دانشکده کامپیوتر

تمرین سری اول

دکتر میثم عبدالهی

الناز رضایی ۹۸۴۱۱۳۸۷

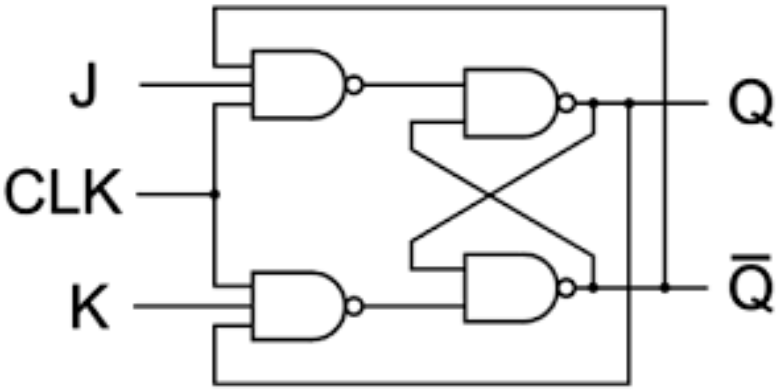
اسفند ۱۴۰۰

سوال ۱ :

برنامه ای به زبان Verilog و به صورت Structural برای K-J flip flop بنویسید و شبیه سازی بن را با استفاده Waveform انجام دهید.

پاسخ ۱:

در این سوال ما نیاز داریم تا طراحی JK-FF را بلد باشیم. همانطور که در شکل زیر نشان داده شده، می توان آن را با استفاده از گیت‌های nand پیاده سازی کرد.



طبق کدهای نشان داده شده در شکل زیر، j و k و clk را به عنوان input تعریف کرده و q و qnot هم خروجی‌های فلیپ فلاپ می‌باشند. همچنین ۲ wire out۱ و out۲ هم خروجی‌های دو nand داخلی شکل بالا می‌باشند.

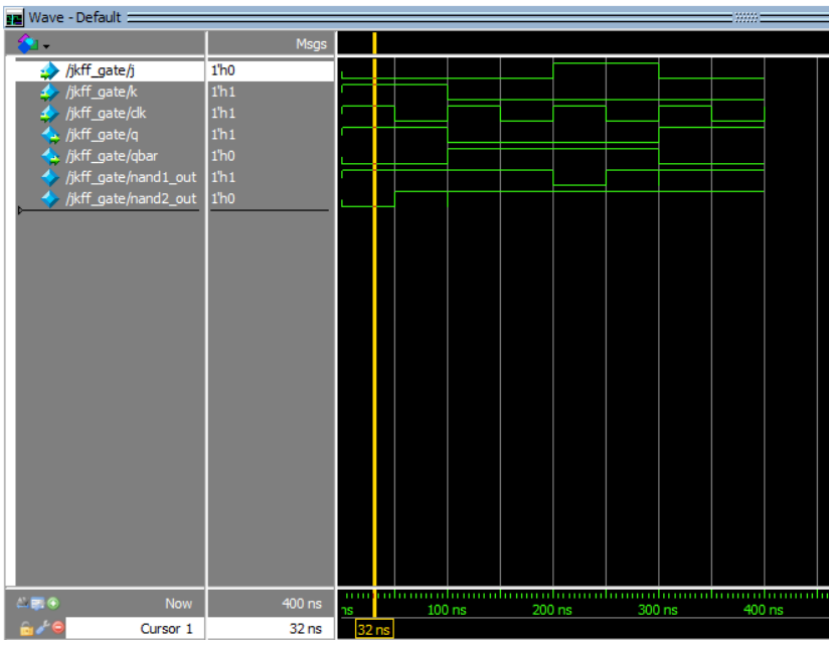
```
1  module JK_FF(
2          j,
3          k,
4          clk,
5          q,
6          qnot
7  );
8
9  input j, k, clk;
10 output q, qnot;
11
12 wire out1;
13 wire out2;
14
15 nand(out1, j, clk, qnot);
16 nand(out2, k, clk, q);
17 nand(q, qnot, out1);
18 nand(qnot, q, out2);
19
20 endmodule
```

طبق جدول JK-FF که در زیر مشاهده می‌کنید، باید wave آن هم خروجیی‌های متناظر با هر ورودی را به ما بدهد.

