به نام خداوند بخشنده مهربان

طراحی سیستم های دیجیتال

دانشکده مهندسی کامپیوتر

دکتر فصحتی
بهار ۱۴۰۳

مهدی علی نژاد، ۴۰۱۱۰۶۲۶۶

تمرین چهارم

سوال ۱

مداری طراحی کنید که ۳ عدد صحیح ۳۲ بیتی (A، B و C) را از ورودی بگیرد و با فشردن دکمه Start دو عدد A و B را با روش جمع متوالی در هم ضرب کند و در صورتی که حاصلضرب کوچکتر از عدد C باشد، حاصلضرب را در خروجی و در صورتی که بزرگتر باشد، عدد C را در خروجی قرار دهد.

الف) مدار خود را با یک ASM Chart طراحی کنید.

ب) از روی ASM Chart، منابع لازم برای DataPath را بدست آورده و بنویسید.

پ) سپس هر کدام از این منابع را در وریلاگ به صورت رفتاری توصیف کنید (برای هر منبع یک ماژول) و با اتصال ساختاری این ماژول‌ها به هم DataPath خود را تکمیل کنید.

ت) در انتها نمودار حالت طراحی خود را بدست آورید.

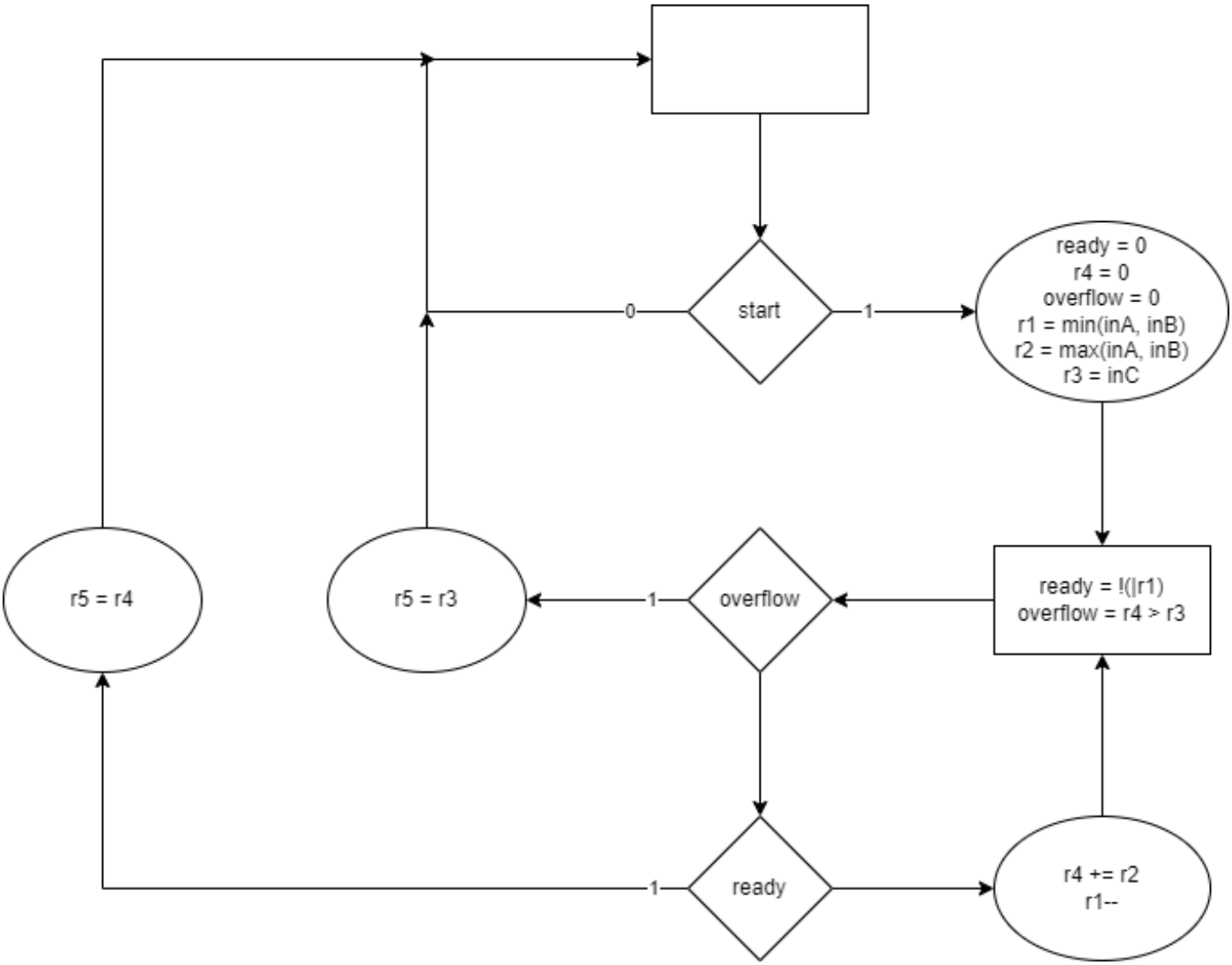
ث) از روی نمودار حالت بخش ControlUnit طراحی خود را در وریلاگ توصیف کنید.

ج) برای توصیف وریلاگ خود Testbench بنویسید.

به نکات زیر در طراحی مدار دقت کنید:

۱- طراحی شما باید در تعداد پالاس ساعت کمتر پاسخ را محاسبه کند.
۲- مدار شما باید از لحاظ سخت افزار بهینه باشد.
۳- سیگنال‌های کنترلی (Start و Ready) و قوانین طراحی (مانند عدم نشت مقادیر میانی مدار) در طراحی مدار رعایت گردد.

(آ) برداشتم از ضرب کننده ی مورد نظر، یک ضرب کننده است که از همان الگوریتم ذکر شده در اسلاید ها برای محاسبه ی ضرب در تعداد کلاک کمتر استفاده می کند با این تفاوت که اگر حاصل ضرب از یک مقداری overflow کرد، به جای نتیجه، همان مقدار را بازگردانیم، این حالت را با سیگنال overflow در طراحی خود مدل کرده ام، همچنین برای تعداد حلقه های کمتر، از دو ماژول min و max بهره برده ام. مقادیر میانی نیز از بیرون قابل مشاهده نیست و سخت افزار تا حد امکان بهینه شده است.



(ب) دستوراتی که این Datapath باید توانایی انجام آنها را داشته باشد به صورت زیر است:

```
ready <= 0
r4 <= 0
overflow <= 0
r1 <= min(inA, inB)
r2 <= max(inA, inB)
r3 <= inC
ready <= !(|r1)
overflow <= r4 > r3
r4 <= r4 + r2
r1–
r5 <= r3
r5 <= r3
```

و برای اجرای این دستورات به قطعات زیر در دیتاپف نیاز است:

register with reset and enable for ready, overflow, r1 $\sim$ r4 and state
min and max modules
reduction nor for r1
adder, decrement
2x1 mux for r5 and r1
comparator

البته این min و max به صورت خاصی عمل می کنند تا اعداد علامت دار را نیز هندل کنند. ماژول min فاصله ی اعداد تا صفر را می سنجد و قدرمطلق عددی که فاصله ی کمتری دارد را خروجی می دهد، و ماژول max نیز به عددی که فاصله ی کمتری دارد نگاه می کند، اگر آن عدد منفی بود، قرینه ی عدد دوم و در غیر این صورت همان عدد دوم را خروجی می دهد. با انجام این تغییرات، همواره عددی که به صفر نزدیک تر است، در رجیستر ۱ قرار می گیرد و اگر منفی بود، عدد دوم را قرینه می کند و عدد دوم را نیز در رجیستر ۲ قرار می دهد.

(ج) از آنجایی که کد های توصیف رفتاری تا حد خوبی ساده و قابل فهم هستند، تنها به نشان دادن آنها اکتفا می کنم.

```verilog
module decrementer #(
    parameter n
) (
    a,
    res
);

input [n-1:0] a;
output reg [n-1:0] res;

always @(a) begin
    res <= a-1;
end

endmodule //decrementer
```

```verilog
module adder #(
    parameter n
) (
    a, b,
    overflow, result
);

input [n-1:0] a, b;
output reg overflow;
output reg [n-1:0]result;

always @(a or b) begin
    {overflow, result} <= a + b;
end

endmodule //adder
```

```verilog
module two_to_one_mux #(
    parameter n
) (
    a1, a2, sel,
    res
);

input [n-1:0]a1, a2;
input sel;
output reg [n-1:0] res;

always @(a1 or a2 or sel) begin
    if (sel == 1)
        res <= a2;
    else
        res <= a1;
end

endmodule //two_to_one_mux
```

```verilog
module compartor #(
    parameter n
) (
    a, b,
    isGreater
);

input[n-1:0] a, b;
output reg isGreater;


always @(a, b) begin
    if ($signed(a) > $signed(b))
        isGreater <= 1;
    else
        isGreater <= 0;
end

endmodule //compartor
```

