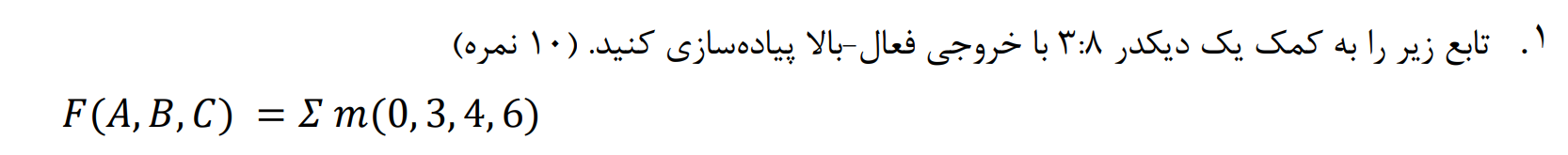
به نام خدا

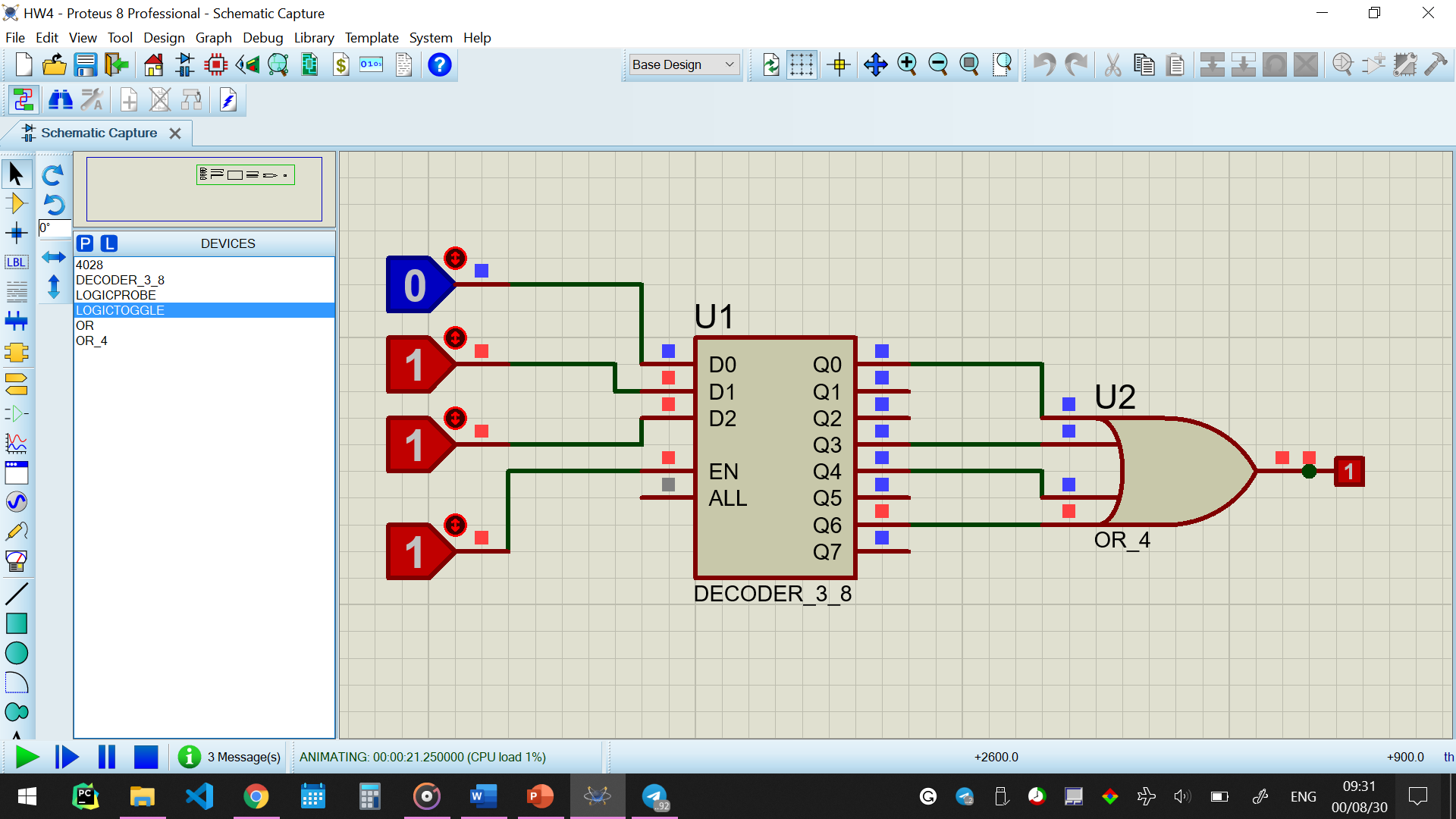
تمرین چهارم مدارهای منطقی

چمران معینی

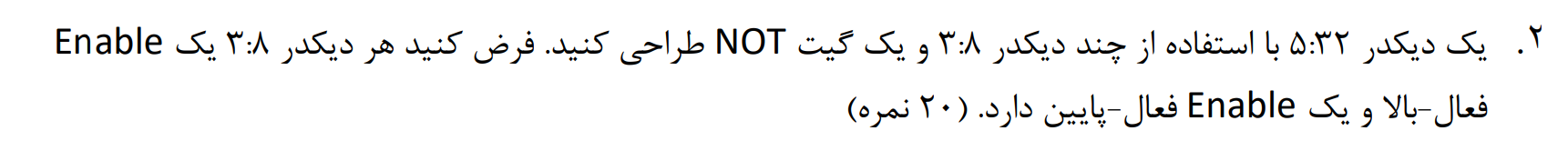
۹۹۳۱۰۵۳



ابتدا به کمک دیکدر،‌ تشخیص می‌دهیم که متغیرهای تابع‌مان چه مقداری را نشان می‌دهند، سپس اگر آن مقدار یکی از مین‌ترم‌ها بود، خروجیِ ما صحیح خواهد بود، پس خروجی‌هایی که با مین‌ترم‌ها هم‌اندیس 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)



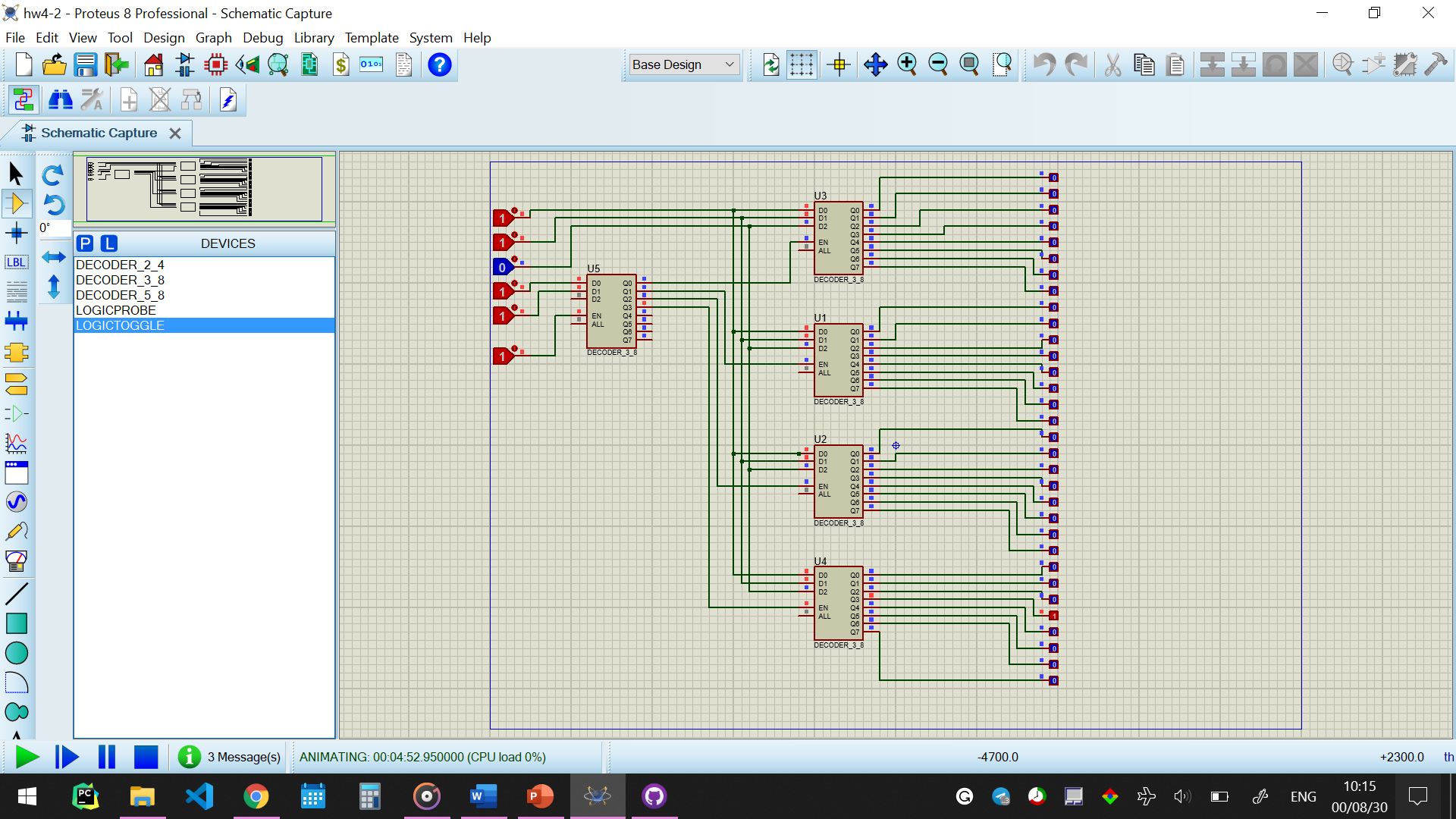
می‌دانیم که دیکدرِ ۵:۳۲‌ ، ۵ ورودی دارد و ۳۲ خروجی، پس باید پنج ورودی و سی و دو خروجی را هندل کنیم. ورودی‌هایمان را i0 تا i4 و خروجی‌هایمان را i0 تا i31 می‌نامیم.

