‌سازی:

پس از دو دوره کلاک، rstN به 1 تنظیم می‌شود تا بازنشانی لغو شده و عملیات عادی آغاز شود.

پاپ از پشته‌ی خالی:

سیگنال pop به مدت سه دوره کلاک (#(CLK\_PERIOD\*3)) در حالی که پشته هنوز خالی است به 1 تنظیم می‌شود. این حالت، واکنش پشته را به عملیات پاپ روی پشته‌ی خالی آزمایش می‌کند.

عملیات پوش:

سیگنال push به 1 تنظیم می‌شود تا داده‌ها به پشته فشرده شوند.

مقادیر 13 (4'b1101)، 15 (4'b1111)، 2 (4'b0010)، و 9 (4'b1001) به ترتیب و هر کدام برای یک دوره کلاک به پشته فشرده می‌شوند.

عملیات پاپ:

سیگنال push به 0 تنظیم شده (push = 0) و سیگنال pop به 1 تنظیم می‌شود (pop = 1) تا مقدار بالای پشته پاپ شود. این حالت به مدت یک دوره کلاک اتفاق می‌افتد.

تناوب پوش و پاپ:

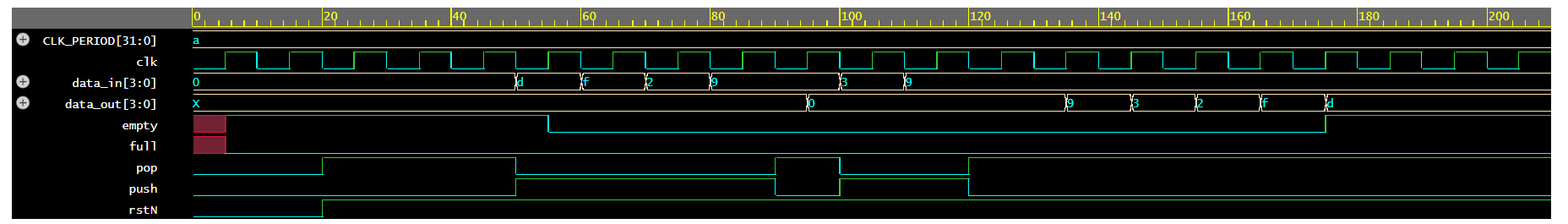
سیگنال push دوباره به 1 تنظیم شده (push = 1) و pop به 0 تنظیم می‌شود (pop = 0) تا مقدار 3 (4'b0011) به پشته فشرده شود.

مقدار دیگری 9 (4'b1001) به پشته فشرده می‌شود.

سیگنال push به 0 تنظیم شده و سیگنال pop به 1 تنظیم می‌شود تا مقدار بالای پشته دوباره پاپ شود.

پایان شبیه‌سازی:

شبیه‌سازی برای 8.5 دوره کلاک دیگر ادامه می‌یابد و سپس با فراخوانی $finish متوقف می‌شود.



در بالا نتیجه نمایش فایل vcd تولید شده از تست‌بنچ را میبینیم که با توضیحات مذکور مطابقت دارد.

## منابع و مراجع

* وبسایت گیکزفورگیکز
* وبسایت یوتیوب
* وبسایت ویکی‌پدیا