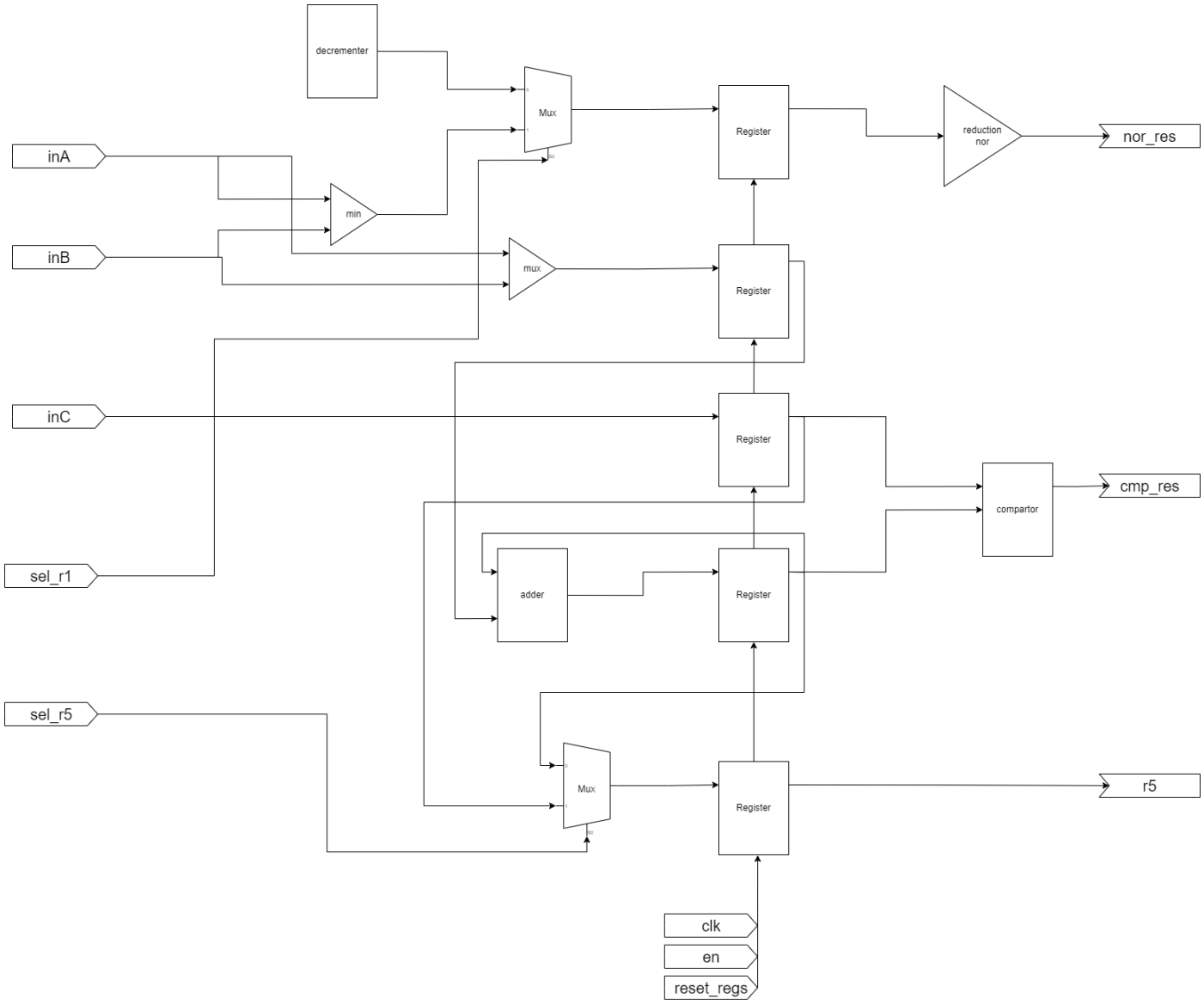
```verilog
module reduction_nor #(
    parameter n
) (
    in,
    rn
);


input [n-1:0] in;
output reg rn;

always @(in) begin
    rn <= ~(|in);
end

endmodule //reduction_nor
```

```verilog
module register #(
    parameter n
) (
    data_in, reset, clk, enable,
    data_out
);
input [n-1:0]data_in;
input  reset, clk, enable;
output reg [n-1:0] data_out;

always @(posedge clk or negedge reset) begin
    if (!reset)
        data_out <= 0;
    else begin
        if (enable) begin
            data_out <= data_in;
        end
    end
end

endmodule //register
```

```
module max #(
    parameter n
) (
    a, b,
    res
);


input [n-1:0]a, b;
output reg [n-1:0]res;

always @(a or b) begin
    if ($signed(a) < 0) begin
        if($signed(b) < 0) begin
            if ($signed(a) < $signed(b))
                res = -a;
            else
                res = -b;
        end
        else begin
            if (-a < b)
                res = -b;
            else
                res = a;
        end
    end
    else begin
        if($signed(b) < 0) begin
            if (-b < a)
                res = -a;
            else
                res = b;
        end
        else begin
            if (a < b)
                res = b;
            else
                res = a;
        end
    end
end

endmodule //max
```

```
module min #(
    parameter n
) (
    a, b,
    res
);


input [n-1:0]a, b;
output reg [n-1:0]res;

always @(a or b) begin
    if ($signed(a) < 0) begin
        if($signed(b) < 0) begin
            if ($signed(a) < $signed(b))
                res = -b;
            else
                res = -a;
        end
        else begin
            if (-a < b)
                res = -a;
            else
                res = b;
        end
    end
    else begin
        if($signed(b) < 0) begin
            if (-b < a)
                res = -b;
            else
                res = a;
        end
        else begin
            if (a < b)
                res = a;
            else
                res = b;
        end
    end
end

endmodule //min
```

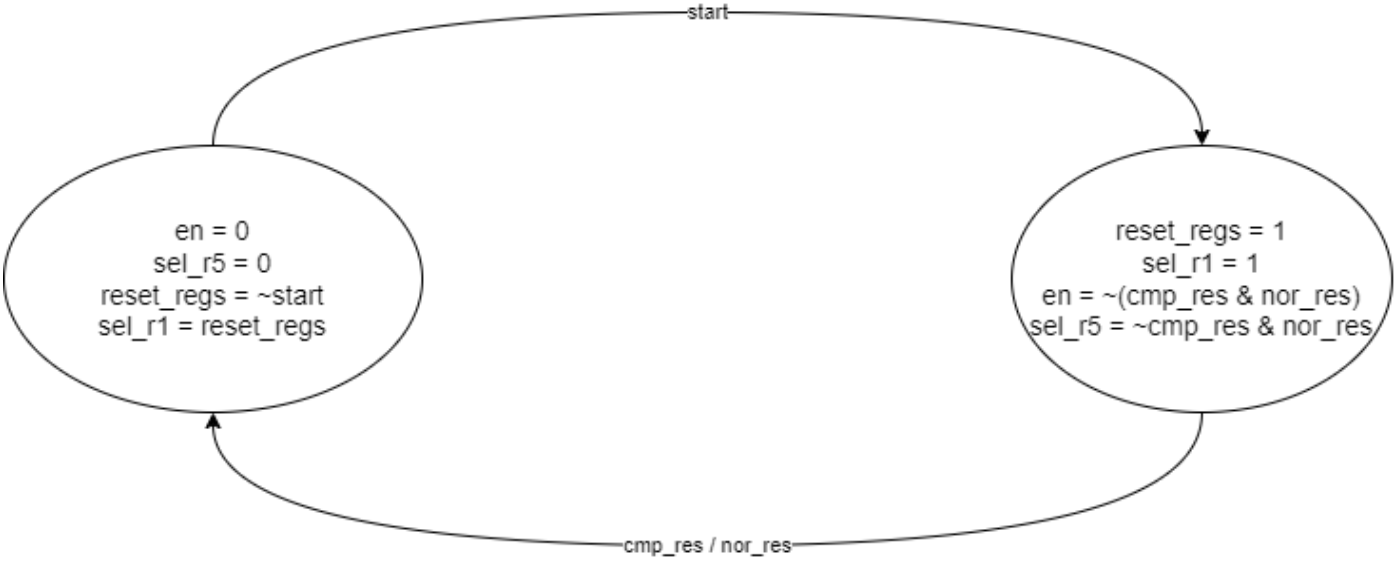
سپس آنها را در ماژول دیتاپف به یکدیگر متصل می کنیم تا همچین مداری تشکیل شود.


decrementer
Mux
Register
reduction nor
nor_res
inA
min
inB
mux
Register
inC
Register
compartor
cmp_res
sel_r1
adder
Register
sel_r5
Mux
Register
r5
clk
en
reset_regs

```verilog
module datapath #(
    parameter n
) (
    inA, inB, inC, reset_regs, sel_r1, sel_r5, clk, en,
    cmp_res, nor_res, r5
);

input [n-1:0]inA, inB, inC;
input reset_regs, sel_r1, sel_r5, clk, en;
output cmp_res, nor_res;
output [n-1:0]r5;

wire [n-1:0]r1, r2, r3, r4, adder_res, r1_mux_in, r5_mux_in, dec_r1, min_in, max_in;
wire nor_res, cmp_res, r1_mux_sel, r5_mux_sel;

reduction_nor #(n) reduction_nor_instance (
    .in(r1) ,
    .rn(nor_res)
);

compartor #(n) compartor_ins (
    .a(r4),
    .b(r3),
    .isGreater(cmp_res)
);

adder #(n) adder_inst (
    .a(r4),
    .b(r2),
    .result(adder_res)
);

register #(n) r4_register (
    .reset(reset_regs),
    .clk(clk),
    .enable(en),
    .data_in(adder_res),
    .data_out(r4)
);

register #(n) r3_register (
    .reset(1'b1),
    .enable(1'b1),
    .clk(clk),
    .data_in(inC),
    .data_out(r3)
);
```

```
max #(n) max_inst (
    .a(inA),
    .b(inB),
    .res(max_in)
);

min #(n) min_inst (
    .a(inA),
    .b(inB),
    .res(min_in)
);

register #(n) r2_register (
    .reset(1'b1),
    .enable(1'b1),
    .clk(clk),
    .data_in(max_in),
    .data_out(r2)
);

decrementer #(n) dec_r1_inst (
    .a(r1),
    .res(dec_r1)
);

two_to_one_mux #(n) r1_mux_inst (
    .a1(min_in),
    .a2(dec_r1),
    .sel(sel_r1),
    .res(r1_mux_in)
);

register #(n) r1_register (
    .reset(1'b1),
    .enable(1'b1),
    .clk(clk),
    .data_in(r1_mux_in),
    .data_out(r1)
);

two_to_one_mux #(n) r5_mux_inst (
    .a1(r3),
    .a2(r4),
    .sel(sel_r5),
    .res(r5_mux_in)
);

register #(n) r5_register (
    .reset(1'b1),
    .enable(nor_res | cmp_res),
    .clk(clk),
    .data_in(r5_mux_in),
    .data_out(r5)
);

endmodule //datapath
```

(د) نمودار حالت مدار کنترلی آن به این صروت در میاید:



(۵) و ماژول کنترلی ما نیز به این صورت است:

```
module control_unit (
    cmp_res, nor_res, start, clk,
    reset_regs, sel_r1, sel_r5, en
);

input cmp_res, nor_res, start, clk;
output reset_regs, sel_r1, sel_r5, en;

reg state = 0;

always @(posedge clk) begin
    if (state == 0) begin
        if(start == 1) begin
            state <= 1;
        end
    end
    else begin
        if(cmp_res) begin
            state <= 0;
        end
        else if (nor_res) begin
            state <= 0;
        end
    end
end

assign reset_regs = state | ~start;
assign sel_r1 = reset_regs;
assign en = state & (~cmp_res | ~nor_res);
assign sel_r5 = state & ~cmp_res & nor_res;

endmodule //control_unit
```

```
module mult_and_min #(
    parameter n
) (
    inA, inB, inC, start, clk,
    ready, overflow, result
);

input [n-1:0]inA, inB, inC;
input start, clk;
output ready, overflow;
output [n-1:0]result;

wire reset_regs, sel_r1, sel_r5, en, nor_res, cmp_res;

control_unit cn_inst (
    .cmp_res(cmp_res),
    .nor_res(nor_res),
    .start(start),
    .clk(clk),
    .reset_regs(reset_regs),
    .sel_r1(sel_r1),
    .sel_r5(sel_r5),
    .en(en)
);

datapath #(n) dp_inst (
    .inA(inA),
    .inB(inB),
    .inC(inC),
    .reset_regs(reset_regs),
    .sel_r1(sel_r1),
    .sel_r5(sel_r5),
    .clk(clk),
    .en(en),
    .cmp_res(cmp_res),
    .nor_res(nor_res),
    .r5(result)
);

register #(1) ready_register (
    .reset(reset_regs),
    .clk(clk),
    .enable(en),
    .data_in(nor_res),
    .data_out(ready)
);

register #(1) overflow_register (
    .reset(reset_regs),
    .clk(clk),
    .enable(en),
    .data_in(cmp_res),
    .data_out(overflow)
);

endmodule //mult_and_min
```

(و) تست بنچ نوشته شده برای این مدار حالت هایی که در آنها ضرب بیشتر از عدد سوم می شود، حالتی که عدد بزرگتر در a قرار دارد و همچنین عدد بزرگتر در b قرار دارد را مدل می کند.

```
module testbench ();
parameter n = 32;
reg signed [n-1:0]a, b, c;
reg start, clk;
wire [n-1:0]result;
wire overflow, ready;

mult_and_min #(n) mam (
    .inA(a),
    .inB(b),
    .inC(c),
    .start(start),
    .clk(clk),
    .overflow(overflow),
    .ready(ready),
    .result(result)
);

initial begin
    clk = 0;
end

always #5 clk = ~clk;

initial begin
    a = 13; b = 12; c = 100; start = 1;
    #20;
    start = 0;
    #200;
    a = 4; b = 8; c = 59; start = 1;
    #20;
    start = 0;
    #200;
    a = 10; b = -2; c = 38; start = 1;
    #20;
    start = 0;
    #200;
    a = -10; b = -2; c = 38; start = 1;
    #20;
    start = 0;
    #200;
    a = -1; b = -2; c = 38; start = 1;
    #20;
    start = 0;
    #200;
    $stop();
end

initial begin
    $monitor($time, ": value of a is: ", a, " value of b is: ", b, " value of c is: ", c,
     " and we see result ", $signed(result), " with signals ready and overflow in order being ", ready, overflow);
end

initial begin
    $dumpfile("mult_and_min.vcd");
    $dumpvars();
end

endmodule //testbench
```

```
#                0: value of a is:          13 value of b is:          12 value of c is:         100 and we see result           x with signals ready and overflow in order being 00
#               95: value of a is:          13 value of b is:          12 value of c is:         100 and we see result         100 with signals ready and overflow in order being 01
#              220: value of a is:           4 value of b is:           8 value of c is:          59 and we see result         100 with signals ready and overflow in order being 00
#              275: value of a is:           4 value of b is:           8 value of c is:          59 and we see result          32 with signals ready and overflow in order being 10
#              440: value of a is:          10 value of b is:          -2 value of c is:          38 and we see result          32 with signals ready and overflow in order being 00
#              475: value of a is:          10 value of b is:          -2 value of c is:          38 and we see result         -20 with signals ready and overflow in order being 10
#              660: value of a is:         -10 value of b is:          -2 value of c is:          38 and we see result         -20 with signals ready and overflow in order being 00
#              695: value of a is:         -10 value of b is:          -2 value of c is:          38 and we see result          20 with signals ready and overflow in order being 10
#              880: value of a is:          -1 value of b is:          -2 value of c is:          38 and we see result          20 with signals ready and overflow in order being 00
#              905: value of a is:          -1 value of b is:          -2 value of c is:          38 and we see result           2 with signals ready and overflow in order being 10
```

فایل vcd نیز جهت مشاهده ی دقیق تر در دسترس شما قرار داده شده است.