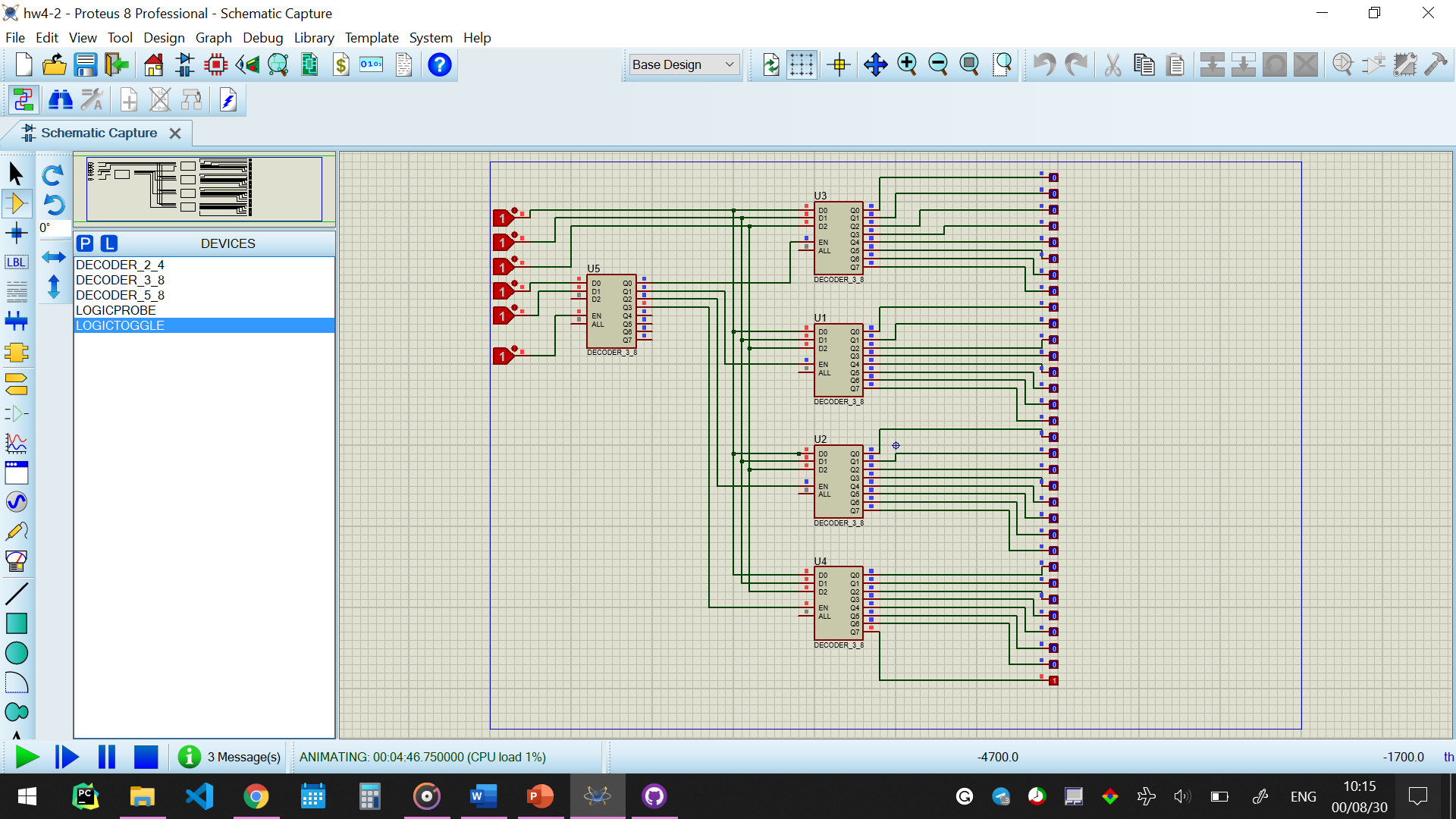
با توجه به این که ۳۲ خروجی داریم،‌ می‌فهمیم که باید ۴ دیکدر ۳:۸ را در کنار یکدیگر قرار دهیم تا این خروجی‌ها را تولید کنند.

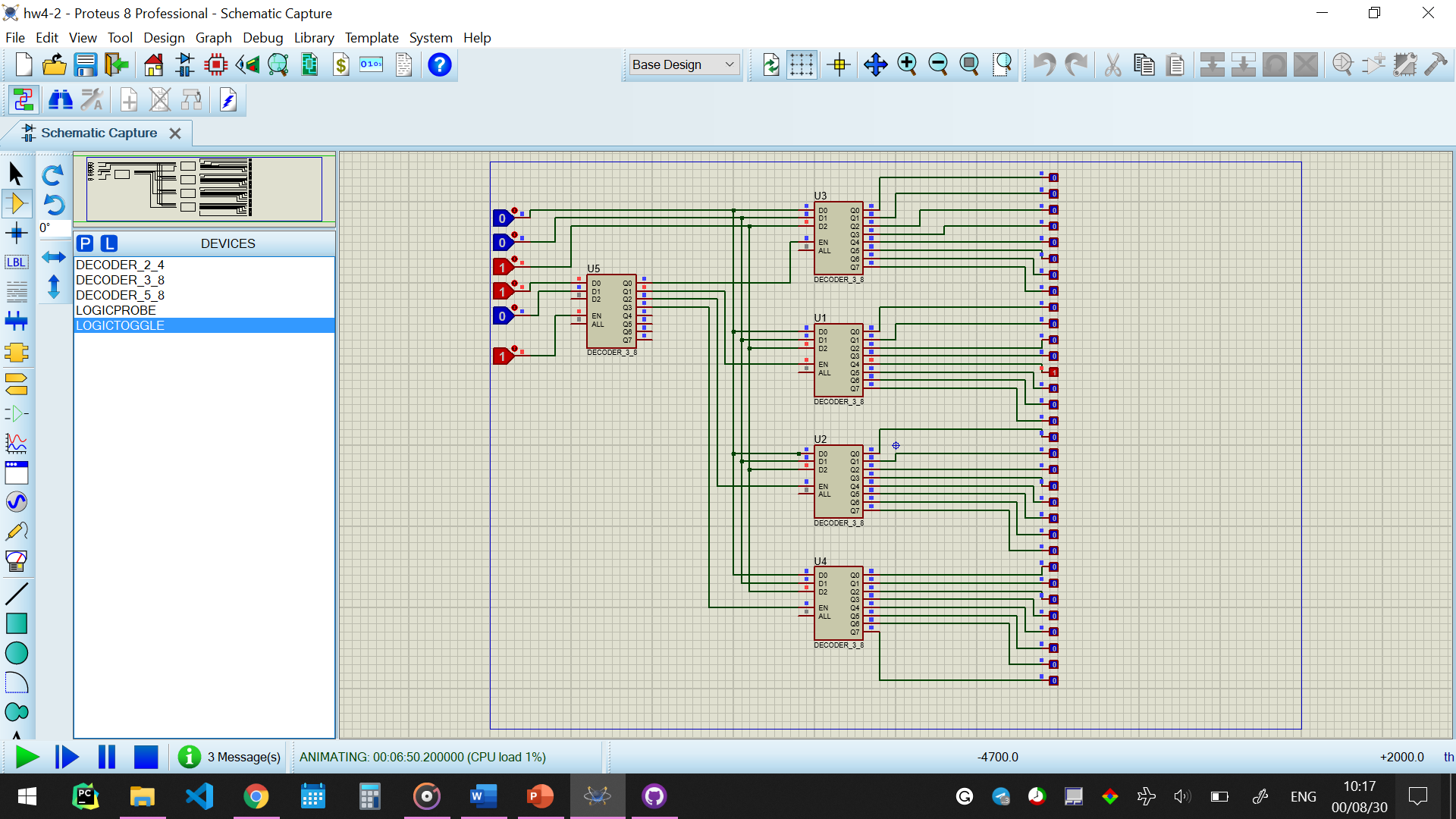
هنگامی که i4i3 مقدار 00 را داشته باشد، خروجیِ صحیح یکی از i0 تا i7 خواهد بود، هنگامی که i4i3 مقدار 01 را داشته باشد، خروجیِ صحیح یکی از i8 تا i15 خواهد بود و... پس از یک دیکدر برای یافتن مقدار i4i3 استفاده می‌کنیم، و خروجی‌های آن را به ترتیب به Enable های فعال‌بالا و Notشان را به فعال‌پائين‌ها متصل می‌کنیم.

