گزارش پروژه مدار منطقی

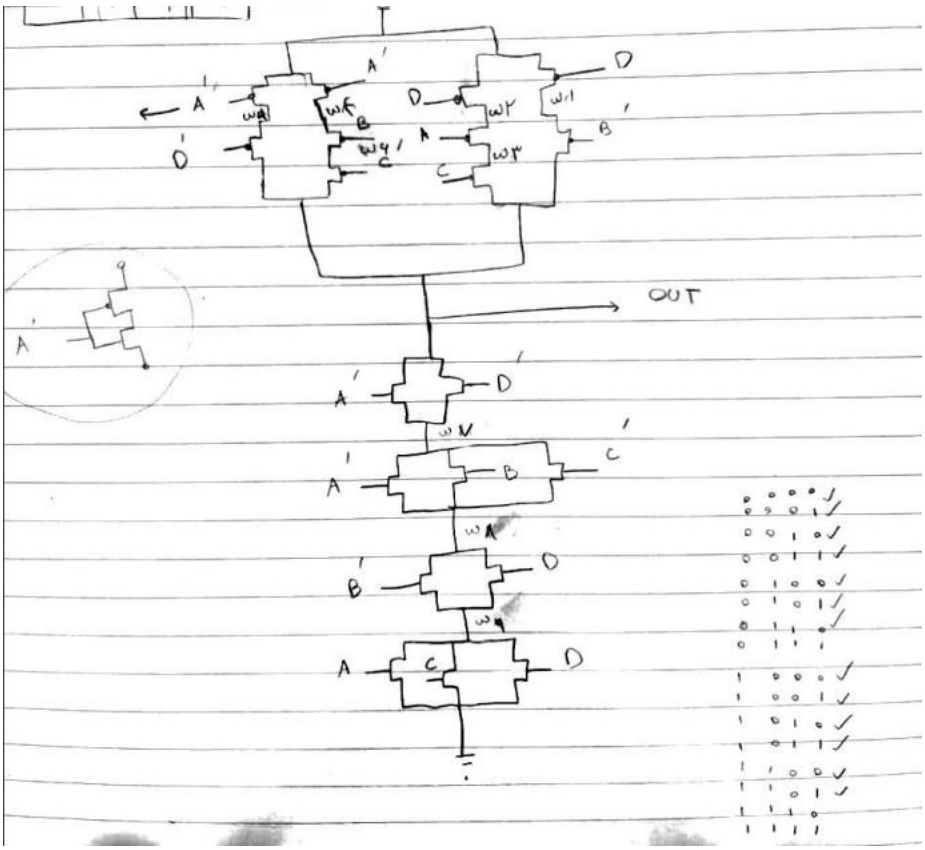
سوال اول:

برای کشیدن فانکشن سوال با ترانزیستور از روش زیر استفاده میکنیم.

1. محاسبه f’ و رسم قسمت nmos با استفاده از آن (مجموعه nmos به زمین وصل میشود)
2. محاسبه دوگان f’ و رسم قسمت pmos با استفاده از آن (مجموعه pmos به power وصل میشود)

$$f' = (A'+D')(A'+B+C')(B'+D)(A+C+D)$$

$$D(f') = A'D' + A'BC' + B'D + ACD$$



کد وریلاگ:

```
module firstQ(a,b,c,d,out);
    input a,b,c,d;
    output out;
    supply0 gnd;
    supply1 power;
    wire wa,wb,wc,wd;
    wire w1,w2,w3,w4,w5,w6,w7,w8,w9;

    pmos(wa,power,a);
    nmos(wa,gnd,a);

    pmos(wb,power,b);
    nmos(wb,gnd,b);

    pmos(wc,power,c);
    nmos(wc,gnd,c);

    pmos(wd,power,d);
    nmos(wd,gnd,d);
```

1. تعاریف

- تعداد wire ها و محل آنها در شکل صفحه قبل نشان داده شده.
- جفت های pmos و nmos نشان داده شده در خط های بعد، برای تعریف a’ و b’ و c’ و d’ است. (شکل در صفحه قبل)

2. رسم بقیه تابع با استفاده از 9 wire که محل قرارگیری آنها در شکل صفحه قبل نشان داده شده است.

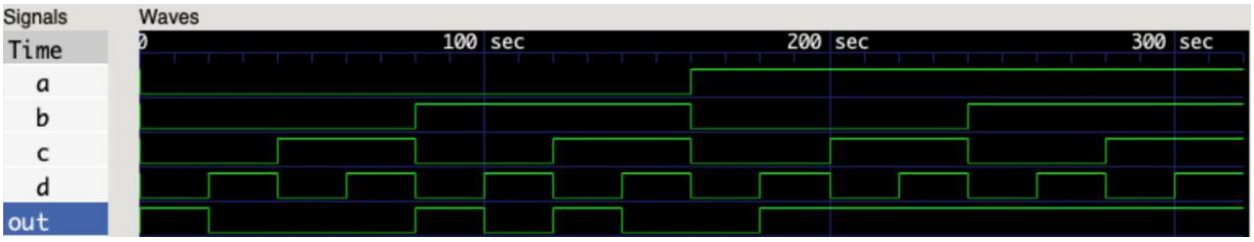
```
    pmos(w5,power,wa);
    pmos(out,w5,wd);

    pmos(w4,power,wa);
    pmos(w6,w4,b);
    pmos(out,w6,wc);


    pmos(w2,power,d);
    pmos(w3,w2,a);
    pmos(out,w3,c);

    pmos(w1,power,d);
    pmos(out,w1,wb);


    nmos(out,w7,wa);
    nmos(out,w7,wd);

    nmos(w7,w8,wa);
    nmos(w7,w8,b);
    nmos(w7,w8,wc);

    nmos(w8,w9,wb);
    nmos(w8,w9,d);

    nmos(w9,gnd,a);
    nmos(w9,gnd,c);
    nmos(w9,gnd,d);

endmodule
```

تست بنچ:

در قسمت تست بنچ همه حالات a,b,c,d بررسی شد. (16 حالت)

```
14      initial begin
15          a = 0;
16          b = 0;
17          c = 0;
18          d = 0;
19          #20
20
21          a = 0;
22          b = 0;
23          c = 0;
24          d = 1;
25          #20
26
27          a = 0;
28          b = 0;
29          c = 1;
30          d = 0;
31          #20
32
33          a = 0;
34          b = 0;
35          c = 1;
36          d = 1;
37          #20
38
39          a = 0;
40          b = 1;
```

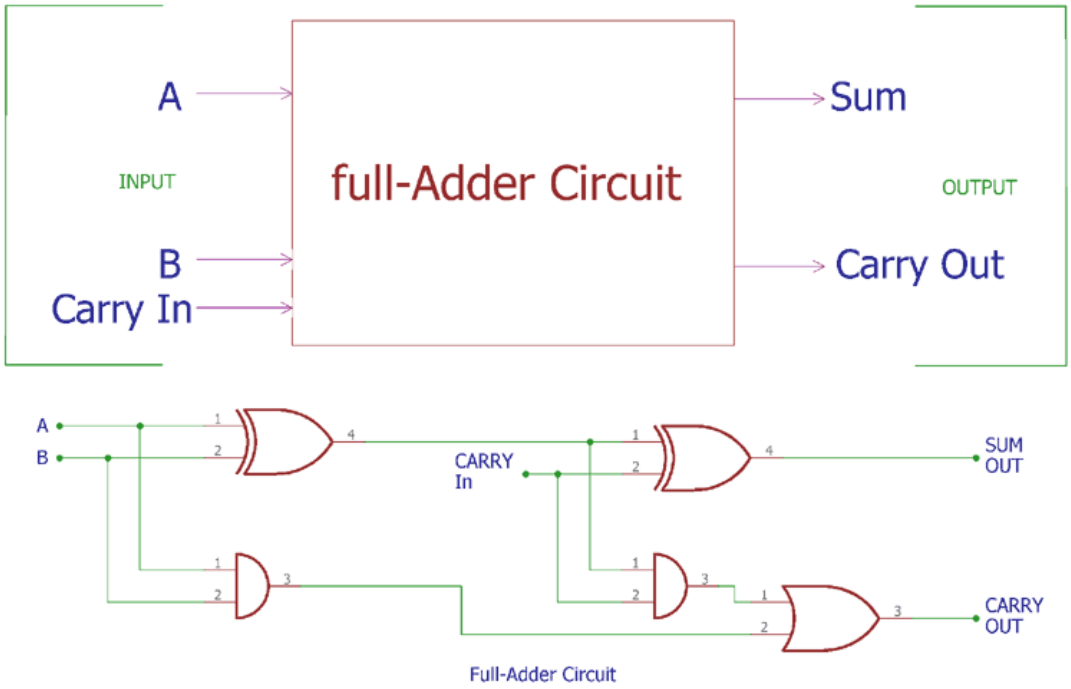
```
39          a = 0;
40          b = 1;
41          c = 0;
42          d = 0;
43          #20
44
45          a = 0;
46          b = 1;
47          c = 0;
48          d = 1;
49          #20
50
51          a = 0;
52          b = 1;
53          c = 1;
54          d = 0;
55          #20
56
57          a = 0;
58          b = 1;
59          c = 1;
60          d = 1;
61          #20
62
63          a = 1;
64          b = 0;
65          c = 0;
```



