**به نام خدا**

**تمرین پنجم مدارهای منطقی**

**چمران معینی**

**۹۹۳۱۰۵۳**

■ بخش دوم: سوالات اصلی

۱. با استفاده از یک مالتی‌پلکسر ۴ به ۱ و حداقل تعداد گیت‌های دیگر، تابع زیر را پیاده‌سازی کنید.

راهنمایی: سعی کنید ورودی‌ها را هوشمندانه انتخاب کنید تا مجبور نشوید زمان زیادی را صرف امتحان کردن همه حالات ممکن کنید.

$$F(A,B,C,D) = \Sigma m(2,3,6,9,10,13,15) + \Sigma d(0,12,14)$$

می‌دانیم که هر مالتی‌پلکستر ۴:۱ دو سلکتور دارد، پس می‌توانیم دوتا از متغیرهایمان را به عنوان سلکتور انتخاب کنیم. همچنین می‌دانیم که در هر یک از چهار حالتِ سلکتور، دو متغیرِ دیگر نیز می‌توانند چهار حالتِ مختلف داشته باشند.

می‌توانیم این حالت‌های مختلف را به کمک جدول کارنو بررسی کنیم.

| AB / CD | 00 | 01 | 11 | 10 | |
|---|---|---|---|---|---|
| 00 | X | 0 | X | 0 | 0 |
| 01 | 0 | 0 | 1 | 1 | A |
| 11 | 1 | 0 | 1 | 0 | C XNOR D |
| 10 | 1 | 1 | X | 1 | 1 |
| | C | CD’ | 1 | C XOR D | |

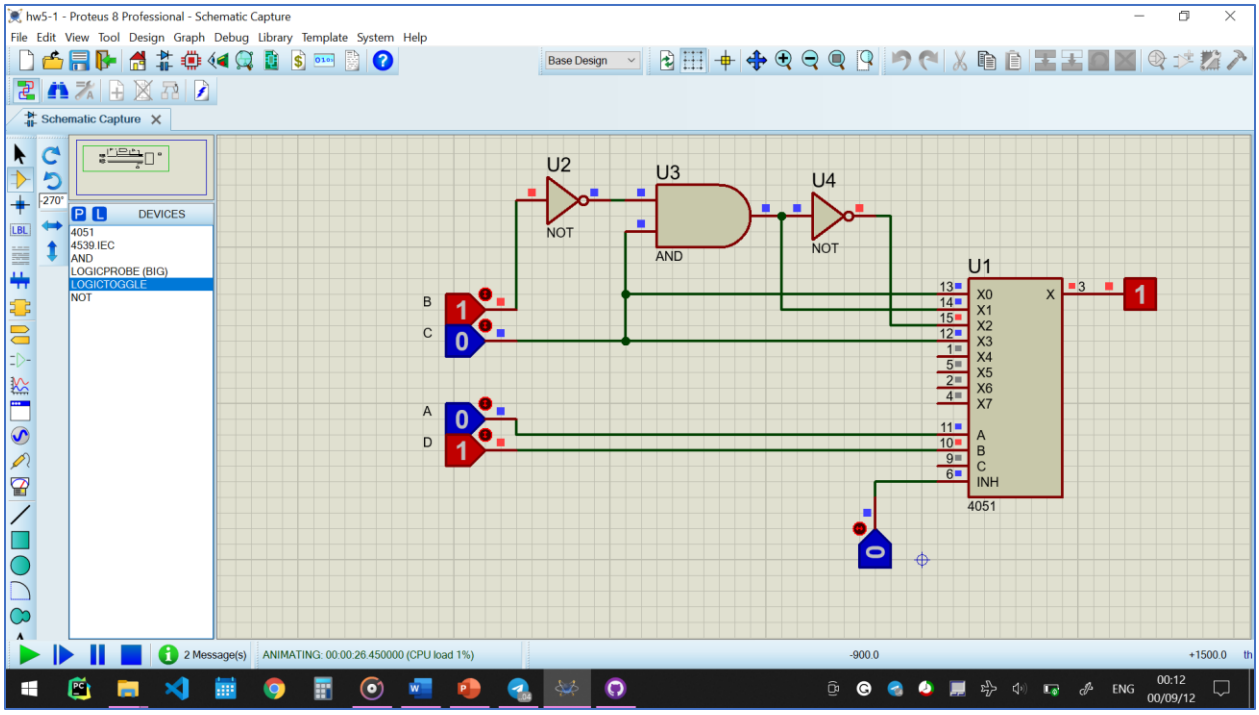
در هر یک از سطرها، و هم چنین در هر یک از سطرها، مقدار دو متغیر ثابت است که می‌توانیم آن دو متغیر را سلکتورها فرض کنیم که در یکی از چهار حالتِ خود هستند. سپس دو متغیرِ دیگر را بررسی می‌کنیم تا ببینیم در آن حالت، به واسطه‌ی چند گیت ورودیِ موردنظر را تولید می‌کنند.

همین کار را برای شش (تعداد حالات ممکن برای انتخابِ دو سلکتور از چهار متغیر) حالتِ دیگر نیز بررسی می‌کنیم.

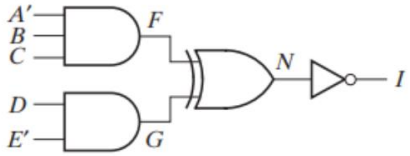
| AC / BD | 00 | 01 | 11 | 10 | |
|---|---|---|---|---|---|
| 00 | X | 1 | 1 | 0 | C |
| 01 | 0 | 1 | 0 | 1 | A XOR C |
| 11 | 0 | 0 | 1 | 1 | A |
| 10 | 0 | 1 | X | X | C |
| | 0 | B’+D’ | B+D’ | D | |

| AD / BC | 00 | 01 | 11 | 10 | |
|---|---|---|---|---|---|
| 00 | X | 0 | 1 | 0 | AD |
| 01 | 1 | 1 | 0 | 1 | A’+D’ |
| 11 | 1 | 0 | 1 | X | A’+D |
| 10 | 0 | 0 | 1 | X | A |
| | C | B’C | (B’C)’ | C | |

با مقایسه‌ی این شش حالت، می‌بینیم بهترین حالت هنگامی‌ست که AD به عنوان سلکتور انتخاب شود، در این حالت تنها به دو گیت نات و یک گیت اند نیاز داریم.

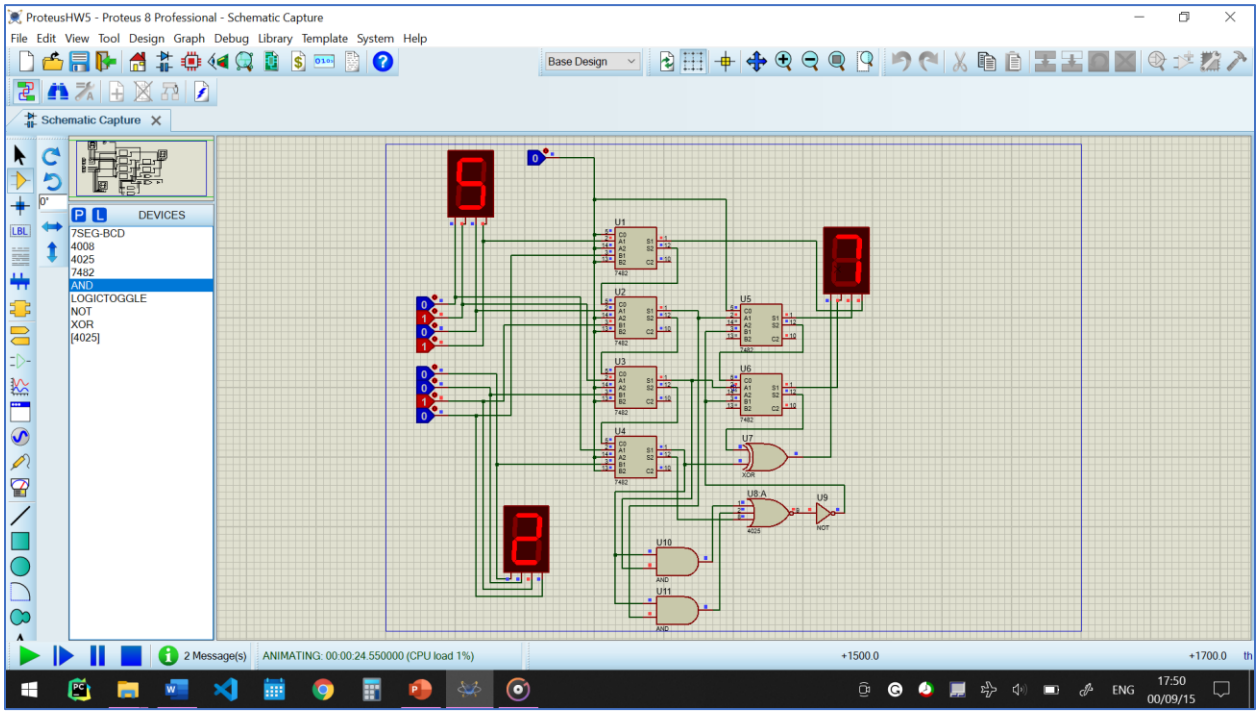


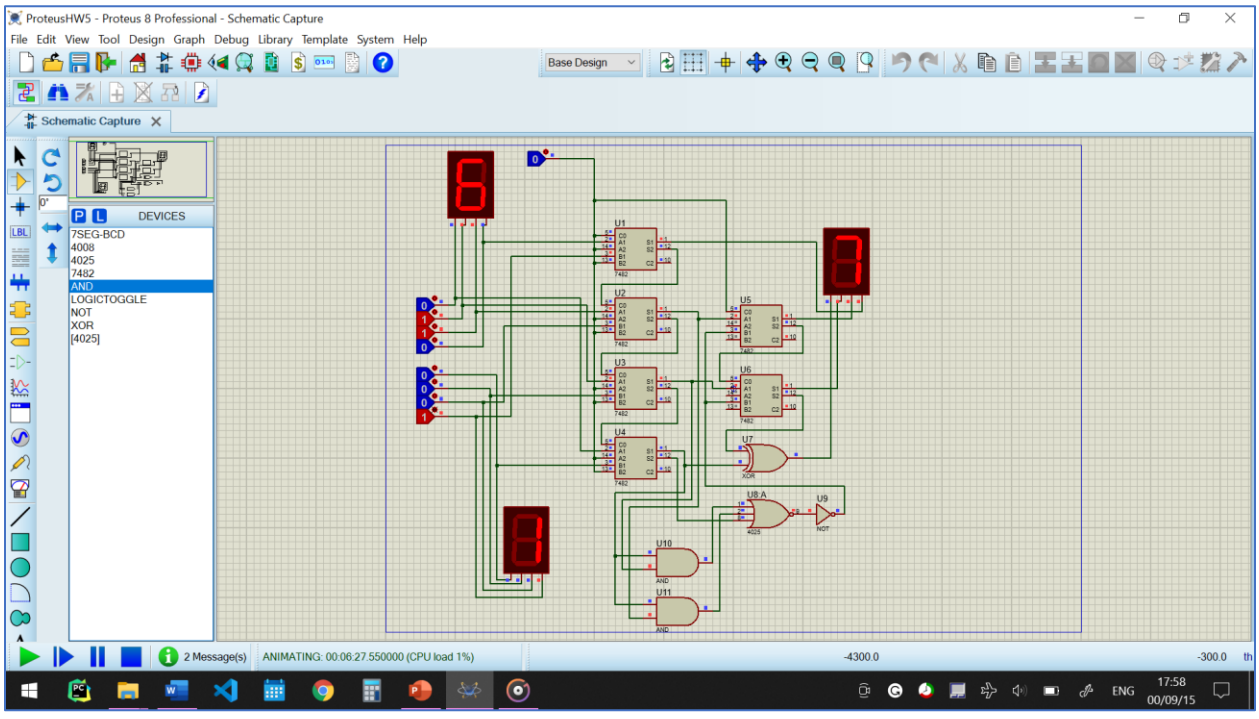
۲. عملکرد مدار زیر را با زبان وریلاگ (Verilog) بصورت dataflow توصیف کنید. (۱۰ نمره)




```
module Two(I, A, B, C, D, E);

    output I;
    input A, B, C, D, E;
    wire F, G, N;

    assign F = (~A) & B & C;
    assign G = D & (~E);
    assign I = ~ (F ^ G);


endmodule
```

۳. به کمک جمع‌کننده BCD که در اسلایدها بررسی شد، یک مدار جمع‌کننده-تفریق‌کننده BCD طراحی کنید. (۲۰ نمره)




این یک مدار جمع‌کننده‌ی BCD است، مانند همان چیزی که در اسلایدها بررسی شد، با این تفاوت که چون FA مناسب برای جمع دو تک بیتی در پروتئوس پیدا نکردم، از FA هایی برای جمع دو عددِ دوبیتی استفاده شده که از s1 های‌شان به عنوان Cout استفاده شده و به A2 B2 های‌شان هم مقدار صفر داده شده است تا مشابه فول‌ادرهایی که در اسلایدها برای جمع دو تک‌بیتی داشتیم، عمل کنند.

