گزارش کار آزمایشگاه DSD

آزمایش شماره 9

27 اردیبهشت 1400
عرشیا اخوان
محمدحسین عبدی
علیرضا ایلامی

شماره آزمایش: **9**	موضوع: پیاده سازی حافظه های شرکت پذیر نوع سه گانه	تاریخ آزمایش: 27 اردیبهشت 1400
عرشیا اخوان **97110422**	محمدحسین عبدی **97110285**	علیرضا ایلامی **97101286**

مقدمه: حافظه های شرکت پذیر (Content Addressable Memory) از نوع سه گانه (Ternary) موسوم به TCAM، در بسیاری از کاربردها مانند فشرده سازی و بانک داده ها و سیستم های هوشمند، به کار میروند.

تفاوت TCAM های سه گانه با CAM های عادی در این است که در CAM فقط مقادیر باینری 0 و 1 را میتوان ذخیره کرد. اما در TCAM یک مقدار سوم به نام X داریم که به معنای این است که آن بیت مقایسه نمیشود و برابر در نظر گرفته میشود. به همین دلیل به این نوع CAM ها، سه گانه گفته میشود.
به عنوان مثال، داده 01101110 با XXXX0110 و ،0X1X11X0 هم منطبق است و یک Match در نظر گرفته میشود.

شرح آزمایش:
در این آزمایش یک پترن (Pattern) به عنوان ورودی داده میشود. هدف این است که Maximal Match را پیدا کنیم. منظور از Maximal Match، آن Match ای است که کمترین تعداد X را دارد.

ماژول TCAM:

```
module TCAM (word,
             address,
             rwn,
             clk);
    parameter WORD_LEN = 8;
```

```
    parameter ADDR_LEN = 5;

    inout [ADDR_LEN - 1 : 0] address;
    input [WORD_LEN - 1 : 0] word;
    input rwn ,clk;

    reg [WORD_LEN - 1 : 0] memory [(1 << ADDR_LEN) - 1 : 0];
    reg [ADDR_LEN - 1 : 0] read_candidate;

    assign address = (rwn) ? read_candidate : 'bz;


    reg [ADDR_LEN: 0] i;
    always @ (posedge clk) begin
        if (rwn) begin
            read_candidate = 'bx;
            for (i = 0 ; i < (1 << ADDR_LEN) ; i = i + 1) begin
                if (equals(memory[i] , word)) begin
                    if (read_candidate === 'bx ||
(number_of_x(memory[read_candidate]) > number_of_x(memory[i]))) begin
                        read_candidate = i;
                    end
                end
            end
        end
        else begin
            memory[address] = word;
        end
    end


    // this function get number of x elements in given bit-vector
    function [WORD_LEN - 1 : 0] number_of_x;
        input [WORD_LEN - 1 : 0] c;
        reg [WORD_LEN - 1 : 0] number, j;
        begin
            number = 0;
            for(j = 0 ; j < WORD_LEN ; j = j + 1) begin
                if (c[j] === 1'bx) begin
                    number = number + 1;
                end
            end
```

```
            number_of_x = number;
        end
    endfunction

    // this function checks equality of two bit-vectors with ignoring x's
    function equals;
        input [WORD_LEN - 1 : 0] a,b;
        reg [WORD_LEN - 1 : 0] j;
        reg flag;
        begin
            flag = 1;
            for(j = 0 ; j < WORD_LEN ; j = j + 1) begin
                if (a[j] !== 1'bx && b[j] !== 1'bx && a[j] != b[j]) begin
                    flag = 0;
                end
            end
            equals = flag;
        end
    endfunction
endmodule
```

توضیحات کد:

در ابتدا یک word به عنوان input میگیریم. یک address از نوع in/out داریم. یک سیگنال rwn داریم که در واقع read/write not است. که در هنگام خواندن برابر با 1 میشود.

به همین دلیل آدرس in/out است.

اگر rwn = 0 بود و می خواستیم بنویسیم، کافیست word را در خانه address از memory قرار دهیم.

وقتی rwn = 1 است، استفاده از word به این دلیل است که پترن ورودی را با خانه های مختلف ممووری مقایسه کنیم و در صورتی که تشابه سه گانه برقرار بود، آدرس شروع آن خانه از memory را خروجی دهیم.

همانطور که در خطوط 19 تا 33 مشاهده می شود، یکی یکی روی خانه های memory پیمایش میکنیم و تشابه را بررسی میکنیم. برای این کار، از دو تابع کمکی استفاده میکنیم. یکی تابع number_of_x که تعداد x های یک رشته باینری ورودی را برمی گرداند.

تابع دیگر هم equals است که تساوی دو ورودی (رجیستر) را با یکدیگر مقایسه می کند. به این صورت که X ها را ignore میکند.

حال در always block اصلی، می آییم یکی یکی روی خانه های حافظه iterate میکنیم.

آنهایی که با word مان equal بودند، در صورتی که تعداد x هایش کمتر از تعداد x های کاندید قبلی بود، read_candidate را که اندیس فعلی را نگه میدارد، برابر i قرار میدهیم و آپدیت میکنیم.
به طور خلاصه، میخواهیم اندیس بخشی از عبارت را نگه داریم که اولا با word مان equal باشد و ثانیا کمینه تعداد X را داشته باشد. زیرا این یعنی Maximal Match.

توضیحات in/out address :
در TCAM موقع read کردن، address برابر با out میشود
برای هندل کردن حالت in از طریق تست بنچ مقدار را ست کرده ایم.
در خط 20 تست بنچ، در صورتی که حالت write باشیم، address را برابر out ست میکنیم.
به عبارت دیگر، وقتی در حالت write هستیم، خود TCAM کاری به address ندارد و آن را ignore میکند. و در تست بنچ مقدار آن را برابر با out ست میکنیم.
اما در حالت read، برعکس است. و TCAM مقدار address را برابر in ست میکند و تست بنچ address را ignore میکند.

کد تست بنچ:

Testbench.v

```
`define NULL 0
module testbench();

    parameter WORD_LEN = 16;
    parameter ADDR_LEN = 4;

    parameter clk_c = 10;
    reg rwn, clk;
    reg [WORD_LEN-1:0] word;
    wire [ADDR_LEN-1:0] address;
    reg [ADDR_LEN-1:0] w_addr;

    TCAM #(.WORD_LEN(WORD_LEN), .ADDR_LEN(ADDR_LEN)) tcam (
    .word(word),
```

```
    .address(address),
    .rwn(rwn),
    .clk(clk)
    );

    assign address = (!rwn) ? w_addr : 'bz;

    initial begin
        $dumpfile("report/waveform.vcd");
        $dumpvars(0,tcam);
    end

    initial begin
        clk         = 0;
        forever clk = #(clk_c/2) ~clk;
    end

    initial
        $monitor($time," rwn = %d  word = %b address =
%d",rwn,word,address);

    initial begin
        #clk_c  rwn = 0; word = 16'b0111xx11_0111xx11; w_addr = 0;
        #clk_c  rwn = 0; word = 16'b11x01001_11x01001; w_addr = 1;
        #clk_c  rwn = 0; word = 16'b01110x11_01110x11; w_addr = 2;
        #clk_c  rwn = 0; word = 16'b1101xx11_1101xx11; w_addr = 3;
        #clk_c  rwn = 1; word = 16'b0111xx11_0111xx11;
        #clk_c  rwn = 1; word = 16'bxxxxxxxx_xxxxxxxx;
        #clk_c  rwn = 1; word = 16'b11xxxx11_11xxxx11;
        #clk_c  rwn = 0; word = 16'b11110101_11110101; w_addr = 0;
        #clk_c  rwn = 0; word = 16'b101xxxxx_101xxxxx; w_addr = 1;
        #clk_c  rwn = 0; word = 16'b10110x11_10110x11; w_addr = 2;
        #clk_c  rwn = 0; word = 16'b0100xx01_0100xx01; w_addr = 3;
        #clk_c  rwn = 1; word = 16'b11xx0101_11xx0101;
        #clk_c  rwn = 1; word = 16'b10011011_10011011;
        #clk_c  rwn = 1; word = 16'b10xxxxxx_10xxxxxx;
        #clk_c  $finish;
    end

endmodule
```

نتایج تست بنچ به شرح زیر است:

Result.txt

```
VCD info: dumpfile report/waveform.vcd opened for output.
                0 rwn = x  word = xxxxxxxxxxxxxxxx address =  x
               10 rwn = 0  word = 0111xx110111xx11 address =  0
               20 rwn = 0  word = 11x0100111x01001 address =  1
               30 rwn = 0  word = 01110x1101110x11 address =  2
               40 rwn = 0  word = 1101xx111101xx11 address =  3
               50 rwn = 1  word = 0111xx110111xx11 address =  x
               55 rwn = 1  word = 0111xx110111xx11 address =  2
               60 rwn = 1  word = xxxxxxxxxxxxxxxx address =  2
               65 rwn = 1  word = xxxxxxxxxxxxxxxx address =  1
               70 rwn = 1  word = 11xxxx1111xxxx11 address =  1
               75 rwn = 1  word = 11xxxx1111xxxx11 address =  3
               80 rwn = 0  word = 1111010111110101 address =  0
               90 rwn = 0  word = 101xxxxx101xxxxx address =  1
              100 rwn = 0  word = 10110x1110110x11 address =  2
              110 rwn = 0  word = 0100xx010100xx01 address =  3
              120 rwn = 1  word = 11xx010111xx0101 address =  3
              125 rwn = 1  word = 11xx010111xx0101 address =  0
              130 rwn = 1  word = 1001101110011011 address =  0
              135 rwn = 1  word = 1001101110011011 address =  x
              140 rwn = 1  word = 10xxxxxx10xxxxxx address =  x
              145 rwn = 1  word = 10xxxxxx10xxxxxx address =  2
```

Waveform.png: