آزمایشگاه طراحی سیستم‌های دیجیتال

آزمایش شماره ۹: پیاده‌سازی حافظه‌های شرکت‌پذیر نوع سه‌گانه

اعضای گروه:

روژین تقی‌زادگان ۴۰۱۱۰۵۷۷۵
رادین شاه‌دایی ۴۰۱۱۰۶۰۹۶
باربد شهرآبادی ۴۰۱۱۰۶۱۲۵

استاد: دکتر انصاری

در این آزمایش یک ماژول TCAM را به صورت پارامتریک ساخته‌ایم. یعنی طول داده و حجم حافظه را با پارامتر به ماژول داده‌ایم تا بتوان با تغییر این پارامترها، TCAM با عرض و عمق دلخواه داشته باشیم. برای تهیه TCAM طول داده را برابر با ۱۶ می‌گذاریم و چون ۱۶ ثبات داریم، به ۴ خط آدرس احتیاج داریم.

توضیح ماژول TCAM:

```
`define data_size 16
`define address_line 4

module TCAM (input_data, clk, match_address, match_found);

parameter data_size = `data_size;
parameter address_line = `address_line;

input clk;
input [data_size-1:0] input_data;
output reg [address_line-1:0] match_address;
output reg match_found;

reg [data_size-1:0] register_file [(1 << address_line)-1:0];
reg [data_size-1:0] dont_care_register_file [(1 <<
address_line)-1:0];
reg [data_size-1:0] data;
reg [data_size-1:0] mask;
```

این ماژول ۴ ورودی دارد:
۱- داده ورودی که می‌خواهیم خانه حافظه‌ای که با آن match می‌شود را پیدا کنیم. (input)
۲- سیگنال کلاک (input)
۳- آدرس خانه‌ای که با داده ورودی match شده است. (output)
۴- سیگنال found که اگر خانه‌ای از حافظه با داده ورودی match شود، یک و در غیر این صورت صفر می‌شود. (output)

همانطور که در قبل مطرح شد، طول داده و تعداد بیت‌های آدرس به صورت پارامتر مشخص شده‌اند.

دو رجیسترفایل در مدار وجود دارد که رجیستر فایل اصلی شامل مقادیر ذخیره شده در حافظه است و رجیسترفایل don’t care مربوط به بیت‌های X مقادیر ذخیره شده در حافظه است. به این صورت که اگر بیتی در رجیسترفایل اصلی برابر X باشد، بیت متناظر آن در رجیسترفایل don’t care برابر با صفر است و بالعکس.

منطق پیاده‌سازی:

```
integer i, j, match_count;

always @(posedge clk) begin
    match_address = 4'b0;
    match_found = 0;

    for(i = 0; i < (1 << address_line); i = i+1) begin
        if(!match_found) begin
            match_count = 0;
            data = register_file[i];
            mask = dont_care_register_file[i];

            for (j = 0; j < data_size; j = j+1) begin
                if(!mask[j])
                    match_count = match_count+1;

                else if(data[j] == input_data[j])
                    match_count = match_count+1;
            end

            if(match_count == data_size) begin
                match_address = i;
                match_found = 1;
            end
        end
    end
end

endmodule
```

در این بخش ماژول TCAM، منطق کار این ماژول مشخص شده است:
در ابتدا آدرس خانه match شده و سیگنال found برابر با صفر می‌شود.
حلقه for اول روی همه خانه‌های حافظه پیمایش می‌کند. مقدار data برابر با مقدار خانه iام رجیسترفایل اصلی و مقدار mask برابر با مقدار خانه iام رجیسترفایل don’t care است.
حلقه for دوم روی همه بیت‌های خانه iام رجیسترفایل اصلی و رجیسترفایل don’t care پیمایش می‌کند. اگر بیت jام mask برابر با صفر باشد، به این معنی است که این بیت در رجیسترفایل اصلی برابر با X است، پس با داده ورودی مدار match می‌شود و match_count بالا می‌رود. در صورتی که بیت jام mask یک باشد، یعنی این بیت X نیست و باید بیت‌های متناظر در حافظه و داده ورودی با هم چک شوند. در صورتی که این بیت‌ها match شوند، match_count بالا می‌رود.
در نهایت اگر match_count برابر با طول داده ذخیره‌شده در هر خانه حافظه شود، یعنی تمامی بیت‌های داده ورودی و یک خانه حافظه match شده‌اند، بنابراین آدرس خانه حافظه در خروجی match_address قرار می‌گیرد و سیگنال match_found هم برابر با یک می‌شود.
چون پس از یک شدن سیگنال match_found، جستجو در رجیسترفایل متوقف می‌شود، این ماژول در واقع آدرس اولین خانه‌ای که داده ورودی با آن match شده است را برمی‌گرداند.

مقداردهی اولیه:

مقداردهی اولیه را در ماژول TCAM انجام داده‌ایم:

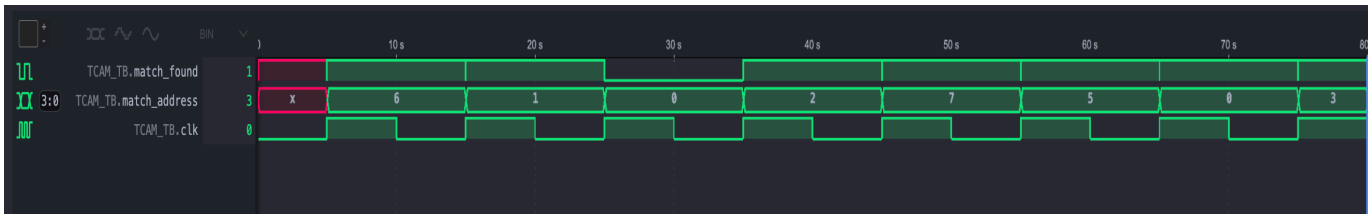
```
initial begin
    register_file[0] = 16'b1010101010101010;
    register_file[1] = 16'b1001001001001001;
    register_file[2] = 16'b1100110011001100;
    register_file[3] = 16'b1110001110001110;
    register_file[4] = 16'b1001110010100011;
    register_file[5] = 16'b0101010101010101;
    register_file[6] = 16'b1111111111111111;
    register_file[7] = 16'b0000000000000000;
    register_file[8] = 16'b1010010110100101;
    register_file[9] = 16'b1001100110011001;
    register_file[10] = 16'b0011110000111100;
    register_file[11] = 16'b1010101001010101;
    register_file[12] = 16'b0011011110110110;
    register_file[13] = 16'b1100111000111100;
    register_file[14] = 16'b1001011001011001;
    register_file[15] = 16'b0011010011010011;

    dont_care_register_file[0] = 16'b1111111111111111;
    dont_care_register_file[1] = 16'b1111111111111111;
    dont_care_register_file[2] = 16'b1100111111001111;
    dont_care_register_file[3] = 16'b1100111111001111;
    dont_care_register_file[4] = 16'b0011110000111100;
    dont_care_register_file[5] = 16'b0011110000111100;
    dont_care_register_file[6] = 16'b1000111110001111;
    dont_care_register_file[7] = 16'b1000111110001111;
    dont_care_register_file[8] = 16'b1111111111111111;
    dont_care_register_file[9] = 16'b1111111111111111;
    dont_care_register_file[10] = 16'b1100111111001111;
    dont_care_register_file[11] = 16'b1100111111001111;
    dont_care_register_file[12] = 16'b0011110000111100;
    dont_care_register_file[13] = 16'b0011110000111100;
    dont_care_register_file[14] = 16'b1000111110001111;
    dont_care_register_file[15] = 16'b1000111110001111;
end
```

Test Bench:

```
`define data_size 16
`define address_line 4

module TCAM_TB ();

parameter data_size = `data_size;
parameter address_line = `address_line;

reg [data_size-1:0] input_data;
reg clk = 0;
wire [address_line-1:0] match_address;
wire match_found;

TCAM tcam1(input_data, clk, match_address, match_found);

always
    #5 clk = ~clk;

initial begin
    input_data = 16'b1111111111111111;
    #10 $display("match address: %d, found: %b", match_address, match_found);

    input_data = 16'b1001001001001001;
    #10 $display("match address: %d, found: %b", match_address, match_found);

    input_data = 16'b0101101001011010;
    #10 $display("match address: %d, found: %b", match_address, match_found);

    input_data = 16'b1111110011001100;
    #10 $display("match address: %d, found: %b", match_address, match_found);

    input_data = 16'b0111000001110000;
    #10 $display("match address: %d, found: %b", match_address, match_found);

    input_data = 16'b1001011010010110;
    #10 $display("match address: %d, found: %b", match_address, match_found);

    input_data = 16'b1010101010101010;
    #10 $display("match address: %d, found: %b", match_address, match_found);

    input_data = 16'b1111001110011110;
    #10 $display("match address: %d, found: %b", match_address, match_found);
    $finish;
end

endmodule
```

```
match address:  6, found: 1
match address:  1, found: 1
match address:  0, found: 0
match address:  2, found: 1
match address:  7, found: 1
match address:  5, found: 1
match address:  0, found: 1
match address:  3, found: 1
```

خروجی test bench



شکل موج test bench