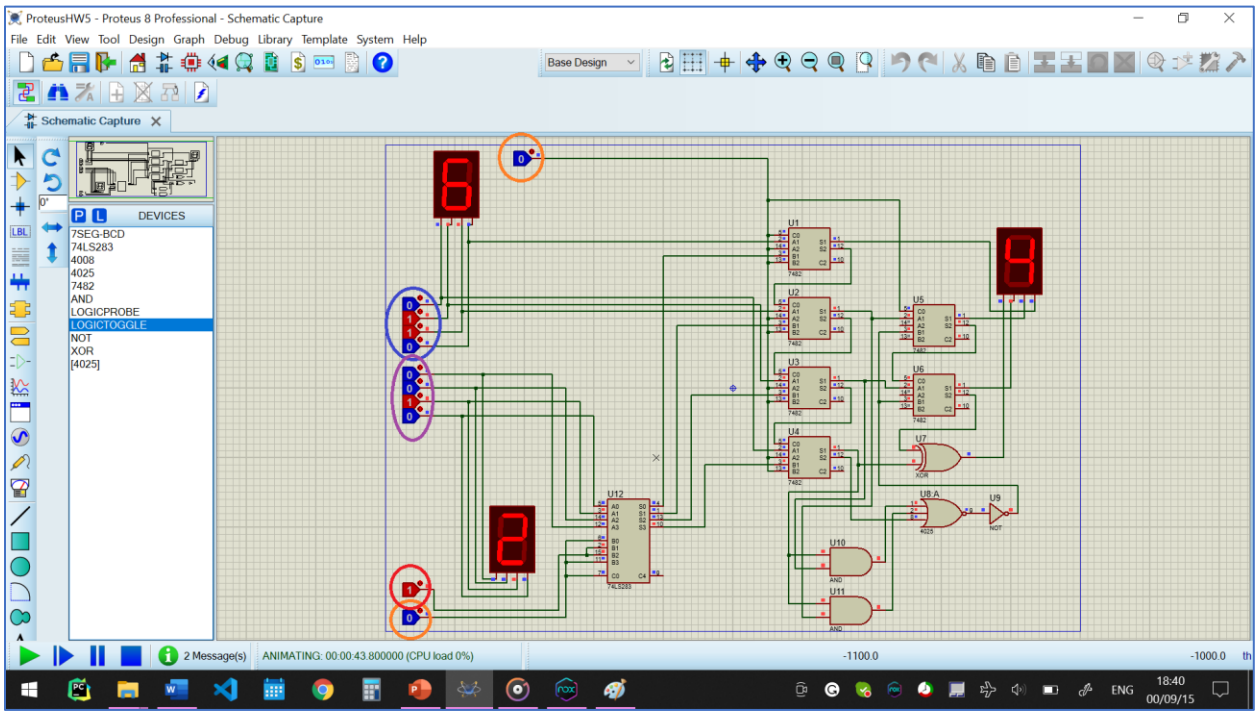
حال قصد داریم عملگر تفریق را نیز به قابلیت‌های این مدار اضافه بکنیم. می‌دانیم که اگر به تفریق این دو عدد ده واحد اضافه کنیم، چیزی که در خروجی می‌بینیم تغییر نخواهد کرد، یعنی:

$$A - B = A - B + 10 = A + (10 - B)$$

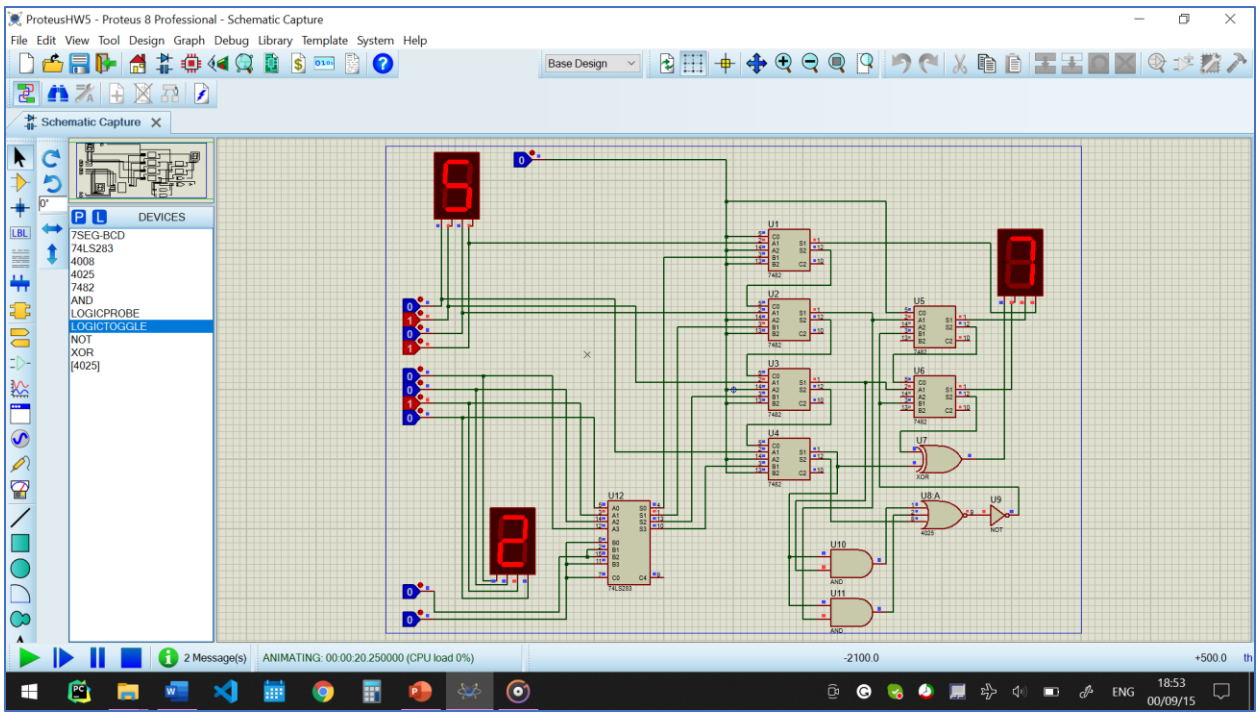
پس می‌توانیم در ورودی‌مان یک سلکتورِ تک‌بیتی اضافه کنیم. اگر صفر بود، یعنی حاصل برابر جمع خواهد بود و همان فرآیندِ قبلی را ادامه می‌دهیم. اگر سلکتور برابرِ یک بود، به جای $B$ از $10 - B$ استفاده می‌کنیم.

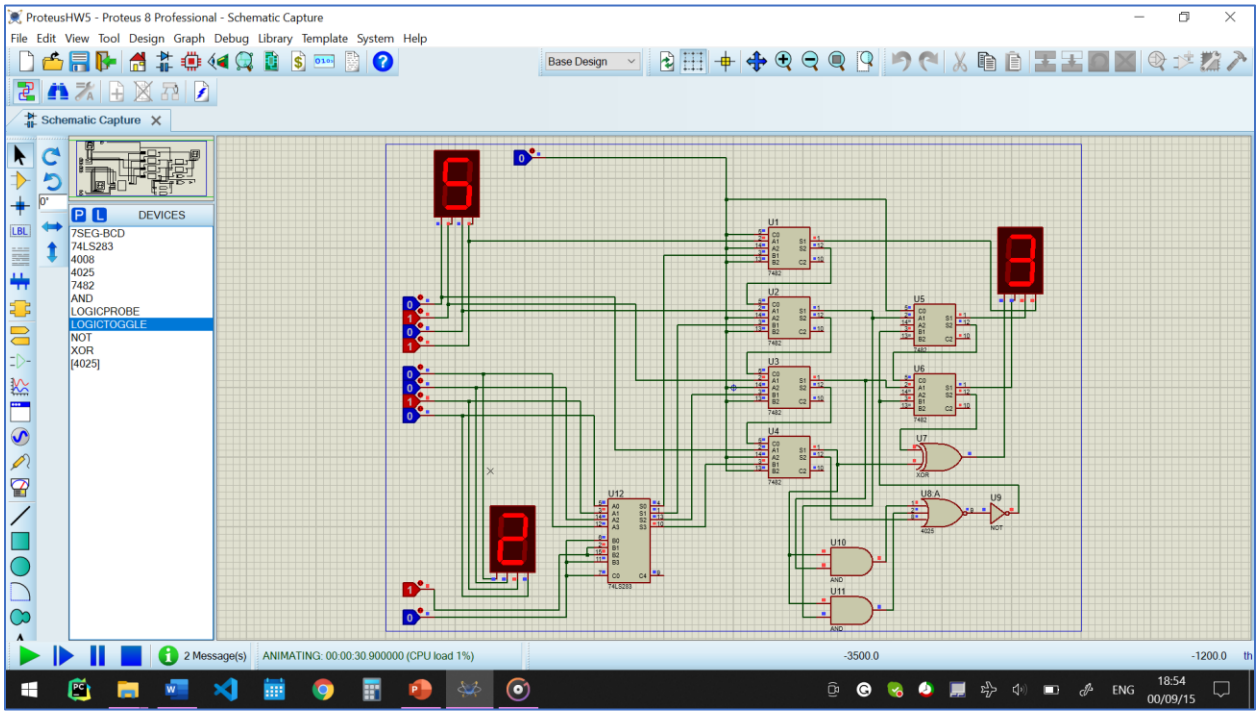
برای به دست آوردنِ $10 - B$ کافی‌ست که از یک ادر استفاده کنیم و عدد $B$ را با مکملِ دوی $1010$ (دهِ باینری) که برابر است با $0110$ جمع بکنیم.

با منطقِ گفته شده، مدار مطلوب به این شکل خواهد بود:




دو 0 ای که با دایره‌ی نارنجی مشخص شده‌اند، صفرهای منطقی هستند. 1 ای که با دایره‌ی قرمز مشخص شده است، سلکتوری‌ست که اگر 1 باشد، حاصلمان $A - B$ خواهد شد و اگر 0 باشد، حاصل‌مان $A + B$ خواهد شد. دایره‌های آبی و بنفش هم به ترتیب اعداد A و B را نشان می‌دهند که هر ارقامِ باارزش‌تر بالاتر قرار گرفته‌اند.

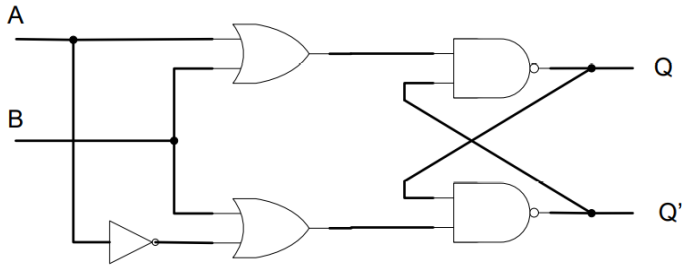





اگر قصد داشته باشیم بیشتر از دو عدد را هم با یکدیگر جمع و تفریق کنیم، می‌توانیم از Cin ادری که کم‌ارزش‌ترین ارقام اعداد را جمع می‌کند، به عنوان Cin استفاده کنیم، برای Cout هم از خروجی گیتِ NOT استفاده می‌کنیم.

۴. الف) نشان دهید مدار زیر مانند یک لچ عمل می‌کند.

