به نام خدا

آزمایش نهم

آزمایشگاه طراحی سیستم‌های دیجیتال

دانشکده مهندسی کامپیوتر

دانشگاه صنعتی شریف

نویسندگان:

رادین چراغی ۴۰۱۱۰۵۸۱۵

امیرمحمد محفوظی ۴۰۱۱۰۶۴۶۹

سیدعلی جعفری ۴۰۰۱۰۴۸۸۹

تاریخ ارائه تکلیف:

۱۴۰۳/۰۵/۱۵

مقدمه

هدف از انجام این آزمایش طراحی و پیاده‌سازی یک Ternary Content Addressable Memory یا همان TCAM به اندازه ۱۶ ثبات ۱۶ بیتی می‌باشد. حافظه‌های شرکت پذیر سه گانه در بسیاری از کاربردها از جمله فشرده سازی، بانک داده ها، سیستم‌های هوشمند و ... به کار گرفته می‌شوند. تفاوت اصلی TCAM با CAM عادی در این می‌باشد که علاوه بر ۰ و ۱ می‌توانیم مقدار X را نیز در TCAM ذخیره سازی انجام بدهیم که به این معنی است که مقایسه فقط برای محل‌هایی صورت می‌گیرد که ۰ یا ۱ می‌باشند و برای بیت‌های X مقایسه‌ای صورت نمی‌گیرد و بیت‌ها برابر فرض می‌شوند. به عنوان مثال داده ۰۱۱۰۱۱۱۰ اگر مورد جستجو قرار بگیرد با هر کدام از داده‌های روبرو به عنوان انطباق (match) در نظر گرفته می‌شود: 0110XXXX، 0X1X11X0 و X1101XXX.

شرح آزمایش

ماژول TCAM را پیاده‌سازی می‌کنیم. ابتدا به بررسی ورودی‌ها و خروجی‌های این ماژول می‌پردازیم.

ورودی‌ها

- clk: همان کلاک کلی مدار می‌باشد.
- rstN: ریست مدار بوده که فعال پایین (active low) است.
- r_e: سیگنال read_enable بوده که با فعال بودن آن مدار در حالت خواندن و جستجو قرار می‌گیرد.
- w_e: سیگنال write_enable بوده که با فعال بودن آن مدار در حالت نوشتن قرار می‌گیرد.
- data_in: داده‌ای ۱۶ بیتی که به عنوان داده ورودی به مدار جهت جستجو یا نوشتن داده می‌شود.
- Mask: یک ماسک ۱۶ بیتی که مشخص کننده جایگاه بیت‌های don’t care در ورودی data_in می‌باشد. روش مشخص کردن بدین صورت است که در صورتی که بیت iام در mask فعال باشد، بیت iام در data_in برابر با X می‌باشد. در غیر این‌صورت این بیت don’t care نخواهد بود.
- addr_in: آدرس ورودی که محل نوشتن داده در TCAM را مشخص می‌کند.

خروجی‌ها

- matched_num: در صورت اتفاق افتادن انطباق، آخرین داده در حافظه که data_in با آن منطبق می‌شود را مشخص می‌کند.
- match: فعال بودن آن به معنی انطباق data_in با یکی از داده‌های موجود در TCAM است.

ابتدا ماژول را تعریف کرده و ورودی‌ها و خروجی‌های آن را مشخص می‌کنیم. تصویر زیر این عملیات را نشان می‌دهد.

```
module TCAM
(
    input clk, rstN, r_e, w_e,
    input [15:0] data_in, mask, // if mask[i] == 1 then data_in[i] = x
    input [3:0] addr_in,
    output reg [15:0] matched_num,
    output reg match
);
```

حال دو آرایه با با عمق و پهنای ۱۶ تعریف می‌کنیم. آرایه اول (data_mem)، حافظه داده TCAM بوده که داده‌ها بدون توجه به mask خود در آن نوشته می‌شوند. آرایه دوم (mask_mem)، حافظه ماسک مربوط به هر داده بوده که هر درایه آن مشخص کننده جایگاه بیت‌های don’t care در درایه متناظر با آن درایه در data_mem می‌باشد. سپس یک رجیستر ۱۶ بیتی (data_flag) تعریف می‌کنیم که هر بیت آن نشان‌دهنده ولید بودن یا نبودن خانه متناظر با آن بیت در data_mem و mask_mem می‌باشد. به عبارتی دیگر تا زمانی که داده‌ای در یک خانه از حافظه نوشته شده باشد بیت متناظر با آن خانه در data_flag صفر خواهد بود. در نهایت یک integer تعریف می‌کنیم تا بعد برای حلقه for از آن استفاده کنیم. تصویر زیر این عملیات را نشان می‌دهد.

```
reg [15:0] data_mem [0:15];
reg [15:0] mask_mem [0:15];

// Vector for checking data validation
reg [15:0] data_flag;

integer i;
```

حال با استفاده از بلاک always، دسترسی به حافظه را پیاده‌سازی می‌کنیم. لیست حساسیت این بلاک از لبه بالارونده کلاک و لبه پایین رونده ریست تشکیل شده است. در ابتدای این بلاک صفر بودن ورودی rstN بررسی می‌شود. در صورتی که rstN صفر باشد، خروجی‌های matched_num و matched را صفر کرده و همچنین تمام خانه‌های آرایه‌های data_mem و mask_mem و رجیستر data_flag را نیز برابر با صفر می‌کنیم. تصویر زیر این عملیات را نشان می‌دهد.

```
always @(posedge clk or negedge rstN) begin
    if(~rstN) begin
        matched_num = 16'b0;
        match = 1'b0;
        for(i = 0; i <= 15; i = i + 1) begin
            data_mem[i] = 16'b0;
            mask_mem[i] = 16'b0;
            data_flag[i] = 1'b0;
        end
    end
```

در ادامه ورودی w_e که همان فعال‌ساز فرایند نوشتن می‌باشد را بررسی می‌کنیم. در صورتی که سیگنال w_e فعال باشد صرف نظر از ورودی r_e مدار در حالت نوشتن بوده و data_in در خانه حافظه data_mem با آدرس addr_in نوشته می‌شود. سپس ورودی mask نیز در خانه حافظه mask_mem با آدرس addr_in نوشته می‌شود. در نهایت بیت با شماره addr_in در رجیستر data_flag را یک می‌کنیم تا ولید بودن خانه حافظه با آدرس مذکور را نشان بدهیم. تصویر زیر این عملیات را نشان می‌دهد.

```
else if(w_e) begin
    data_mem[addr_in] = data_in;
    mask_mem[addr_in] = mask;
    data_flag[addr_in]  =1'b1;
end
```

در نهایت ورودی r_e که همان فعال‌ساز فرایند جستجو می‌باشد را بررسی می‌کنیم. در صورتی که این ورودی فعال باشد، ابتدا خروجی‌ها را صفر می‌کنیم. سپس با استفاده از یک حلقه for تمام خانه‌های حافظه data_mem را بررسی می‌کنیم. در خانه iام در صورتی که این خانه مقدار مجاز داشته باشد و در صورتی که بیت‌های غیر X در این خانه از حافظه که با mask_mem[i] مشخص می‌شوند با بیت‌های متناظرشان در data_in برابر باشند، match اتفاق افتاده و خروجی matched_num را برابر با این خانه از حافظه می‌کنیم. برای بررسی برابری بیت‌های غیر X در یک خانه از حافظه با بیت‌های متناظرشان در data_in کافی است برابری data_in & ~mask_mem[i] را با data_mem[i] & ~mask_mem[i] بررسی کنیم. تصویر زیر این عملیات را نشان می‌دهد.

```
else if(r_e) begin
    match = 1'b0;
    matched_num = 16'b0;
    for(i = 0; i <= 15; i = i + 1) begin
        if(data_flag[i] && (data_in & ~mask_mem[i]) == (data_mem[i] & ~mask_mem[i])) begin
            matched_num = data_mem[i];
            match = 1'b1;
        end
    end
end
```

حال برای این مدار یک تست بنچ طراحی می‌کنیم. ابتدا سیگنال‌ها و رجیسترهای مورد نیاز برای نمونه‌گیری از ماژول TCAM را تعریف می‌کنیم و سپس با استفاده از بلاک‌های always و initial کلاک مدار را شبیه‌سازی می‌کنیم. تصویر زیر این عملیات را نشان می‌دهد.

```
reg clk, rstN, r_e, w_e;
reg [15:0] data_in, mask;
reg [3:0] addr_in;
wire [15:0] matched_num;
wire match;

TCAM tc(clk, rstN, r_e, w_e, data_in, mask, addr_in, matched_num, match);

initial clk = 1'b0;
always #5 clk = ~clk;
```

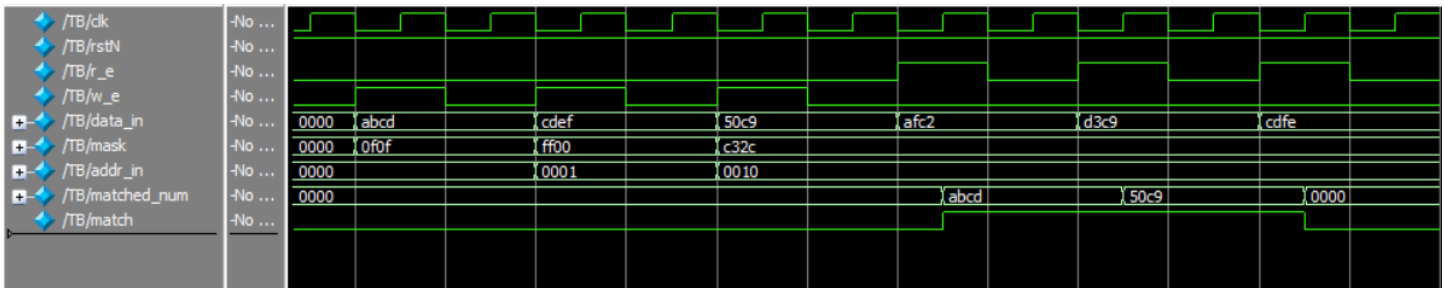
سپس در یک بلاک initial تست‌های مدار را قرار می‌دهیم. ابتدا با فعال نمودن w_e سه داده ABCD با ماسک 0F0F، CDEF با ماسک FF00 و 50C9 با ماسک C32C را به ترتیب در خانه‌های ۰، ۱ و ۲ حافظه می‌نویسیم. سپس به ترتیب ورودی‌های AFC2، D3C9 و CDFE را از به عنوان داده ورودی به TCAM داده، سیگنال r_e را فعال کرده و w_e را غیر فعال می‌کنیم. تصویر زیر این عملیات را نشان می‌دهد.

```
initial begin
    rstN = 1'b0;
    r_e = 1'b0; w_e = 1'b0; data_in = 16'h0000; mask = 16'h0000; addr_in = 4'h0;

    #10 rstN = 1'b1;

    #10 r_e = 1'b0; w_e = 1'b1; data_in = 16'hABCD; mask = 16'h0F0F; addr_in = 4'h0;
    #10 w_e = 1'b0;

    #10 r_e = 1'b0; w_e = 1'b1; data_in = 16'hCDEF; mask = 16'hFF00; addr_in = 4'h1;
    #10 w_e = 1'b0;

    #10 r_e = 1'b0; w_e = 1'b1; data_in = 16'h50C9; mask = 16'hC32C; addr_in = 4'h2;
    #10 w_e = 1'b0;

    #10 r_e = 1'b1; w_e = 1'b0; data_in = 16'hAFC2;
    #10 r_e = 1'b0;
```

```
#10 r_e = 1'b1; w_e = 1'b0; data_in = 16'hAFC2;
#10 r_e = 1'b0;

if (match) begin
    $display("Matched: %h", matched_num);
end else begin
    $display("Not matched!");
end

#10 r_e = 1'b1; w_e = 1'b0; data_in = 16'hD3C9;
#10 r_e = 1'b0;

if (match) begin
    $display("Matched: %h", matched_num);
end else begin
    $display("Not matched!");
end

#10 r_e = 1'b1; w_e = 1'b0; data_in = 16'hCDFE;
#10 r_e = 1'b0;

if (match) begin
    $display("Matched: %h", matched_num);
end else begin
    $display("Not matched!");
end

#10 $stop;
```

حال مدار را با استفاده از نرم افزار ModelSim شبیه‌سازی می‌کنیم. داده اول یعنی AFC2 با ABCD انطباق پیدا خواهد کرد زیرا با توجه به ماسک آن یعنی FF00، داده ABCD در حقیقت 1010XXXX1100XXXX بوده که با AFC2 (1010111111000010) منطبق است. به طور مشابه داده دوم یعنی D3C9 با 50C9 انطباق پیدا می‌کند. اما داده سوم یعنی CDFE با هیچ داده‌ای انطباق پیدا نمی‌کند. نزدیکترین داده به آن CDEF می‌باشد که با توجه ماسک آن یعنی FF00 انطباق اتفاق نمی‌افتد.

تصاویر زیر خروجی waveform و transcript را نشان می‌دهند.




```
# Matched: abcd
# Matched: 50c9
# Not matched!
# ** Note: $stop    : C:/Users/ideapad 5/Desktop/E9/TB.v(55)
#    Time: 140 ps  Iteration: 0  Instance: /TB
```

خروجی **flow summary**

Flow Summary	
Flow Status	Successful - Sat Aug 03 17:29:54 2024
Quartus II 64-Bit Version	13.1.0 Build 162 10/23/2013 SJ Web Edition
Revision Name	TCAM_E9
Top-level Entity Name	TCAM
Family	Cyclone IV GX
Total logic elements	686 / 21,280 (3 %)
Total combinational functions	531 / 21,280 (2 %)
Dedicated logic registers	545 / 21,280 (3 %)
Total registers	545
Total pins	57 / 167 (34 %)
Total virtual pins	0
Total memory bits	0 / 774,144 (0 %)
Embedded Multiplier 9-bit elements	0 / 80 (0 %)
Total GXB Receiver Channel PCS	0 / 4 (0 %)
Total GXB Receiver Channel PMA	0 / 4 (0 %)
Total GXB Transmitter Channel PCS	0 / 4 (0 %)
Total GXB Transmitter Channel PMA	0 / 4 (0 %)
Total PLLs	0 / 4 (0 %)
Device	EP4CGX22CF19C6
Timing Models	Final

خروجی RTL Viewer

تصویر زیر خروجی RTL Viewer مدار را نشان می‌دهد. فایل pdf آن نیز در پیوست آورده شده است.