Truth Table

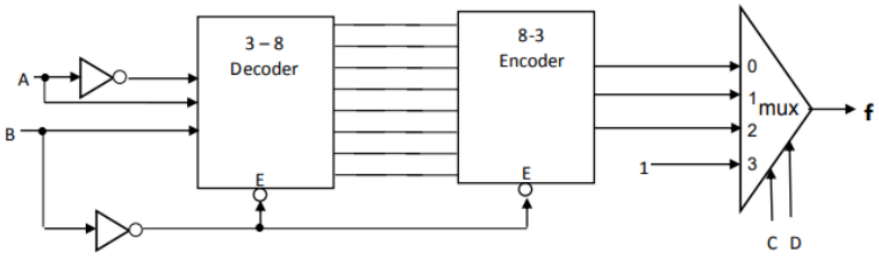
J	K	CLK	Q
0	0	↑	Q_0 (no change)
1	0	↑	1
0	1	↑	0
1	1	↑	$\overline{Q}_0$ (toggles)

wave مشاهده شده در modelsim :



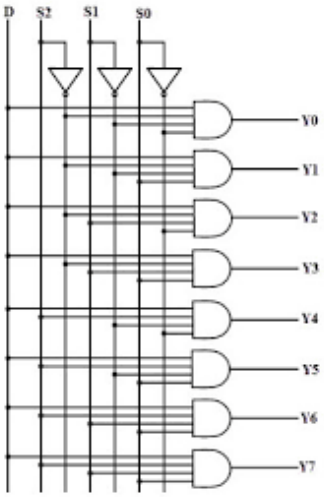
سوال ۲ :

مدار زیر را طراحی کنید بصورتی که A در بن MSB و D در بن LSB است و E سیگنال Enable است و اگر فعال نباشد مقدار خروجی سیگنال های encoder و decoder مقدار صفر خواهند داشت.
می توانید ورودی را یک سیگنال ۵ بیتی در نظر بگیرید و طبق گفته ی سوال A را MSB و D را LSB در نظر بگیرید و یا ۵ سیگنال تک بیتی جداگانه به عنوان ورودی در نظر بگیرید.
برای این سوال شبیهسازی با استفاده از Waveform انجام دهید.



پاسخ ۲:

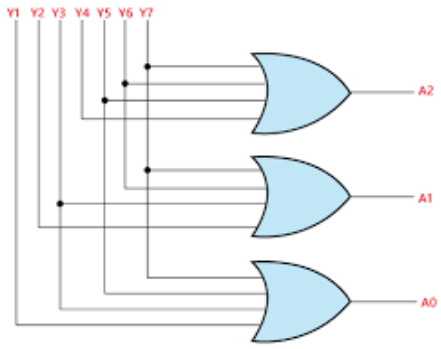
در این سوال، ما نیاز به طراحی دیکودر ۳ به ۸، انکودر ۳ به ۸ و مالتی پلکسر ۴ به ۱ داریم.
دیاگرام داخلی دیکودر ۳ به ۸ به شکل زیر می‌باشد:



اکنون با توجه به شکل بالا، کدهای آن را به شرح زیر پیاده سازی می‌کنیم:

```
module dec(
        A,
        q,
        enable
);

//define variables
input [2 : 0] A;
wire [2 : 0] Abar;
output [7 : 0] q;
input enable;
wire enablebar;

//not inputs & enable
not (Abar[0], A[0]);
not (Abar[1], A[1]);
not (Abar[2], A[2]);
not (enablebar, enable);

//implement decoder
and(q[0], Abar[2], Abar[1], Abar[0], enablebar);
and(q[1], Abar[2], Abar[1], A[0], enablebar);
and(q[2], Abar[2], A[1], Abar[0], enablebar);
and(q[3], Abar[2], A[1], A[0], enablebar);
and(q[4], A[2], Abar[1], Abar[0], enablebar);
and(q[5], A[2], Abar[1], A[0],enablebar);
and(q[6], A[2], A[1], Abar[0], enablebar);
and(q[7], A[2], A[1], A[0], enablebar);

endmodule