ب) معادله مشخصه آن را بدست آورید. (۳۰ نمره)




الف)

در این مدار، هنگامی که مقدار B برابر با صفر باشد، مقدار Q برابر با A′ می‌شود، و هنگامی که مقدار B برابر با ۱ شود، مدار وارد حالتِ HOLD می‌شود و هرچقدر که مقدار A را تغییر دهیم، مقدار Q و Q′ همان مقدار قبلی‌شان خواهند ماند.

چرا؟ وقتی که مقدار B برابر با یک می‌شود، هر دو گیتِ OR خروجیِ یک می‌دهند، به این ترتیب یکی از ورودی‌های هر دو NAND برابر با یک می‌شود، که در این حالت می‌دانیم که X NAND 1 برابر است با X′ ، که می‌دانیم در NAND بالایی، X برابر است با Q′ ، در نتیجه X′ برابر با همان Q خواهد شد و مقدار جدید Q ، همان Q خواهد بود.

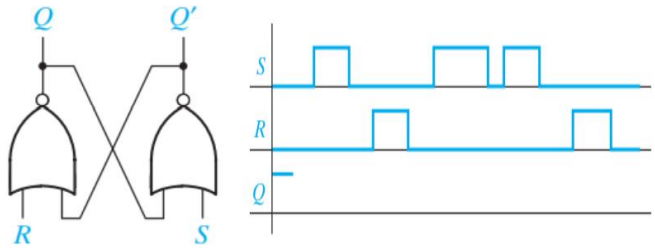
به همین ترتیب در NAND پائینی هم، مقدار X برابر با Q است، پس X′ هم برابر با Q′ می‌شود، یعنی مقدار جدیدی که به Q′ نسبت داده می‌شود، همان Q′ است.

تا این‌جا ثابت کردیم که چرا وقتی B برابر با یک باشد، مدار وارد حالتِ هلد می‌شود و همواره همان پاسخ قبلی را دوباره نشان می‌دهد. حال توضیح می‌دهیم که چرا وقتی B برابر با صفر باشد، مقدار Q همواره برابر با A′ می‌شود. وقتی که مقدار B برابر با صفر می‌شود، هر دو گیتِ OR خروجی‌شان برابر با ورودیِ دیگر خواهد شد چون0 X OR همارزش با X است، به این ترتیب یکی از ورودی‌های دو NAND ، به ترتیب از بالا به پایین برابر با A و A′ خواهد شد. اگر A برابر با یک باشد، مقدار Q برابر با صفر می‌شود و در نتیجه Q′ برابر با یک می‌شود. همچنین اگر A برابر با صفر شد، مقدار Q′ برابر با صفر می‌شود و مقدار Q برابر با یک می‌شود.

ب)

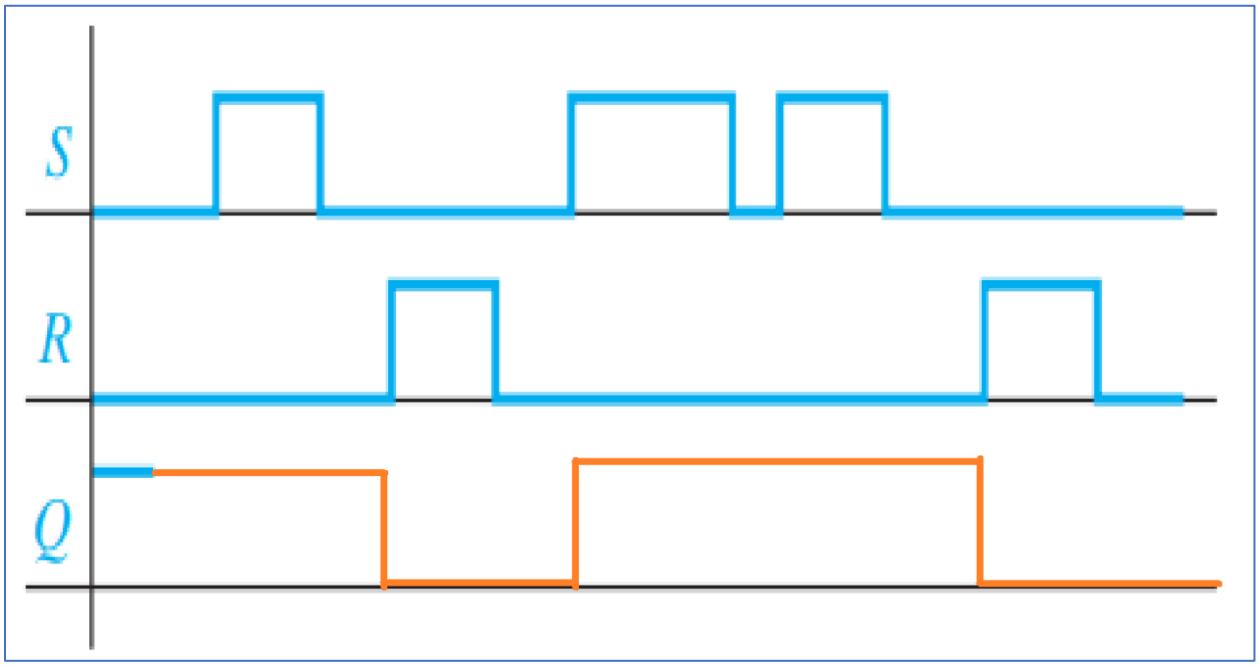
| A | B | Q | Q′ |
|---|---|---|---|
| 0 | 0 | 1 | 0 |
| 0 | 1 | HOLD | HOLD |
| 1 | 0 | 0 | 1 |
| 1 | 1 | HOLD | HOLD |

۵. دیاگرام زمانی لچ SR زیر را رسم کنید. فرض کنید Q در ابتدا برابر با یک است. (۲۰ نمره)

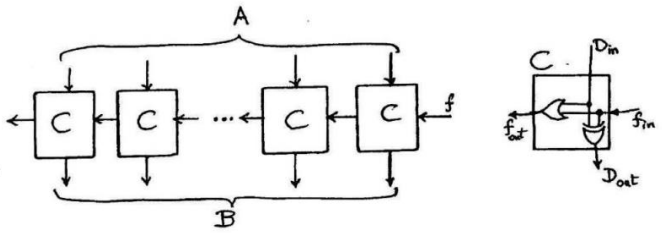



این که در ابتدا مقدار Q برابر با یک است، به این معناست که مدتی پیش، مقدار S یک بوده است.

هنگامی که دوباره مقدار S برابر با یک و دوباره صفر می‌شود، تغییری در مقدار Q ایجاد نمی‌شود، اما هنگامی که R برابر با یک می‌شود، مقدار Q برابر با صفر خواهد شد و این بار Q′ برابر یک می‌شود. بعد از صفر شدن R نیز، Q همچنان همان مقدار را در خود نگه خواهد داشت، تا زمانی که دوباره مقدار S برابر با یک می‌شود که این باعث می‌شود دوباره Q برابر با یک شود و این مقدار را تا زمانی که R برابر با صفر است، در خود نگه می‌دارد و نهایتا هنگامی که دوباره R برابر با یک می‌شود، دوباره مقدارQ برابر با صفر می‌شود و تا زمانی که مقدار S برابر با صفر است، این مقدار را در خود نگه می‌دارد.



۶. عملکرد مدار زیر را شرح دهید. (۲۰ نمره)




هنگامی که **f برابر با یک** باشد، هر یک از بیت‌های B برابر با متضادِ بیت متناظر از A خواهد بود، به بیانِ دیگر مدارمان مکملِ یکِ ورودی‌اش را خروجی می‌دهد.

هنگامی که **f برابر با صفر** باشد، تا جایی که بیت‌های A (از شروع به پایان / از راست به چپ) برابر با ۰ باشند، هر یک از بیت‌های B برابر با بیت متناظر از A خواهند بود، اولین بیتِ A که برابر با ۱ بشود نیز، خروجیِ متناظرش برابر آن خواهد بود و مقدارِ یک را خواهد داشت، اما از آن‌جا به بعد تمام بیت‌های B برابر با مقدار متضاد بیتِ متناظر از A خواهند شد. می‌بینیم که این همان روشی‌ست که برای یافتنِ مکمل دوی اعداد استفاده می‌کردیم، که از سمت راست شروع می‌کردیم و تمام صفرها و اولین یک را رد می‌کردیم و بعد از آن تمام ۱ ها را ۰ ، و تمام ۰ ها را ۱ می‌کردیم. پس در این حالت، مدار مکملِ دوی ورودی‌اش را خروجی می‌دهد.

اما چطور و چگونه؟

در حالتی که **f برابر با یک** باشد، fout در اولین C مان هم برابر یک خواهد شد. به این ترتیب، fin بعدی و پس از آن تمام fin ها و تمام fout ها برابر با یک می‌شوند. هنگامی که همه‌ی fin ها برابر با یک هستند، مقدار B برابر با A xor 1 یا همان A’ خواهد شد، به این ترتیب مکمل یک تولید می‌شود.

در حالتی که **f برابر با صفر** باشد، تا جایی که بیت‌های A هم برابر با صفر هستند، هم 0 xor 0 برابر با صفر است و هم 0 and 0 ، در نتیجه تمام خروجی‌ها و fin و fout ها نیز 0 خواهند بود. هنگامی که اولین بیتِ A برابر با یک شود، آن‌گاه 1 xor 0 هم برابر با یک می‌شود و از آن‌جا که

1 and 0 برابر با 1 است، از این‌جا به بعد تمام fin ها و fout ها برابر با ۱ خواهند شد و مطابقِ همان منطقِ بندِ قبلی، هر یک از بیت‌های B برابر با NOT بیتِ متناظرِ A خواهد شد.

پس به طور خلاصه می‌توان گفت این مدار می‌تواند مکملِ یک و مکملِ دوی اعداد را تولید کنند.