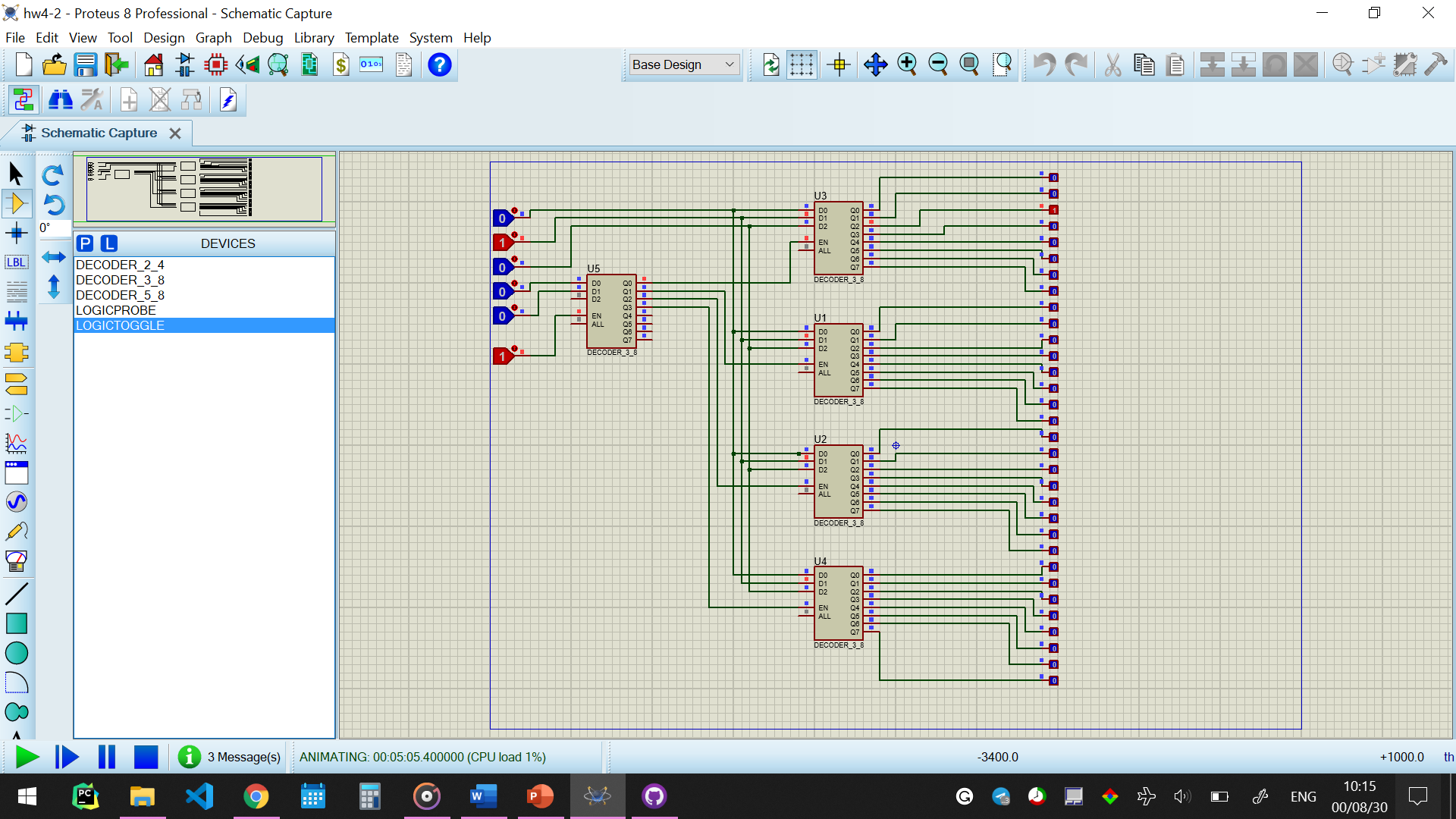
تا این‌جای کار موفق شدیم که دیکدری که قرار است خروجی ۱ را نشان دهد را پیدا کنیم. حال باید ورودی‌های دیکدرها را طوری مقداردهی کنیم که اگر Enable‌شان فعال بود، خروجی صحیح را ۱ کنند. در این مرحله، دیگر نیازی به i4i3 نداریم و i0 تا i2 در این بخش تاثیر گذارند، با متصل کردنِ آن‌ها به ترتیب به هر سه ورودیِ چهار دیکدری که قرار است خروجی نهایی را نشان دهند، هنگامی که Enable‌های یک دیکدر در حالت فعال بودند، دیکدر ما مقدار صحیح نهایی را نشان خواهد داد.

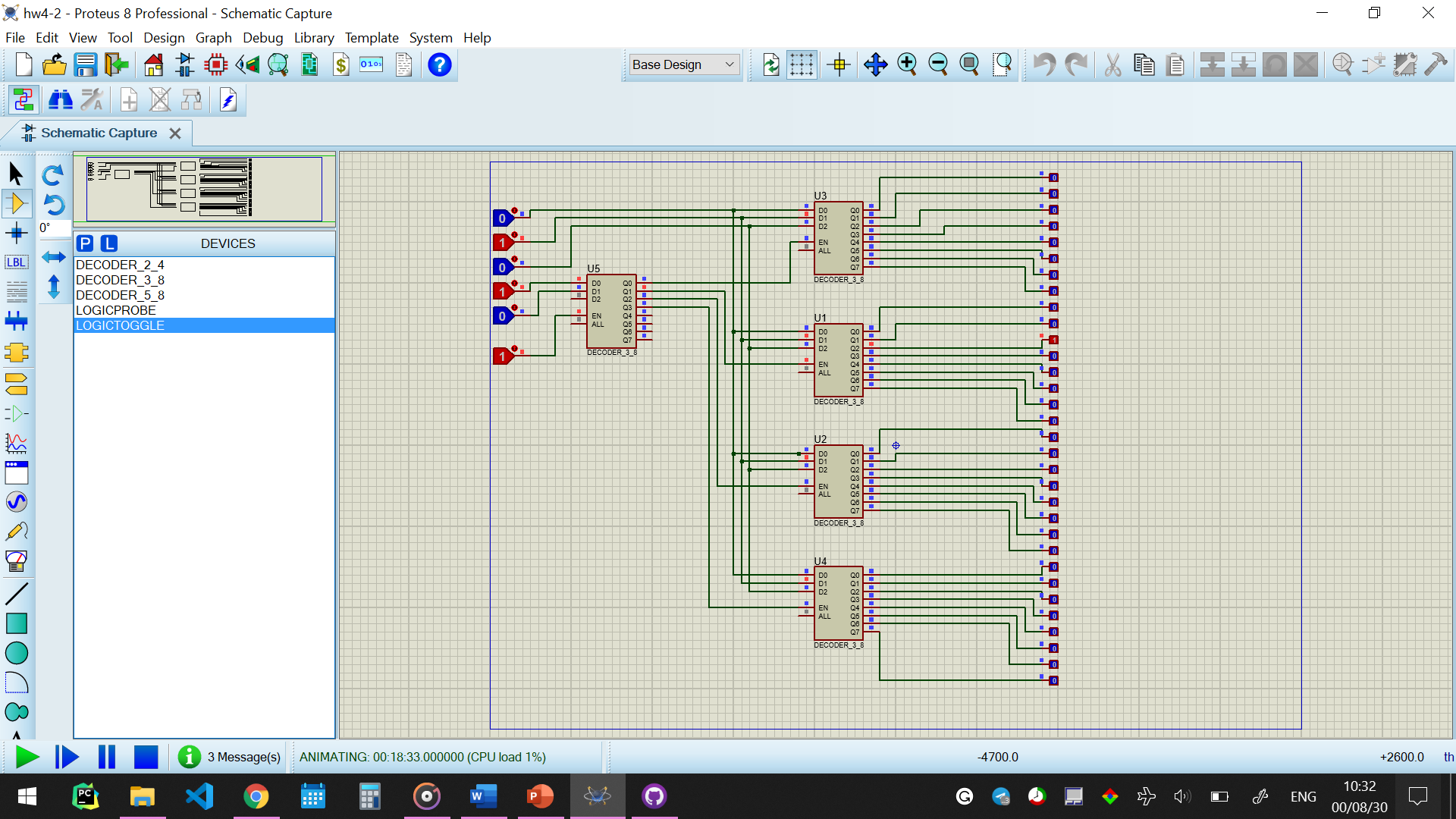
در شماتیک‌های زیر، Enable فعال‌پایین لحاظ نشده چون متاسفانه Device مناسب برای نمایش آن را پیدا نکردم، اما به کمک گیت Not برعکس ورودی‌های Enable فعال‌بالا را به Enable های فعال‌پایین می‌دهیم.

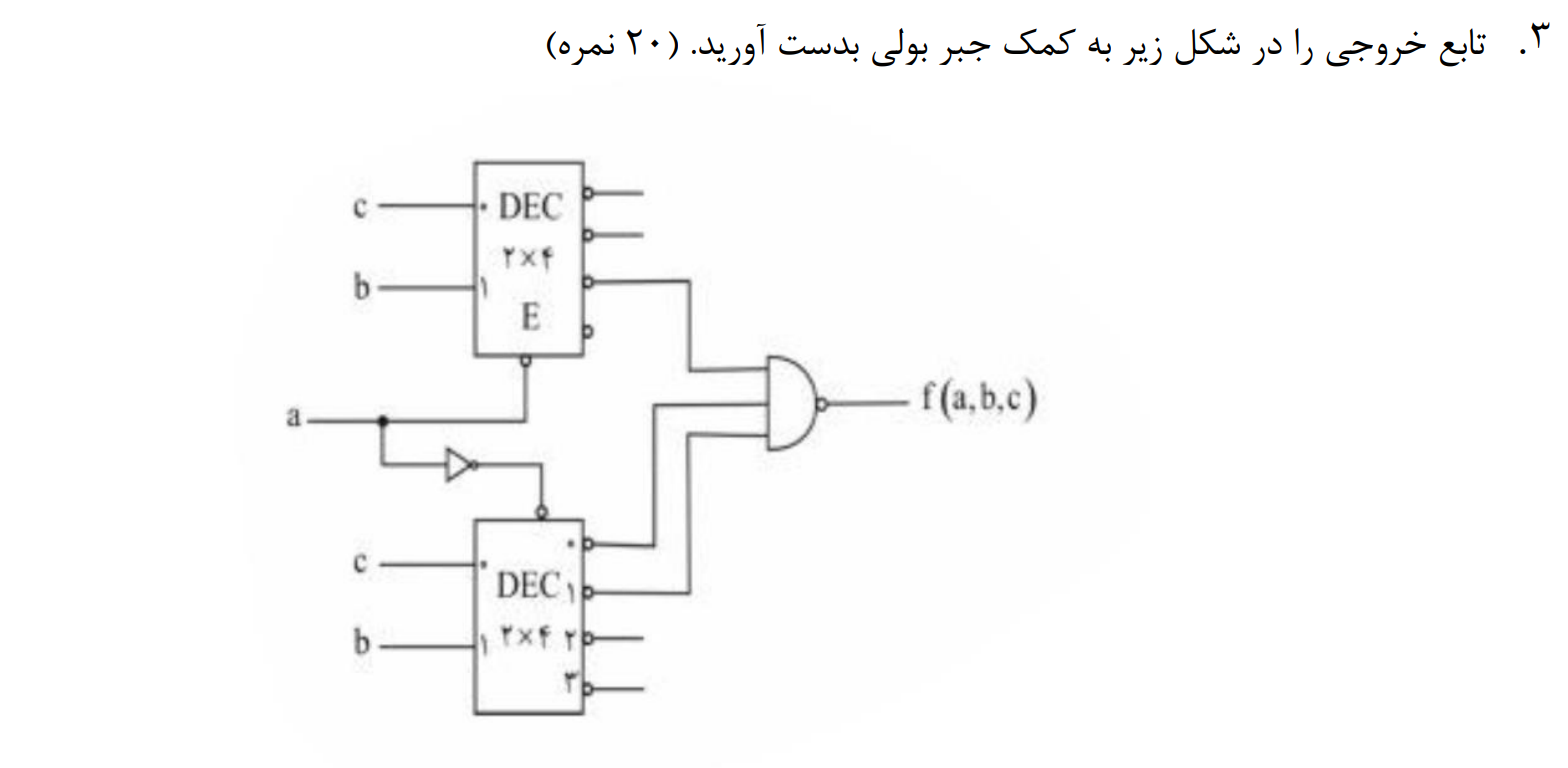




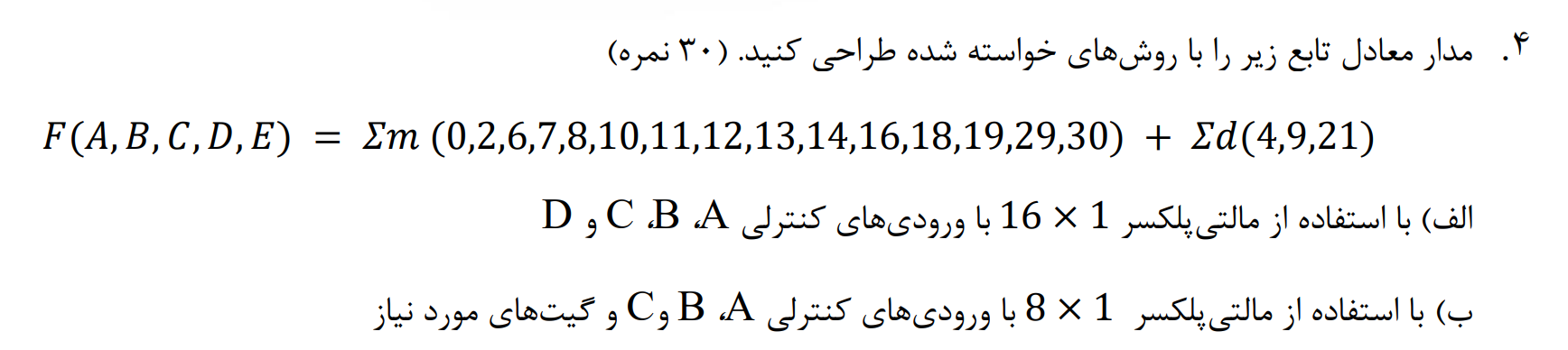








گیت‌نهایی‌مان سه ورودی دارد که آن‌ها را به ترتیب از بالا به پائین x y z می‌نامیم، هر یک از این سه ورودی خود تابعی بر حسب a b c ست، این توابع را جداگانه محاسبه می‌کنیم و نهایتا آن‌ها را با یکدیگر AND می‌کنیم.



الف)

می‌توانیم از روشی مشابهِ همان روشی استفاده کنیم که برای طراحی مدار با مالتی‌پلکسر برای توابع سه متغیره داشتیم.

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

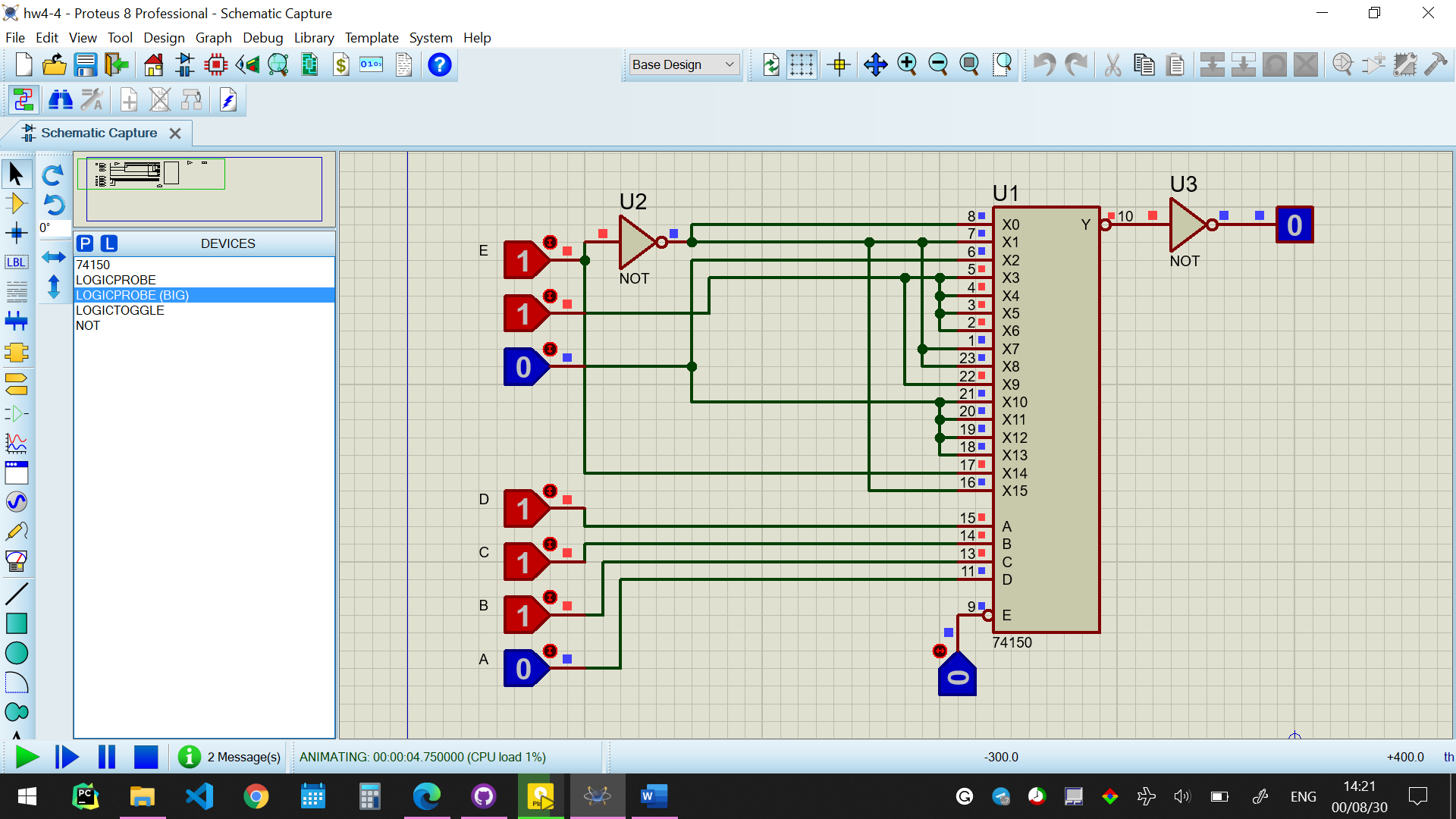
نکته‌ی مهم این است که هر یک از این ۱۶ حالت، شاملِ دو حالتِ مختلف برای متغیر E می‌باشد، اما با توجه به این نکته که مقدار خروجی در این دو حالت، برابرِ یکی از چهار حالتِ 1, 0, E, E’ خواهد بود، می‌توانیم این شانزده ورودی را مقدار دهی کنیم.

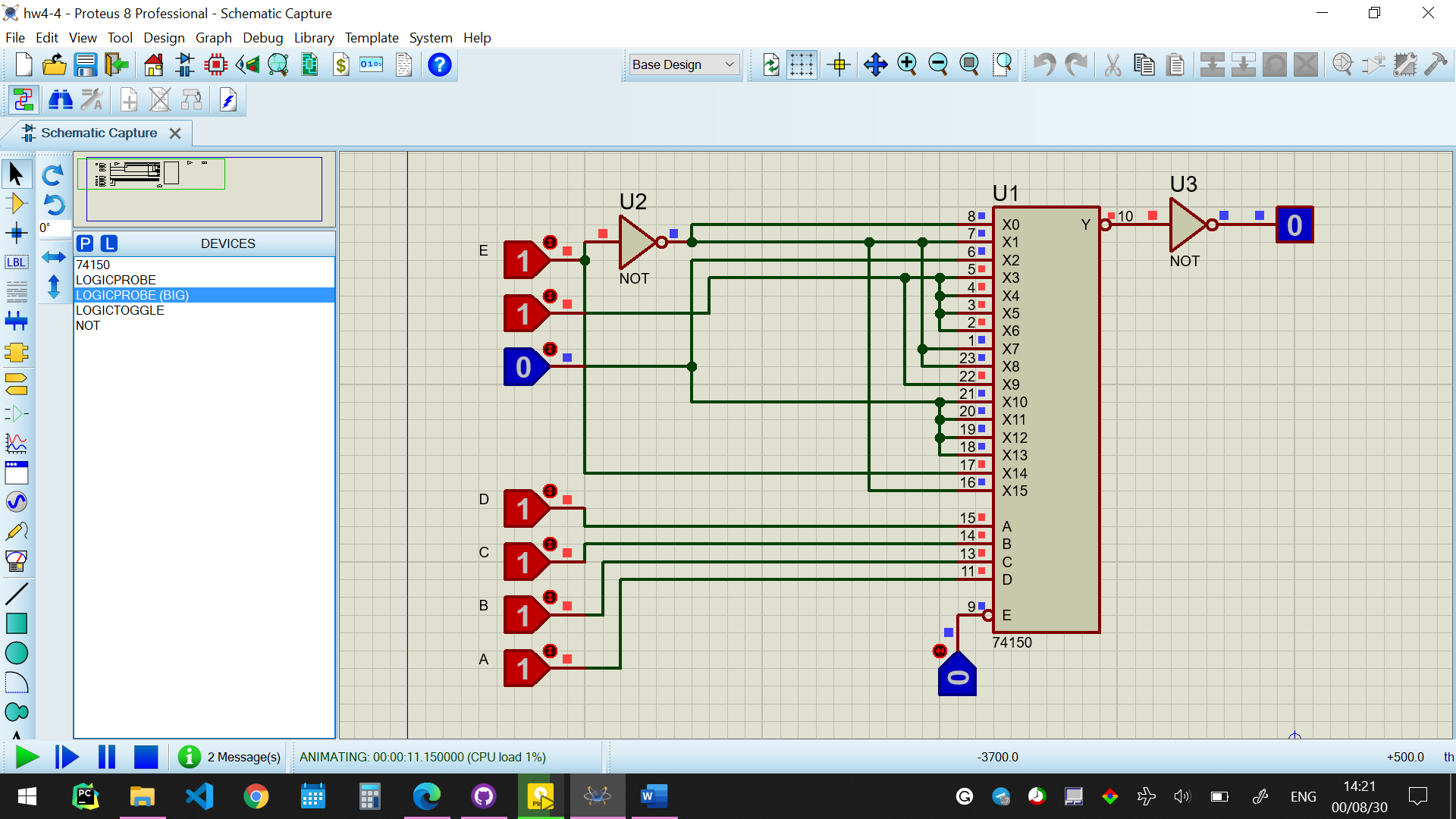
(ورودی‌های مالتی‌پلکسر را X0 تا X15 می‌نامیم)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| X | F | E | D | C | B | A |
| X0 = E’ | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 |
| X1 =E’ | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 |
| X2 = 0 | X | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 |
| X3 = 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 |
| X4 = 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| X | 1 | 0 | 0 | 1 | 0 |
| X5 = 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 | 0 |
| X6 = 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 |
| X7= E’ | 1 | 0 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 0 |
| X8 = E’ | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 | 1 |
| X9 = 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 0 | 1 |
| X10 = 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| X | 1 | 0 | 1 | 0 | 1 |
| X11 = 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | 1 |
| X12 = 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 |
| X13 = 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 1 |
| X14 = E | 0 | 0 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| X15 = E’ | 1 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 |

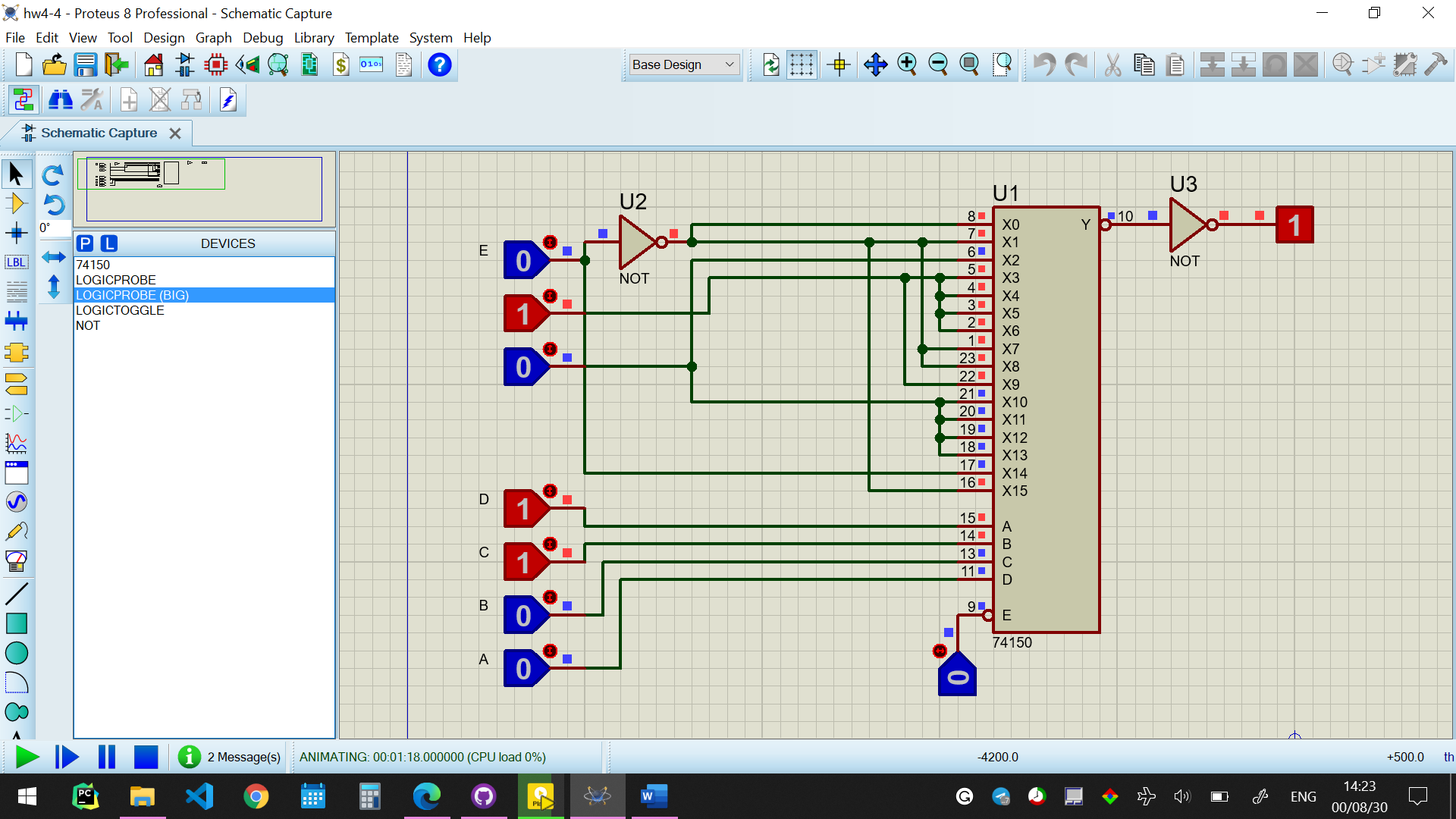
(مالتی‌پلکسری که از آن استفاده شده فعال پائین است، پس ورودی E را 0 می‌دهیم و بعد از خروجی یک NOT می‌گذاریم تا خروجی را بهتر درک کنیم.)

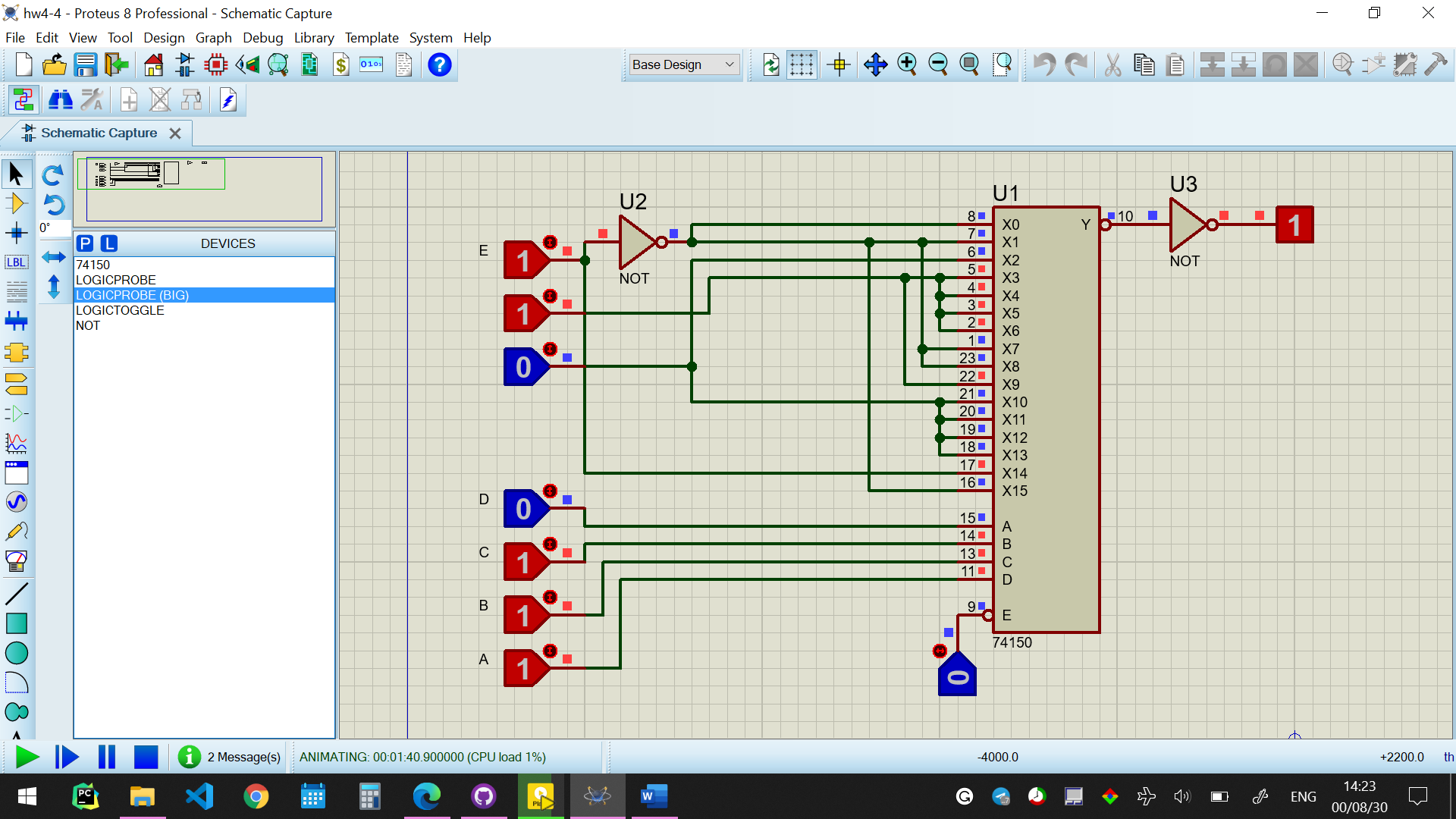
ماکس‌ترم ۱۵ و ۳۱ :





مین‌ترم ۶ و ۲۹:



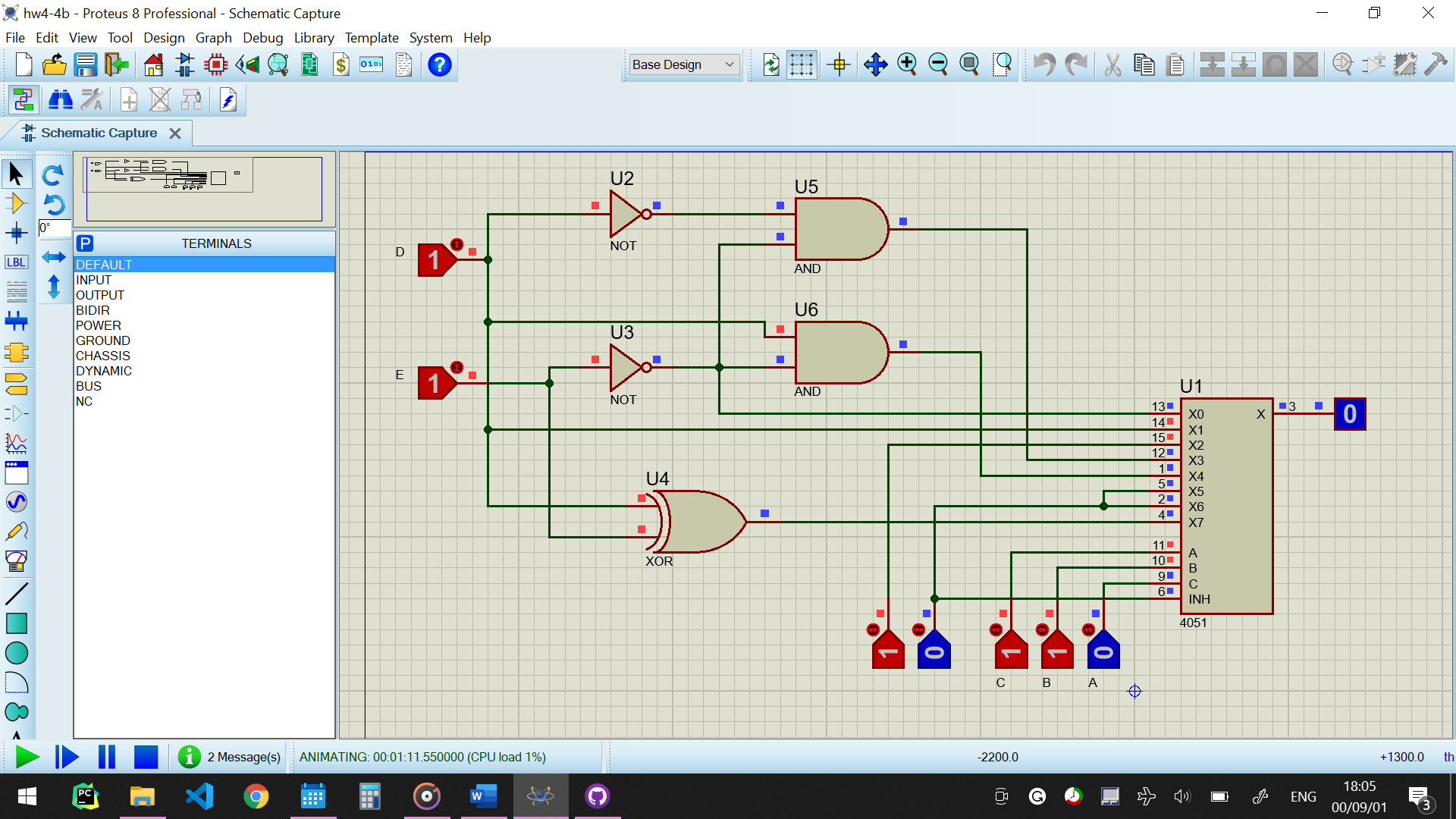


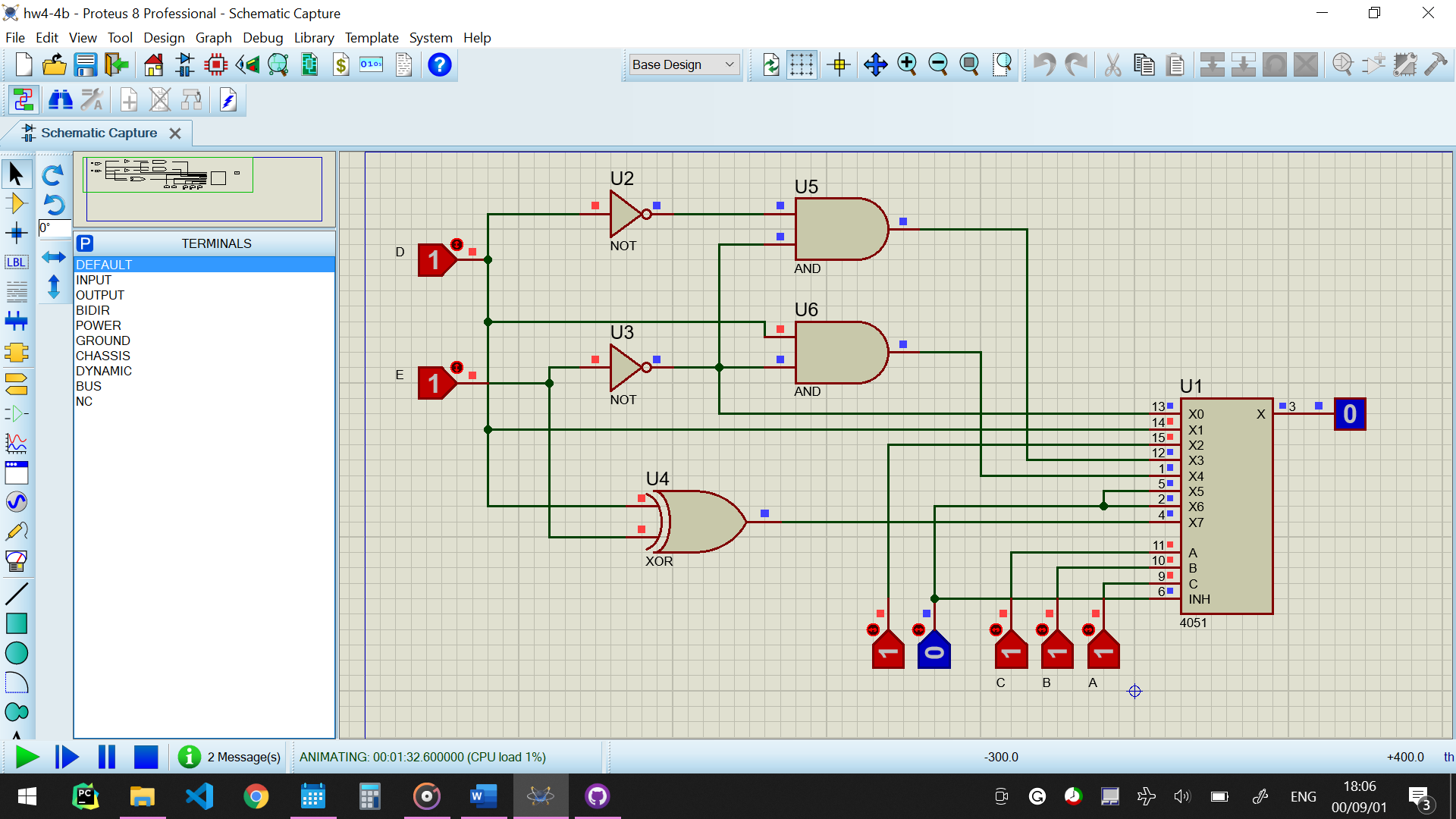
ب)

مشابه حالت قبل عمل می‌کنیم، با این تفاوت که باید X ها را 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.

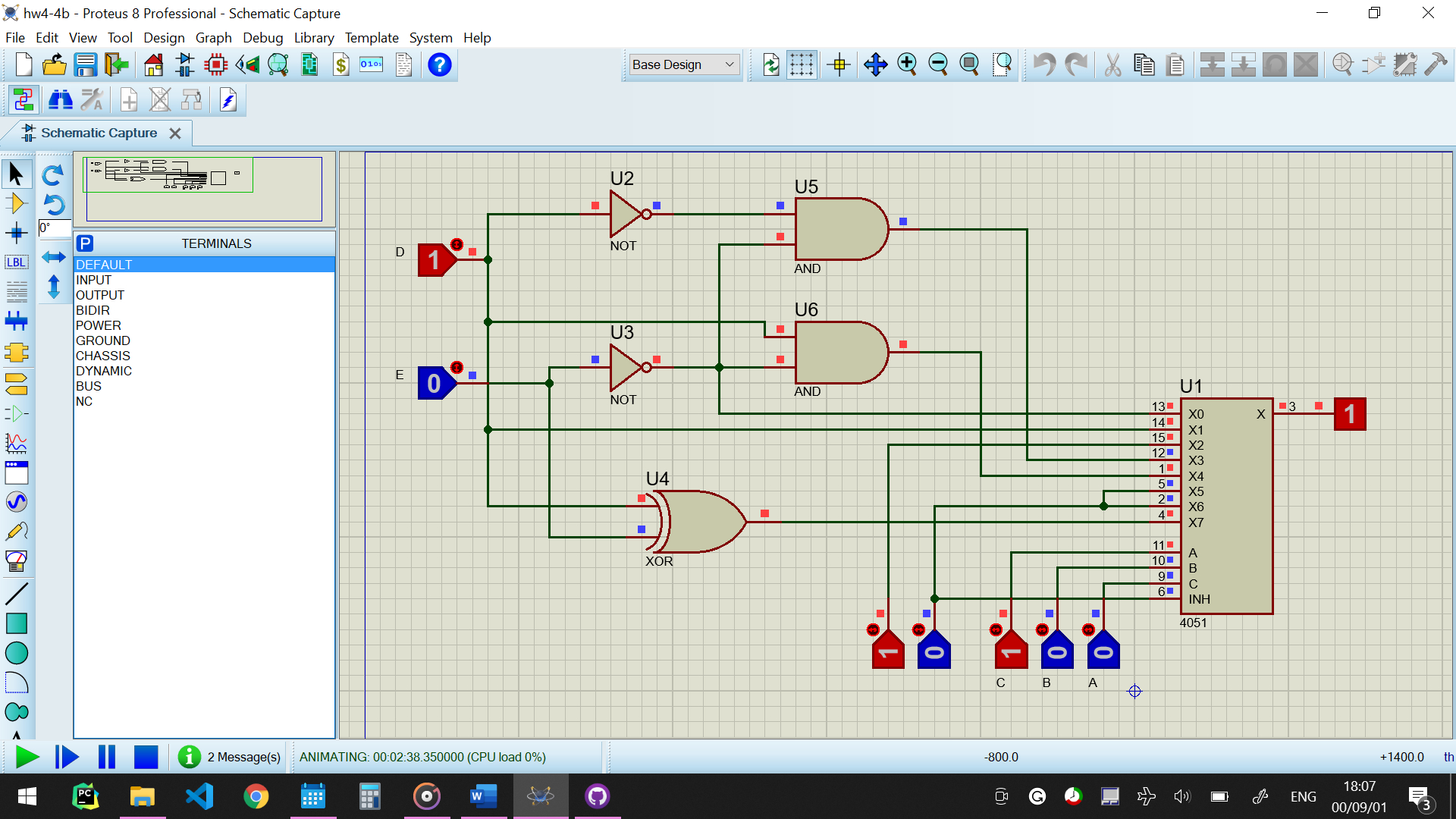
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| X | F | E | D | C | B | A |
| X0 = E’ | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 |
| X1 = D | X | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 |
| X2 = 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| X | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 | 0 |
| X3 = D’ + E’ | 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 0 |
| X4 = D + E’ | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 0 | 1 |
| X5 = 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| X | 1 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | 1 |
| X6 = 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 1 |
| X7 =  D E | 0 | 0 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 |

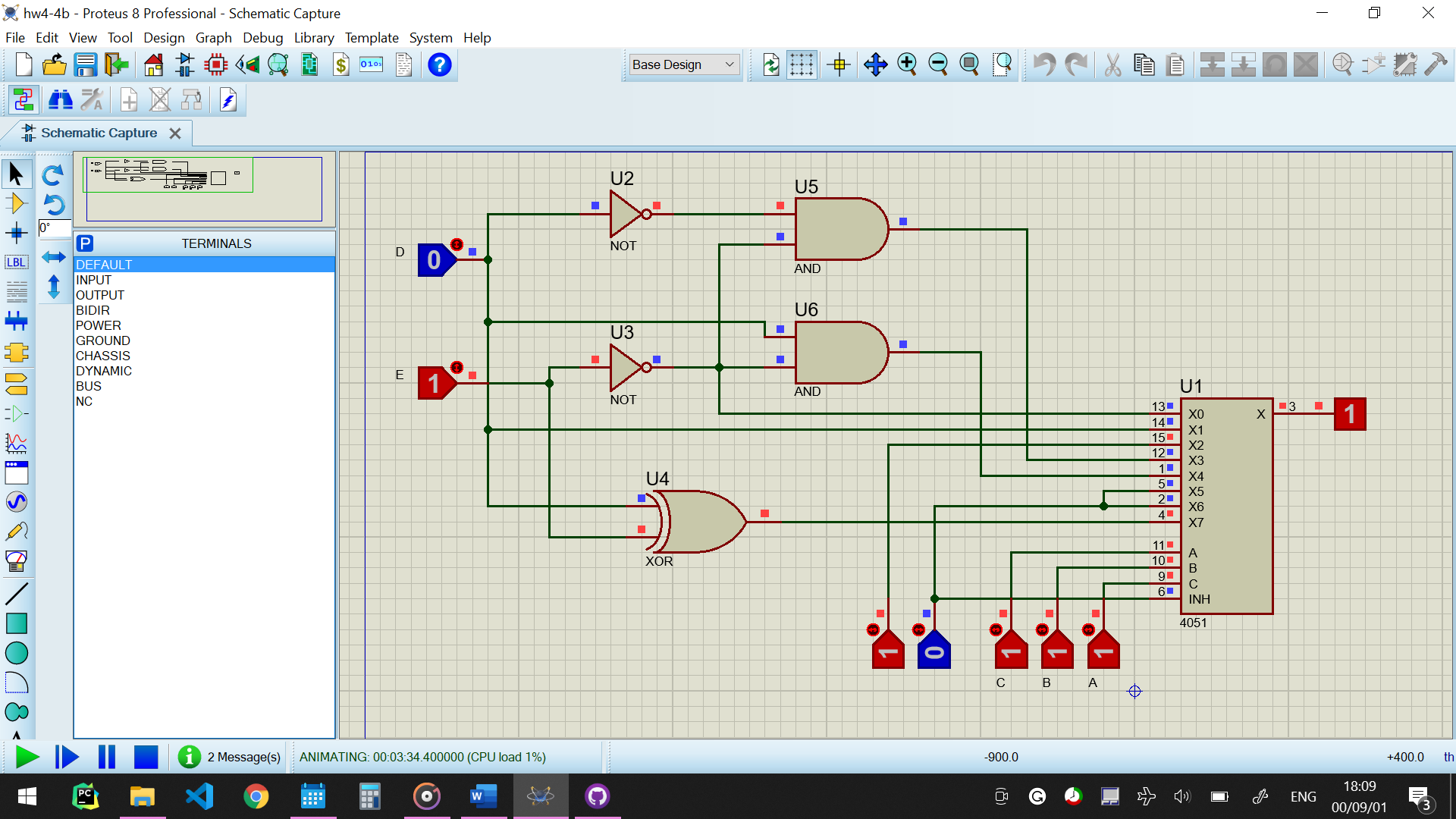
ماکس‌ترم ۱۵ و ۳۱ :

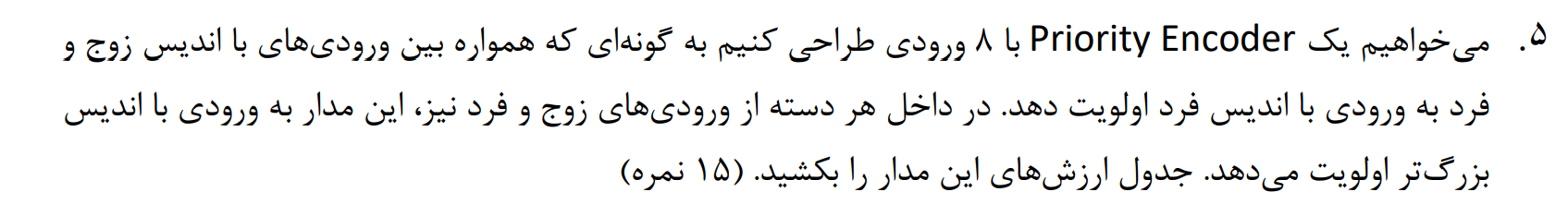




مین‌ترم ۶ و ۲۹:

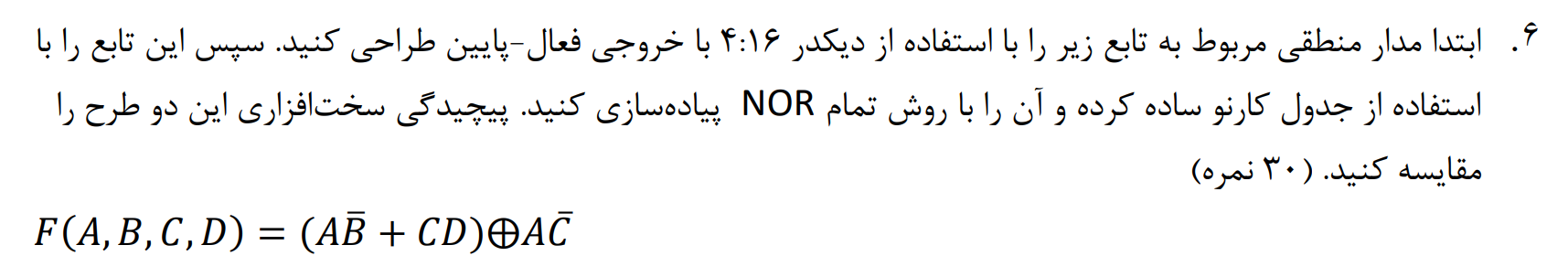




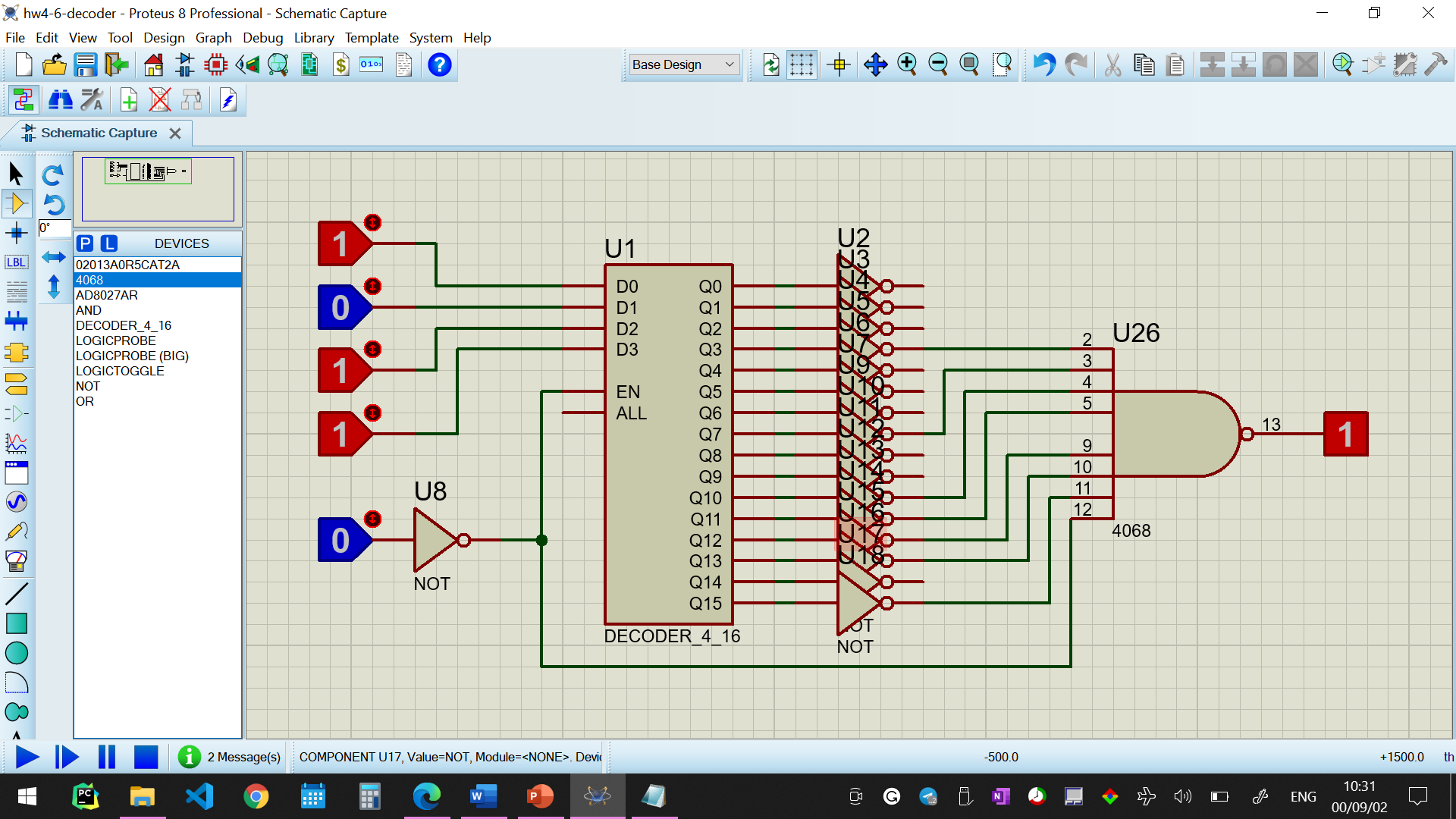


ورودی‌های این مدار را d0 تا d7 می‌نامیم و خروجی‌های آن را A0 تا A3 و V می‌نامیم.

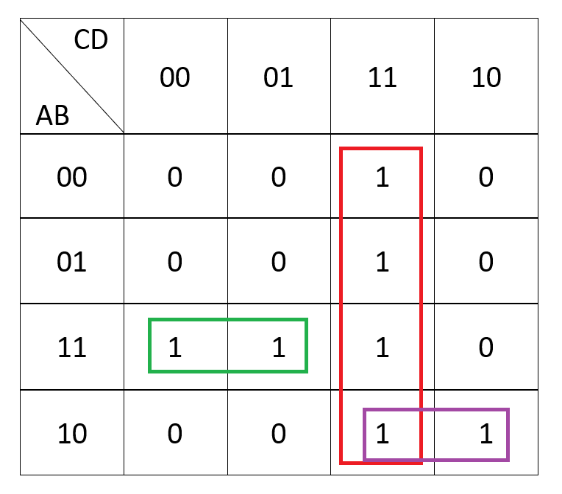
|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| outputs | | | | inputs | | | | | | | |
| V | A0 | A1 | A2 | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 |
| 0 | X | X | X | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | X | X | X | X | X |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

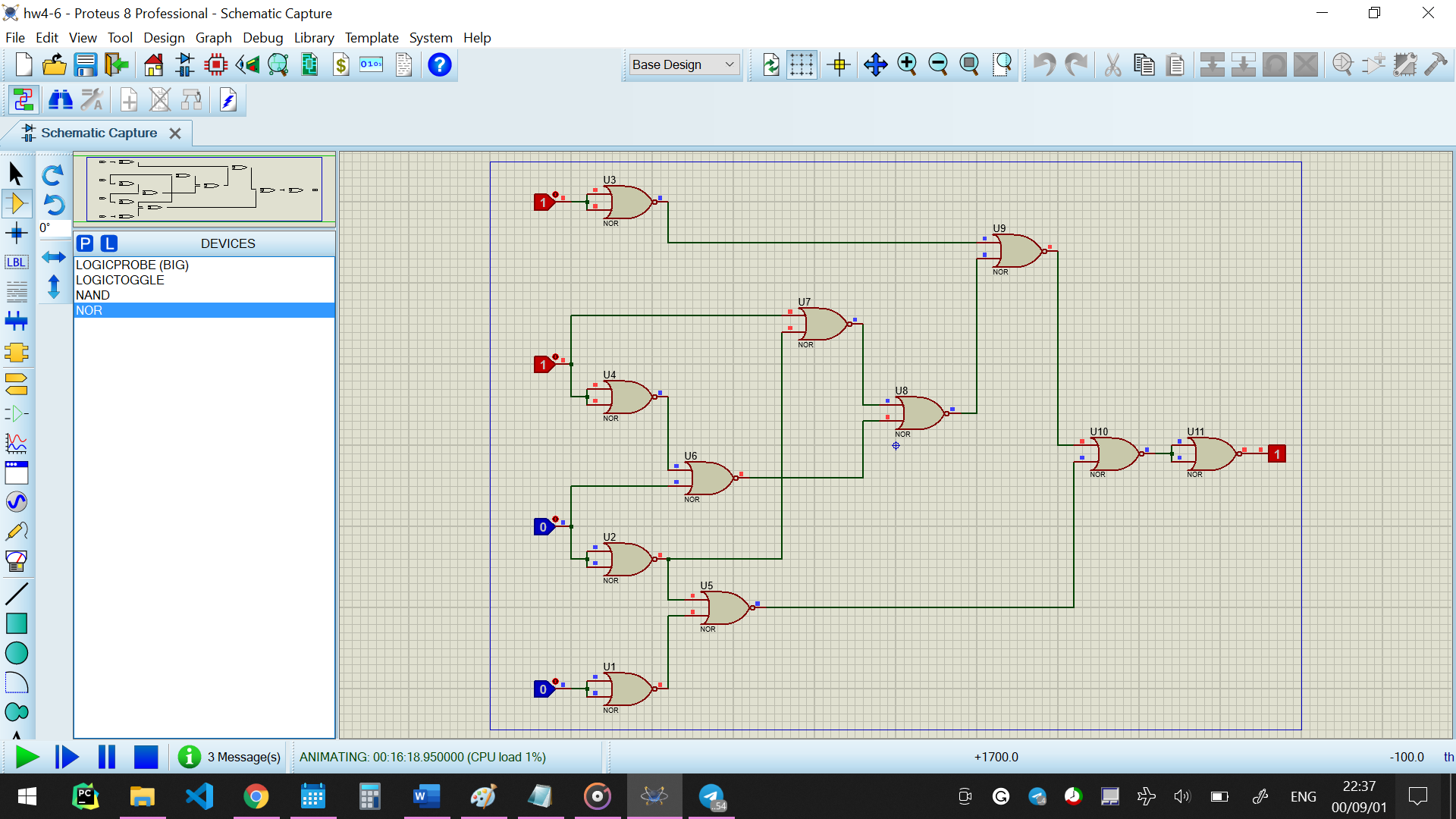
ابتدا محاسبه می‌کنیم که در چه نقاطی

(با توجه به فعال‌پایین بودنِ دیکدر، باید به Enable آن مقدار 0 داده شود، تا خروجی‌های آن مقداری غیر از 111…1 داشته باشند. همچنین در خروجی‌ها نیز، همه‌ی خروجی‌ها ۱ هستند و فقط خروجیِ هم‌اندیس با عدد ورودی ۰ است. متاسفانه دیکدر فعال‌پایین در پروتئوس پیدا نکردم، که با متصل کردن NOT به تمام ورودی‌ها و خروجی‌ها، فرض می‌کنیم که NOT ها بخشی از دیکدر هستند.)

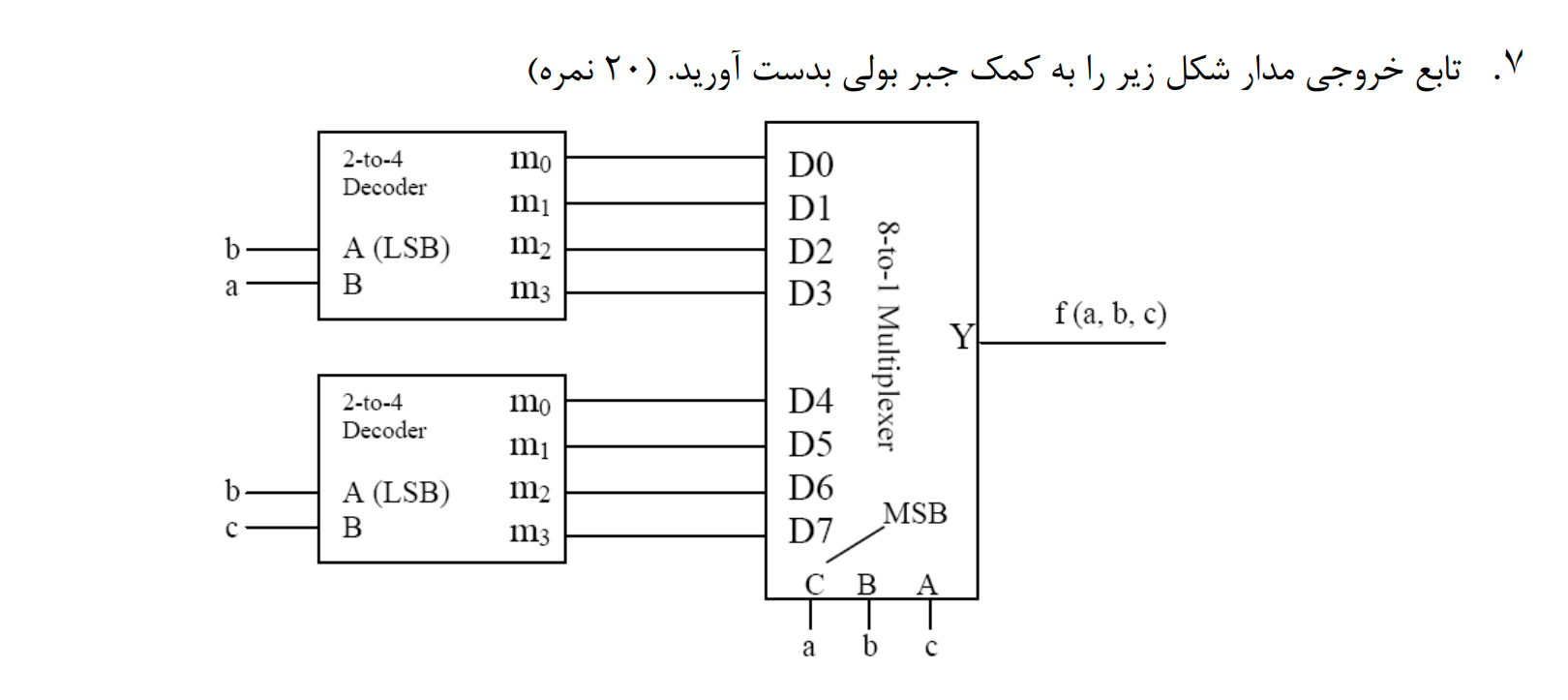


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 10 | 11 | 01 | 00 | CD  AB |
| 0 | 1 | 0 | 0 | 00 |
| 0 | 1 | 0 | 0 | 01 |
| 0 | 1 | 1 | 1 | 11 |
| 1 | 1 | 0 | 0 | 10 |





در مقایسه‌ی 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 نهایتا هم بستن مدار پیچیده‌تر شود و هم طراحی آن.

 براساس مولتی‌پلکسر، تابع را می‌نویسیم:

حال باید توابع D0 تا D7 را بیابیم:

حال مقادیر به دست آمده را جایگذاری می‌کنیم:

