به نام خداوند بخشنده مهربان

طراحی سیستم های دیجیتال

دانشکده مهندسی کامپیوتر

استاد فصحتی
بهار ۱۴۰۳

مهدی علی نژاد، ۴۰۱۱۰۶۲۶۶

تمرین اول

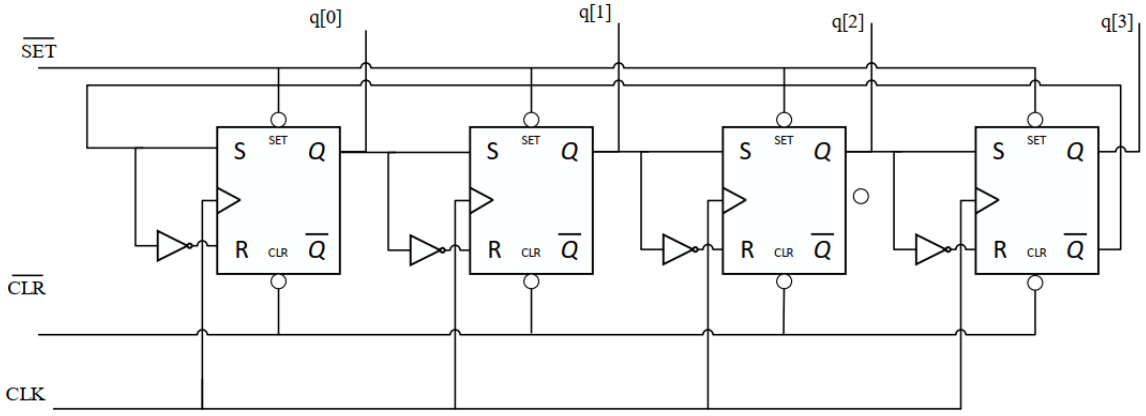
سوال ۱

۱- الف) سخت‌افزار زیر را در وریلاگ توصیف کنید (ورودی‌های SET و CLR به‌صورت ناهم‌زمان هستند).

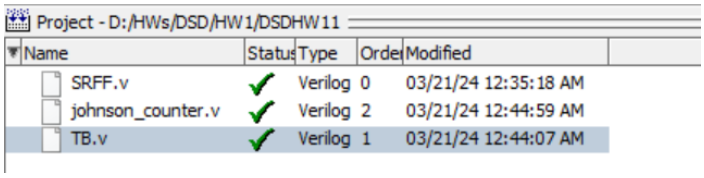
ب) خروجی‌های ممکن سخت‌افزار زیر را مشخص کنید.

پ) برای آن یک ماژول بسترآزمون[1] (ماژول تحریک[2]) نوشته که تمامی حالات را در ابزار شبیه‌سازی ModelSim مورد آزمون قرار دهد.

توجه: سیگنال‌های ورودی و خروجی در توصیف وریلاگ با نام سخت‌افزار یکسان باشند.



(آ) .



```v
module SRFF (
    S, R, clk, SET, CLR,
    Q, Q_not
);

input  wire S, R, clk, SET, CLR;
output reg Q, Q_not;

always @ (posedge clk or negedge CLR or negedge SET)begin
    if(~CLR)
        Q <= 0;
    else if(~SET)
        Q <= 1;
    else begin
        case({S,R})
            2'b00:
                Q <= Q;
            2'b01:
                Q <= 0;
            2'b10:
                Q <= 1;
            2'b11:
                Q <= 1'bx;
            default:
                Q <= Q;
        endcase
    end
    Q_not <= ~Q;
end

endmodule
```

HW1 > johnson_counter.v

```v
module johnson_counter (
    input CLK, SET, CLR,
    output [3:0]q
);

wire inner_c;
SRFF srff0(inner_c, ~inner_c, CLK, SET, CLR, q[0]);
SRFF srff1(q[0], ~q[0], CLK, SET, CLR, q[1]);
SRFF srff2(q[1], ~q[1], CLK, SET, CLR, q[2]);
SRFF srff3(q[2], ~q[2], CLK, SET, CLR, q[3], inner_c);

endmodule //johnson_counter

```

برای طراحی شماتیک ابتدا نیاز است SRFF را طراحی کنیم. همانطور که گفته شده بود در لیست حساسیت حلقه ی always متغیر های CLR و SET نیز آمده است که باعث می شود این سیگنال ها به طور ناهمزمان کار کنند. سپس اگر این سیگنال ها یک باشند، بر روی حالت های مختلف SR سوییچ کیس می زنیم و Q را مقدار دهی می کنیم.
و سپس برای خود طراحی نیز ۴ نمونه از SRFF طراحی شده می گیریم و سیم ها را همانطور که نمایش داده شده متصل می کنیم.

(ب) این توصیف در واقع سخت افزار یک شمارنده ی جانسون برعکس است. از حالت ۰۰۰۰ شروع می کند و در هر کلاک، نات MSB به جای LSB می نشیند و همچنین همه ی بیت ها به سمت چپ شیفت می خورند پس شمارنده به اینگونه می شمارد:

$$0000^{1}$$
$$0001$$
$$0011$$
$$0111$$
$$1111^{2}$$
$$1110$$
$$1100$$
$$1000$$
$$0000$$
$$\text{repeat}$$

علاوه بر این سیگنال های SET و CLR را نیز داریم که به صورت ناهمزمان و active low عمل می کنند و هنگام فعال سازی شمارنده را به ترتیب به وضعیت ۱ و وضعیت ۲ می برند.

(ج) .

HW1 > TB_johnson_counter.v

```v
module TB ();

reg clk, CLR, SET;
wire [3:0]q;

johnson_counter tb(clk, SET, CLR, q);

integer i;
always #5 clk = ~clk;

initial begin
    $monitor($time, ": q= %b ", q,
     " while CLR is: ", CLR, " and SET is: ", SET);
    clk = 0;
    for(i = 0; i < 4; i = i+1) begin
        {CLR, SET} = i;
        #85;
    end
    $stop();
end



endmodule //TB

```

```
VSIM 99> run -all
#                    0: q= 0000  while CLR is: 0 and SET is: 0
#                   85: q= 0000  while CLR is: 0 and SET is: 1
#                  170: q= 1111  while CLR is: 1 and SET is: 0
#                  255: q= 1110  while CLR is: 1 and SET is: 1
#                  265: q= 1100  while CLR is: 1 and SET is: 1
#                  275: q= 1000  while CLR is: 1 and SET is: 1
#                  285: q= 0000  while CLR is: 1 and SET is: 1
#                  305: q= 0001  while CLR is: 1 and SET is: 1
#                  315: q= 0011  while CLR is: 1 and SET is: 1
#                  325: q= 0111  while CLR is: 1 and SET is: 1
#                  335: q= 1111  while CLR is: 1 and SET is: 1
# Break in Module TB at D:/HWs/DSD/HW1/TB_johnson_counter.v line 19
```

بستر آزمون نوشته شده با نرخ کلاک ۱۰ واحد زمانی کار می کند و هر ۸۵ ثانیه یک بار به یکی از ۴ حالت ورودی های ممکن می رود، دو بیت CLR و SET ورودی های ما هستند. در ۸۵ واحد اول که هر دوی آنها صفر هستند، مدار در وضعیت صفر قرار می گیرد، در ۸۵ واحد بعدی SET در حالت غیر فعال قرار می گیرد و به دلیل فعال بودن CLR همچنان خروجی ها در وضعیت صفر باقی می مانند. در ۸۵ واحد زمانی سوم سیگنال CLR غیرفعال و سیگنال SET فعال می شود و مدار در وضعیت تمام یک قرار می گیرد. و در بخش آخر شبیه سازی که هردو سیگنال غیرفعال هستند از وضعیت تمام یک شروع به شمردن می کند.

سوال ۲

۲- یک رمزکننده اولویت‌دار[۳] ۴ در ۲ را با زبان برنامه‌نویسی وریلاگ با دستورات تخصیص پیوسته[۴] توصیف کنید و برای آن یک ماژول بسترآزمون (ماژول تحریک) نوشته که تمامی حالات را در ابزار شبیه‌سازی ModelSim مورد آزمون قرار دهد.

```
module encoder_4to2 (
    input [3:0]d ,
    output [1:0]q, v
);

assign q[1] = d[3] | d[2];
assign q[0] = d[3] | (~d[2] & d[1]);
assign v = d[3] | d[2] | d[1] | d[0];


endmodule //encoder_8to3

```

```
module TB_encoder ();
reg [3:0]d;
wire [1:0]q, v;
encoder_4to2 tb0 (d, q, v);
integer i;
initial begin
    $monitor($time, ": q = %b", q ,
     " for d = %b", d, " v = ", v);
    d = 8'b0;
    for (i = 0; i < 16 ; i = i+1) begin
        #10;
        d = d + 1;
    end
    $stop();
end

endmodule //TB_encoder

```

در بالا کد خود سخت افزار و بستر آزمون آن آمده است. خود سخت افزار همانطور که گفته شده بود از دستور assign که دستور تخصیص پیوسته است استفاده شده است. d ورودی های رمزکننده ، q خروجی رمزکننده است و v سیگنال معتبر بودن خروجی است. در بستر آزمون نیز خروجی ها را مانیتور می کنیم و روی تمام حالات ممکن d ها حلقه می زنیم و خروجی را بررسی می کنیم، در زیر بخشی از این خروجی آمده است:

```
  0: q = 00 for d = 0000 v = 0
 10: q = 00 for d = 0001 v = 1
 20: q = 01 for d = 0010 v = 1
 30: q = 01 for d = 0011 v = 1
 40: q = 10 for d = 0100 v = 1
 50: q = 10 for d = 0101 v = 1
 60: q = 10 for d = 0110 v = 1
 70: q = 10 for d = 0111 v = 1
 80: q = 11 for d = 1000 v = 1
 90: q = 11 for d = 1001 v = 1
100: q = 11 for d = 1010 v = 1
110: q = 11 for d = 1011 v = 1
120: q = 11 for d = 1100 v = 1
130: q = 11 for d = 1101 v = 1
140: q = 11 for d = 1110 v = 1
150: q = 11 for d = 1111 v = 1
```

۳- الف) کد وریلاگ زیر ضرب دو عدد علامت‌دار ۸ بیتی را توصیف می‌کند. برای آن یک ماژول بسترآزمون نوشته و آن را در ابزار ModelSim شبیه‌سازی کنید. سپس نتایج شبیه‌سازی را به ۳ روش (چاپ در خروجی، نمایش موج‌ها و فایل vcd) گزارش کنید.

ب) درصورتی‌که مدار دارای اشکال است، ورودی آزمونی که اشکال را تشخیص می‌دهد، مشخص کنید؛ اگر نیست، توضیح دهید چگونه از صحت مدار خود اطمینان یافتید.

```
`define width 8
`timescale 1ns/1ps

module mult (p, x, y);

  parameter width=`width;
  parameter N = `width/2;
  input[width-1:0]x, y;
  output[width+width-1:0]p;
  reg [2:0] cc[N-1:0];
  reg [width:0] pp[N-1:0];
  reg [width+width-1:0] spp[N-1:0];
  reg [width+width-1:0] prod;
  wire [width:0] inv_x;
  integer kk,ii;

  assign inv_x = {~x[width-1],~x}+1;

  always @ (x or y or inv_x)
  begin
    cc[0] = {y[1],y[0],1'b0};
    for(kk=1;kk<N;kk=kk+1)
      cc[kk] = {y[2*kk+1],y[2*kk],y[2*kk-1]};

    for(kk=0;kk<N;kk=kk+1) begin
      case(cc[kk])
        3'b001 , 3'b010 : pp[kk] = {x[width-1],x};
        3'b011 : pp[kk] = {x,1'b0};
        3'b100 : pp[kk] = {inv_x[width-1:0],1'b0};
        3'b101 , 3'b110 : pp[kk] = inv_x;
        default : pp[kk] = 0;
      endcase
      spp[kk] = $signed(pp[kk]);
      for(ii=0;ii<kk;ii=ii+1)
        spp[kk] = {spp[kk],2'b00};
    end //for(kk=0;kk<N;kk=kk+1)

    prod = spp[0];
    for(kk=1;kk<N;kk=kk+1)
      prod = prod + spp[kk];
  end

  assign p = prod;

endmodule
```

(آ) مدل شبیه سازی شده به این صورت است:

HW1 > multiplier.v

```
1  `define width 8
2  `timescale 1ns/1ps
3
4  module multiplier (
5      p, x, y
6  );
7
8  parameter width = `width;
9  parameter N = `width/2;
10 input  [width-1:0]x, y;
11 output [width+width-1:0] p;
12 reg [2:0] cc[N-1:0];
13 reg [width:0] pp[N-1:0];
14 reg [width+width-1:0] spp[N-1:0];
15 reg [width+width-1:0] prod;
16 wire [width:0] inv_x;
17 integer kk, ii;
18
19 assign inv_x = {~x[width-1], ~x} + 1;
20
21 always @(x or y or inv_x) begin
22     cc[0] = {y[1], y[0], 1'b0};
23     for (kk = 1; kk < N; kk = kk+1)
24         cc[kk] = {y[2*kk+1], y[2*kk], y[2*kk-1]};
25
26     for (kk = 0 ; kk < N; kk = kk + 1) begin
27         case (cc[kk])
28             3'b001 , 3'b010 : pp[kk] = {x[width-1], x};
29             3'b011 : pp[kk] = {x, 1'b0};
30             3'b100 : pp[kk] = {inv_x[width-1:0], 1'b0};
31             3'b101 , 3'b110 : pp[kk] = inv_x;
32             default : pp[kk] = 0;
33         endcase
34
35         spp[kk] = $signed (pp[kk]);
36         for (ii = 0; ii<kk ; ii = ii + 1)
37             spp[kk] = {spp[kk], 2'b00};
38     end
39
40     prod = spp[0];
41     for (kk = 1; kk < N; kk = kk+1) begin
42         prod = prod + spp[kk];
43     end
44 end
45
46 assign p = prod;
47
48 endmodule //multiplier
```

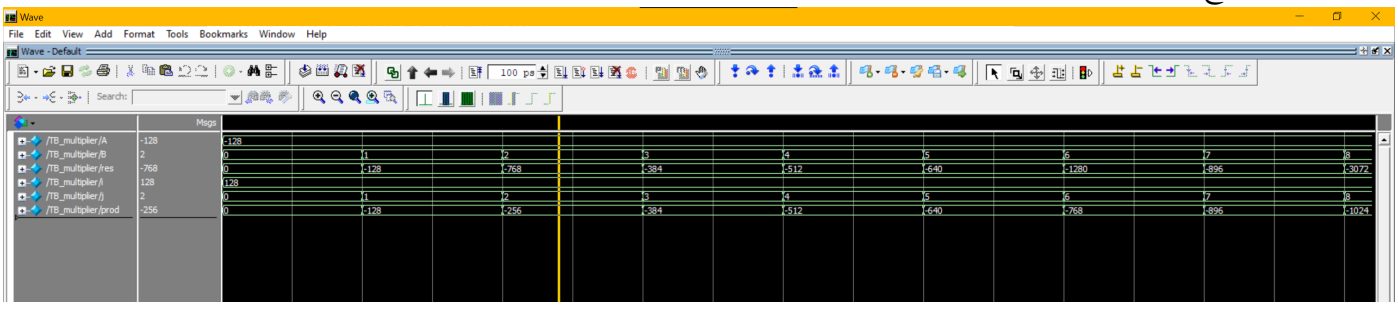
HW1 > TB_multiplier.v

```
1  module TB_multiplier ();
2
3  reg signed [7:0] A, B;
4  reg signed [15:0] prod;
5  wire signed [15:0] res;
6
7  multiplier tb0 (res, A, B);
8  integer signed i, j;
9  initial begin
10
11     for (i = 0; i < 256; i=i+1)
12         for (j = 0; j < 256; j=j+1) begin
13         A = i; B = j;
14         prod = A * B;
15         #10;
16         if (prod - res != 0)
17             $display("result of the verilog program is ",  res
18             ,"\n but product is ", prod, "\n and A and B are    "
19             , A, "     ", B,
20             "\n================================================================================");
21         end
22
23
24     $stop();
25 end
26
27 initial begin
28
29     $dumpfile("TB_multiplier.vcd");
30     $dumpvars;
31 end
32
33
34
35 endmodule //TB_multiplier
36
```

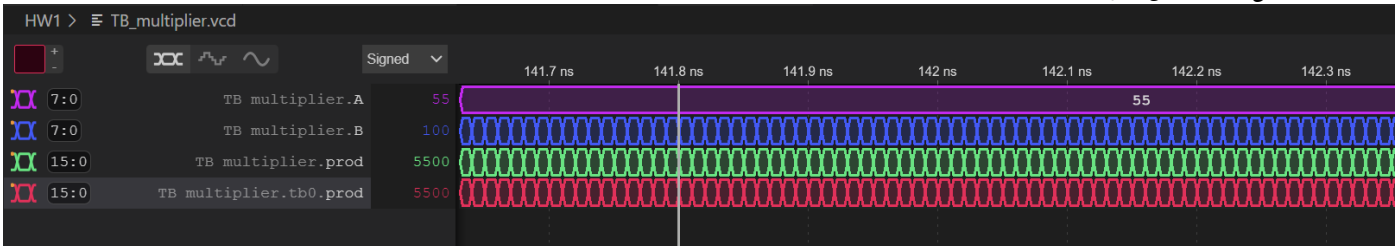
بستر آزمون ساخته شده تمام حالات ضرب را برای دو عدد ۸ بیتی علامت دار شبیه سازی می کند و اگر حاصل مدار با حاصل ضرب برابر نبود این تفاوت را برای ما نشان می دهد. این نمایش در خروجی به این صورت است:

```
Transcript
File  Edit  View  Bookmarks  Window  Help
Transcript
#  but product is   4992
#  and A and B are    -128      -39
# ================================================================================
# result of the verilog program is   4352
#  but product is   4864
#  and A and B are    -128      -38
# ================================================================================
# result of the verilog program is   3840
#  but product is   4352
#  and A and B are    -128      -34
# ================================================================================
# result of the verilog program is  -4096
#  but product is   4096
#  and A and B are    -128      -32
# ================================================================================
# result of the verilog program is  -4224
#  but product is   3968
#  and A and B are    -128      -31
# ================================================================================
# result of the verilog program is  -4864
#  but product is   3840
#  and A and B are    -128      -30
# ================================================================================
# result of the verilog program is  -4480
#  but product is   3712
#  and A and B are    -128      -29
# ================================================================================
# result of the verilog program is  -4608
#  but product is   3584
#  and A and B are    -128      -28
# ================================================================================
# result of the verilog program is  -4736
#  but product is   3456
#  and A and B are    -128      -27
# ================================================================================
# result of the verilog program is  -5376
#  but product is   3328
#  and A and B are    -128      -26
# ================================================================================
# result of the verilog program is  -4992
#  but product is   3200
#  and A and B are    -128      -25
# ================================================================================
# result of the verilog program is   1024
#  but product is   3072
#  and A and B are    -128      -24
# ================================================================================
# result of the verilog program is    896
#  but product is   2944
#  and A and B are    -128      -23
# ================================================================================
# result of the verilog program is   2304
#  but product is   2816
#  and A and B are    -128      -22
# ================================================================================
# result of the verilog program is   1792
#  but product is   2304
#  and A and B are    -128      -18
# ================================================================================
# result of the verilog program is   1280
#  but product is   1792
```

در نمایش موج به این صورت است:



و در فایل vcd به این صورت است:



(ب) همانطور که دیدید این مدار کاملا صحیح نیست و زمانی که $A = -۱۲۸$ است دچار مشکل می شود. به طور خاص می توان به حالت $A = -۱۲۸, B = -۳۸$ اشاره کرد.

```
# ================================================================================
# result of the verilog program is   4352
#  but product is   4864
#  and A and B are    -128      -38
# ================================================================================
```