گزارش آزمایش نهم طراحی سیستم‌های دیجیتال

گروه شش

اعضا:

احمد سلیمی

همیلا میلی

درنا دهقانی

شرح آزمایش

در این آزمایش می‌خواهیم که پیمانه ۱۶ ثباتی با ثبات‌های ۱۶ بیتی از حافظه TCAM بسازیم. تفاوت این حافظه‌ها با CAM در این است که در TCAM می‌توان مقادیر ۰، ۱ و X را ذخیره کرد. یعنی مقایسه بیت‌ها هنگامی صورت می‌گیرد که ۰ یا ۱ باشند و X با هر دو انطباق می‌یابد. مثلاً داده‌ی ۱۰۱۰۱۰۱۰ با داده‌های 10XX1010, 1X1X1010, XXXX1010 انطباق می‌یابد.

ماژول‌ها

Comperator

ورودی‌ها:

- A: داده‌ی اول که ۱۶ بیت است.
- B: داده‌ی دوم که ۱۶ بیت است.

خروجی:

- Match: در صورت انطباق دو داده‌ی a و b برابر با ۱ می‌شود.

از یک وکتور ۱۶ بیتی برای نشان دادن انطباق بیت‌های متناظر a و b استفاده می‌کنیم.

در یک generate for، بیت‌های متناظر دو داده‌ی a و b مقایسه می‌شوند و در صورتی که حداقل یکی از این دو برابر با x بود یا برابر بودند، مقدار matches برای این خانه‌ها برابر با ۱ و در غیر این صورت برابر با ۰ می‌شود.

نهایتاً خروجی match برابر با حاصل AND تمام بیت‌های matches می‌شود.

کد وریلاگ این ماژول به شرح زیر است:

```
module comparator (
    input   [15:0]  a,
    input   [15:0]  b,
    output          match
);

wire    [15:0]  matches;

genvar i;
generate
    for(i = 0; i < 16; i = i + 1) begin
        assign matches[i] = a[i] === 1'bx || b[i] === 1'bx || ~(a[i] ^ b[i]);
    end
endgenerate

assign match = &matches;

endmodule
```

Team

ورودی‌ها:

- Clk
- rstN
- we: اگر ۱ باشد، مقدار data در خانه‌ی waddr نوشته می‌شود.
- search: اگر ۱ باشد، به دنبال داده‌ای در حافظه می‌گردیم که با data منطبق باشد.
- waddr: خانه‌ای از حافظه که می‌خواهیم در آن بنویسیم.
- data: مقدار ورودی که یا در حافظه نوشته می‌شود و یا انطباق با آن بررسی می‌شود.

خروجی‌ها:

- sdata: مقدار خانه‌ای از حافظه که با data منطبق است.
- saddr: آدرس خانه‌ای از حافظه که با data منطبق است.
- found: در صورت پیدا شدن انطباق برابر با ۱ می‌شود.

این ماژول اصلی آزمایش است. برای نمایش حافظه از یک آرایه ۱۶تایی به نام mem استفاده می‌کنیم که هر خانه‌ی آن ۱۶ بیت است. در یک وکتور ۱۶ بیتی به نام matches، به ازای هر اندیس اگر آن خانه از حافظه با مقدار ورودی data انطباق داشت، مقدار آن خانه برابر با ۱ می‌شود.

خروجی saddr، اگر ورودی search فعال باشد، مقداری برابر با search_res که در always block مقداردهی می‌شود، می‌گیرد. در غیر این صورت مقدار نامعتبر می‌گیرد.

خروجی sdata، اگر ورودی search فعال باشد، مقداری برابر با مقدار خانه‌ای با اندیس search_res از حافظه می‌گیرد. در غیر این صورت مقدار نامعتبر می‌گیرد.

خروجی found برابر با حاصل OR بیت‌های آرایه matches می‌شود. یعنی اگر حداقل یک خانه از حافظه با ورودی data انطباق یافت، برابر با ۱ می‌شود.

سپس در یک always block که به لبه بالارونده clk و لبه پایین‌رونده rstN حساس است، به پیاده‌سازی حافظه می‌پردازیم. اگر rstN برابر با صفر باشد، تمامی خانه‌های حافظه به صفر مقداردهی می‌کنیم و search_res نیز برابر با صفر می‌شود.

در غیر این صورت اگر ورودی we برابر با ۱ باشد، یعنی باید در خانه‌ای از حافظه نوشته شود. پس مقدار data ورودی در خانه‌ی waddr (که از ورودی گرفته می‌شود.) از mem نوشته می‌شود.

در غیر این صورت، اگر مقدار found و search برابر با ۱ بود، یعنی دنبال داده‌ای از حافظه بگردیم که با data منطبق باشد (search=1) و چنین داده‌ای موجود باشد (found=1)، بر مقادیر موجود در matches پیمایش می‌کنیم تا زمانی که به مقدار ۱ برسیم، یعنی این مقدار خانه با مقدار ورودی منطبق شود. اندیس این خانه را در search_res قرار می‌دهیم. اگر چنین داده‌ای موجود نبود مقدار search_res نامعتبر می‌شود.

کد وریلاگ این ماژول به شرح زیر است:

```
module tcam (
    input           clk,
    input           rstN,
    input           we,
    input   [3:0]   waddr,
    input   [15:0]  data,
    input           search,
    output  [3:0]   saddr,
    output  [15:0]  sdata,
    output          found);

reg     [15:0]  mem [15:0];
wire    [15:0]  matches;
reg     [3:0]   search_res;

assign saddr = search ? search_res : 4'bx;
assign sdata = search ? mem[search_res] : 16'bx;
assign found = |matches;

genvar i;
generate
    for (i = 0; i < 16; i = i + 1)
        comparator cmp(mem[i], data, matches[i]);
endgenerate

integer j, k;
always @(posedge clk or negedge rstN) begin
    if (~rstN) begin
        for (j = 0; j < 16; j = j + 1)
            mem[j]  = 0;
        search_res = 0;
    end
    else if (we)
        mem[waddr] = data;
    else begin
        search_res = 4'bx;
        if (search && found)
            for (k = 0; search_res === 4'bx && k < 16; k = k + 1)
                if (matches[k]) search_res = k;
    end
end
endmodule
```

Tcam_TB

در این ماژول به بررسی کارکرد ماژول tcam می‌پردازیم. کد وریلاگ این ماژول به شرح زیر است:

```
module tcam_TB;

reg             clk = 1, rstN = 0;
reg             we = 0;
reg             [3:0] waddr;
reg             [15:0] data;
reg             search = 0;
wire    [3:0]   saddr;
wire    [15:0]  sdata;
wire            found;

tcam TCAM(clk, rstN, we, waddr, data, search, saddr, sdata, found);

always #5 clk = ~clk;

initial
    begin

    #25 rstN = 1;
    #10 we = 1;
    waddr = 0;
    data = 16'b00000101x1010101;

    #10;
    waddr = 4;
    data = 16'b00000101010x0101;

    #10;
    waddr = 10;
    data = 16'bx0110101010x1x01;

    #10;
    waddr = 13;
    data = 16'b101101010101xxx1;

```

```
    #10;
    $display(" 0:%b\n 1:%b\n 2:%b\n 3:%b\n 4:%b\n 5:%b\n 6:%b\n 7:%b\n 8:%b\n 9:%b\n10:%b\n11:%b\n12:%b\n13:%b\n14:%b\n15:%b\n",
        TCAM.mem[0],
        TCAM.mem[1],
        TCAM.mem[2],
        TCAM.mem[3],
        TCAM.mem[4],
        TCAM.mem[5],
        TCAM.mem[6],
        TCAM.mem[7],
        TCAM.mem[8],
        TCAM.mem[9],
        TCAM.mem[10],
        TCAM.mem[11],
        TCAM.mem[12],
        TCAM.mem[13],
        TCAM.mem[14],
        TCAM.mem[15]);


    #10 we = 0;
    data = 16'b000001010101xx01;
    search = 1;

    #10;
    $display("finding %b: found? %d, mem[%d] = %b", data, found, saddr, sdata);

    data = 16'bx0110101010xx101;

    #10;
    $display("finding %b: found? %d, mem[%d] = %b", data, found, saddr, sdata);

    data = 16'b0011010101111101;

    #10;
    $display("finding %b: found? %d, mem[%d] = %b", data, found, saddr, sdata);
    $stop;
    end
endmodule
```

همانطور که مشاهده می‌شود، ابتدا در خانه‌هایی از حافظه مقادیری ذخیره شده و مانیتور می‌شوند، سپس به بررسی انطباق مقادیر ورودی با مقادیر حافظه می‌پردازیم.