```

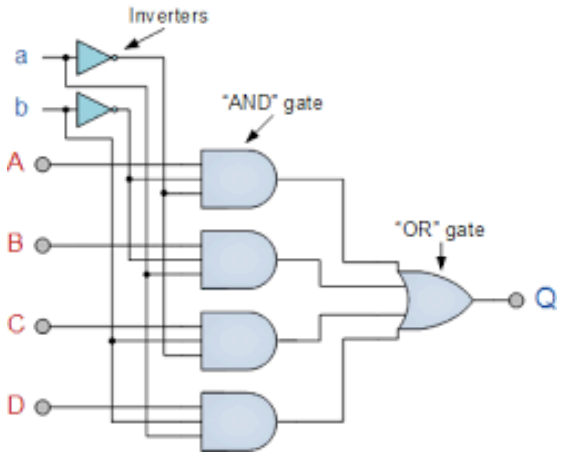
حال به سراغ انکودر می‌رویم. دیاگرام داخلی انکودر نیز در تصویر زیر موجود می‌باشد:



سپس با توجه به شکل بالا، با استفاده از گیت‌های or و همچنین گیت‌های and به منظور active low کردن enable به پیاده سازی encoder خود می پردازیم که کدهای آن در تصویر زیر موجود می‌باشد:

```
1   module enc(
2           A,
3           q,
4           enable
5   );
6
7   //define variables
8   input [7 : 0] A;
9   output [2 : 0] q;
10  input enable;
11  wire [2 : 0] outp;
12  wire enablebar;
13
14  //implement encoder
15  not(enot, en);
16  or(outp[0], A[1], A[3], A[5], A[7]);
17  or(outp[1], A[2], A[3], A[6], A[7]);
18  or(outp[2], A[4], A[5], A[6], A[7]);
19
20  //make it low active
21  and(q[0] ,outp[0], enablebar);
22  and(q[1] ,outp[1], enablebar);
23  and(q[2] ,outp[2], enablebar);
24
25  endmodule
26
```

در اینجا کافیست تا مالتی پلکسر ۴ به ۱ را نیز طراحی کنیم تا مدارمان کامل شود. دیاگرام داخلی مالتی پلکسر نیز در عکس زیر نمایش داده شده است:



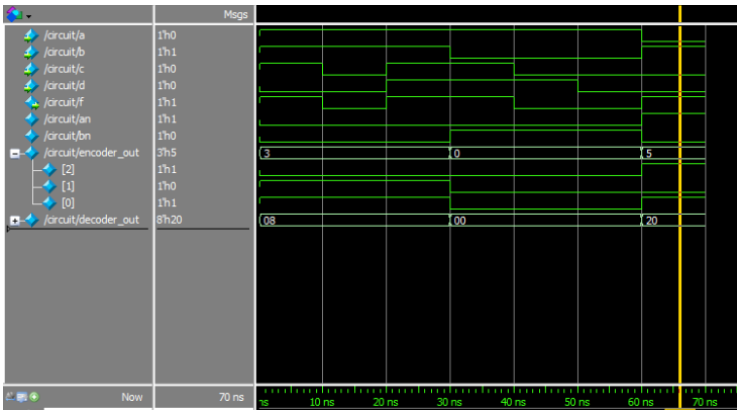
به منظور پیاده سازی شکل بالا، از گیت‌های and و or استفاده می‌کنیم که کد آن در زیر نمایش داده شده است.

```
module mux(
        A,
        selector,
        q
);

//define variables
input [3 : 0] A;
input [1 : 0] selector;
wire [1 : 0] selectorbar;
output q;
wire [3 : 0] outp;

//not selectors
not(selectorbar[0], selector[0]);
not(selectorbar[1], selector[1]);

//implement mux
and(outp[0], A[0], selectorbar[1], selectorbar[0]);
and(outp[1], A[1], selectorbar[1], selector[0]);
and(outp[2], A[2], selector[1], selectorbar[0]);
and(outp[3], A[3], selector[1], selector[0]);
or(q, outp[0], outp[1], outp[2], outp[3]);

endmodule
```

حال با استفاده از گیت‌های طراحی شده در بالا، مدار نهایی را پیاده سازی می‌کنیم:

```
module final(
        a,
        b,
        c,
        d,
        f
);

//define variables
input a, b, c, d;
wire abar, bbar;
output f;
wire [2 : 0] outp1;
wire [7 : 0] outp2;
//implement circuit
not(abar, a);
not(bbar, b);
dec a0(
        .A({abar, a, bbar}),
        .q(outp2),
        .enable(bbar)
);
enc a1(
        .A(outp2),
        .q(outp1),
        .enable(bbar)
);
mux a2(
        .A({1'b1, outp1[0], outp1[1], outp1[2]}),
        .q(f),
        .selector({c,d})
);

endmodule
```

wave مشاهده شده توسط این بخش نیز به شکل زیر می‌باشد:

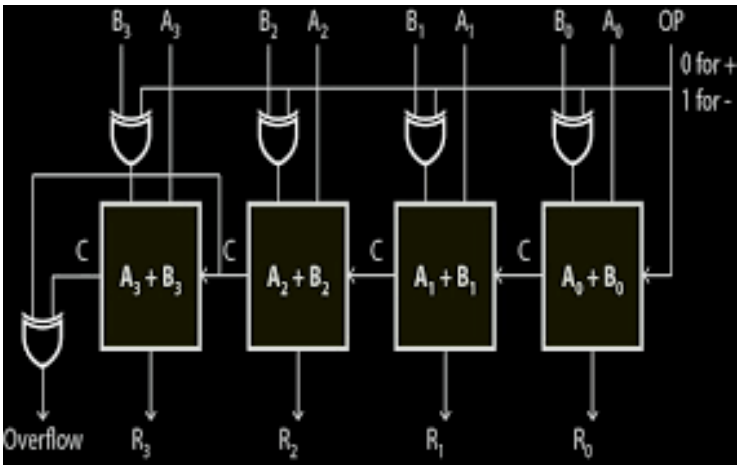


سوال ۳ :

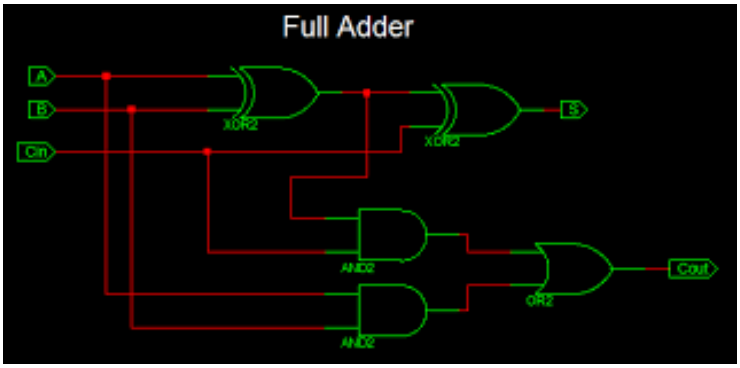
برنامهای به زبان Verilog و به صورت Structural برای یک Subtractor/Adder ۴ بیتی بنویسید که دارای یک سیگنال ورودی به نام select است که با توجه به بن نتیجه جمع یا تفریق را به عنوان خروجی انتخاب میکند. برای این سوال شبیهسازی با استفاده از Waveform انجام دهید.

پاسخ ۳:

در این بخش برای پیاده سازی مدار Subtractor/Adder باید مدار زیر را در نظر بگیریم:



سپس، همانطور که در شکل بالا نیز مشاهده می‌شود، ابتدا نیاز داریم تا adder full را پیاده سازی کنیم که دیاگرام داخلی آن مطابق شکل زیر می‌باشد:



حال برای پیاده سازی full adder فوق کافیست تا با استفاده از گیت‌های xor و and و or کد زیر را در در modelsim بنویسیم:

```
module full_adder(
        cin,
        num1,
        num2,
        sum,
        cout
);

//define variables
input cin, num1, num2;
output sum, cout;
wire outp1, outp2, outp3;

//implement full adder
xor (outp1, num1, num2);
and (outp2, num1, num2);
and (outp3, outp1, cin);
xor (sum, outp1, cin);
or (cout, outp3, outp2);

endmodule

```

سپس به کمک full adder بالا adder/subtracor را با قطعه کد زیر پیاده سازی می‌کنیم.

```
module adder_subtractor(
        num1,
        num2,
        selector,
        sum,
        cout
);

//define variables
input [3 : 0] num1;
input [3 : 0] num2;
input selector;
output [3 : 0] sum;
output cout;
wire [3 : 0] outp1;
wire [3 : 0] outp2;

//implement adder/subtractor
xor (outp1[0], selector, num2[0]);
xor (outp1[1], selector, num2[1]);
xor (outp1[2], selector, num2[2]);
xor (outp1[3], selector, num2[3]);
full_adder a0(selector, num1[0], outp1[0], sum[0], outp2[0]);
full_adder a1(outp2[0], num1[1], outp1[1], sum[1], outp2[1]);
full_adder a2(outp2[1], num1[2], outp1[2], sum[2], outp2[2]);
full_adder a3(outp2[2], num1[3], outp1[3], sum[3], cout);

endmodule
```

شکل موج رسم شده نیز در شکل زیر آورده شده است: