گزارش کار پروژه امتیازی جبرانی پایانترم

استاد:

دکتر امین فصحتی

نویسنده:

درسا شریفی قمبوانی

401170604

سوال 7 میانترم:

در این سوال از ما خواسته شده تا یک پردازنده‌ی آرایه‌ای با ویژگی‌های زیر را بسازیم:

۱. دارای بخش محاسباتی با قابلیت عملیاتهای ضرب و تقسیم علامت‌دار
۲. دارای بخش رجیسترفایل شامل 4 آرایه به ظرفیت 512 بیت
۳. دارای بخش حافظه به عمق 512 و عرض 32 بیت

حال برای پیاده سازی هر مرحله یک ماژول تعریف می‌کنیم و در مورد آن توضیحات بیشتر ارائه خواهیم داد.

ALU :

```
module alu (input [1:0] opcode, input [511:0] A1, input [511:0] A2,
    output reg [1023:0] dataOut
);

always @(*) begin
    case(opcode)
        2'b00:
            dataOut = $signed(A1 + A2);
        2'b01:
            dataOut = $signed(A1 * A2);
    endcase

end
endmodule
```

در این بخش باید عملیات‌های ضرب و جمع اعداد علامت‌دار را هندل کنیم. چون هر یک از رجیسترها 512 بیت هستند پس جواب نهایی در حداکثر حالت خود 1024 خواهد بود و dataOut مطابق کد بالا تعریف می‌شود. حال برحسب نوع opcode که در تست بنچ تغییر می‌کند، نوع عملیات مشخص شده و با استفاده از سینتکس $signed اطمینان حاصل می‌کنیم که عملیاتهای محاسباتی در محدوده اعداد صحیح انجام شوند.

Register file:

در این بخش باید همزمان عملیات‌های خواندن از آرایه‌ها و نوشتن روی آنها را هندل کنیم. برای اینکار اگر سیگنال reset فعال بود، مقدار تمام رجیسترها صفر می‌شود. در غیراینصورت اگر سیگنال writeFlag فعال بود، آنگاه باید کد داده شده تحت عنوان addr1 را دیکود کنیم. به ازای هر یک از مقادیر باینری 00 الی 11 به ترتیب داخل رجیسترهای A1 الی A4 داده را ذخیره می‌کنیم. از طرفی به ازای هر تغییری، این دیکود به ازای addr1, addr2 انجام شده و مقادیر آنها در دو رجیستر که خروجی مدار هستند ذخیره خواهند شد. توجه داشته باشید که ما در بخش ALU باید دو رجیستر A1, A2 را به عنوان ورودی

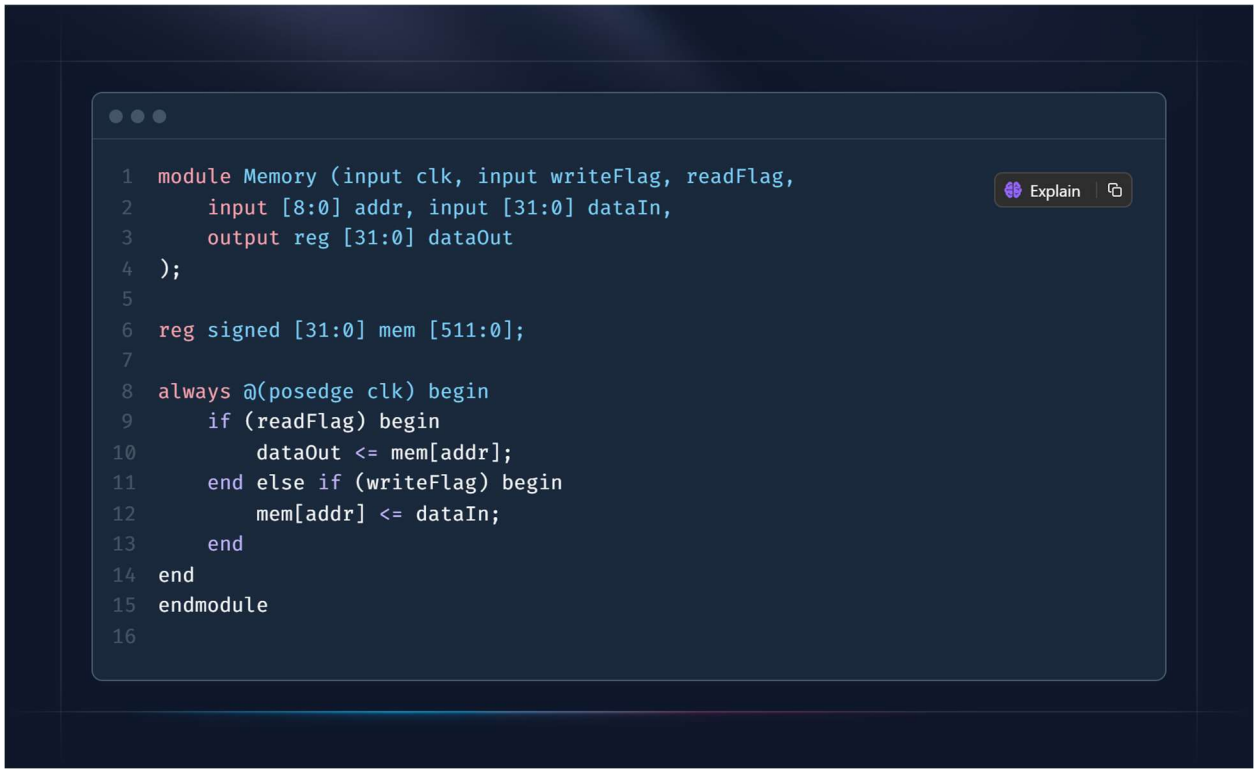
از کاربر بگیریم و در A3, A4 خروجی دهیم. پس این بخش از مدار ما دو ورودی نیاز دارد. این دو ورودی توسط خروجی بخش رجیسترفایل مقداردهی می‌شوند.

```
module registerFile (
    input clk, reset, writeFlag, input [511:0] dataIn,
    input [1:0] addr1, addr2, output reg [511:0] dataOut1, dataOut2
);
reg signed[511:0] A1, A2,A3,A4;


always @(*) begin
    case (addr1)
        2'b00: dataOut1 <= A1;
        2'b01: dataOut1 <= A2;
        2'b10: dataOut1 <= A3;
        2'b11: dataOut1 <= A4;
    endcase
    case (addr2)
        2'b00: dataOut2 <= A1;
        2'b01: dataOut2 <= A2;
        2'b10: dataOut2 <= A3;
        2'b11: dataOut2 <= A4;
    endcase
end

always @(posedge clk, posedge reset) begin
    if (reset == 1) begin
        A1 <= 512'b0; A2 <= 512'b0;
        A3 <= 512'b0; A4 <= 512'b0;
    end else if (writeFlag) begin
        case (addr1)
            2'b00: A1 <= dataIn;
            2'b01: A2 <= dataIn;
            2'b10: A3 <= dataIn;
            2'b11: A4 <= dataIn;
        endcase
    end
end
endmodule

```

Memory:

در این بخش باید هم بتوانیم در حافظه بنویسیم هم از آن بخوانیم. پس دو سیگنال ورودی readFlag و writeFlag احتیاج داریم. از طرفی طبق صورت سوال باید یک مموری به عمق 512 بیت و عرض 32 بیت طراحی کنیم که در داخل ماژول

اینکار را انجام دادیم. حال در لبه بالارونده ساعت، هرگاه که سیگنال readFlag فعال شد، از حافظه در آدرس داده شده می‌خوانیم و در رجیستر خروجی dataOut قرار می‌دهیم. هرگاه که سیگنال writeFlag فعال شد، در آدرس داده شده مقدار ورودی dataIn را قرار می‌دهیم.



Processor:

در این بخش باید از 3 بخش بالا که اجزای مدار را تشکیل می‌دهند، نمونه‌گیری کنیم و سیم‌های بین تمام بخش‌ها را به شکل صحیح به هم متصل کنیم. نکته‌ی قابل توجه در این بخش آن است که ورودی بخش رجیستر همواره توسط خروجی مموری مقداردهی خواهد شد. در اینصورت اگر خروجی مموری تغییر کند، یعنی از مموری می‌خواهیم داده‌ای را در رجیسترفایل ذخیره کنیم که assignment اشاره شده اینکار را انجام می‌دهد.

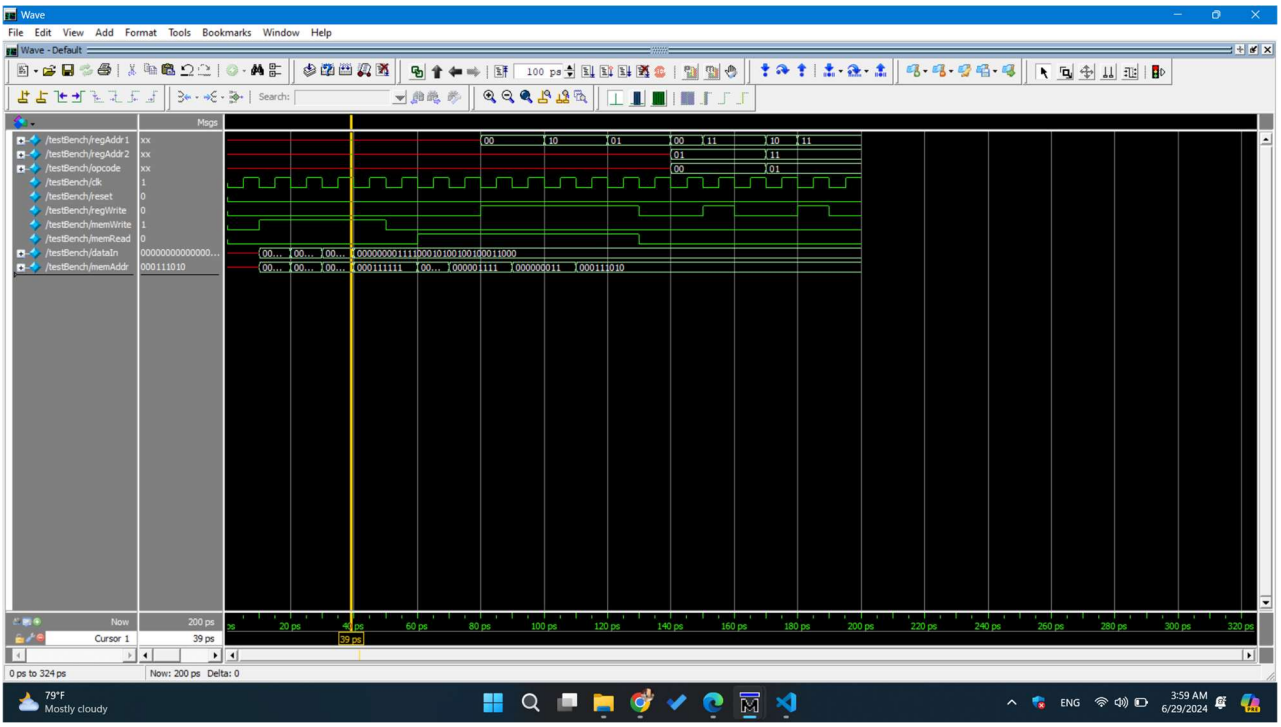
```
module processor (input clk,reset,memRead,regWrite, memWrite,
    input [8:0] memAddr,
    input [1:0] regAddr1,
    input [1:0] regAddr2,
    input [1:0] opcode,
    input [31:0] dataIn, output [31:0] dataOut,
    output reg signed [1023:0] answerOfAlu
);

wire signed [511:0] dataOut1, dataOut2;
wire signed [1023:0] answerOfALU;
reg signed [511:0] registerDataIn;

assign answerOfAlu = answerOfALU;
assign registerDataIn = dataOut;
alu myAlu (opcode,dataOut1,dataOut2, answerOfALU);

registerFile regFile (clk, reset, regWrite, registerDataIn,regAddr1, regAddr2,
    dataOut1, dataOut2);

Memory mem (clk, memWrite, memRead, memAddr, dataIn, dataOut);


endmodule

```

TestBench:

برای اطمینان از صحت کارکرد مدار، تست بنچ زیر را ساختیم تا حالات مطلوب را نمایش دهد. در اینجا ابتدا 4 مقدار متفاوت در آدرس‌های متفاوتی از حافظه باید ذخیره شوند. پس باید سیگنال memWrite به طرز صحیح مقداردهی شود. پس از اینکار، مقادیر ذخیره شده را از حافظه می‌خوانیم و در رجیسترفایل ذخیره می‌کنیم. اینکار را در رجیسترهای 1 الی 3 انجام می‌دهیم. در نهایت روی رجیستر 1و2 عملیات جمع و ضرب را انجام می‌دهیم. اینگونه تمام حالات مطلوب مدار در تست بنچ خواسته شده هندل می‌شوند.

```
module testBench;
    reg [1:0] regAddr1, regAddr2, opcode;
    reg clk, reset, regWrite, memWrite, memRead;
    reg [31:0] dataIn;
    wire [31:0] dataOut;
    reg [8:0] memAddr;
    wire [1023:0] answerOfALU;

    processor pr (clk, reset,memRead, regWrite, memWrite,memAddr, regAddr1, regAddr2,opcode, dataIn, dataOut, answerOfALU);
    always #5 clk = ~clk;

    initial begin
        clk = 0;
        reset = 0;
        regWrite = 0;
        memWrite = 0;
        memRead = 0;
    end

    initial begin
        //Test1 : saving data inside memory
        #10;
        memWrite = 1;
        memAddr = 9'b000000011;
        dataIn = 32'h000000ba;
        #10;
        memWrite = 1;
        memAddr = 9'b000001111;
        dataIn = 32'h00000645;
        #10;
        memWrite = 1;
        memAddr = 9'b000111010;
        dataIn = 32'h00002918;
        #10;
        memWrite = 1;
        memAddr = 9'b000111111;
        dataIn = 32'h000F14918;
        #10;
        memWrite = 0;
        //Test2 : reading data From memory and transfering to regFile
        #10;
        memRead = 1;
        memAddr = 9'b01101001;
        #10;
        memRead = 1;
        memAddr = 9'b000001111;
        #10;
        regWrite = 1;
        regAddr1 = 2'b00;
        #10;
        memRead = 1;
        memAddr = 9'b000000011;
        #10;
        regWrite = 1;
        regAddr1 = 2'b10;
        #10;
        memRead = 1;
        memAddr = 9'b000111010;
        #10;
        regAddr1 = 2'b01;
        #10;
        regWrite = 0;
        memRead = 0;
        // Test 3 : Addition
        #10;
        regAddr1 = 2'b00;
        regAddr2 = 2'b01;
        opcode = 2'b00;
        #10;
        regWrite = 1;
        regAddr1 = 2'b11;
        #10;
        regWrite = 0;
```

حال مدار خود را شبیه‌سازی میکنیم :



waveform بدست آمده به شکل بالا خواهد بود و خروجی پرینت شده نیز مطابق عکس زیر است:

مشاهده می‌کنیم در ابتدا مقادیر مختلفی در آدرس‌های متفاوتی از مموری ذخیره می‌شوند. سپس، مقادیر ذخیره شده در آدرس 000001111 و 000111010 در A1, A2 قرار می‌گیرند و باید با هم جمع شوند. مقادیر این دو به ترتیب 645 و 2918 می‌باشد که در اثر جمع باید خروجی 2f5d دهند که این خروجی را در زمان 140 به عنوان خروجی alu مشاهده می‌کنیم. حال در مرحله بعد این مقدار در A4 ذخیره می‌شود. برای تست ضرب نیز مقدار ذخیره شده در A3 در A4 ضرب می‌شود. یعنی باید ba را که قبلا از طریق مموری به A3 منتقل کردیم در جواب مرحله قبل یعنی 2f5d ضرب کنیم و جواب باید 1ddb70 شود که در زمان 170 آن را مشاهده می‌کنیم. در نتیجه مدار ما درست است.