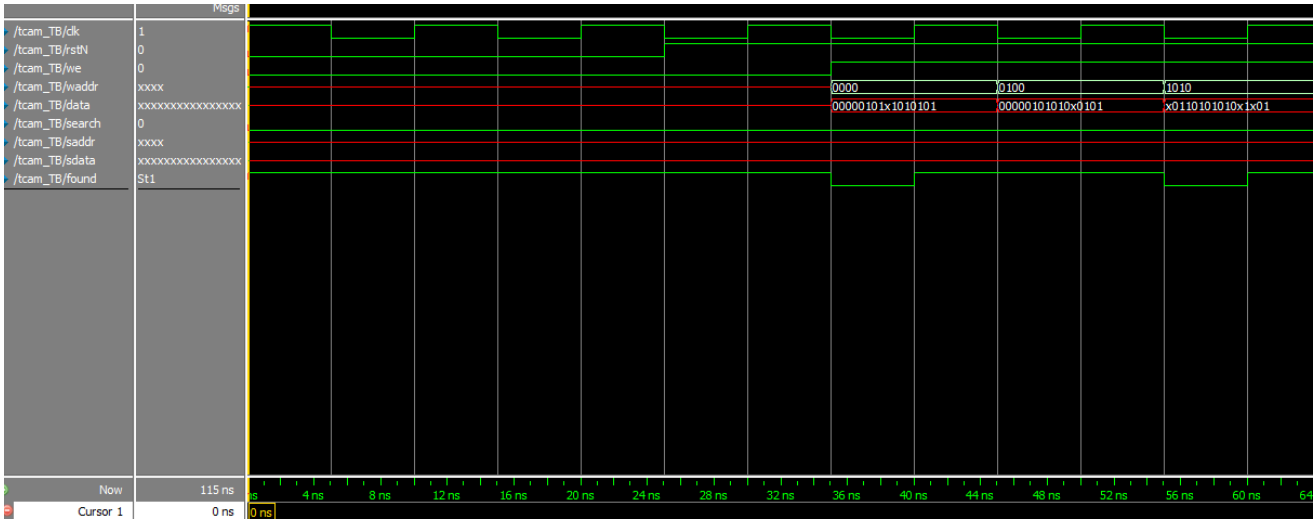
خروجی transcript این test bench به شرح زیر است:

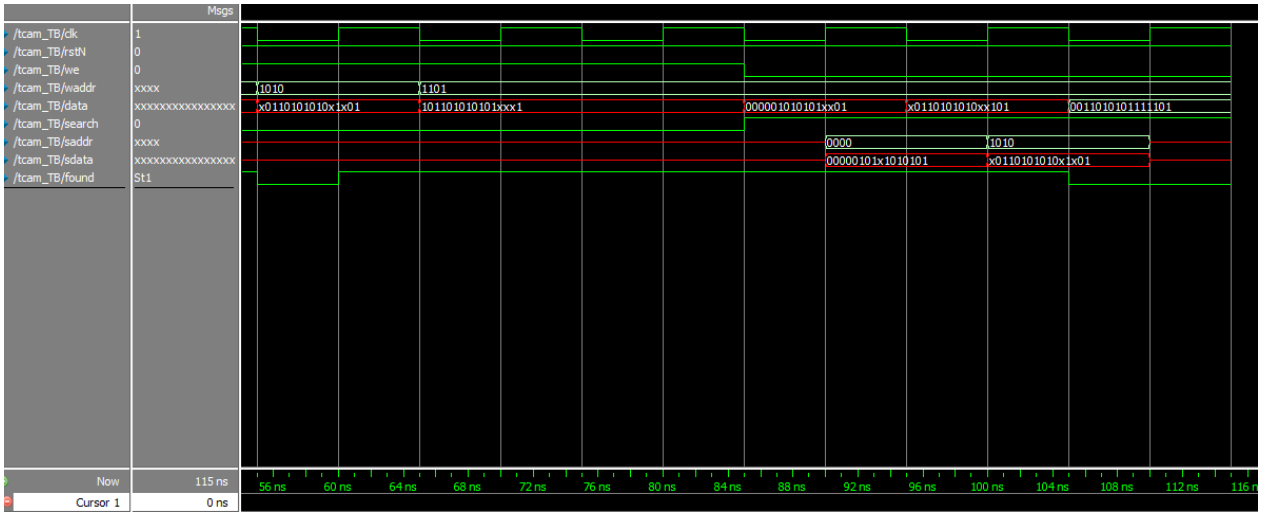
```
VSIM 11> run 2000
#  0:00000101x1010101
#  1:0000000000000000
#  2:0000000000000000
#  3:0000000000000000
#  4:00000101010x0101
#  5:0000000000000000
#  6:0000000000000000
#  7:0000000000000000
#  8:0000000000000000
#  9:0000000000000000
# 10:x0110101010x1x01
# 11:0000000000000000
# 12:0000000000000000
# 13:1011010101011xxx1
# 14:0000000000000000
# 15:0000000000000000
#
# finding 0000010101011xx01: found? 1, mem[ 0] = 00000101x1010101
# finding x0110101010xx101: found? 1, mem[10] = x0110101010x1x01
# finding 0011010101111101: found? 0, mem[ x] = xxxxxxxxxxxxxxxx
# Break in Module tcam_TB at F:/Documents/DSD-az/9/tcam_TB.v line 81
```

مشاهده می‌شود که طبق مقادیر موجود در حافظه، در هنگام جستجو ورودی اول با خانه‌ی اول و ورودی دوم با خانه‌ی یازدهم حافظه منطبق می‌شود و ورودی سوم در حافظه با هیچ داده‌ای منطبق نمی‌شود.

Waveform

حاصل شبیه‌سازی test bench به شکل زیر است:





(چون در ورودی data و مقادیر موجود در حافظه x وجود دارد، توسط modelsim نامعتبر انگاشته شده و به رنگ قرمز به نمایش در می‌آیند.)