آزمایشگاه طراحی سیستم های دیجیتال

پیش گزارش آزمایش نهم

اعضای گروه:

صادق محمدیان: ۴۰۱۱۰۹۴۷۷

متین محمدی: ۴۰۱۱۱۰۳۲۹

امیرحسین ملک محمدی:۴۰۱۱۰۶۵۷۷

شرح آزمایش:

پیاده سازی حافظه های شرکت پذیر نوع سه گانه (TCAM)

میدانیم که در ساخت این نوع از حافظه ها، علاوه بر ۰ و ۱ میتوان x به معنی don’t care را هم ذخیره کرد. بدین صورت یک عدد ذخیره شده در یک واحد از حافظه میتواند با اعداد مختلفی match شود. علاوه بر این تفاوت CAM ها با حافظه های معمول این است که داده به عنوان ورودی داده میشود و در خروجی اولین آدرسی که با این داده مچ شده است خروجی داده میشود.

ورودی :

داده ای که میخواهیم آن را در حافظه جست و جو کنیم. سیگنال کلاک و ریست نیز خواهیم داشت.

خروجی:

سیگنال matchکه نشان می دهد داده ی مورد نظر پیدا شده است یا پیدا نشده است.

عدد چهاربیتی match_addr که نشا ن می دهد داده مورد نظر در کدام رجیستر قرار دارد.

کد زیر طراحی ما برای یک TCAM را نشان میدهد:

```
module TCAM (input [15:0] search_data, input clk, input reset, output reg [3:0] match_addr, output reg match);

    reg [15:0] tcam_memory [0:15];   // TCAM memory array
    reg [15:0] tcam_mask [0:15];     // TCAM mask array (1: compare, 0: don't care)

    integer i;
    reg match_found;

    always @(posedge clk or posedge reset)
    begin
        match_found = 0;

        if (reset)
        begin
        match_addr = 4'b0;  // No match address
        match = 0;              // No match
        tcam_memory[0] = 16'h0000; tcam_mask[0] = 16'hFFFF;
        tcam_memory[1] = 16'h0001; tcam_mask[1] = 16'hFFFF;
        tcam_memory[2] = 16'h1111; tcam_mask[2] = 16'hFFFF;
        tcam_memory[3] = 16'h0011; tcam_mask[3] = 16'hFFFF;
        tcam_memory[4] = 16'h0100; tcam_mask[4] = 16'hFFFF;
        tcam_memory[5] = 16'h0100; tcam_mask[5] = 16'hF0FF;
        tcam_memory[6] = 16'h0110; tcam_mask[6] = 16'hFFFF;
        tcam_memory[7] = 16'h0111; tcam_mask[7] = 16'hFFFF;
        tcam_memory[8] = 16'habcd; tcam_mask[8] = 16'hFFFF;
        tcam_memory[9] = 16'h1234; tcam_mask[9] = 16'hFFFF;
        tcam_memory[10] = 16'h1a1a; tcam_mask[10] = 16'hFFFF;
        tcam_memory[11] = 16'hbcbc; tcam_mask[11] = 16'hFFFF;
        tcam_memory[12] = 16'hbccb; tcam_mask[12] = 16'hFFF0;
        tcam_memory[13] = 16'h1101; tcam_mask[13] = 16'hFFFF;
        tcam_memory[14] = 16'h1110; tcam_mask[14] = 16'hFFFF;
        tcam_memory[15] = 16'haaaa; tcam_mask[15] = 16'hFFFF;

        end
        else
        begin
            match = 0;
            match_addr = 4'b0;  // Default to no match
            for (i = 0; i < 16; i = i + 1)
            begin
                if (!match_found && ((search_data & tcam_mask[i]) == (tcam_memory[i] & tcam_mask[i])))
                begin
                    match_addr = i;
                    match = 1;
                    match_found = 1;
                end
            end
        end
    end


endmodule
```

توضیحات:

این نوع از حافظه ها اکثرا برای پیاده سازی LUT ها مورد استفاده قرار میگیرند، پس مقادیر ذخیره شده در این حافظه ها به صورت Hard-wired میباشد و قابل تغییر نیست. لذا مقادیر ذخیره شده دلخواه خود را با mask های آنها (به ازای هر واحد از حافظه که X میباشد بیت ماسک آنرا 0 میکنیم و اگر mask نیست، بیت مربوط به آنرا ۱ میکنیم:

در صورتی که مدار ما ریست شود مقادیر اولیه زیر در حافظه قرار می گیرند و همچنین match_addr و match برابر با صفر می شوند.

```
if (reset)
begin
match_addr = 4'b0;  // No match address
match = 0;                // No match
tcam_memory[0] = 16'h0000; tcam_mask[0] = 16'hFFFF;
tcam_memory[1] = 16'h0001; tcam_mask[1] = 16'hFFFF;
tcam_memory[2] = 16'h1111; tcam_mask[2] = 16'hFFFF;
tcam_memory[3] = 16'h0011; tcam_mask[3] = 16'hFFFF;
tcam_memory[4] = 16'h0100; tcam_mask[4] = 16'hFFFF;
tcam_memory[5] = 16'h0100; tcam_mask[5] = 16'hF0FF;
tcam_memory[6] = 16'h0110; tcam_mask[6] = 16'hFFFF;
tcam_memory[7] = 16'h0111; tcam_mask[7] = 16'hFFFF;
tcam_memory[8] = 16'habcd; tcam_mask[8] = 16'hFFFF;
tcam_memory[9] = 16'h1234; tcam_mask[9] = 16'hFFFF;
tcam_memory[10] = 16'h1a1a; tcam_mask[10] = 16'hFFFF;
tcam_memory[11] = 16'hbcbc; tcam_mask[11] = 16'hFFFF;
tcam_memory[12] = 16'hbccb; tcam_mask[12] = 16'hFFF0;
tcam_memory[13] = 16'h1101; tcam_mask[13] = 16'hFFFF;
tcam_memory[14] = 16'h1110; tcam_mask[14] = 16'hFFFF;
tcam_memory[15] = 16'haaaa; tcam_mask[15] = 16'hFFFF;
```

برای جست و جو از یک حلقه ی forاستفاده می کنیم که به صورت زیر می باشد:

```
for (i = 0; i < 16; i = i + 1)
begin
    if (!match_found && ((search_data & tcam_mask[i]) == (tcam_memory[i] & tcam_mask[i])))
    begin
        match_addr = i;
        match = 1;
        match_found = 1;
    end
end
```

در صورتی که بیت ماسک مورد نظر ۰ باشد دو داده را برابر فرض می کند و در غیر اینصورت مقدار های متناظر را با یکدیگر مقایسه می کند و اگر برابر بودند مقادیر خروجی را آپدیت می کند.

ماژول تست بنچ:

برای تست کردن مدل، یک فایل test bench مینویسیم و در آن به ازای ورودی های مختلف (اعداد ۱۶ بیتی، خروجی های حافظه TCAM را که شامل یک بیت match (که در صورت مچ شدن یک میشود) و ۴ بیت آدرس برای آدرس دهی ۱۶ خانهٔ حافظه هستند با مقادیر موجود در حافظه تطبیق میدهیم:

```
module TB;

    reg [15:0] search_data;
    reg clk = 0;
    reg reset;
    wire [3:0] match_addr;
    wire match;

    TCAM  uut(search_data, clk, reset, match_addr, match);
   always
    #5 clk = ~clk;

    // Test sequence
    initial begin
        // Apply reset
        reset = 1;
        #10;
        reset = 0;

    search_data = 16'h0000;
    #10
    $display("Search: %h, Match: %b, Match Address: %h", search_data, match, match_addr);

    search_data = 16'h0001;
    #10
    $display("Search: %h, Match: %b, Match Address: %h", search_data, match, match_addr);

    search_data = 16'h1111;
    #10
    $display("Search: %h, Match: %b, Match Address: %h", search_data, match, match_addr);

    search_data = 16'h0011;
    #10
    $display("Search: %h, Match: %b, Match Address: %h", search_data, match, match_addr);
```

