دانشکده کامپیوتر

تمرین سری اول

دکتر میثم عبدالهی

الناز رضایی ۹۸۴۱۱۳۸۷

فروردین ۱۴۰۱

سوال ۱ :

در این مسئله قصد داریم با بهم‌بندی ماژول‌های ساخته‌شده، یک ALU بسیار ساده با قابلیت انجام چند عملیات پردازشی محدود را انجام دهد. در ابتدا ماژول‌های زیر را طراحی و پیاده‌سازی کنید:

- ماژولی طراحی کنید که با دریافت دو عدد صحیح، حاصل‌جمع این دو را خروجی دهد.
- ماژولی طراحی کنید که با دریافت دو عدد صحیح، حاصل تفریق عدد اول از عدد دوم را خروجی دهد.
- ماژولی طراحی کنید که با دریافت دو عدد صحیح، حاصل ضرب آن دو را خروجی دهد.
- ماژولی طراحی کنید که با دریافت دو عدد صحیح، حاصل تقسیم عدد اول بر عدد دوم را خروجی دهد.
- ماژولی طراحی کنید که با دریافت دو عدد صحیح، ب.م.م آن دو را خروجی دهد.
- ماژولی طراحی کنید که با دریافت دو عدد صحیح، حاصل عدد اول به توان عدد دوم را خروجی دهد.

در ادامه یک مالتیپلکسر ۳*۸ پیاده سازی کنید. توجه فرمایید که این مالتیپلکسر باید با استفاده از روش behavioral پیاده‌سازی شده باشد نه با استفاده از گیت‌های منطقی.

در نهایت با توجه به جدول زیر، ALU را طوری پیاده‌سازی کنید که با دریافت ۲ عدد صحیح و ۳ bit به‌عنوان عملیاتselector مناسب را انجام دهد:

عملیات	selector
جمع	۰۰۰
تفریق	۰۰۱
ضرب	۰۱۰
تقسیم	۰۱۱
ب.م.م	۱۰۰
توان	۱۰۱

توجه فرمایید که اعداد ورودی و خروجی نمایش بیتی ندارند و اعداد صحیح هستند و این سوال صرفا یک شبیه‌سازی است.

پاسخ ۱:

در ابتدا توابع مورد نظر را پیاده‌سازی می‌کنیم.برای چهار عمل اصلی از توابع آماده خود زبان وریلاگ استفاده می‌کنیم. برای ساخت تابع ب.م.م یک ماژول جدبد پیاده سازی می‌کنیم. طبق الگوریتم گفته شده در دبیرستان، مقدارش را به صورت behavourial محاسبه می‌کنیم. سپس با استفاده از سوییچ کیس، یک مالتیپلکسر طراحی نموده و به کمک آن ماژول‌ها را به هم وصل می‌کنیم.

```
module ALU (input [31:0] inp1,input [31:0] inp2,input [2:0] select,output reg [31:0] res);

reg [31:0] tmp1;
reg [31:0] tmp2;
reg [31:0] swap;
reg done ;


wire [31:0] gcd ;
GCD g0 (inp1,inp2,gcd);
always @(*)
begin
        case(select)
                3'b000 : res <= inp1+inp2;
                3'b001 : res <= inp2-inp1;
                3'b010 : res <= inp1*inp2;
                3'b011 : res <= inp1/inp2;
                3'b100 : res <=gcd;
                3'b101 : res <=inp1**inp2 ;
                default : res <= inp1+inp2 ;
                endcase

end

endmodule
```

```
module GCD (
input [31:0] a, b,
output reg [31:0] res
);
reg [31:0] A, B, swap;
integer done;
always @(*)
begin
done = 0;
A = a; B = b;
while ( !done )
begin
if ( A < B )
begin
swap = A;
A = B;
B = swap;
end
else if ( B != 0 )
A = A - B;
else
done = 1;
end

res = A;
end



endmodule
```

```
module ALU_tb;
reg [31:0] a;
reg [31:0] b;
wire [31:0] res;
reg [2:0] s;



localparam period = 20;



ALU a0(a, b,  s,res);



initial
begin

a = 8;
b = 4;
s = 0;
#period;



a = 4;
b = 8;
s = 1;
#period;
```

```
a = 8;
b = 4;
s = 2;
#period;



a = 8;
b = 4;
s = 3;
#period;



a = 8;
b = 4;
s = 4;
#period;



a = 8;
b = 4;
s = 5;
#period;



end

initial
$monitor("INPUT VALUES: \t a = %b b = %b s = %b \t OUTPUT VALUE: \t res = %b", a, b, s, res);

endmodule
```

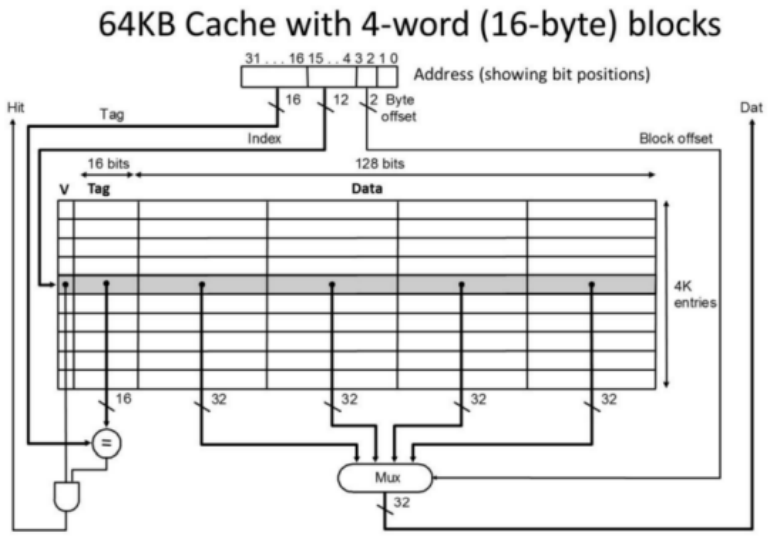
خروجی:

```
INPUT VALUES:    a = 00000000000000000000000000001000 b = 00000000000000000000000000000100 s = 000       OUTPUT VALUE:   res = 00000000000000000000000000001100
INPUT VALUES:    a = 00000000000000000000000000000100 b = 00000000000000000000000000001000 s = 001       OUTPUT VALUE:   res = 00000000000000000000000000000100
INPUT VALUES:    a = 00000000000000000000000000001000 b = 00000000000000000000000000000100 s = 010       OUTPUT VALUE:   res = 00000000000000000000000001000000
INPUT VALUES:    a = 00000000000000000000000000001000 b = 00000000000000000000000000000100 s = 011       OUTPUT VALUE:   res = 00000000000000000000000000000010
INPUT VALUES:    a = 00000000000000000000000000001000 b = 00000000000000000000000000000100 s = 100       OUTPUT VALUE:   res = 00000000000000000000000000000100
INPUT VALUES:    a = 00000000000000000000000000001000 b = 00000000000000000000000000000100 s = 101       OUTPUT VALUE:   res = 00000000000000000001000000000000
```

سوال ۲ :

یک حافظه Cache و یک Cache Controller را با توجه به نکات زیر پیاده سازی نمایید.

- اندازه حافظه Cache برابر ۶۴ کیلوبایت خواهد بود.
- هر Block حافظه ۴ word را ذخیره میکند. (هر word ۳۲ بیت را در خود جای میدهد.)
- حافظه Cache دارای valid bit میباشد تا نشان دهد داده درون یک Block معتبر است یا خیر.
- Address bus حافظه اصلی ۳۲ بیت است و Data Bus حافظه اصلی ۱ Word میباشد.
- به ازای هر عملیات Read از Cache میبایست مرتبه Hit Signal را تعیین نمایید. (سیگنال hit یک سیگنال خروجی است و وقتی داده درون Cache پیدا شد این سیگنال باید ۱ شود و در غیر این صورت ۰)



پاسخ ۲:

در ابتدا یک cache ۱۴۴ تایی تعریف می‌کنیم. سپس ۱۶ بیت برای بخش tag و ۱۲ بیت برای index در نظر می‌گیریم. به علاوه به ۲ بیت برای offset byte و ۲ بیت برای offset block نیاز داریم.

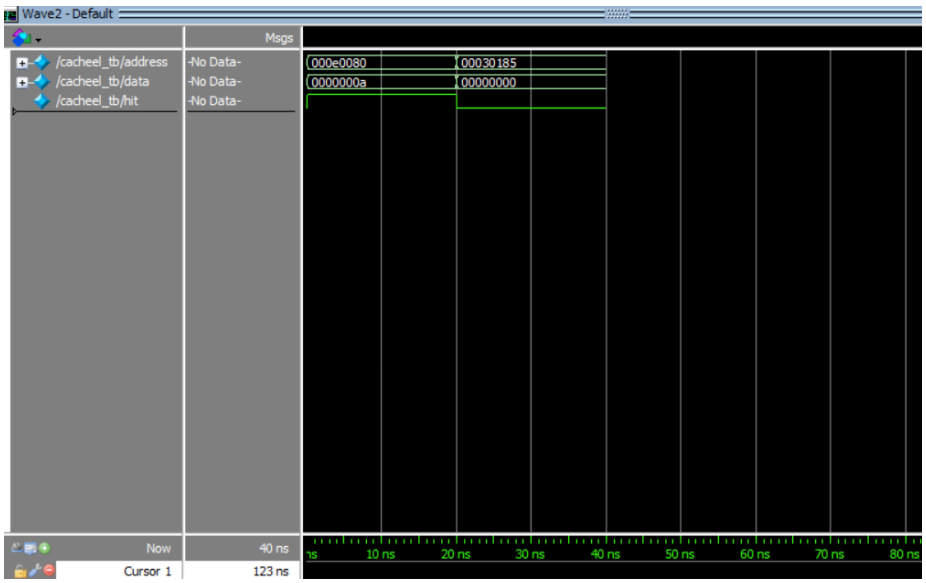
سپس متغیرهای تعریف شده در بخش بالا را مقدار دهی اولیه می‌کنیم و شرایط را با استفاده از if چک می‌کنیم که اگر آدرس tag بلوک با tag خانه حافظه‌مان یکی بود و bit valid برابر با یک بود، hit رخ دهد و مقدار متغیر hit را برابر با ۱ قرار می‌دهیم و دیتا موجود در آن را به خروجی انتقال می‌دهیم. به علاوه نیاز به طراحی تست بنچ هم با توجه به خواسته سوال داریم.

کدهای آن:

```
module cache_el (       input [31:0] addres,    output reg [31:0] data,output reg hit);

        reg [144:0]  cache_4k [11:0];
        reg [15:0] tag;
        reg [11:0] index;
        reg [1:0] byteofset;
        reg [1:0] blockofset;


        always @(*)
                begin
                byteofset <= addres[1:0];
                blockofset  <= addres [3:2];
                index <= addres[15:4];
                tag <= addres[31:16];
                data =0;
                hit =0;
                if(cache_4k[index][143:128] == tag && cache_4k[index][144] == 1 )
                        begin
                                hit =1;
                                case(blockofset)
                                        0'b00:data <= cache_4k[index][31:0] ;
                                        0'b01: data <= cache_4k[index][63:32]   ;
                                        0'b10: data <= cache_4k[index][95:64]   ;
                                        0'b11 : data <= cache_4k[index][127:96] ;
                                        endcase
                end
                end

                initial
                        begin
                                cache_4k[8][31:0] = 10;
                                cache_4k [8] [128+16] =1 ;
                                cache_4k [8] [127+16:128]= 14;

```

```
                                cache_4k[24][63:32] = 5;
                                cache_4k [24] [128+16] =0 ;
                                cache_4k [24] [127+16:128]= 3;

                        end


        endmodule


        module cacheel_tb;
        reg [31:0] address;
         wire [31:0] data;
         wire hit;
         localparam period = 20;
         cache_el cach (address,data,hit);

         initial
                begin
                        address [1:0] <=0;
                        address [3:2] <=0;
                        address [15:4] <=8;
                        address [31:16] <= 14;
                        #period;
                        address [1:0] <=1;
                        address [3:2] <=1;
                        address [15:4] <=24;
                        address [31:16] <= 3;
                        #period;


                        end

                  initial
                        $monitor("address: \t addres = %b \t OUTPUT VALUE: \t data = %b hit = %b  ", address, data, hit);

                endmodule
```

نتایج خروجی توسط تست بنچ:



سوال ۳ :

با استفاده از زبان Verilog یک صف اولویت بسازید. این صف ۱۶ خانه دارد که هرکدام از خانه‌ها ۸ بیت را در خود ذخیره میکند. این صف دو سیگنال جهت عملیات‌های Enqueue و Dequeue به منظور وارد کردن داده و خارج کردن داده از صف را دارد. در صف اولویت، هر بسته دارای اولویت است؛ بدین شکل که بسته ای با اولویت بیشتر جلوتر از بسته با اولویت کمتر قرار میگیرد. مقدار قطعی هر بسته اولویت این را تعیین میکند.(هر چه عدد بزرگتر، اولویت بیشتر) فرض کنید مقادیر بسته‌ها اعداد Unsigned هستند.

پاسخ ۳:

در صف عادی از روش خروج به ترتیب ورود (FIFO) استفاده می‌شود. در این تکنیک داده‌ها به ترتیب ورود پشت سر هم در صف قرار می‌گیرند؛ بنابراین اولین داده ورودی اولین داده خروجی نیز خواهد بود. اما در صف اولویت‌دار برای هر داده اولویتی - نه لزوماً منحصربه‌فرد - مشخص می‌شود. صف‌های عملیات دو عمل enqueue و dequeue را پشتیبانی می‌کنند. در این سوال برای پیاده‌سازی صف اولویت از روش آرایه استفاده می‌کنیم:

ابتدا آرایه‌ای با ظرفیت ۱۶ خانه ۸ بیتی تعریف می‌کنیم. سپس if را برای دستور read و write می‌نویسیم و عملیات queue و dequeue را در آن‌ها پیاده سازی می‌کنیم. در انتها مانند دو بخش قبل testbench را پیاده سازی می‌کنیم.

کدهای مربوطه:

```
module queue(
        inout unsigned[7:0] data,
        input read, write
);
reg unsigned[8:0] m [15:0];
integer i ,max;
reg unsigned[7:0] data1, max_data;
reg hitw,hitr;
reg[7:0] data_reg;
always @(*)
        begin
                data1 = data;
                hitw=0;
                hitr=0;
                max=0;
                if(write)
                        begin
                                for (i=0 ; i < 16; i=i+1)
                                        begin
                                                if(!hitw)
                                                        if(!mem[i][8])
                                                                begin
                                                                        m[i][7:0]<=data1;
                                                                        m[i][8]<=1'b1;
                                                                        hitw=1;
                                                                end
                                        end
                        end
                if(read)
                        begin
```

```
                        for (i=0 ; i < 16; i=i+1)
                                begin
                                        if(!hitr)
                                                begin
                                                        if(m[i][8])
                                                                begin
                                                                        max_data=m[i][7:0];
                                                                        max=i;
                                                                end
                                                end
                                                        else if(m[i][8])
                                                                begin
                                                                        if(m[i][7:0]>max_data)
                                                                                begin
                                                                                        max_data=m[i][7:0];
                                                                                        max=i;
                                                                                end
                                                                end
                                        end
                                m[max][8] = 0;
                                data_reg = max_data;
                        end
        end
        assign data = data_reg;
endmodule

module queue_tb;
        reg enq;
        reg deq;
        wire unsigned [7:0] inw;
        reg unsigned [7:0] in;
        assign inw=in;
        queue uut (
                .write(enq),
                .read(deq),
                .data(inw)
        );
        initial begin
                enq = 0;
                deq = 0;
                in = 0;
                #10;
                enq = 1;
                deq = 0;
                in = 2;
                #10;
                enq = 1;
                deq = 0;
                in = 5;
                #10;
                enq = 1;
                deq = 1;
                in = 10;
                #10;
                enq = 1;
                deq = 0;
                in = 0;
                #10;
        end
endmodule
```

نتایج تست بنچ: