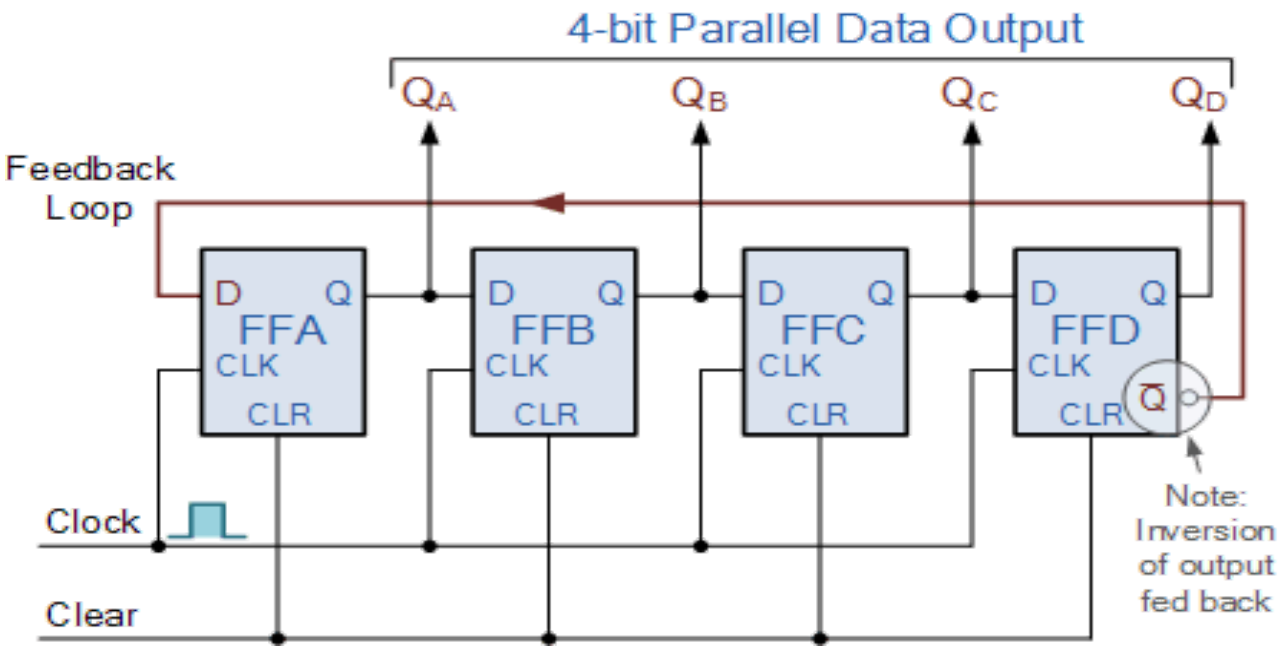
من سوالات ۶ و ۷ رو انتخاب کردم.

سوال ۶)

من ابتدا مدار D - flip flop را طراحی کردم که کد آن به شکل زیر است:

```
module D_FlipFlop(input D,clk,rst,output reg Q );
    always @(posedge clk or negedge rst) begin
        if(!rst)
            Q<=0;
        else
            Q<=D;
    end
endmodule
```

سپس با استفاده از آن شمارنده جانسون را طبق مدار زیر طراحی کردم:



کد این شمارنده به شکل زیر است:

```
module Johnson_counter #(parameter N=4)(input rst,clk,output [N-1:0]Q);
    genvar ii;
    generate
        for (ii=0;ii<N;ii=ii+1) begin
            if(ii==0) begin
                D_FlipFlop d(~Q[N-1],clk,rst,Q[ii]);
            end
            else begin
                D_FlipFlop d(Q[ii-1],clk,rst,Q[ii]);
            end
        end
    endgenerate
endmodule
```

سپس طبق خواسته سوال برای آن تست بنچ نوشتم:

برای N =4 :

```
endmodule
module TB;
    parameter N=4;
    reg rst;
    reg clk;
    wire [N-1:0]Q;
    always begin
        #5 clk=~clk;
    end
    Johnson_counter #(.N(N)) sample (rst,clk,Q);
    initial begin
        clk=0;
        rst<=0;
        #2 rst<=1;
        #500 $finish;
    end
    initial begin
        $monitor("%b", Q);
    end
    initial begin
        $dumpfile("Johnson_counter.vcd");
        $dumpvars(0,TB);
    end
endmodule
```

که خروجی آن به شکل زیر است:

```
0000
0001
0011
0111
1111
1110
1100
1000
0000
0001
0011
0111
1111
1110
1100
1000
0000
0001
0011
0111
1111
1110
1100
1000
0000
0001
0011
0111
1111
1110
```

برای N=8:

```
00000000
00000001
00000011
00000111
00001111
00011111
00111111
01111111
11111111
11111110
11111100
11111000
11110000
11100000
11000000
10000000
00000000
00000001
00000011
00000111
00001111
00011111
00111111
01111111
11111111
11111110
11111100
11111000
11110000
11100000
```

برای N=16 :

```
0000000000000000
0000000000000001
0000000000000011
0000000000000111
0000000000001111
0000000000011111
0000000000111111
0000000001111111
0000000011111111
0000000111111111
0000001111111111
0000011111111111
0000111111111111
0001111111111111
0011111111111111
0111111111111111
1111111111111111
1111111111111110
1111111111111100
1111111111111000
1111111111110000
1111111111100000
1111111111000000
1111111110000000
1111111100000000
1111111000000000
1111110000000000
1111100000000000
1111000000000000
```

برای N=32 :

```
VCD info: dumpfile Johnson_counter.vcd opened for output.
00000000000000000000000000000000
00000000000000000000000000000001
00000000000000000000000000000011
00000000000000000000000000000111
00000000000000000000000000001111
00000000000000000000000000011111
00000000000000000000000000111111
00000000000000000000000001111111
00000000000000000000000011111111
00000000000000000000000111111111
00000000000000000000001111111111
00000000000000000000011111111111
00000000000000000000111111111111
00000000000000000001111111111111
00000000000000000011111111111111
00000000000000000111111111111111
00000000000000001111111111111111
00000000000000011111111111111111
00000000000000111111111111111111
00000000000001111111111111111111
00000000000011111111111111111111
00000000000111111111111111111111
00000000001111111111111111111111
00000000011111111111111111111111
00000000111111111111111111111111
00000001111111111111111111111111
00000011111111111111111111111111
00000111111111111111111111111111
```

کد این سوال در پیوست خدمتتون تقدیم میشود.

سوال ۷)

برای این سوال ابتدا طبق خواسته های سوال من به ترتیب (از سمت چپ)
RegisterFile, ALU, Memory , Processor را طراحی کردم که کد آنها به همراه توضیحاتشان به شرح زیر است:

برای

برای RegisterFile من سیگنال های کنترلی برای اینکه شروع به نوشتن کنیم و از کدام رجیسترها بخوانیم و بنویسیم،برای کلاک، تعریف کردم و همچنین ورودی و خروجی هایی برای نوشتن و خواندن دیتا.

```
module RegisterFile (
    input wire clk,
    input wire [1:0] read_reg1, read_reg2, write_reg1,write_reg2,
    input wire [511:0] write_data1,write_data2,
    input wire reg_write_enable1,reg_write_enable2,
    output reg [511:0] read_data1, read_data2
);
    reg [511:0] registers [0:3];
    always @(posedge clk) begin
                if(reg_write_enable1)
                    registers[write_reg1] <= write_data1;
                if(reg_write_enable2)
                    registers[write_reg2] <= write_data2;
    end
    always @(*) begin
        read_data1 = registers[read_reg1];
        read_data2 = registers[read_reg2];
    end
endmodule
```

برای ALU ،سیگنال کلاک،دو سیگنال کنترلی برای مشخص کردن دستور و دو ورودی و دو خروجی برای ورودی دادن اطلاعات و گرفتن خروجی عملیات تعریف کردم:

```
module ArithmeticUnit (
    input wire [511:0] A1, A2,
    input wire [1:0] operation, // 00: no-op, 01: addition, 10: multiplication
    output reg [511:0] A3, A4
);
    always @(*) begin
        case (operation)
            2'b10: begin
                // A4=512'b0;
                {A4, A3} = A1 + A2;
            end
            2'b11: begin
                {A4, A3} = A1 * A2;
            end
            default: begin
                A3 = 0;
                A4 = 0;
            end
        endcase
    end
endmodule
```

\
\

برای Memory سیگنال کلاک، سیگنال فعال کردن توانایی نوشتن، ورودی دادن آدرس ، ورودی دادن دیتا و خروجی گرفتن دیتا تعریف کردم:

```
module Memory (
    input wire clk,
    input wire mem_write_enable,
    input wire [8:0] mem_addr,
    input wire [511:0] mem_write_data,
    output reg [511:0] mem_read_data
);
    reg [31:0] memory [0:511];

    integer i;

    always @(posedge clk) begin
        if (mem_write_enable) begin
            for (i = 0; i < 16; i = i + 1) begin
                memory[mem_addr + i] <= mem_write_data[(i*32) +: 32];
            end
        end
    end

    always @(*) begin
        for (i = 0; i < 16; i = i + 1) begin
            mem_read_data[(i*32) +: 32] = memory[mem_addr + i];
        end
    end
endmodule
```

حالا میرسیم به اسمبل کردن این مدول ها و طراحی پردازنده:
من از بیرون سیگنال دستور، سیگنال برای اینکه روی کدام رجیسترها بنویسم و بخوانم، سیگنال آدرس، سیگنال فعال کردن نوشتن روی ورودی، سیگنال ورودی دادن دیتا برای نوشتن در حافظه، (توانایی نوشتن مستقیم بر روی رجیسترها نداریم)، سیگنال اینکه آیا باید دیتای روی حافظه از بیرون نوشته شود یا
(control)، سیگنال فعال کردن نوشتن روی رجیسترها و همچنین چهار سیگنال برای اینکه ببینم نتایج عملیات ها موفقیت آمیز بوده یا نه تعریف کردم:

```
module Processor (
    input wire clk,
    input wire [1:0] instruction, // 00: load, 01: store, 10: add, 11: multiply
    input wire [1:0] reg_select_read1, reg_select_read2, reg_select_write1, reg_select_write2,
    input wire [8:0] mem_addr,
    input wire [511:0] mem_write_data, // input of processor
    input wire control,
    input wire reg_write_enable1,
    input wire reg_write_enable2,
    output wire [511:0] A1, A2, A3, A4 // for checking content of A!, A2, A3, A4
);
    wire [511:0] alu_result1, alu_result2;
    wire [511:0] reg_read_data1, reg_read_data2;
    wire [511:0] mem_read_data;
    RegisterFile rf (
        .clk(clk),
        .read_reg1(reg_select_read1),
        .read_reg2(reg_select_read2),
        .write_reg1(reg_select_write1),
        .write_reg2(reg_select_write2),
        .write_data1((instruction == 2'b00) ? mem_read_data : alu_result1),
        .write_data2((instruction == 2'b00) ? 512'bz : alu_result2),
        .reg_write_enable1(reg_write_enable1),
        .reg_write_enable2(reg_write_enable2),
        .read_data1(reg_read_data1),
        .read_data2(reg_read_data2)
    );
    ArithmeticUnit alu (
        .A1(reg_read_data1),
        .A2(reg_read_data2),
        .operation(instruction[1:0]),
        .A3(alu_result1),
        .A4(alu_result2)
    );
    Memory mem (
        .clk(clk),
        .mem_write_enable(instruction == 2'b01),
        .mem_addr(mem_addr),
        .mem_write_data((control==1)? mem_write_data:reg_read_data1),
        .mem_read_data(mem_read_data)
    );
    assign A1 = reg_read_data1;
    assign A2 = reg_read_data2;
    assign A3 = alu_result1;
    assign A4 = alu_result2;

endmodule
```

حال میرسیم به تست بنچ نوشتن:

متاسفانه نشد که کل کد را در یک صفحه جا دهم. لطفا به کد الصاقی مراجعه کنید:

خلاصه تست بنچی که نوشتم اینه که ابتدا در دو خانه متفاوت حافظه دو دیتای متفاوت نوشته ام و سپس این دو دیتا را از حافظه روی رجیسترهای A1 , A2 (که جون وریلاگ صفر بیس است، یک واحد از شماره رجیسترها کم میشود) سپس این دیتاها را به ALU میدهیم و سپس برای اینکه مطمین شوم که پردازنده به درستی کار میکند این دو خروجی را روی حافظه نوشتم و دوباره میخوانم که کد ورودی و خروجی گرفتنم را میتوانید در زیر ببینید:

این بلاک برای `TestBench1` است: جمع دو عدد ۲ و ۴ :

```
    initial begin
    reg_write_enable1 = 0;
    reg_write_enable2 = 0;
    clk = 0;
    instruction = 2'b01;
    mem_addr = 9'b0;
    control = 1;
    mem_write_data = 512'b10; // 2 in binary
    #10
    mem_addr = 9'b010000000; // 128 in binary
    mem_write_data = 512'b100; // 4 in binary
    #10
    control = 0;
    instruction = 2'b00;
    mem_addr = 9'b0;
    reg_select_write1 = 2'b00;
    reg_write_enable1 = 1;
    #10
    mem_addr = 9'b010000000;
    reg_select_write1 = 2'b01;
    #10
    reg_write_enable1 = 0;
    reg_select_read1 = 2'b00;
    reg_select_read2 = 2'b01;
    #10
    instruction = 2'b10; // add
    #50
    $display("A2= %d\n A1= %d", A2, A1);
    $display("A4= %d\n A3= %d", A4, A3);
    reg_select_write1=2'b10;
    reg_select_write2=2'b11;
    reg_write_enable1=1;
    reg_write_enable2=1;
    #10
    instruction=2'b01;
    reg_select_read1=2'b10;
    mem_addr=9'b0;
    #10
    reg_select_read1=2'b11;
    mem_addr = 9'b010000000;
    #10
    instruction = 2'b00;
    mem_addr = 9'b0;
    reg_select_write1 = 2'b00;
    reg_write_enable1 = 1;
    #10
    mem_addr = 9'b010000000;
    reg_select_write1 = 2'b01;
    #10
    reg_write_enable1 = 0;
    reg_select_read1 = 2'b00;
    reg_select_read2 = 2'b01;
    #10
    $display("A1= %d\n A2= %d", A1, A2);
    #20 $finish;
    end
```

```
A2= 4
A1= 2
A4= 0
A3= 6
A1= 6
A2= 0
```

حالا من مدار را برای جمع -1 و دو به توان ۵۱۱ منهای یک حساب میکنم که همانطور که میبنید به درستی اورفلو را در A4 میریزد و مقدار درست را در A3 میریزد : که در ۵۱۲ بیت جا میشود و عدد صفر مینویسم که همانطور که میبینید برای این کار هم به درستی خروجی میدهد:

TestBench2:

```
232     initial begin
233     reg_write_enable1 = 0;
234     reg_write_enable2 = 0;
235     clk = 0;
236     instruction = 2'b01;
237     mem_addr = 9'b0;
238     control = 1;
239     mem_write_data = {1'b0,{511{1'b1}}};
240     #10
241     mem_addr = 9'b010000000;
242     mem_write_data = {512{1'b1}};
243     #10
244     control = 0;
245     instruction = 2'b00;
246     mem_addr = 9'b0;
247     reg_select_write1 = 2'b00;
248     reg_write_enable1 = 1;
249     #10
250     mem_addr = 9'b010000000;
251     reg_select_write1 = 2'b01;
252     #10
253     reg_write_enable1 = 0;
254     reg_select_read1 = 2'b00;
255     reg_select_read2 = 2'b01;
256     #10
257     instruction = 2'b10; // add
258     #50
259     $display("A2= %b\n A1= %b", A2, A1);
260     $display("A4= %b\n A3= %b", A4, A3);
261     #20 $finish;
262     end
263 endmodule
```

خروجی کد :

```
A2= 111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
111111111
 A1= 01111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111
A4= 000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000001
 A3= 01111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111110
steven@steven:~/Downloads/verilog/Question 7$
```

حال مدار را برای جمع بزرگترین عدد مثبت و قرینه آن حساب میکنیم که به درستی عدد صفر را در A3 و اورفلو را در A4 میریزد:

TestBench 5

```verilog
initial begin
reg_write_enable1 = 0;
reg_write_enable2 = 0;
clk = 0;
instruction = 2'b01;
mem_addr = 9'b0;
control = 1;
mem_write_data = {1'b0,{511{1'b1}}};
#10
mem_addr = 9'b010000000;
mem_write_data = -{1'b0,{511{1'b1}}};
#10
control = 0;
instruction = 2'b00;
mem_addr = 9'b0;
reg_select_write1 = 2'b00;
reg_write_enable1 = 1;
#10
mem_addr = 9'b010000000;
reg_select_write1 = 2'b01;
#10
reg_write_enable1 = 0;
reg_select_read1 = 2'b00;
reg_select_read2 = 2'b01;
#10
instruction = 2'b10; // add
#50
$display("A2= %b\n A1= %b", A2, A1);
$display("A4= %b\n A3= %b", A4, A3);
#20 $finish;
end
endmodule
```

خروجی:

```
A2= 10000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001
 A1= 01111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
A4= 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001
 A3= 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
steven@steven:~/Downloads/verilog/Question 7$
```

حالا میخواهیم عملکرد مدار را برای ضرب دو عدد مورد بررسی قرار دهیم:
برای ضرب 10 و 100 :
برای فهم راحت تر اعداد را بر مبنای 10 مینویسم: TestBench3

```
295   initial begin
296   reg_write_enable1 = 0;
297   reg_write_enable2 = 0;
298   clk = 0;
299   instruction = 2'b01;
300   mem_addr = 9'b0;
301   control = 1;
302   mem_write_data = 512'd10;
303   #10
304   mem_addr = 9'b010000000;
305   mem_write_data = 512'd100;
306   #10
307   control = 0;
308   instruction = 2'b00;
309   mem_addr = 9'b0;
310   reg_select_write1 = 2'b00;
311   reg_write_enable1 = 1;
312   #10
313   mem_addr = 9'b010000000;
314   reg_select_write1 = 2'b01;
315   #10
316   reg_write_enable1 = 0;
317   reg_select_read1 = 2'b00;
318   reg_select_read2 = 2'b01;
319   #10
320   instruction = 2'b11; // multiply
321   #50
322   $display("A2= %d\n A1= %d", A2, A1);
323   $display("A4= %d\n A3= %d", A4, A3);
324   #20 $finish;
325   end
```

که همانطور که میبینیم به درستی حاصل که برابر ۱۰۰۰ است را در رجیستر A3 ریخته و A4 را چون ۱۰۰۰ در A3 جا میشد، با صفر مقداردهی کرده است.

```
0000000000000001
A2=                                                    100
A1=                                                     10
A4=                                                      0
A3=                                                   1000
```

حال مدار را برای ضرب دو به توان ۵۱۲ منهای یک و صفر حساب میکنیم که همانطور که میبینیم جواب صفر را به درستی در A3 و A4 خروجی میدهد : TestBench4