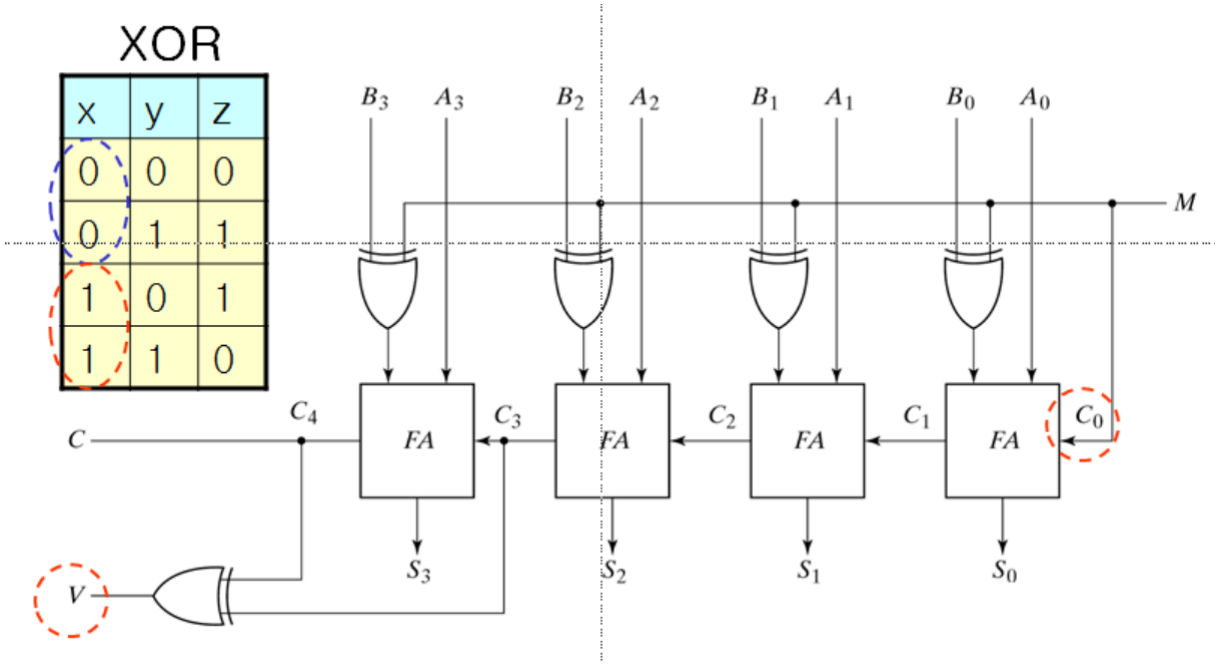
سوال دوم:

نحوه رسم

1. ابتدا 1bit signed full-adder را به شکل زیر ساخته و سپس آن را کسکید میکنیم.



2. نحوه cascade: شکل حاصل از کسکید به 4 بیت مانند زیر میشود.



کد وریلاگ:

1. تعریف متغیرها

```
module secondQ(a0, a1, a2, a3, b0, b1, b2, b3, S0, S1, S2, S3,out,V);
    input a0, a1, a2, a3,b0, b1, b2, b3;
    output S0, S1, S2, S3,out,V;
    wire c1, c2, c3;
    wire X0, X1, X2, X3;
    wire S0, S1, S2, S3;
    supply0 M;
```

2. بدنه اصلی کد: از 4 بخش تشکیل شده. در هر بخش با توجه به فرم کسکید شده full-adder آن را با گیت های منطقی رسم میکنیم.

```
    xor(X0, b0, M);
    wire W1, W2, W3;

    xor(W1, a0, X0);
    and(W2, W1, M);
    and(W3, a0, X0);

    xor(S0, W1, M);
    or(C1, W2, W3);
```

xor اول در حالت کسکید شده وجود دارد (طبق شکل دوم) سپس باید a0 و نتیجه xor (که در اینجا با X0) نشان دادیم ، با استفاده full-adder 1 bit 1 جمع کنیم. پس به بدنه جمع کننده 1 بیتی میرویم تا آن را پیاده سازی کنیم. (طبق شکل اول در صفحه قبل)

درواقع باید هرجا در شکل 1، B میبینیم با X0 جایگذاری کنیم. S0 بیت اول خروجی است.

(برای رسم full-adder تعدادی وایر دیگر هم نیاز بود که با wi تعریف شده است.)