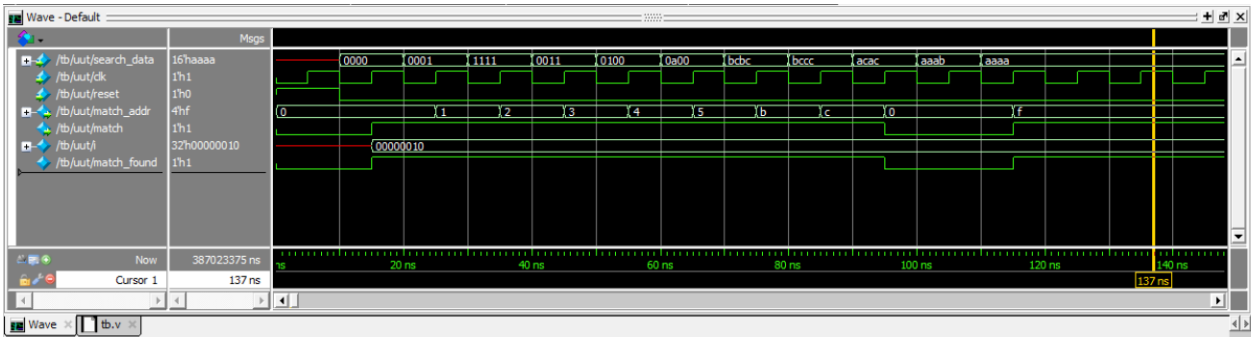
```
35
36      search_data = 16'h0100;
37      #10
38      $display("Search: %h, Match: %b, Match Address: %h", search_data, match, match_addr);
39
40      search_data = 16'h0a00;
41      #10
42      $display("Search: %h, Match: %b, Match Address: %h", search_data, match, match_addr);
43
44      search_data = 16'hbcbc;
45      #10
46      $display("Search: %h, Match: %b, Match Address: %h", search_data, match, match_addr);
47
48      search_data = 16'hbccc;
49      #10
50      $display("Search: %h, Match: %b, Match Address: %h", search_data, match, match_addr);
51
52      search_data = 16'hacac;
53      #10
54      $display("Search: %h, Match: %b, Match Address: %h", search_data, match, match_addr);
55
56      search_data = 16'haaab;
57      #10
58      $display("Search: %h, Match: %b, Match Address: %h", search_data, match, match_addr);
59
60      search_data = 16'haaaa;
61      #10
62      $display("Search: %h, Match: %b, Match Address: %h", search_data, match, match_addr);
63
64      end
65
66  endmodule
```

پس از شبیه سازی ماژول TB، توسط نرم افزار modelsim خروجی به صورت زیر خواهد بود:

```
# Search: 0000, Match: 1, Match Address: 0
# Search: 0001, Match: 1, Match Address: 1
# Search: 1111, Match: 1, Match Address: 2
# Search: 0011, Match: 1, Match Address: 3
# Search: 0100, Match: 1, Match Address: 4
# Search: 0a00, Match: 1, Match Address: 5
# Search: bcbc, Match: 1, Match Address: b
# Search: bccc, Match: 1, Match Address: c
# Search: acac, Match: 0, Match Address: 0
run
# Search: aaab, Match: 0, Match Address: 0
# Search: aaaa, Match: 1, Match Address: f
```

نتایج حاصل نشان می دهد که مدار ما به درستی کار می کند.

همچنین شکل موج متغیر ها به صورت زیر میباشد:



نتیجه گیری

با استفاده از mask، با دوبرابر کردن تعداد واحد های حافظه میتوانیم علاوه بر صفر و یک X را هم به عنوان mask ذخیره کنیم و بدین ترتیب یک TCAM پیاده سازی کنیم.