۷. حداکثر با چهار عدد FA و چهار عدد HA مداری طراحی کنید که یک عدد ۵-بیتی را در ورودی دریافت نموده و آن را در عدد ثابت ۳۷ ضرب کند. استفاده از هیچ عنصر دیگری غیر از چهار عدد FA و چهار عدد HA مجاز نیست. (۳۰ نمره)

باید ۳۷ را به شکلِ جمعی از توان‌های دو در بیاوریم:

$$37 = 32 + 4 + 1 = 2^5 + 2^2 + 2^0$$

می‌دانیم که با ضرب شدنِ دو در یک عددِ باینری، یک صفر به سمتِ راستِ عددمان اضافه می‌شود، یعنی حاصل ضرب هر عدد ۵-بیتی در ۳۷ را می‌توان این‌گونه به دست آورد:

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | A[4] | A[3] | A[2] | A[1] | A[0] |
| | | | | A[4] | A[3] | A[2] | A[1] | A[0] | | |
| | A[4] | A[3] | A[2] | A[1] | A[0] | | | | | |
| ANSWER | A[4] | A[3] | A[2]<br>+ Cout | A[3] + A[0]<br>+ Cout(A[4] + A[2]) | A[3] + A[0]<br>+ Cout(A[4] + A[2]) | A[4] + A[2]<br>+ Cout(A[3] + A[1]) | A[3] + A[1]<br>+ Cout(A[2] + A[0]) | A[2]<br>+<br>A[0] | A[1] | A[0] |

تفاوت FA و HA در این است که در HA دو بیت را با یکدیگر جمع می‌کنیم و یک بیت کری و یک بیتِ سام را در خروجی می‌دهیم، اما در FA علاوه بر این‌ها، همچنین می‌توانیم یک Cin هم بگیریم، یعنی توانایی جمعِ سه بیت را داریم، پس برای بخش‌های بنفشِ جدول که جمع دو متغیرند، از HA و برای بخش‌های آبی که جمع سه متغیرند، از FA استفاده می‌کنیم.

فرض می‌کنیم که HA و FA به این ترتیب ورودی بگیرند:

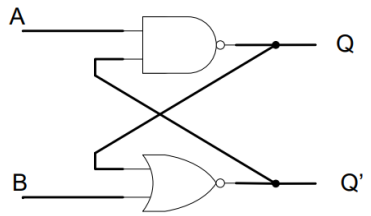
```
module HA(
    output Sum,
    output Cout,
    input x,
    input y
);
```

```
module FA(
    output Sum,
    output Cout,
    input x,
    input y,
    input Cin
);
```

اگر عدد ورودی را A بنامیم و خروجی را B بنامیم، کد وریلاگ مدار طراحی شده، به این شکل خواهد بود:

```
module x37(b, a);

    output b.
    input a;

    wire c0, c1, c2, c3, c4, c5, c6, c7;

    assign b[0] = a[0];
    assign b[1] = a[1];


    HA ha0(b[2], c0, a[2], a[0]);

    FA fa0(b[3], c1, a[3], a[1], c0);
    FA fa1(b[4], c2, a[4], a[2], c1);
    FA fa2(b[5], c3, a[0], a[3], c2);
    FA fa3(b[6], c4, a[1], a[4], c3);

    HA ha1(b[7], c5, a[2], c4);
    HA ha2(b[8], c6, a[3], c5);
    HA ha3(b[9], c7, a[4], c6);

endmodule
```

۸. جدول مشخصه مدار زیر را بدست آورید. (۲۰ نمره)




در تمام حالاتِ تعریف شده‌ی این مدار، مقدار Q برابر با 1 است، برای اثباتِ این موضوع، فرض می‌کنیم که حالتی وجود داشته باشد که در آن Q برابر با صفر باشد. در این حالت، باید هر دو ورودیِ گیتِ NAND برابر با یک باشند، یعنی A و Q’ برابر با یک باشند. اگر Q’ برابر با یک باشد، آن‌گاه باید هر دو ورودیِ گیت NOR برابر با صفر باشد. ورودی‌ای که از Q می‌آید که برابر با صفر فرض شد، پس B هم باید برابر با صفر باشد. پس تنها در صورتی Q برابر با صفر می‌شود که A برابر با یک باشد و B برابر با صفر باشد، که در حالتی که این دو ورودی این مقادیر را داشته باشند، نه یک ورودیِ یک برای فعال شدنِ گیت NAND کافی‌ست و نه یک ورودیِ صفر برای فعال شدنِ گیت NOR کافی‌ست، پس فرضی که داشتیم یعنی صفر شدنِ Q به صورت طبیعی هرگز اتفاق نمی‌افتد، مگر این که یک منبع خارجی در ابتدا مقدار صفر با به Q یا یک را به Q’ متصل کند. به این ترتیب جدول مشخصه‌ی مدار به طور طبیعی به این شکل است:

| A | B | Q | Q’ |
|---|---|---|---|
| 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 |