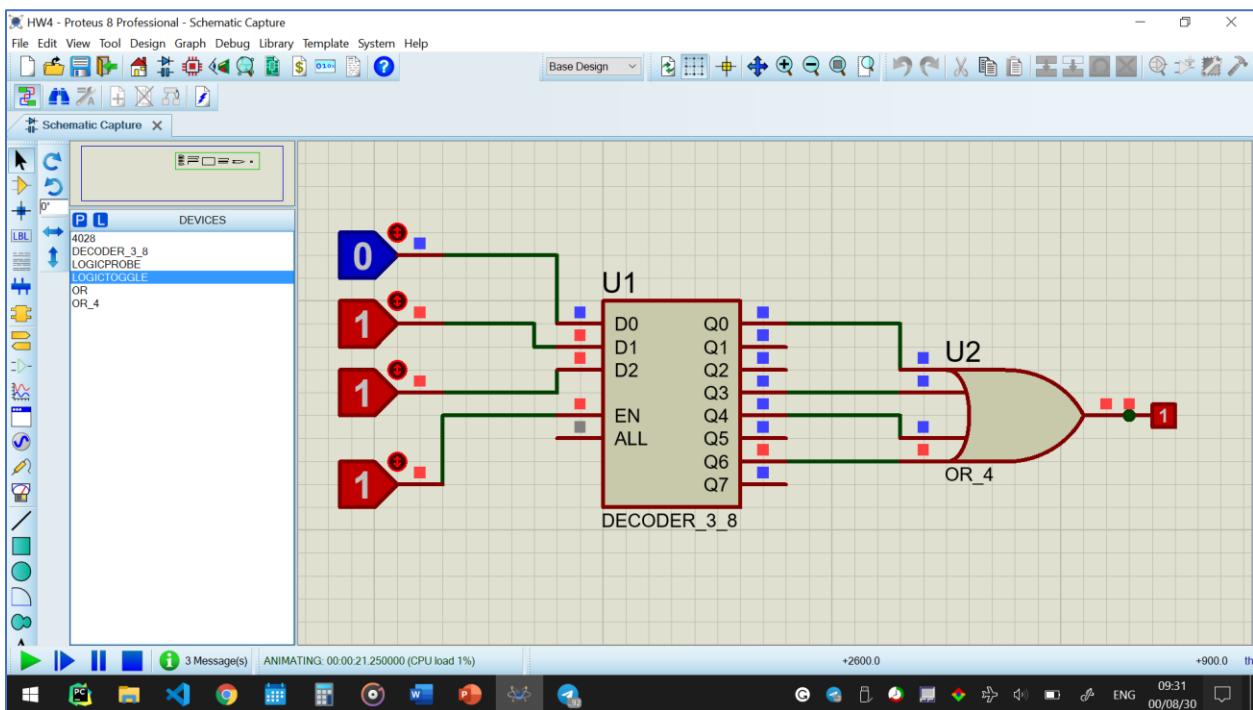


۱. تابع زیر را به کمک یک دیکدر ۳:۸ با خروجی فعال-بالا پیاده‌سازی کنید. (۱۰ نمره)

$$F(A, B, C) = \sum m(0, 3, 4, 6)$$

ابتدا به کمک دیکدر، تشخیص می‌دهیم که متغیرهای تابع مان چه مقداری را نشان می‌دهند، سپس اگر آن مقدار یکی از مین‌ترمها بود، خروجی ما صحیح خواهد بود، پس خروجی‌هایی که با مین‌ترم‌ها هماندیش هستند را با یکدیگر OR می‌کنیم.



(ورودی ALL، در صورتی که صحیح باشد تمام خروجی‌ها را صحیح می‌کند که ما اینجا به آن نیازی نداریم و فرض می‌کنیم وجود ندارد، همچنین می‌توانستیم یک مقدار ۰ را به آن متصل کنیم)

۲. یک دیکدر ۵:۳۲ با استفاده از چند دیکدر ۳:۸ و یک گیت NOT طراحی کنید. فرض کنید هر دیکدر ۳:۸ یک Enable-بالا و یک Enable-پایین دارد. (۲۰ نمره)

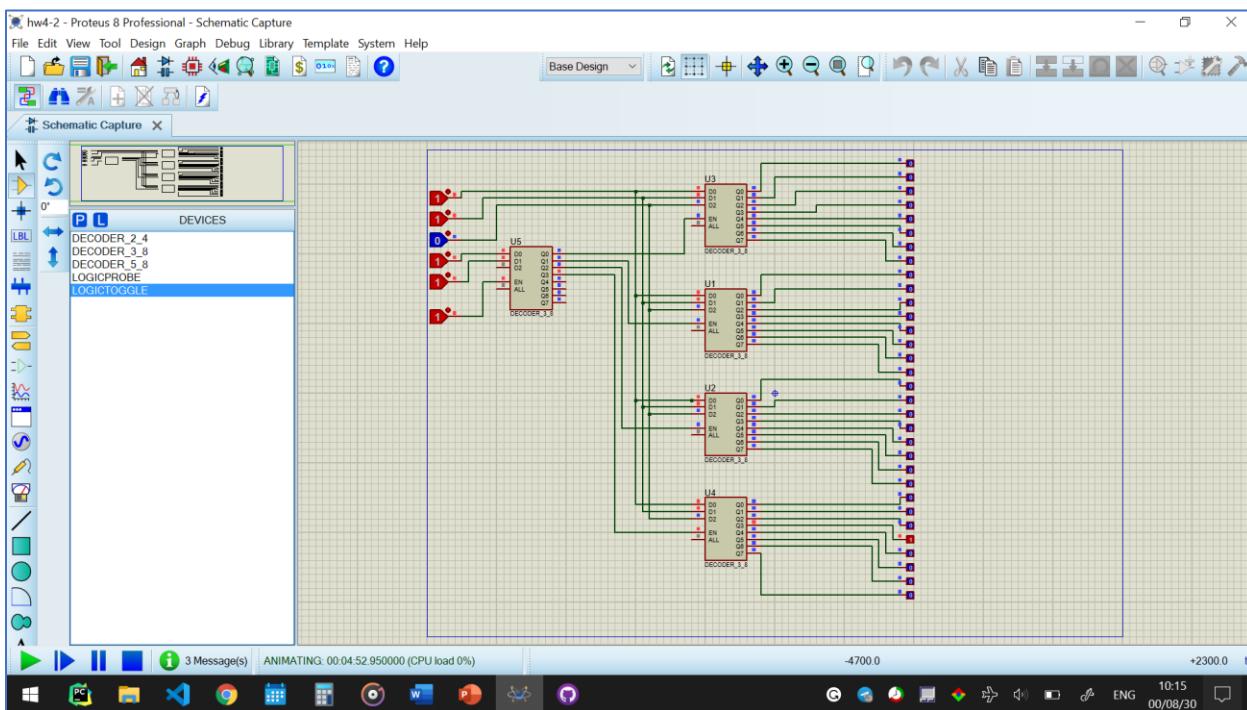
می‌دانیم که دیکدر ۵:۳۲، ۵ ورودی دارد و ۳۲ خروجی، پس باید پنج ورودی و سی و دو خروجی را هندل کنیم. ورودی‌هایمان را $i0$ تا $i4$ و خروجی‌هایمان را $i0$ تا $i31$ می‌نامیم.

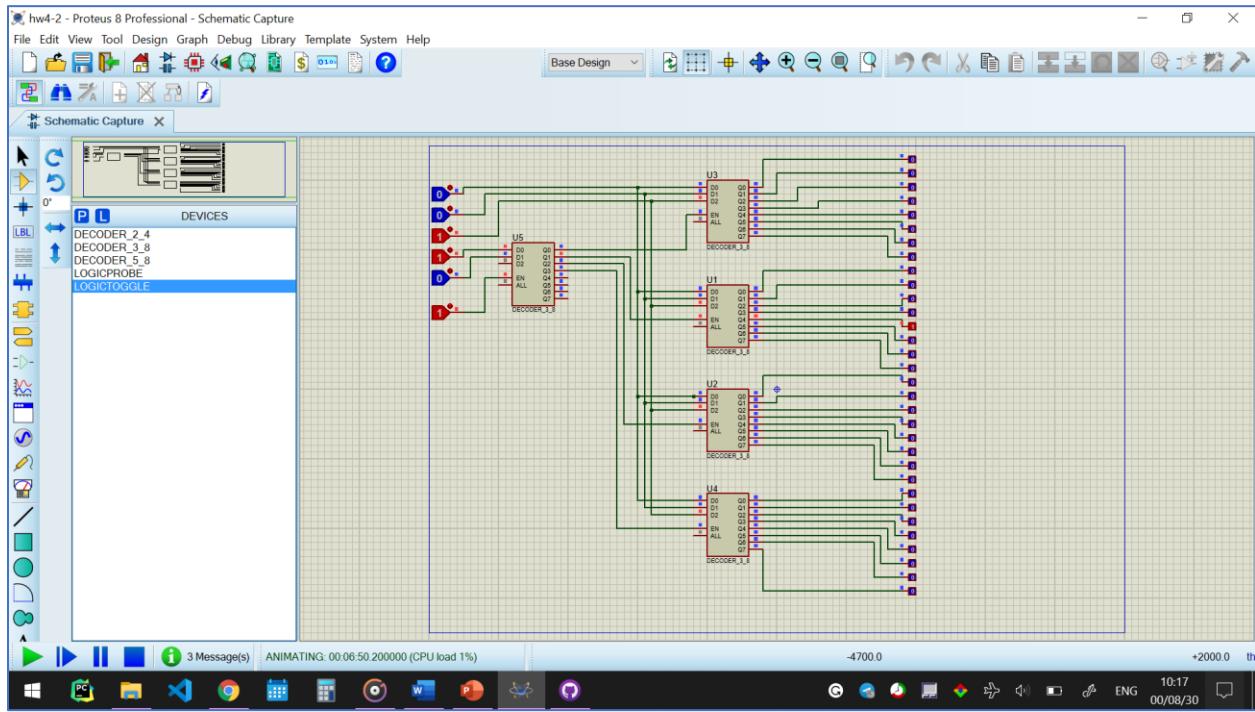
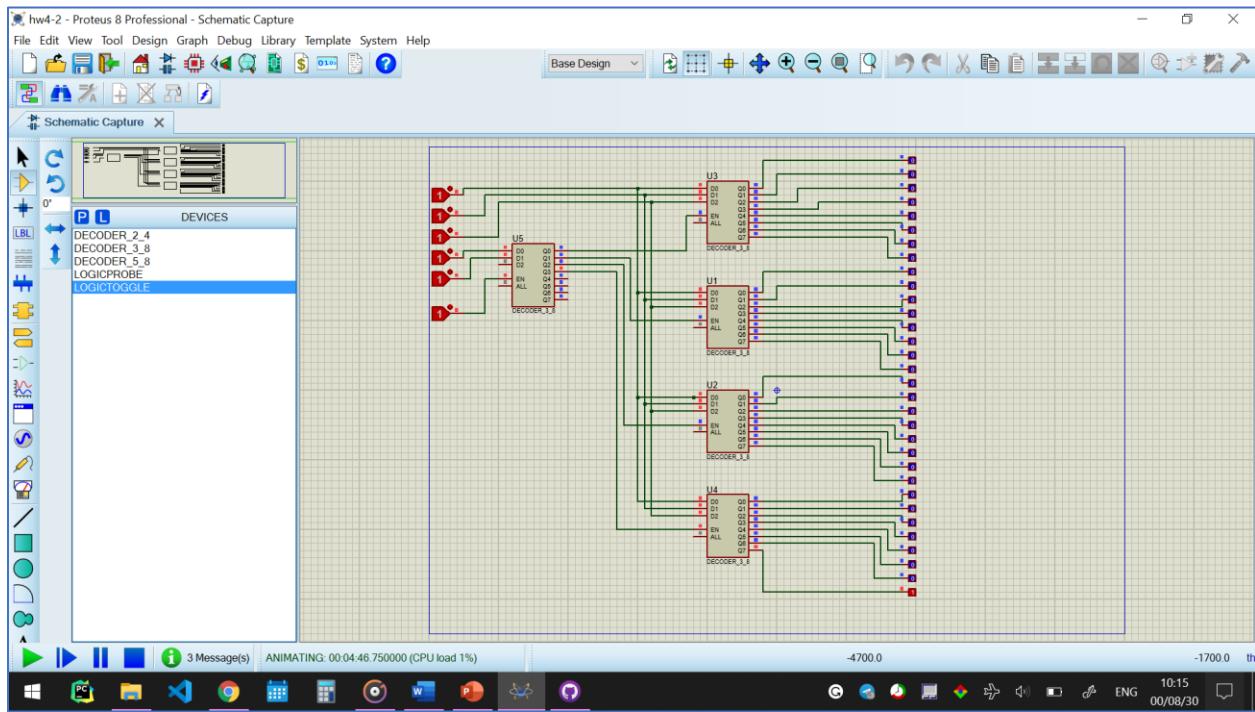
با توجه به این که ۳۲ خروجی داریم، می‌فهمیم که باید ۴ دیکدر ۳:۸ را در کنار یکدیگر قرار دهیم تا این خروجی‌ها را تولید کنند.

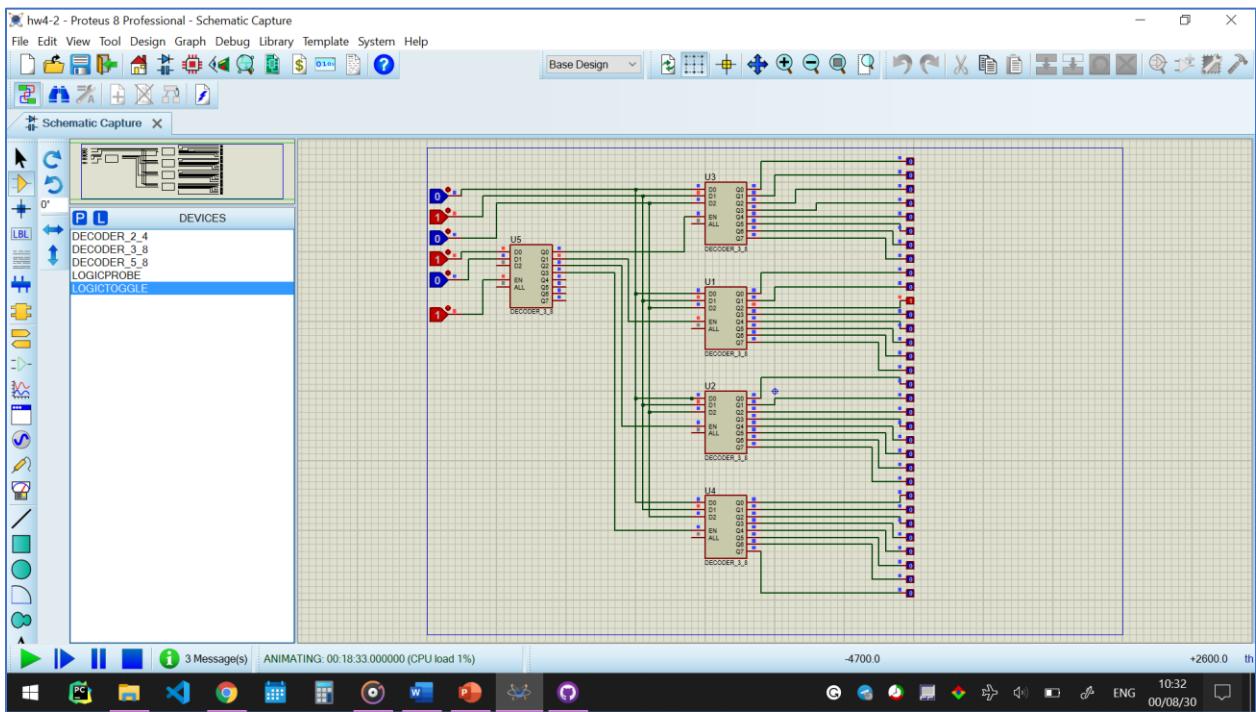
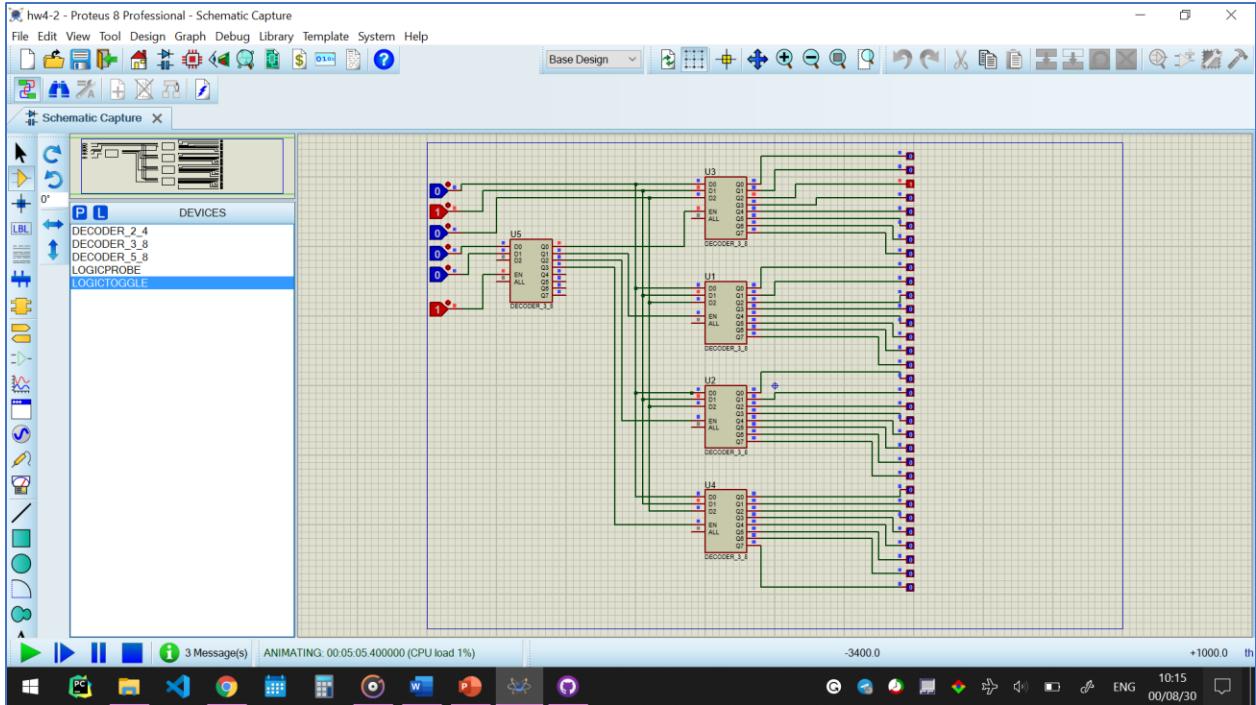
هنگامی که $i3$ مقدار ۰۰ را داشته باشد، خروجی صحیح یکی از $i0$ تا $i7$ خواهد بود، هنگامی که $i3$ مقدار ۰۱ را داشته باشد، خروجی صحیح یکی از $i8$ تا $i15$ خواهد بود و... پس از یک دیکدر برای یافتن مقدار $i3$ استفاده می‌کنیم، و خروجی‌های آن را به ترتیب به Enable‌های فعال‌بالا و Not‌شان را به فعال‌پائین‌ها متصل می‌کنیم.

