به نام خدا

دانشگاه تهران

پردیس دانشکده‌های فنی

دانشکده مهندسی برق و کامپیوتر

طراحی سیستم های نهفته مبتنی بر هسته

تکلیف کامپیوتری ۴

محمد تقی زاده گیوری

۸۱۰۱۹۸۳۷۳

بهار ۱۴۰۲

قسمت اول

ماژول Slave:

```
module Slave(input CS, SCLK, SDI, output SDO, output reg [6 : 0]Red_Leds, output reg [6 : 0]Blue_Leds);

reg [7 : 0] Register [1 : 0];
reg input_sample, address;
reg number_of_got_samples;

always @(negedge SCLK) begin
	if(~CS) begin
		input_sample = SDI;
		address = (number_of_got_samples == 0) ? input_sample : address;
		number_of_got_samples = 1;
	end
end

always @(posedge SCLK) begin
	if(~CS) begin
		Register[address] = {input_sample, Register[address][7 : 1]};
	end
end

always @(negedge CS) begin
	Register[0] = 8'b0;
	Register[1] = 8'b0;
	number_of_got_samples = 0;
end

always @(posedge CS) begin
	Red_Leds = Register[0][7 : 1];
	Blue_Leds = Register[1][7 : 1];
end

endmodule
```

هرگاه CS (Chip Select) از 1 به 0 تغییر کند (لبه پایین رونده سیگنال CS مشاهده شود)، فرایند ارسال داده از سمت Master به Slave آغاز می شود. بنابراین با مشاهده لبه پایین رونده سیگنال CS،

مقداردهی اولیه صورت گرفته و مقادیر موجود در دو رجیستر و تعداد نمونه دریافت شده، صفر می شود تا پس از هر بار ارتباط، مقادیری که از قبل در دو رجیستر هست، پاک شده و بتوان مجدد در این دو رجیستر، مقادیر ارسال شده از سمت Master را ذخیره کرد. برای اینکه در هر بار ارتباط بتوان اولین داده ارسالی از سمت Master را شناسایی کرده و مقدار آن را در address (رجیستر بیان کننده آدرس ثبات مورد نظر) قرار دهیم، نیاز است که در مقداردهی اولیه، number_of_got_samples (تعداد نمونه دریافت شده) را صفر کنیم.

```
always @(negedge CS) begin
	Register[0] = 8'b0;
	Register[1] = 8'b0;
	number_of_got_samples = 0;
end
```

در صورتی که CS صفر باشد (Slave فعال بوده و ارسال داده صورت گیرد)، نیاز است که در هر لبه پایین رونده سیگنال کلاک (SCLK)، داده ارسال شده از سمت Master (SDI) نمونه برداری شده و مقدار آن در رجیستر input_sample ذخیره شود. در صورتی که این، اولین داده ای باشد که از سمت Master به Slave ارسال می شود (به عبارتی اولین نمونه دریافتی بوده و درنتیجه number_of_got_samples صفر باشد)، نیاز است که داده ارسال شده (که همان آدرس ثبات مورد نظر است) در رجیستر address که بیانگر آدرس ثبات مورد نظر است، ذخیره شود. درآخر مقدار number_of_got_samples را 1 کرده تا در دفعات بعد، داده ارسال شده از سمت Master، در address قرار نگیرد و آدرس ثبات مورد نظر به اشتباه تغییر نکند.

```
always @(negedge SCLK) begin
        if(~CS) begin
                input_sample = SDI;
                address = (number_of_got_samples == 0) ? input_sample : address;
                number_of_got_samples = 1;
        end
end
```

در صورتی که CS صفر باشد (Slave فعال بوده و ارسال داده صورت گیرد)، نیاز است که در هر لبه بالا رونده سیگنال کلاک (SCLK)، داده های موجود در ثبات ها یک بیت شیفت یابند تا پس از 8 کلاک، داده ای که در ثبات ها قرار دارد، معادل با همان داده 8 بیتی باشد که Master به Slave ارسال کرده است. برای این منظور، کافیست که input_sample (نمونه ورودی خوانده شده از SDI) را در کنار 7 بیت با ارزش تر ثبات (بیت 7 تا 1) قرار دهیم.

```
always @(posedge SCLK) begin
        if(~CS) begin
                Register[address] = {input_sample, Register[address][7 : 1]};
        end
end
```

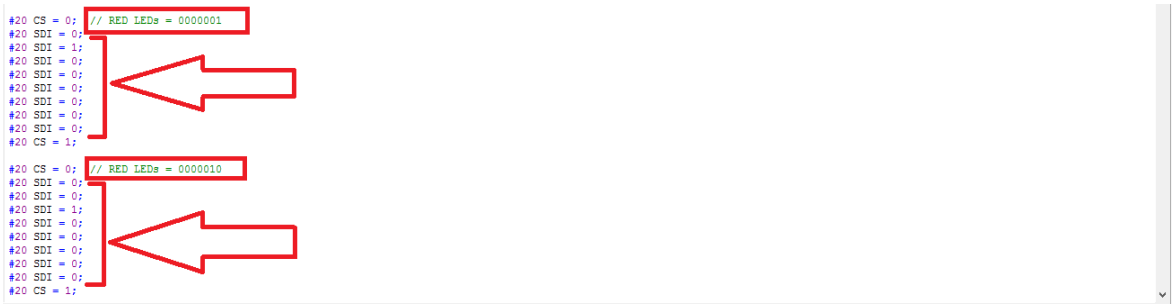
هرگاه CS (Chip Select) از 0 به 1 تغییر کند (لبه بالا رونده سیگنال CS مشاهده شود)، فرایند ارسال داده از سمت Master به Slave خاتمه می یابد. بنابراین با مشاهده لبه بالا رونده سیگنال CS، مقادیر موجود در ثبات ها (7 بیت با ارزش تر) باید به مدارات LED اعمال شوند تا داده ارسال شده از سمت Master به Slave که بیانگر روشن خاموش بودن LED ها بود، به درستی اعمال شده و LED ها مطابق با داده ارسالی از سمت Master، روشن یا خاموش شوند.

```
always @(posedge CS) begin
        Red_Leds = Register[0][7 : 1];
        Blue_Leds = Register[1][7 : 1];
end
```

قسمت دوم

برای هر بار ارتباط با Slave:

- ابتدا CS را صفر می کنیم تا Slave فعال شده و فرآیند ارسال داده به Slave، آغاز شود.
- در کلاک بعدی، آدرس ثبات مورد نظر را روی پورت SDI قرار می دهیم تا آدرس ثبات مورد نظر و اینکه کدام LED ها قرار است روشن یا خاموش شوند، تعیین شود.
- سپس در 7 کلاک، ابتدا بیت های کم ارزش تر (LSB) و سپس بیت های با ارزش تر (MSB) روی پورت SDI قرار داده و به Slave ارسال می شود. (حالت مورد نظر LSB First می باشد)
- درآخر CS را 1 کرده تا فرآیند ارسال داده به Slave خاتمه یابد.



این فرایند را به تعداد ۷ دفعه برای LED های قرمز و ۷ دفعه برای LED های آبی تکرار می کنیم طوری که ابتدا LED های قرمز تک تک روشن و خاموش شوند، و سپس LED های آبی تک تک روشن و خاموش شوند.

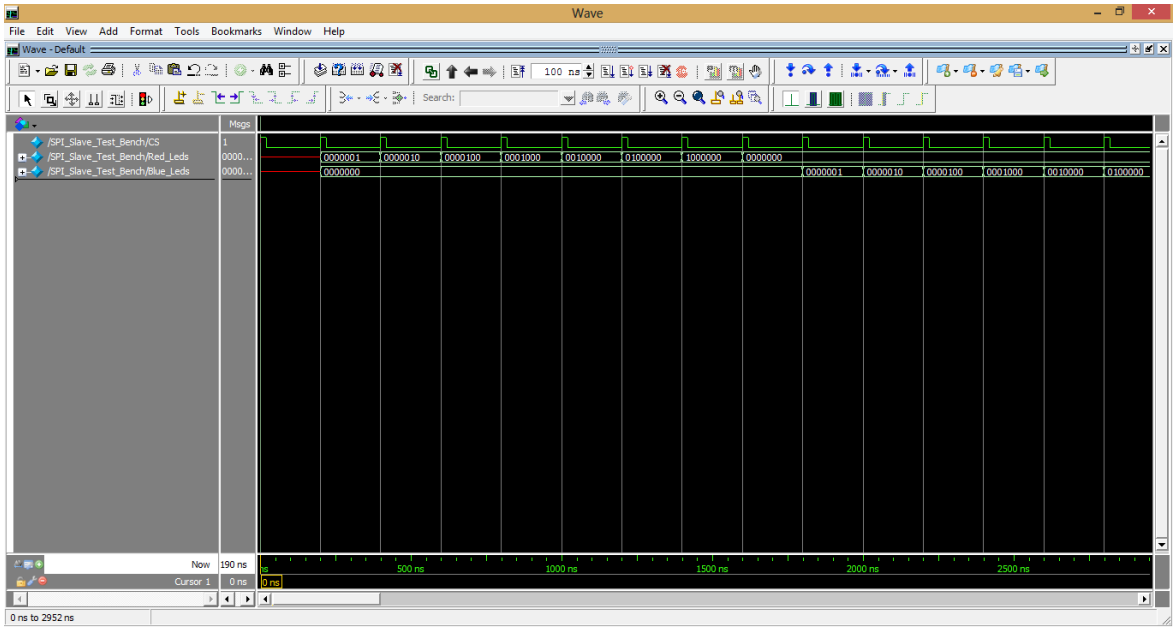
ماژول SPI Slave TestBench:

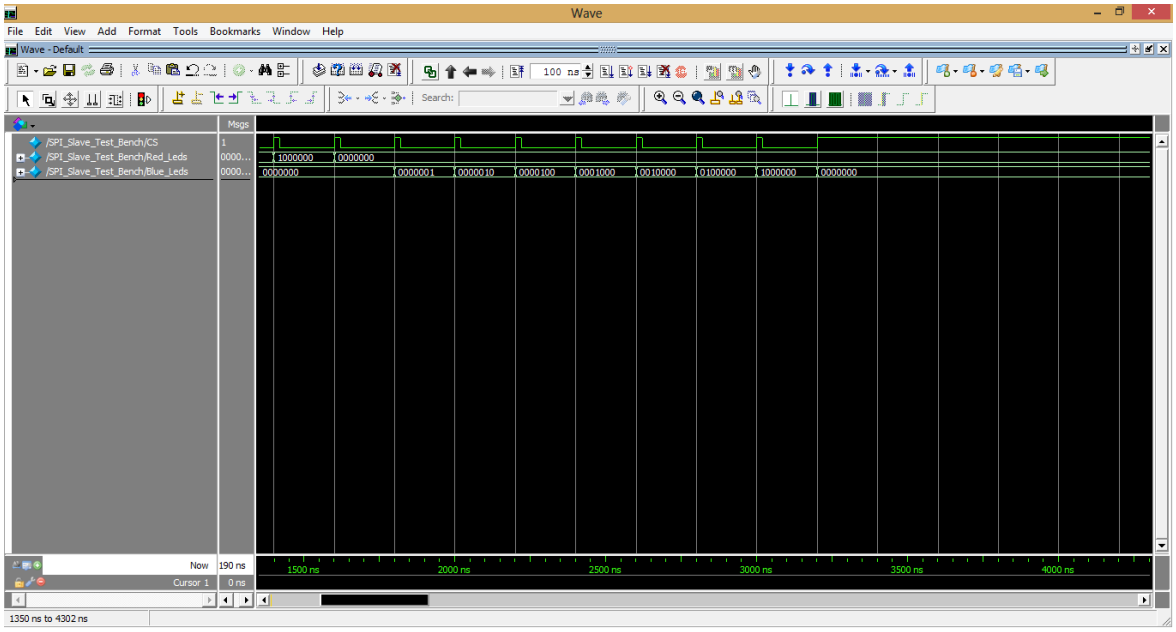
```
`timescale 1ns/1ns

module SPI_Slave_Test_Bench();

reg CS, SCLK, SDI;

wire SDO;
wire [6 : 0]Red_Leds, Blue_Leds;

Slave spi_slave(CS, SCLK, SDI, SDO, Red_Leds, Blue_Leds);

always #10 SCLK = ~SCLK;

initial begin

SCLK = 0; CS = 1;

#20 CS = 0;  // RED LEDs = 0000001
#20 SDI = 0;
#20 SDI = 1;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 CS = 1;

#20 CS = 0;  // RED LEDs = 0000010
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 1;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 CS = 1;
```

```
#20 CS = 0;  // RED LEDs = 0000100
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 1;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 CS = 1;

#20 CS = 0;  // RED LEDs = 0001000
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 1;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 CS = 1;

#20 CS = 0;  // RED LEDs = 0010000
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 1;
#20 SDI = 0;
#20 SDI = 0;
#20 CS = 1;

#20 CS = 0;  // RED LEDs = 0100000
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 1;
#20 SDI = 0;
#20 CS = 1;

#20 CS = 0;  // RED LEDs = 1000000
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 1;
#20 CS = 1;

#20 CS = 0;  // RED LEDs = 0000000
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 CS = 1;

#20 CS = 0;  // Blue LEDs = 0000001
#20 SDI = 1;
#20 SDI = 1;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
```

```
#20 SDI = 0;
#20 SDI = 0;
#20 CS = 1;

#20 CS = 0;  // Blue LEDs = 0000010
#20 SDI = 1;
#20 SDI = 0;
#20 SDI = 1;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 CS = 1;

#20 CS = 0;  // Blue LEDs = 0000100
#20 SDI = 1;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 1;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 CS = 1;

#20 CS = 0;  // Blue LEDs = 0001000
#20 SDI = 1;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 1;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 CS = 1;

#20 CS = 0;  // Blue LEDs = 0010000
#20 SDI = 1;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 1;
#20 SDI = 0;
#20 SDI = 0;
#20 CS = 1;

#20 CS = 0;  // Blue LEDs = 0100000
#20 SDI = 1;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 1;
#20 SDI = 0;
#20 CS = 1;

#20 CS = 0;  // Blue LEDs = 1000000
#20 SDI = 1;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 1;
#20 CS = 1;

#20 SDI = 1;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 1;
#20 SDI = 0;
#20 CS = 1;

#20 CS = 0;  // Blue LEDs = 1000000
#20 SDI = 1;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 1;
#20 CS = 1;

#20 CS = 0;  // Blue LEDs = 0000000
#20 SDI = 1;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 SDI = 0;
#20 CS = 1;

repeat(1000) @(posedge SCLK);
$stop;

end

endmodule
```

قسمت سوم





همان طور که در تصویر فوق مشاهده می کنید، ابتدا LED های قرمز تک تک روشن و خاموش شده، سپس LED های آبی تک تک روشن و خاموش می شوند. هنگامی که LED های قرمز در حال روشن و خاموش شدن هستند، LED های آبی خاموش هستند و همچنین هنگامی که LED های آبی در حال روشن و خاموش شدن هستند، LED های قرمز خاموش می شوند. درنتیجه مدار به درستی کار کرده و داده ها به درستی به Slave ارسال شده و LED ها به درستی روشن و خاموش می شوند.