به همین شکل بخش دوم و سوم و چهارم را نیز رسم میکنیم.

```
    xor(X1, b1, M);
    wire W4, W5, W6;

    xor(W4, a1, X1);
    and(W5, W4, c1);
    and(W6, a1, X1);

    xor(S1, W4, c1);
    or(c2, W5, W6);
```

```
    xor(X2, b2, M);
    wire W7, W8, W9;

    xor(W7, a2, X2);
    and(W8, W7, c2);
    and(W9, a2, x2);

    xor(S2, W7, c2);
    or(c3, W8, W9);
```

```
    xor(X3, b3, M);
    wire W10, W11, W12;

    xor(W10, a3, X3);
    and(W11, W10, c3);
    and(W12, a3, X3);

    xor(S3, W10, c3);
    or(C, W11, W12);
```

در آخر هم برای تشخیص overflow از xor نهایی شکل دوم صفحه قبل استفاده میکنیم.

```
  xor(V, out, c3);
```

تست بنچ :

```
    initial begin
        // a = 5
        a0 = 1;
        a1 = 0;
        a2 = 1;
        a3 = 0;

        // b = 1
        b0 = 1;
        b1 = 0;
        b2 = 0;
        b3 = 0;
        #20;

        // a = 7
        a0 = 1;
        a1 = 1;
        a2 = 1;
        a3 = 0;

        // b = 6
        b0 = 0;
        b1 = 1;
        b2 = 1;
        b3 = 0;
        #20;

        // a = -5
        a0 = 1;
        a1 = 1;
        a2 = 0;
        a3 = 1;

        // b = -1
        b0 = 1;
        b1 = 1;
        b2 = 1;
        b3 = 1;
        #20;

        // a = -7
        a0 = 1;
        a1 = 0;
        a2 = 0;
        a3 = 1;

        // b = -6
        b0 = 0;
        b1 = 1;
        b2 = 0;
        b3 = 1;
        #20
```

سوال سوم:

برای مرحله قبل وریلاگ توضیح خاصی وجود ندارد.

کد وریلاگ:

```
module ALU(a,b,op,out);
    input signed [5:0] a,b;
    input signed [1:0] op;
    output signed [5:0] out;
    assign out = op[1]?(op[0]?((a+a+-b) > 0 ? a+a-b  : b-a-a): -b ) : (op[0]? a+b+b+b: (a <<< 2)+(b >>> 1));
endmodule
```

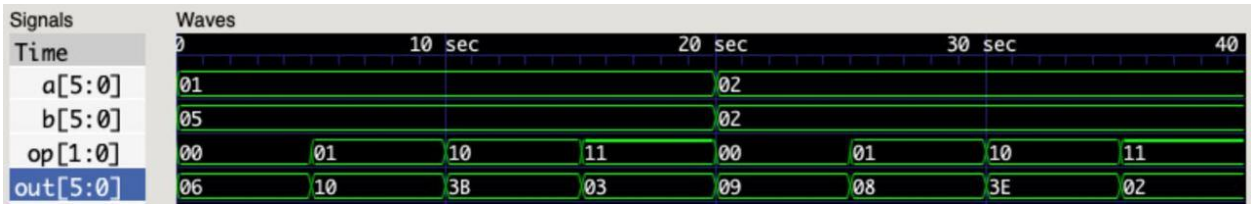
1. ورودی ها: a و b 6 بیتی هستند. (قرار است اعمال روی آنها انجام شود) ، op عملیات را مشخص میکند
2. در قسمت assign به ترتیب حالات زیر از چپ به راست چک شده (ترتیب زمانی مدنظر نیست)

11 ,10, 01,00

تست بنچ:

```
'include "ALU.v";
module ALU_tb;
        reg signed[5:0] a;
        reg signed[5:0] b;
        reg signed[1:0] op;
        wire signed [7:0] out;
        ALU result (a,b,op,out);

        initial begin
                a='d1;
                b='d5;
                op='b00;
                #5;

                a='d1;
                b='d5;
                op=2'b01;
                #5;

                a='d1;
                b='d5;
                op='b10;
                #5;

                a='d1;
                b='d5;
                op='b11;
                #5
```

```
                a='d2;
                b='d2;
                op='b00;
                #5;

                a='d2;
                b='d2;
                op=2'b01;
                #5;

                a='d2;
                b='d2;
                op='b10;
                #5;

                a='d2;
                b='d2;
                op='b11;
                #5

        end
endmodule
```


Signals
Waves
Time
a[5:0]
b[5:0]
op[1:0]
out[5:0]
0
10 sec
20 sec
30 sec
40
01
02
05
02
00
01
10
11
00
01
10
11
06
10
3B
03
09
08
3E
02