تا این جای کار موفق شدیم که دیکدری که قرار است خروجی ۱ را نشان دهد را پیدا کنیم. حال باید ورودی‌های دیکدرها را طوری مقداردهی کنیم که اگر Enable‌شان فعال بود، خروجی صحیح را ۱ کنند. در این مرحله، دیگر نیازی به $i3$ نداریم و $i0$ تا $i2$ در این بخش تاثیر گذارند، با متصل کردن آن‌ها به ترتیب به هر سه ورودی چهار دیکدری که قرار است خروجی نهایی را نشان دهند، هنگامی که یک دیکدر در حالت فعال بودند، دیکدر ما مقدار صحیح نهایی را نشان خواهد داد.

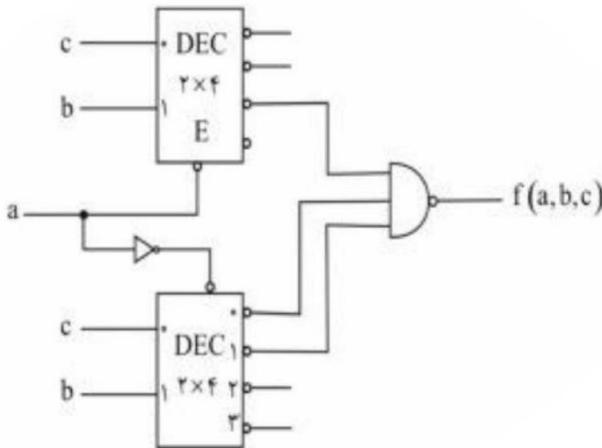
در شماتیک‌های زیر، Enable فعال‌پائین لحظه نشده چون متساقنه Device مناسب برای نمایش آن را پیدا نکرد، اما به کمک گیت Not بر عکس ورودی‌های Enable فعال‌بالا را به Enable‌های فعال‌پائین می‌دهیم.







۳. تابع خروجی را در شکل زیر به کمک جبر بولی بدست آورید. (۲۰ نمره)



گیت‌نهایی مان سه ورودی دارد که آن‌ها را به ترتیب از بالا به پائین $z \leq X$ می‌نامیم، هر یک از این سه ورودی خود تابعی بر حسب $a b c$ است، این توابع را جداگانه محاسبه می‌کنیم و نهایتاً آن‌ها را با یکدیگر NAND می‌کنیم.

$$x = ED_2 = (a'(bc'))'$$

$$y = ED_0 = (a(b'c'))'$$

$$z = ED_1 = (a(b'c))'$$

$$\begin{aligned} f(a, b, c) &= (xyz)' = ((a'(bc'))'(a(b'c'))'(a(b'c))')' \rightarrow DeMorgan \\ &\rightarrow ((a + (bc'))'(a' + (b'c'))'(a' + (b'c))')' \\ &= ((a + b' + c)(a' + b + c)(a' + b + c'))' \\ &= ((ab + ac + a'b' + b'c + a'c + bc + c)(a' + b + c'))' \\ &= ((ab + a'b' + (a' + a)c + (b + b')c + c)(a' + b + c'))' \\ &= ((ab + a'b' + c)(a' + b + c'))' \\ &= (a'ab + a'b' + a'c + ab + a'b'b + bc + abc' + a'b'c' + cc')' \\ &= (a'b' + a'c + ab + bc + abc' + a'b'c')' \\ &= (a'b' + bc + b(ac' + a) + a'(c + b'c'))' \rightarrow \textbf{Absorption Property} \rightarrow f(a, b, c) \\ &= (a'b' + bc + b(a) + a'(c + b'))' = (a'b' + bc + ab + a'c + a'b')' \\ &= (a'b' + bc + ab + a'c)' \rightarrow DeMorgan \rightarrow (a'b')'(bc)'(ab)'(a'c)' \rightarrow DeMorgan \rightarrow \end{aligned}$$

$$\begin{aligned} f(a, b, c) &= (a + b)(b + c)(a' + b')(a + c') = (ab + b + ac + bc)(aa' + ab' + a'c' + b'c') \\ &\rightarrow \textbf{Absorption Property} \rightarrow (b + ac)(aa' + ab' + a'c' + b'c') \\ &= (b + ac)(ab' + a'c' + b'c') = bb' + ba'c' + bb'c' + acb' + aca'c' + acb'c' \\ &= a'bc' + ab'c \end{aligned}$$

۴. مدار معادلتابع زیر را با روش‌های خواسته شده طراحی کنید. (۳۰ نمره)

$$F(A, B, C, D, E) = \Sigma m(0, 2, 6, 7, 8, 10, 11, 12, 13, 14, 16, 18, 19, 29, 30) + \Sigma d(4, 9, 21)$$

الف) با استفاده از مالتی‌پلکسر 1×16 با ورودی‌های کنترلی A، B، C و D

ب) با استفاده از مالتی‌پلکسر 1×8 با ورودی‌های کنترلی A، B، C و دیگر گیت‌های مورد نیاز

(الف)

می‌توانیم از روش مشابه همان روشی استفاده کنیم که برای طراحی مدار با مالتی‌پلکسر 1:4 برای توابع سه متغیره داشتیم.

جدول درستی تابع را می‌کشیم، می‌بینیم که مقادیر ABCD دوردیفدوریف یکسان است، یعنی به طور کلی ABCD می‌تواند ۱۶ مقدار متفاوت داشته باشد. در هر یک از این مقادیر، یکی از ورودی‌های مالتی‌پلکسر به خروجی متصل خواهد شد.

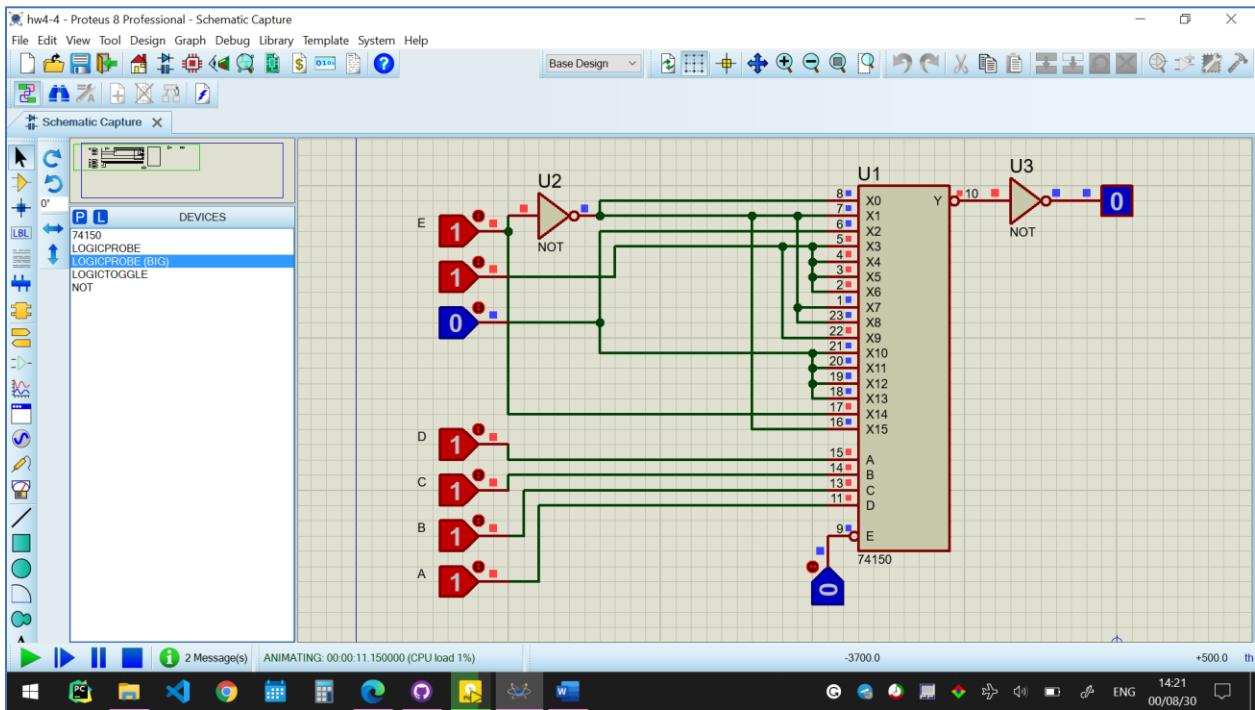
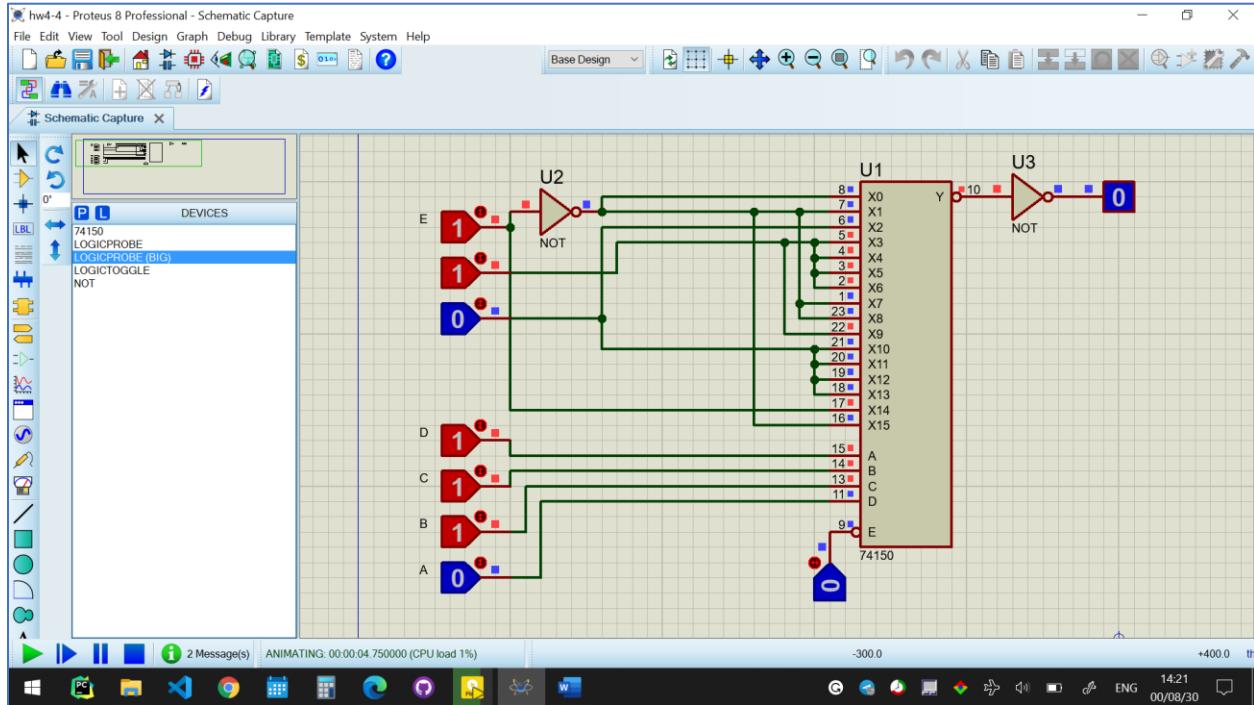
نکته‌ی مهم این است که هر یک از این ۱۶ حالت، شامل دو حالت مختلف برای متغیر E می‌باشد، اما با توجه به این نکته که مقدار خروجی در این دو حالت، برابر یکی از چهار حالت ۱، ۰، E، E' خواهد بود، می‌توانیم این شانزده ورودی را مقدار دهی کنیم.

(ورودی‌های مالتی‌پلکسر را X0 تا X15 می‌نامیم)

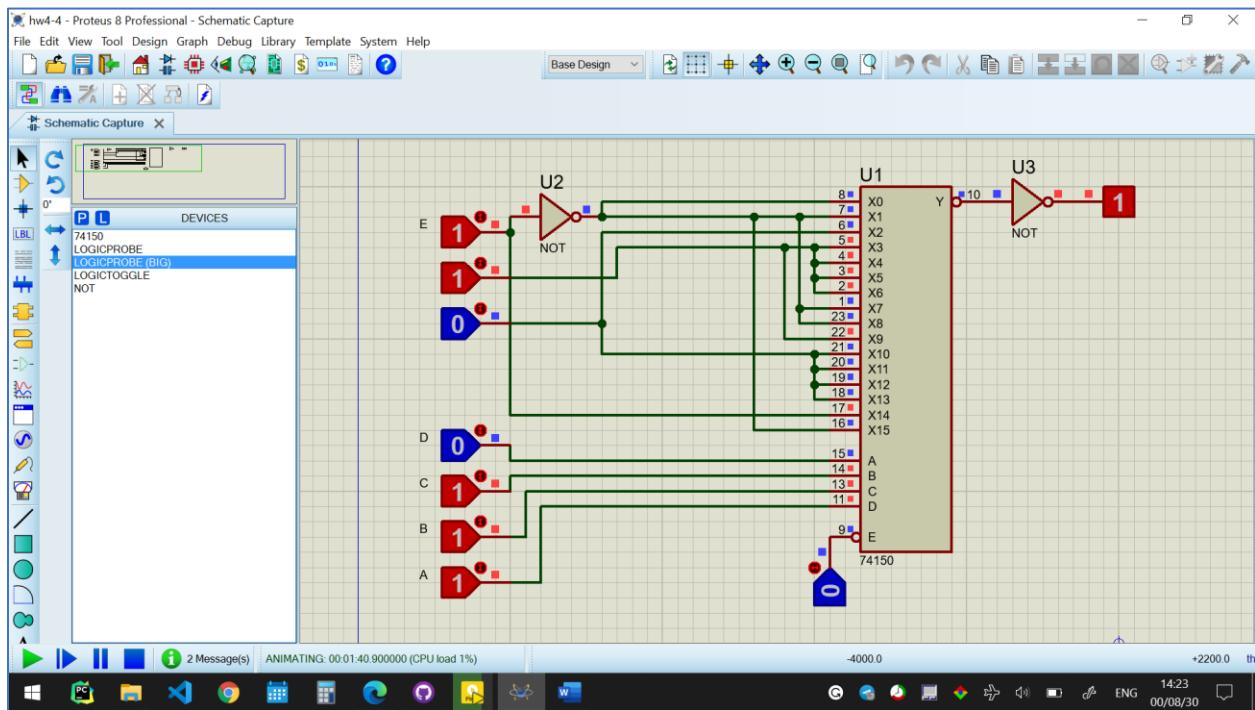
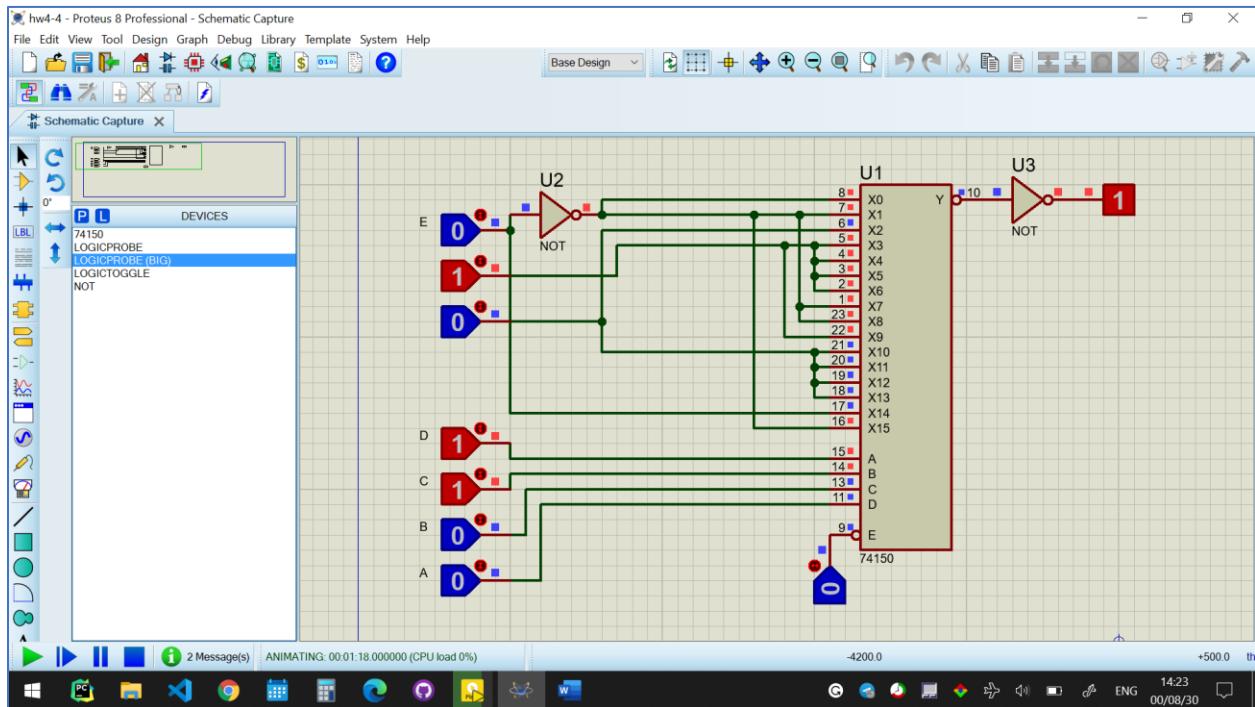
A	B	C	D	E	F	X
0	0	0	0	0	1	$X_0 = E'$
0	0	0	0	1	0	
0	0	0	1	0	1	$X_1 = E'$
0	0	0	1	1	0	
0	0	1	0	0	X	$X_2 = 0$
0	0	1	0	1	0	
0	0	1	1	0	1	$X_3 = 1$
0	0	1	1	1	1	
0	1	0	0	0	1	$X_4 = 1$
0	1	0	0	1	X	
0	1	0	1	0	1	$X_5 = 1$
0	1	0	1	1	1	
0	1	1	0	0	1	$X_6 = 1$
0	1	1	0	1	1	
0	1	1	1	0	1	$X_7 = E'$
0	1	1	1	1	0	
1	0	0	0	0	1	$X_8 = E'$
1	0	0	0	1	0	
1	0	0	1	0	1	$X_9 = 1$
1	0	0	1	1	1	
1	0	1	0	0	0	$X_{10} = 0$
1	0	1	0	1	X	
1	0	1	1	0	0	$X_{11} = 0$
1	0	1	1	1	0	
1	1	0	0	0	0	$X_{12} = 0$
1	1	0	0	1	0	
1	1	0	1	0	0	$X_{13} = 0$
1	1	0	1	1	0	
1	1	1	0	0	0	$X_{14} = E$
1	1	1	0	1	1	
1	1	1	1	0	1	$X_{15} = E'$
1	1	1	1	1	0	

(مالتیپلکسری که از آن استفاده شده فعال پائین است، پس ورودی E را 0 می‌دهیم و بعد از خروجی یک NOT می‌گذاریم تا خروجی را بهتر درک کنیم.)

ماکس ترم ۱۵ و ۳۱ :



مین ترم ۶ و ۲۹

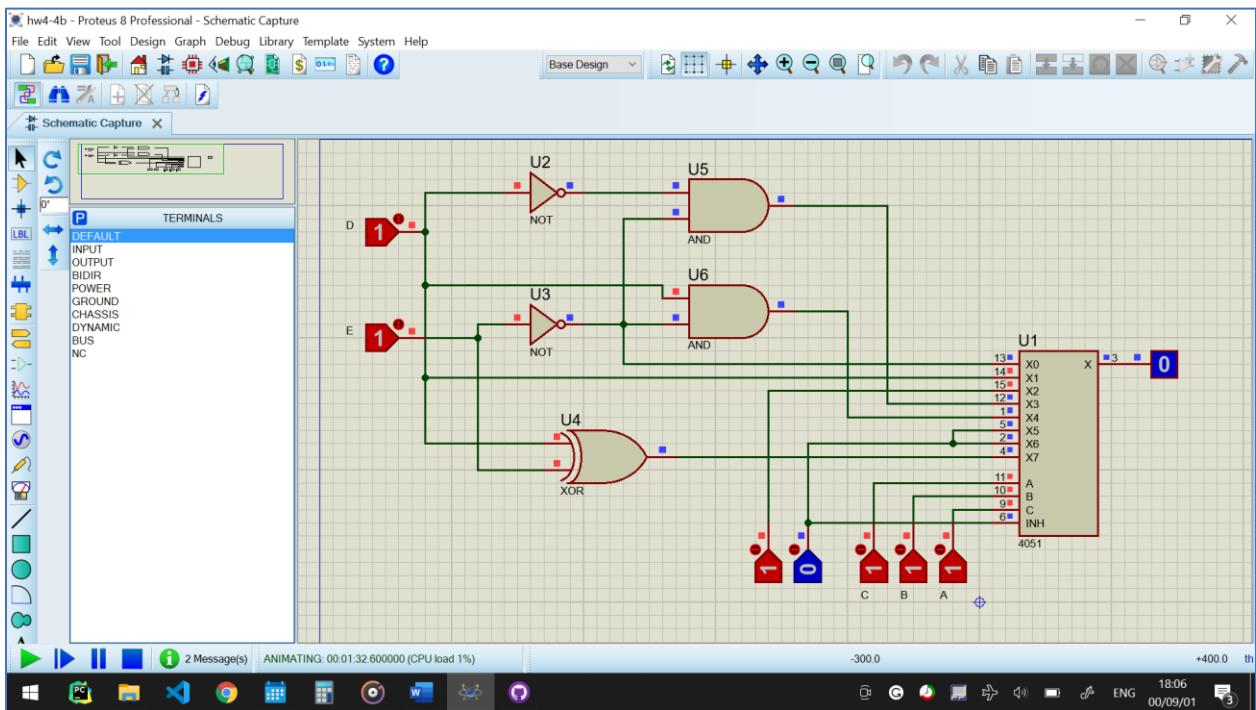
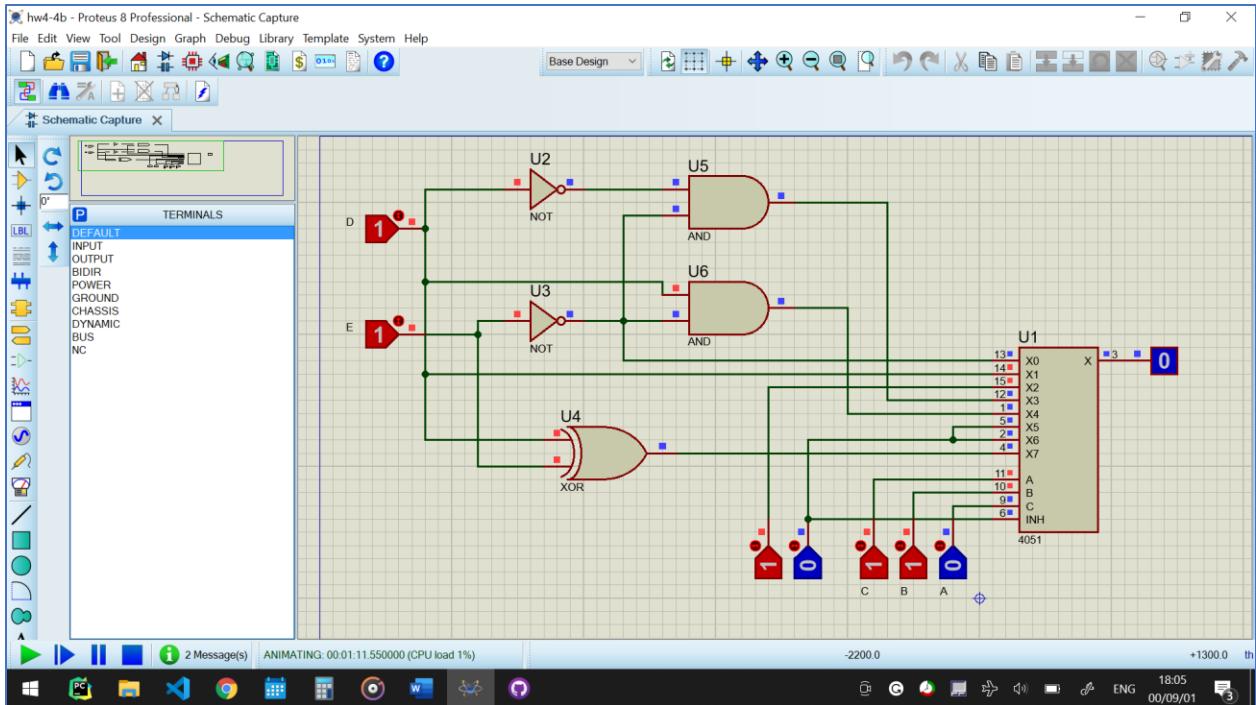


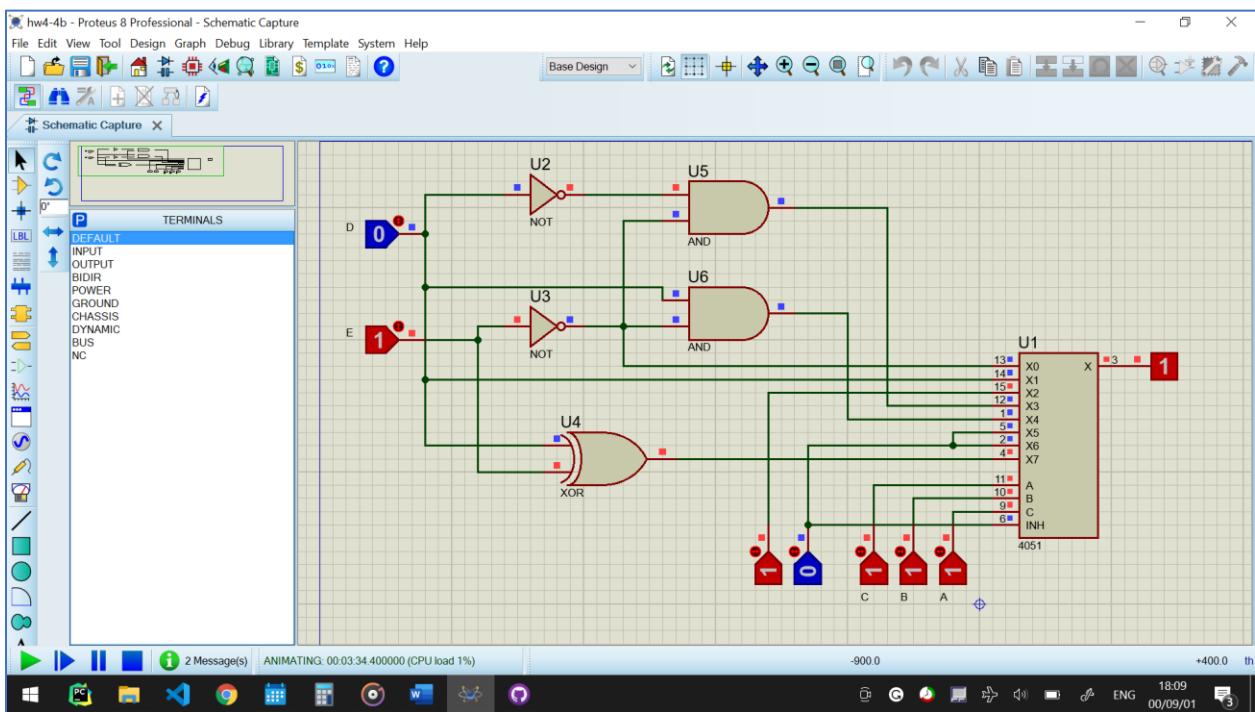
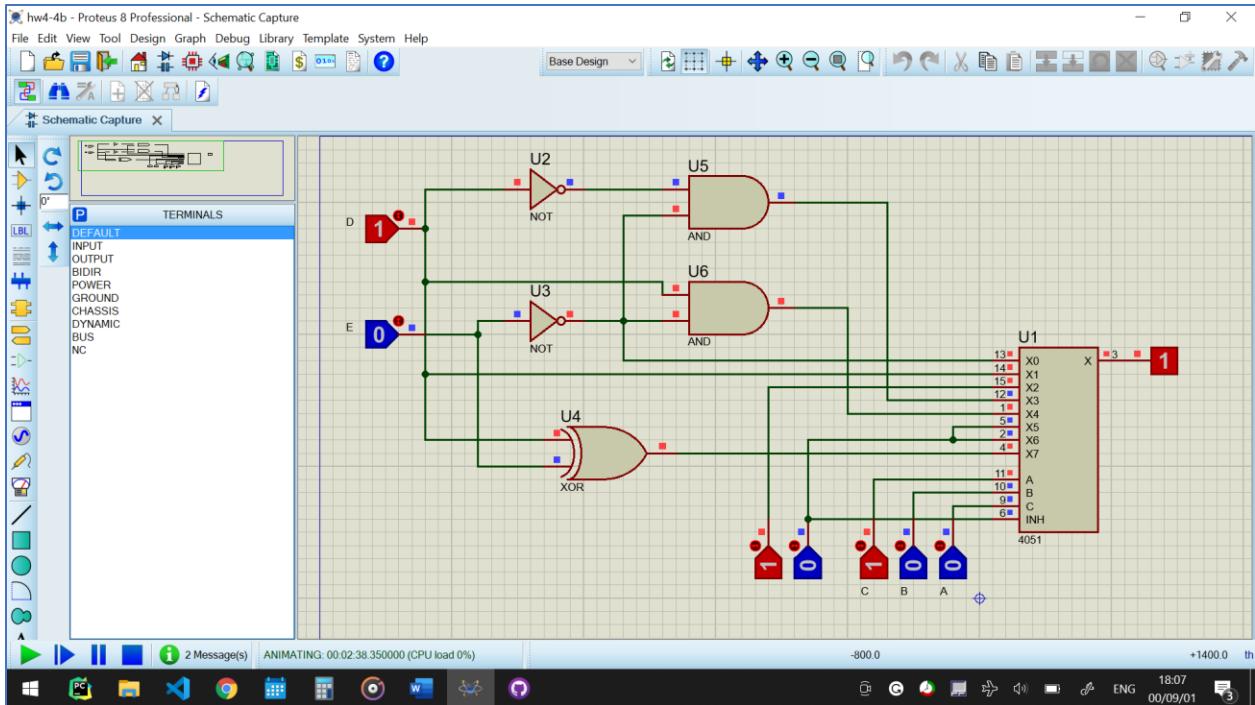
(ب)

مشابه حالت قبل عمل می‌کنیم، با این تفاوت که باید X ها بر اساس توابعی از D و E بنویسیم، نه فقط E ، که البته با توجه به این که مجاز به استفاده از گیت‌های لازم هستیم، مشکلی از این جهت نداریم.

A	B	C	D	E	F	X
0	0	0	0	0	1	$X_0 = E'$
0	0	0	0	1	0	
0	0	0	1	0	1	
0	0	0	1	1	0	
0	0	1	0	0	X	$X_1 = D$
0	0	1	0	1	0	
0	0	1	1	0	1	
0	0	1	1	1	1	$X_2 = 1$
0	1	0	0	0	1	
0	1	0	0	1	X	
0	1	0	1	0	1	$X_3 = D' + E'$
0	1	0	1	1	0	
0	1	1	0	1	1	
0	1	1	1	1	0	
1	0	0	0	0	1	$X_4 = D + E'$
1	0	0	0	1	0	
1	0	0	1	0	1	
1	0	0	1	1	1	$X_5 = 0$
1	0	1	0	0	0	
1	0	1	0	1	X	
1	0	1	1	0	0	
1	0	1	1	1	0	$X_6 = 0$
1	1	0	0	0	0	
1	1	0	0	1	0	
1	1	0	1	0	0	
1	1	0	1	1	0	$X_7 = D \oplus E$
1	1	1	0	0	0	
1	1	1	0	1	1	
1	1	1	1	0	1	
1	1	1	1	1	0	

ماكس ترم ١٥ و ٣١ :





۵. می خواهیم یک Priority Encoder با ۸ ورودی طراحی کنیم به گونه ای که همواره بین ورودی های با اندیس زوج و فرد به ورودی با اندیس فرد اولویت دهد. در داخل هر دسته از ورودی های زوج و فرد نیز، این مدار به ورودی با اندیس بزرگ تر اولویت می دهد. جدول ارزش های این مدار را بکشید. (۱۵ نمره)

ورودی های این مدار را d0 تا d7 می نامیم و خروجی های آن را A0 تا A3 و V می نامیم.

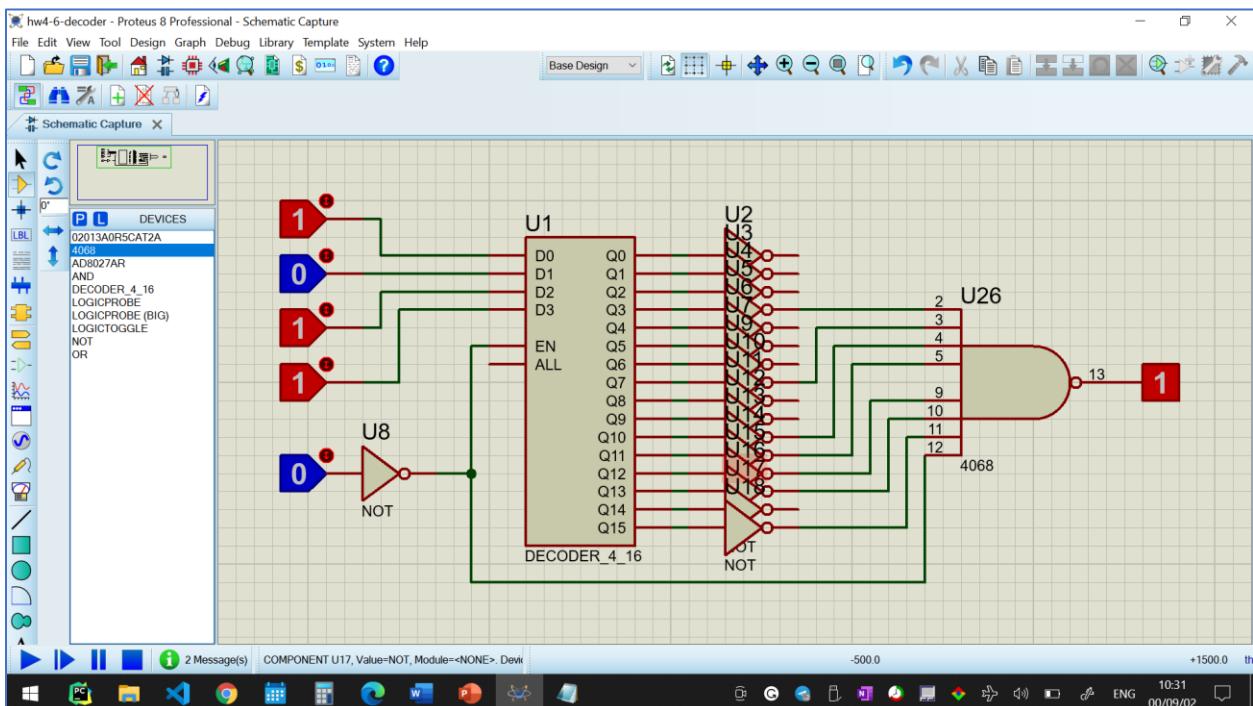
inputs								outputs			
D7	D6	D5	D4	D3	D2	D1	D0	A2	A1	A0	V
0	0	0	0	0	0	0	0	X	X	X	0
1	X	X	X	X	X	X	X	1	1	1	1
0	X	1	X	X	X	X	X	1	0	1	1
0	X	0	X	1	X	X	X	0	1	1	1
0	X	0	X	0	X	1	X	0	0	1	1
0	1	0	X	0	X	0	X	1	1	0	1
0	0	0	1	0	X	0	X	1	0	0	1
0	0	0	0	0	1	0	X	0	1	0	1
0	0	0	0	0	0	0	1	0	0	0	1

۶. ابتدا مدار منطقی مربوط به تابع زیر را با استفاده از دیکدر ۴:۱۶ با خروجی فعال-پایین طراحی کنید. سپس این تابع را با استفاده از جدول کارنو ساده کرده و آن را با روش تمام NOR پیاده‌سازی کنید. پیچیدگی سخت‌افزاری این دو طرح را مقایسه کنید. (۳۰ نمره)

$$F(A, B, C, D) = (A\bar{B} + CD)\oplus A\bar{C}$$

ابتدا محاسبه می‌کنیم که در چه نقاطی

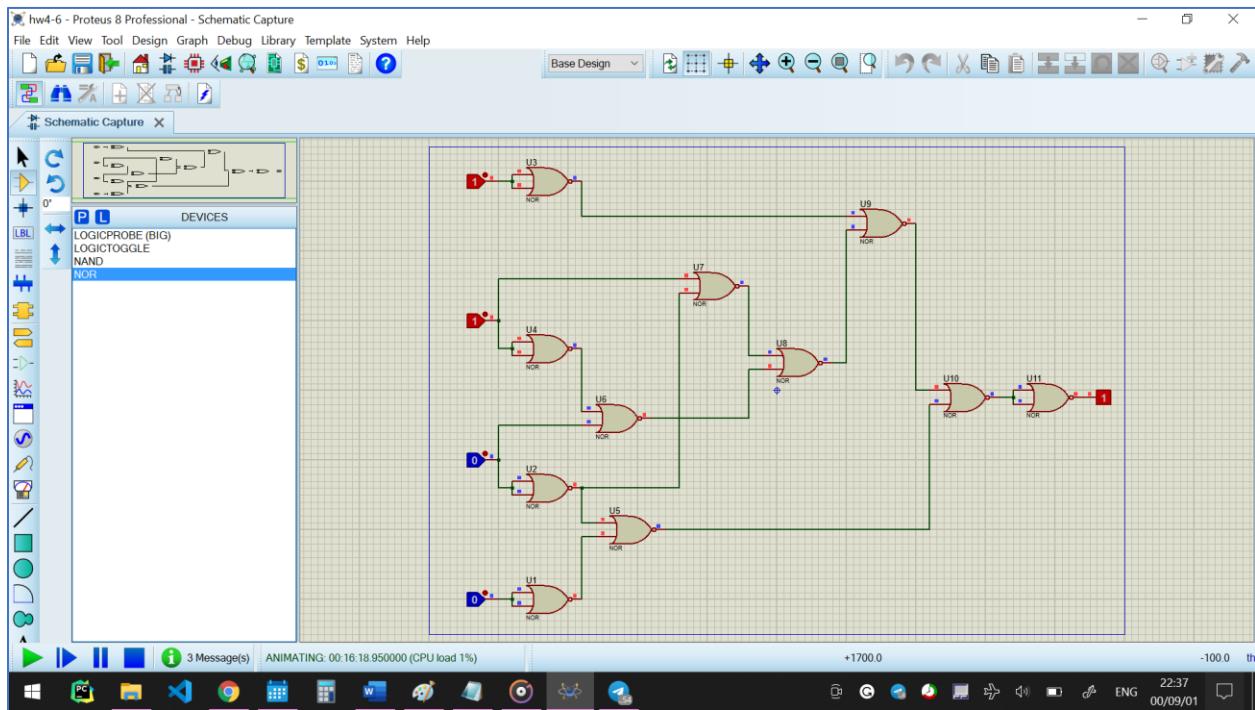
(با توجه به فعال‌پایین بودن دیکدر، باید به آن مقدار ۰ داده شود، تا خروجی‌های آن مقداری غیر از ۱ داشته باشند. همچنین در خروجی‌ها نیز، همه‌ی خروجی‌ها ۱ هستند و فقط خروجی هماندیس با عدد ورودی ۰ است. متناسبانه دیکدر فعال‌پایین در بروتونس پیدا نکردم، که با متصل کردن NOT به تمام ورودی‌ها و خروجی‌ها، فرض می‌کنیم که NOT‌ها بخشی از دیکدر هستند).



CD \\	00	01	11	10
AB	00	0	1	0
	01	0	1	0
	11	1	1	0
	10	0	1	1

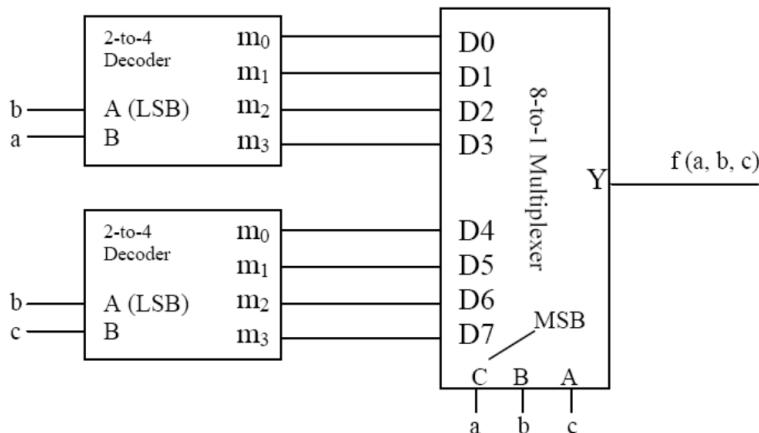
CD \\	00	01	11	10
AB	00	0	1	0
	01	0	1	0
	11	1	1	0
	10	0	1	1

$$\begin{aligned}
 F(A, B, C, D) &= CD + ABC' + AB'C = (C' + D')' + (A(BC' + B'C)) \\
 &= (C' + D')' + (A((B' + C)' + (B + C')')) \\
 &= (C' + D')' + (A' + ((B' + C)' + (B + C')'))'
 \end{aligned}$$



در مقایسه این دو مورد، می‌بینیم که وقتی با دیکدر کار می‌کنیم، روش مستقیم و واضحی وجود دارد، اما با روش تمام NOR نیاز به استفاده از گیت‌های زیاد و انجام محاسبات وجود دارد که باعث می‌شود نهایتاً هم بستن مدار پیچیده‌تر شود و هم طراحی آن.

۷. تابع خروجی مدار شکل زیر را به کمک جبر بولی بدست آورید. (۲۰ نمره)



براساس مولتیپلکسر، تابع را می‌نویسیم:

$$f(a, b, c) = (a'b'c')D_0 + (a'b'c)D_1 + (a'bc')D_2 + (a'bc)D_3 + (ab'c')D_4 + (ab'c)D_5 + (abc')D_6 + (abc)D_7$$

حال باید توابع D_0 تا D_7 را بیابیم:

$$D_0 = b'a'$$

$$D_1 = b'a$$

$$D_2 = ba'$$

$$D_3 = ba$$

$$D_4 = b'c'$$

$$D_5 = b'c$$

$$D_6 = bc'$$

$$D_7 = bc$$

حال مقادیر به دست آمده را جایگذاری می‌کنیم:

$$\begin{aligned} f(a, b, c) &= (a'b'c')b'a' + (a'b'c)b'a + (a'bc')ba' + (a'bc)ba + (ab'c')b'c' + (ab'c)b'c \\ &\quad + (abc')bc' + (abc)bc = (a'b'c') + (a'bc') + (ab'c') + (ab'c) + (abc') + (abc) \\ &= a'c'(b + b') + ab'(c + c') + ab(c + c') = a'c' + ab' + ab = a'c' + a(b + b') \\ &= a + a'c \rightarrow \textbf{Absorption Property} \rightarrow f(a, b, c) = a + c \end{aligned}$$

۸. عدد ۱۱۰۱۱ در کد گری (gray) نوشته شده است، ابتدا این عدد را به فرم دده‌هی تبدیل کنید سپس آن را به فرم BCD بنویسید. با جستجو در اینترنت درمورد نحوه تبدیل کد گری به باینری مطالعه کنید. (۲۰ نمره)

ابتدا عدمن را به فرم باینری در می‌آوریم. می‌دانیم که چپ‌ترین رقم، در هر دو حالت باینری و گری یکی است، پس آن را به عنوان چپ‌ترین رقم می‌نویسیم و به سراغ دومین رقم از چپ می‌رویم.

برای به دست آوردن رقم‌های بعدی، آخرین رقم باینری که نوشته‌ایم و رقمی بعدی از گری‌مان را XOR می‌کنیم، که در این حالت ۱ و ۰ هستند که خواهد شد. حال این ۰ را با ۱ که سومین رقم گری از چپ است XOR می‌کنیم که ۱ خواهد شد. به همین ترتیب تا انتهای ادامه می‌دهیم و به ۱۰۱۱۰۱ خواهیم رسید. این مقدار را به فرم دده‌ی می‌نویسیم:

$$2^5 + 2^3 + 2^2 + 2^0 = 32 + 8 + 4 + 1 = 45$$

حال باید ۴۵ را به فرم BCD بنویسیم. برای این کار کافی است ارقام ۴ و ۵ را، جداگانه به باینری‌های چهاربیتی تبدیل کنیم و در کنار یکدیگر بنویسیم. چهار به باینری برابر ۱۰۰ خواهد شد و ۵ تبدیل به ۱۰۱ خواهد شد، پس فرم BCD برابر است با ۰۱۰۰ ۰۱۰۱

برای رقم‌های کوچک‌تر، می‌توانیم از جدول کارنو که به کمک اعداد گری تشکیل شده نیز استفاده کنیم.

۹. جمع و تفریق های ۴-بیتی زیر را در سیستم مکمل دو انجام دهید و در هر مورد وضعیت پرچم سرریز (overflow) را مشخص کنید. (۲۰ نمره)

- a) $(0101)_2 + (0100)_2$
- b) $(1101)_2 + (0100)_2$
- c) $(1101)_2 - (1010)_2$
- d) $(1010)_2 - (1101)_2$

a)

هر دو عدد مثبت هستند و با یکدیگر جمع شده‌اند، پس نیاز به تبدیلی نداریم.

$$1010 + 0100 = 1001$$

اگر حاصل را به شکل یک عدد **unsigned** نگاه کنیم پاسخ صحیح است و عدد ۹ را نشان می‌دهد، اما از آن جایی که در سیستم مکمل دو کار می‌کنیم، می‌دانیم که این عدد به معنای ۷-است، که از آن‌جا که جمع دو عدد مثبت، منفی شده، پس مشخص است که سرریز رخ داده است.

b)

می‌دانیم که این دو عدد، در سیستم ددهی برابر با -3 و $+4$ خواهند بود و جمع‌شان برابر ۱ خواهد شد.

$$1101 + 0100 = 10001$$

البته از آن‌جا که با چهاربیت کار می‌کنیم، چپ‌ترین رقم به عنوان Carry-out خواهد بود و نتیجه به شکل 0001 خواهد شد که البته صحیح است. به راحتی می‌توان اثبات کرد که در این‌جا $2^n - 1$ که در این‌جا $2^8 - 1 = 255$ خواهد شد، با سرریز شدن آن، جواب صحیح حاصل می‌شود و سرریز نداریم.

c)

می‌دانیم که این دو عدد، در سیستم ددهی -3 و -6 هستند و تفاصل‌شان برابر $+3$ خواهد شد.

$$1101 - 1010 = 0011$$

روش دیگری که برای این سوال وجود دارد، این است که رقم دوم را قرینه کنیم و سپس دو عدد را با یکدیگر جمع کنیم. مکمل دوی رقم دوم برابر با 0110 است، پس می‌توان نوشت:

$$1101 + 0110 = 10011$$

که مشابه مورد قبلی، رقم carry out که در چهار بیت نگنجیده، محاسبه نخواهد شد و دوباره به همین 0011 می‌رسیم. در این مورد هم سرریزی رخ نداد.

d)

می‌دانیم که این دو عدد، در سیستم ددهی -6 و -3 هستند و تفاصل‌شان برابر -9 خواهد شد که مشخص است که در ۴ بیت سیستم مکمل دو، نمی‌توان آن را نشان داد.

مشابه بخش قبلی، از مکمل دوی عدد استفاده می‌کنیم و آن را با عدد اول جمع می‌کنیم:

$$1010 + 0011 = 1001$$

حاصل عدد ۷-را نشان می‌دهد که مشخصاً اشتباه است، پس در این‌جا نیز overflow داشته‌ایم.

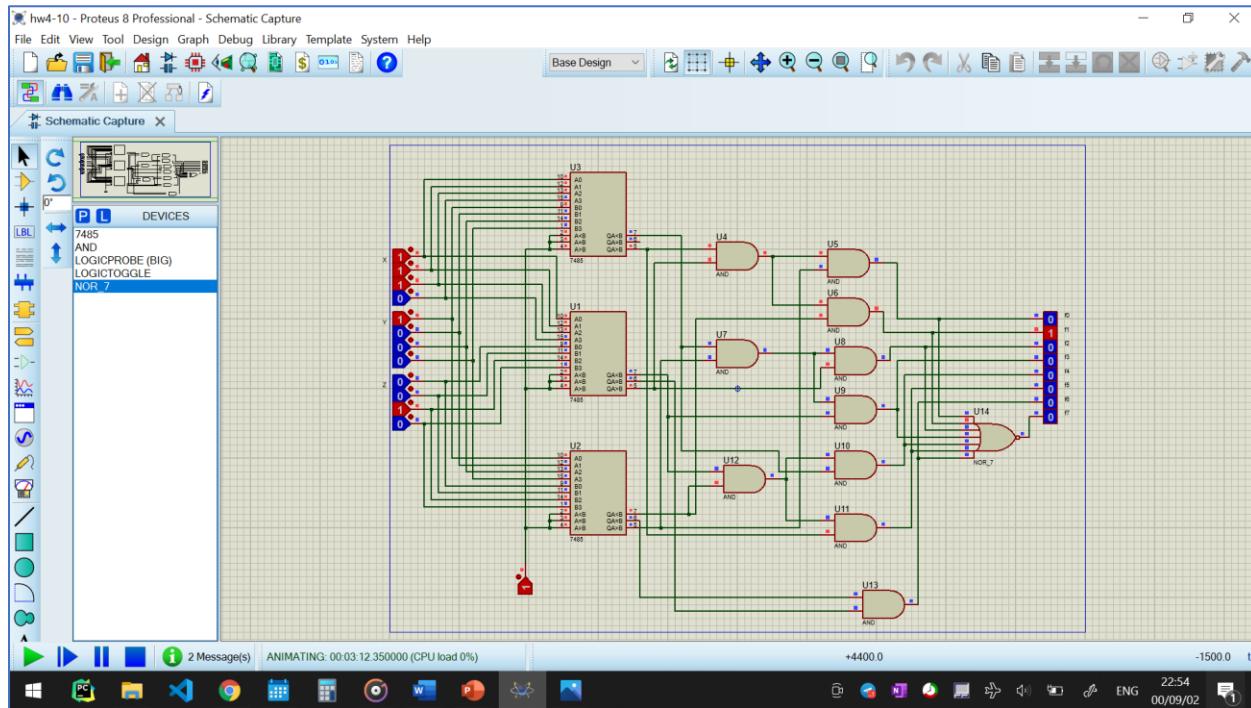
بخش سوم: سوالات امتیازی

۱۰. می خواهیم ۳ عدد ۴-بیتی X , Y و Z را با استفاده از کمترین تعداد ممکن مقایسه کننده 74×85 مقایسه کنیم. مداری طراحی کنید که خروجی آن مطابق جدول درستی زیر باشد. (۲۰ نمره)

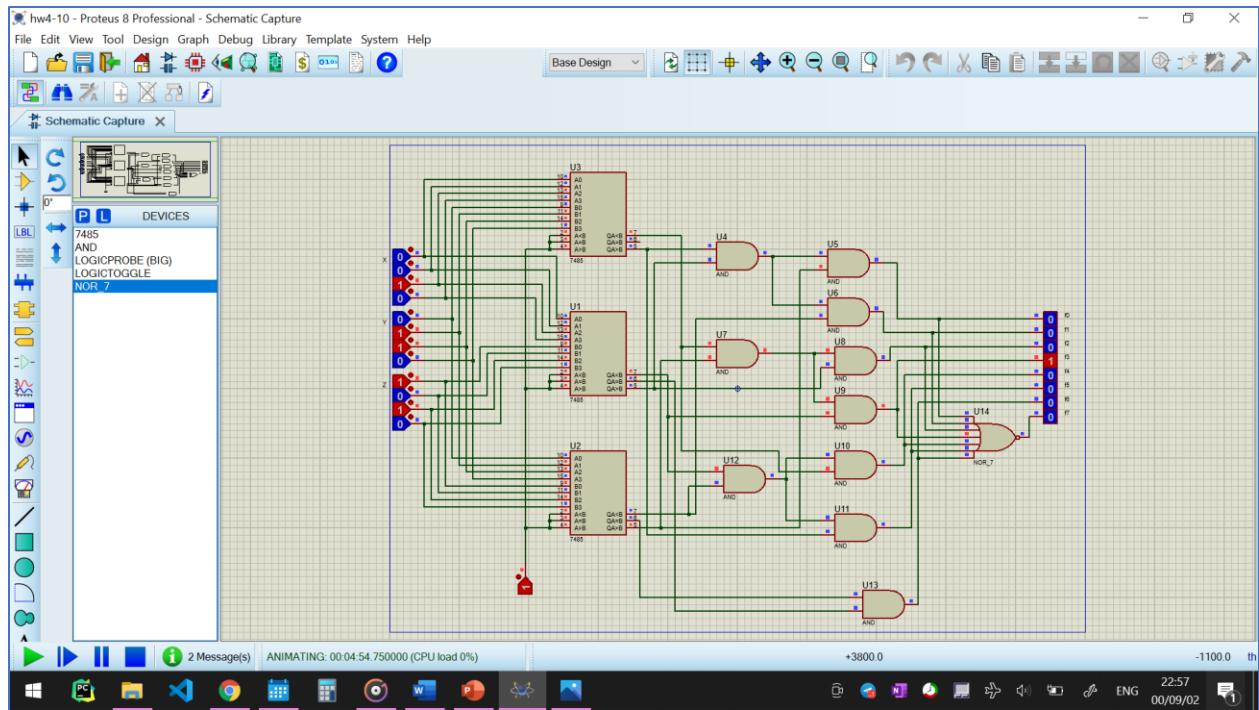
Condition	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7
$X > Y > Z$	1	0	0	0	0	0	0	0
$X > Z > Y$	0	1	0	0	0	0	0	0
$Y > X > Z$	0	0	1	0	0	0	0	0
$Y > Z > X$	0	0	0	1	0	0	0	0
$Z > X > Y$	0	0	0	0	1	0	0	0
$Z > Y > X$	0	0	0	0	0	1	0	0
$X = Y = Z$	0	0	0	0	0	0	1	0
Any other case	0	0	0	0	0	0	0	1

از سه مقایسه کننده استفاده می کنیم و اعدامان را دو به دو با یکدیگر مقایسه می کنیم. سپس با استفاده از گیتهای AND، سه شرط مورد نیاز برای هر یک از حالت های گفته شده را بررسی می کنیم تا اگر صحیح بودند ۱ به خروجی مربوط متصل شود. نهایتا هم همهی هفت حالت اول را با یکدیگر NOR می کنیم تا اگر هیچ یک از آن ها برقرار نبود خروجی شماره‌ی هفت ۱ شود.

$$X = 7, Y = 1, Z = 4 \rightarrow X > Z > Y \rightarrow f_1 = 0$$



$$X = 2, Y = 6, Z = 3, Y > Z > X$$



$$X = 3, Y = 3, Z = 3 \rightarrow X = Y = Z \rightarrow f_6 = 1$$

