آزمایشگاه معماری کامپیوتر

تابستان ۱۴۰۲

آزمایش چهارم

تبدیل سه رقم BCD به باینری

محمدپیام تائبی – ۴۰۰۱۰۴۸۶۷

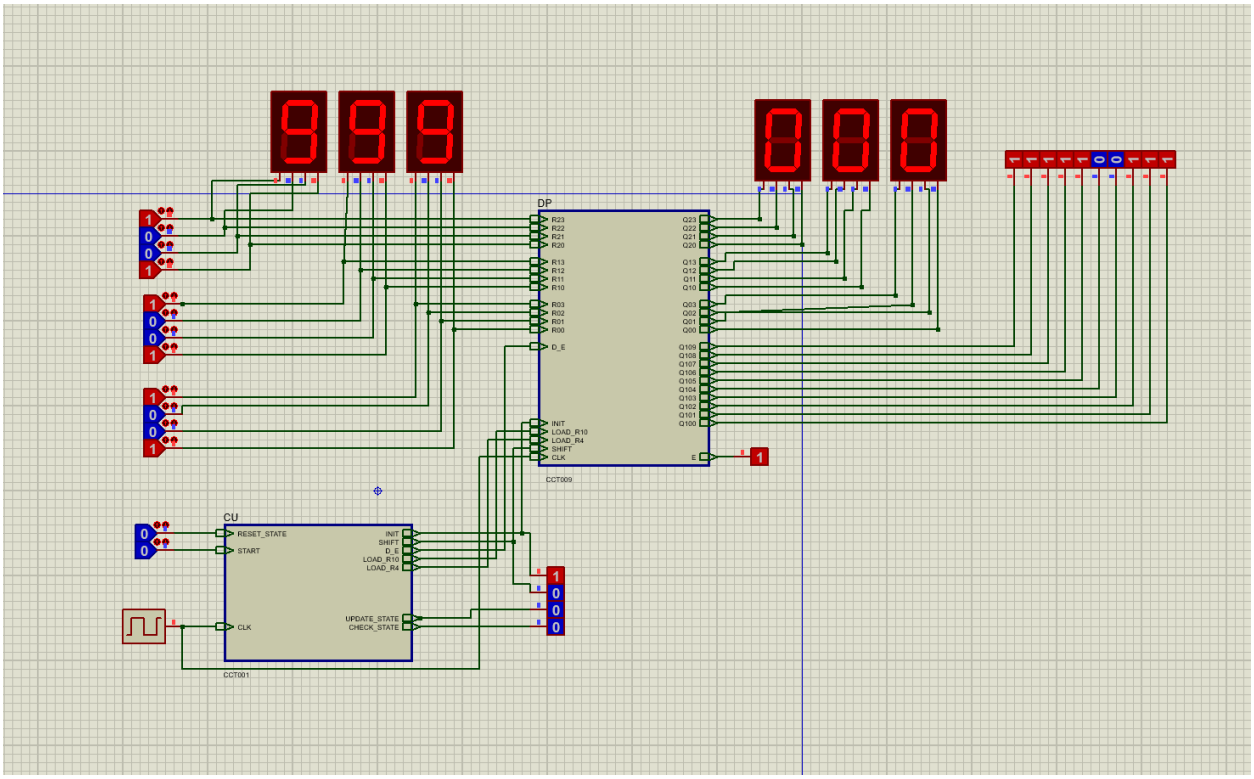
امیرحسین علمدار – ۴۰۰۱۰۵۱۴۴

ماهان بیهقی – ۴۰۰۱۰۴۸۳۴

پیش آزمایش(مربوط به پروتئوس):

شمای کلی آزمایش:

در این آزمایش قصد داریم ۳ رقم BCD که معادل ۱۲ بیت است را ورودی بگیریم و طبق مداری ترتیبی آن را به یک عدد باینری تبدیل کنیم که طبیعتا ۱۰ بیت برای نمایش اعداد ۳ رقمی که بیشترین آن ها ۹۹۹ است کافی است پس ۱۲ بیت ورودی داریم که علاوه بر آن یک بیت start هم داریم که هر وقت یک شود مدار شروع به کار میکند و ۱۰ بیت خروجی و یک بیت هم برای اینکه نشان دهد کار مدار تمام شده است خواهیم داشت .



مثلا در مثال بالا میتوان دید که عدد ۹۹۹ را ورودی گرفته و خروجی ۱۱۱۱۱۰۰۱۱۱ را داده که یک تبدیل درست است.

در ادامه به شرح کار این مدار خواهم پرداخت

شرح الگوریتم آزمایش :

الگوریتم مورد نظر طبق چیزی که در دستور کار آمده است را ابتدا پیاده کردم که این الگوریتم به این شکل است :

الف – عدد دهدهی ورودی را یک بیت به راست شیفت دهید .

ب – اگر با ارزشترین بیت رقم iام یک باشد از آن رقم ۳ تا کم کنید $(1 \le i < r)$.

ج- مراحل الف و ب را آنقدر تکرار کنید تا تمام ارقام دهدهی صفر شوند (حداکثر ۱۰ بار تکرار لازم است).

در پایان بیتهایی که بوسیله شیفت براست بیرون می آیند ، عدد دودویی معادل عدد دهدهی ورودی را تشکیل می‌دهند.

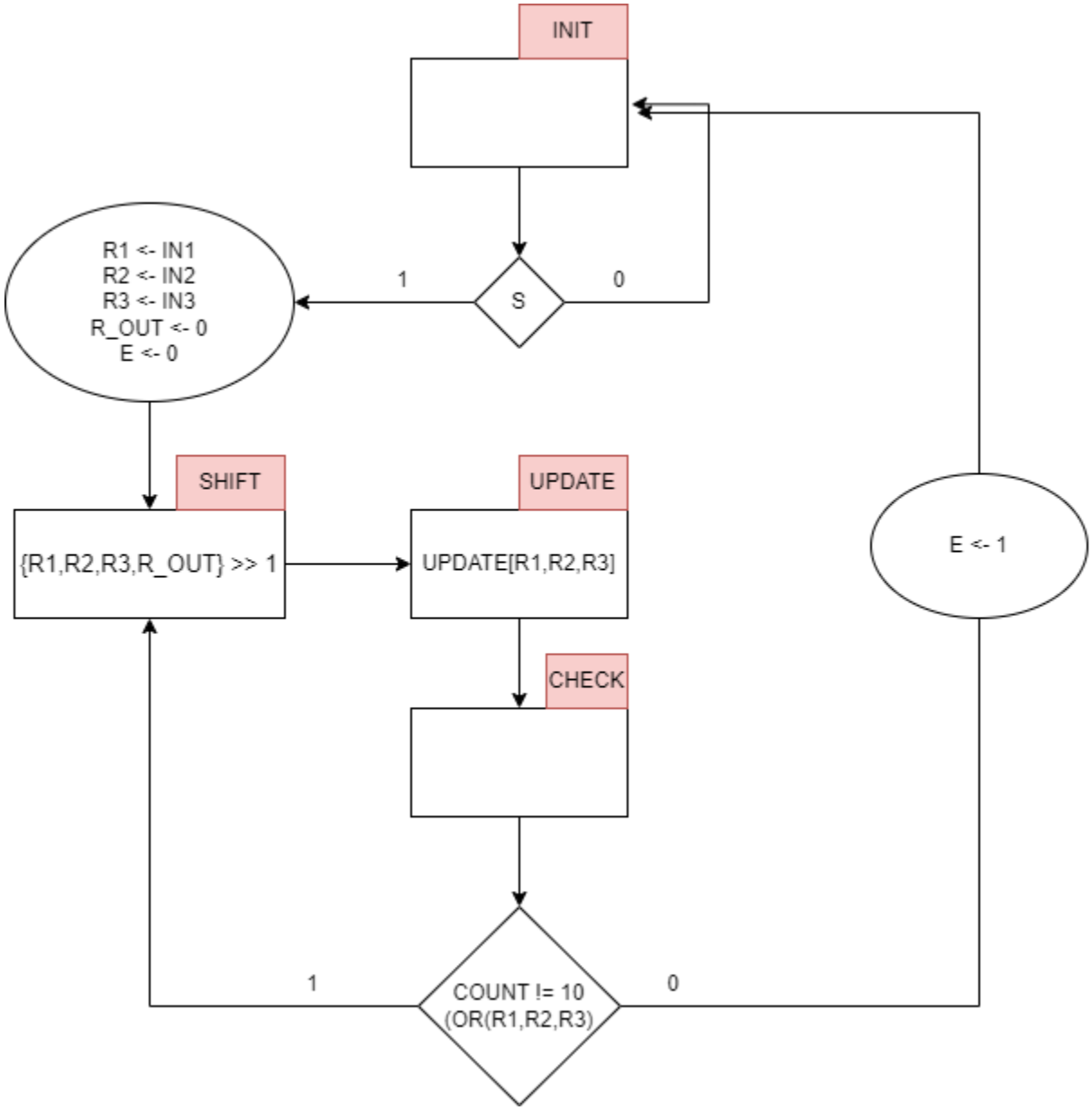
اما بعد از پیاده سازی متوجه شدم این الگوریتم باگ ریزی دارد و اینکه تعداد شیفت ها نباید تا زمانی باشد که OR همه ی بیت های ورودی صفر است بلکه باید به تعداد ۱۰ بار ما شیفت بدیم چونکه در همین مثالی که خود دستور کار زده برای عدد ۱۱۰ ما ۷ شیفت دادیم و ۳ بیت راستی صفر مانده اند انگار برای عدد نهایی باید تقسیم بر ۸ بکنیم که نکرده ایم و جواب در اصل غلط است!

پس همین الگوریتم را پیاده سازی کردم اما با این تفاوت که همواره ۱۰ مرحله باید شیفت تکرار شود و نباید OR ورودی ها چک شود که طبیعتا به یک شمارنده نیاز خواهیم داشت.

پیاده سازی الگوریتم:

برای پیاده سازی این مدار چون یک مدار ترتیبی است از ASM CHART استفاده کردم؛ به طور کلی ۴ حالت برای مدار تعریف میشود یک حالت INIT که حالت اولیه مدار است، یک حالت SHIFT که در آن طبق الگوریتم ۴ رجیستر ما پشت سر هم شیفت میخورند، حالت بعدی UPDATE نام دارد و در این حالت باید ۳ رجیستری که ورودی در آن ها قرار دارد به شرطی که بیت آخر آن ها یک است منهای سه شوند اگر هم بیت آخر صفر هست هیچ تغیری نمیکنند، و استیت آخر CHECK نام دارد که چک میشود آیا ۱۰ بار شیفت انجام شده یا نه، اگر انجام شده بود به INIT میرود و خروجی E که نشان دهنده اماده بودن جواب است را یک میکند در غیر این صورت دوباره باید عمل شیفت تکرار شود

در ادامه میتوانید این ASM CHART را مشاهده کنید:



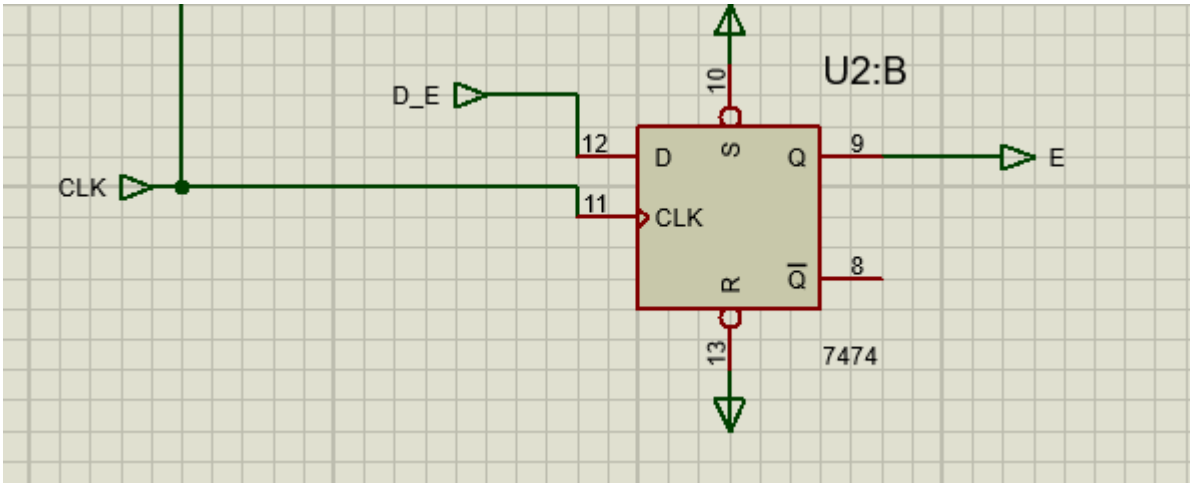
حال باید دیتاپف و کنترل یونیت را طراحی کنیم که ابتدا به طراحی دیتاپف میپردازیم :

طبق نگاهی که به چارت بالا بیاندازیم میتوان فهمید که

برای E یک فلیپ فلاپ ساده نیاز داریم، برای R_out به یک رجیستر ۱۰ بیتی که قابلیت شیفت به راست دارد و برای R1,2,3 به یک رجیستر ۴ بیتی که بتواند شیفت به راست بشود و علاوه بر آن طبق تعدادی بیت کنترلی قابلیت این را داشته باشد که دیتا جدید در خودش وارد کند یا دیتای قبلی را نگه دارد یا اینکه منهای ۳ شود ، و در واقع از استیتی که در آن هستیم خود آن رجیستر ۴ بیتی باید تشخیص دهد که کدام یک از این ۳ باید در خودش نوشته شود، پس به طراحی این مسیر داده خواهیم پرداخت.

DATA PATH:

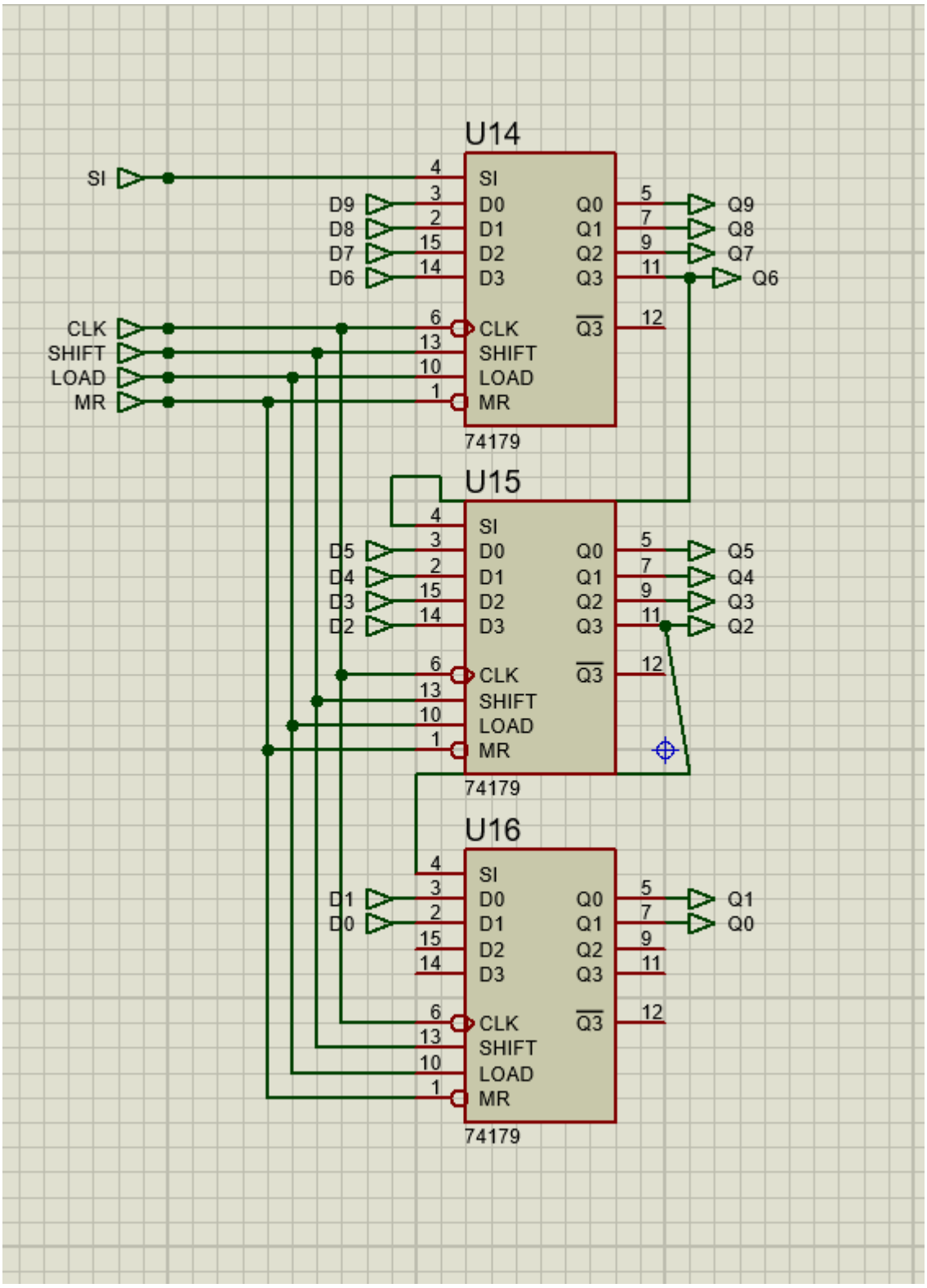
ابتدا برای E یک فلیپ فلاپ در نظر میگیریم (۷۴۷۴)



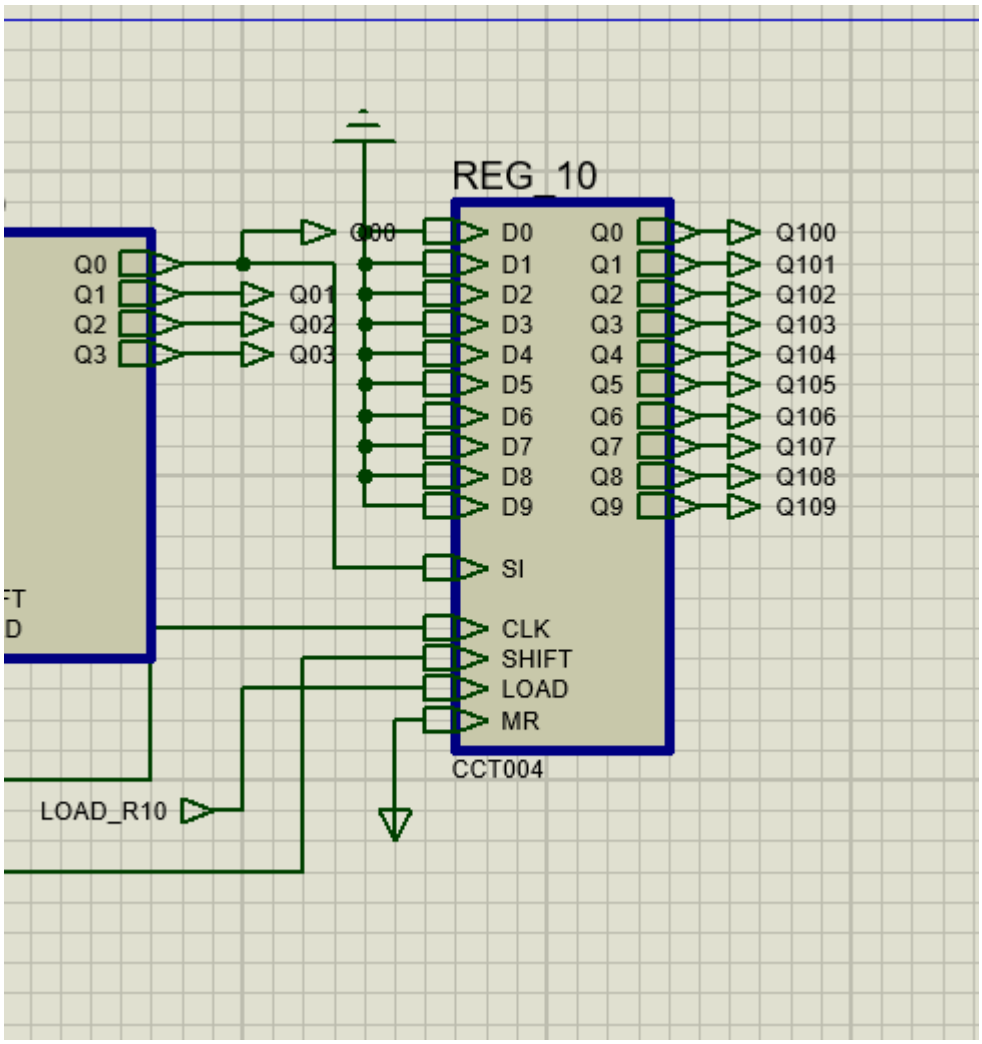
که ورودی آن از کنترل یونیت خواهد آمد علاوه بر آن قرار نیست این بخش مدار به شکل آسنکرون تغیری داشته باشد پس پایه های ست و ریست را یک میدهیم که فعال نشوند.

حال به سراغ طراحی رجیستر ۱۰ بیتی برای نگه داری جواب نهایی میرویم

دقت کنید این رجسیتر باید بتواند به راست شیفت دهد و اینکه بتواند صفر شود یعنی نیاز نیست که هر داده ای رو بتوان در آن لود کرد و همین ۲ کار برای این مدار کافی است؛ طبق قطعات موجود در آزمایشگاه همچین رجیستری نداریم و باید طراحی کنیم، برای این بخش از ۳ رجیستر ۴ بیتی که به هم وصل شده اند استفاده کرده ام که میتوان یک رجیستر ۱۲ بیتی ساخت و ۲ بیت بی ارزش آن را ول گذاشته ام که انگار یک رجیستر ۱۰ بیتی داریم دقت کنید رجیستر های ۴ بیتی ما ۷۴۱۷۹ هستند که ۴ بیت کنترلی ریست ، لود ، شیفت و سریال اینپوت دارند ما باید بیت بی ارزش رجیستر اول را به سریال اینپوت رجیستر بعدی بدیم و همینطور تا آخر تا عمل شیفت درست انجام شود علاوه بر این رجیستر سوم ما ۲ بیت بدون استفاده دارد که از آن ها استفاده نکردیم ، کلاک شیفت و لود همه ی رجیستر ها هم به هم وصل هستند و در نهایت همچین مداری داریم:



که اگر این مدار را به شکل مجتمع نگاه کنیم:



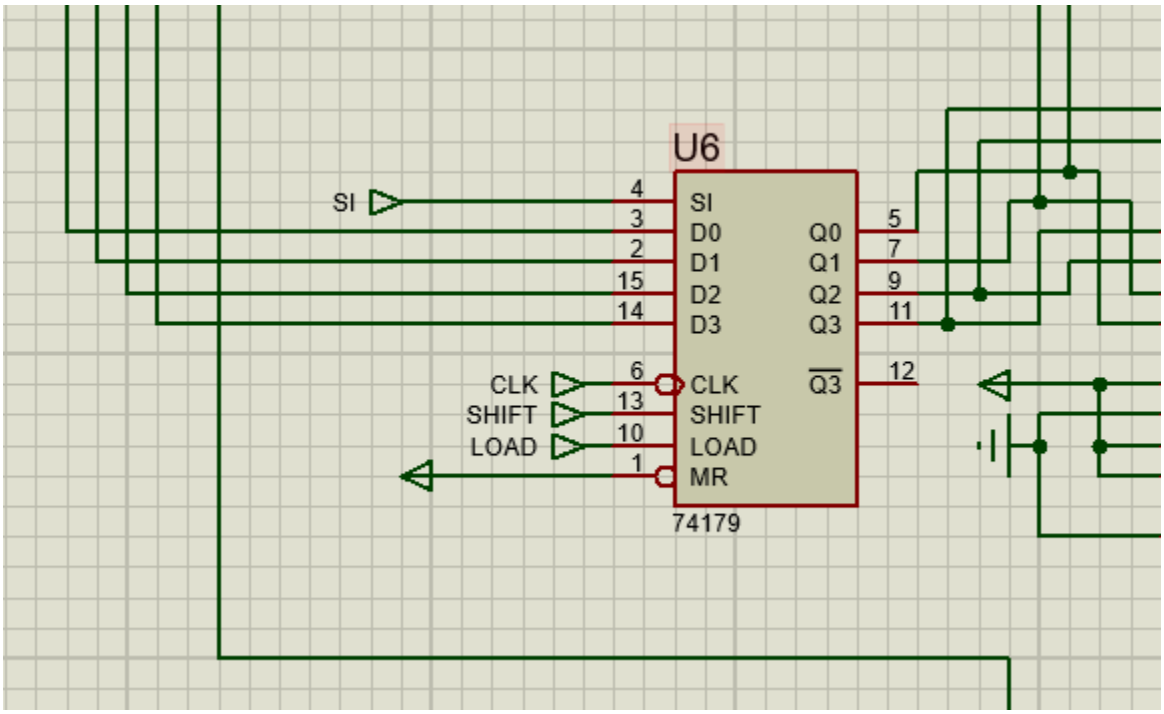
همانطور که میبینید نا یک رجیستر ۱۰ بیتی ساختیم ولی همانطور که گفتم قرار نیست هر عددی در این رجیستر لود شود و فقط ریست میشود یا اینکه شیفت میخورد پس یک راهکار این است که ریست را به یک بزنیم که هیچوقت فعال نشود و برای اینکه صفر را وارد رجیستر کنیم تمام ۱۰ بیت ورودی را به صفر وصل کنیم و هر وقت که لود لجیستر یک باشد این مقدار صفر وارد رجیستر میشود علاوه بر آن هر وقت هم که نیاز به شیفت باشد باید شیفت مدار یک شود تا این رجیستر شیفت بخورد دقت کنید که سریال اینپوت این رجیستر باید از بیت بی ارزش رجیستر R3 بیاید که همینطور وصل شده است و در ادامه به ساختار R3 هم میپردازیم .

پس تا به اینجا یک بیت کنترلی SHIFT و یک بیت کنترلی LOAD_R10 داریم که باید از کنترل یونیت بیاید

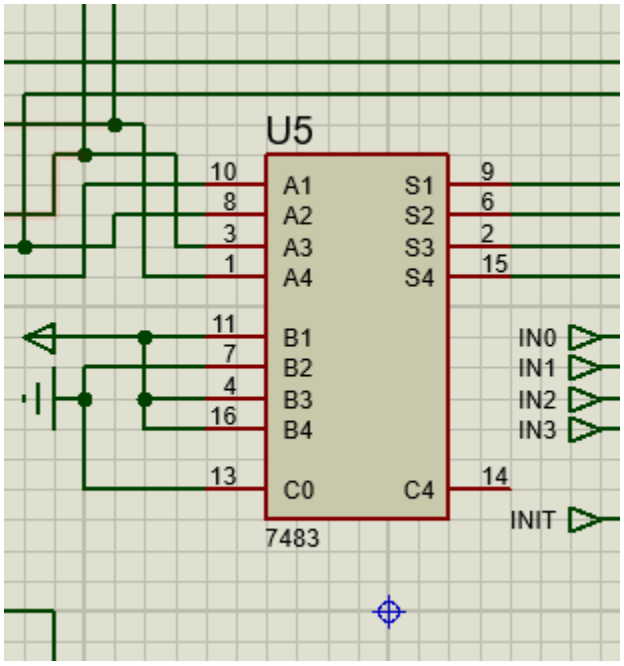
در این بخش باید به طراحی R1,2,3 بپردازیم که طراحی مشابه ای دارند طبق نیاز باید رجیستر ۴ بیتی ای بسازیم که قابلیت شیفت به راست دارد و ۳ قابلیت هولد کردن، منهای سه کردن، لود کردن را داشته باشد

که برای این کار ابتدا از یک رجیستر ۴ بیتی ساده که ۷۴۱۷۹ است استفاده میکنیم سپس خروجی مورد نظر را به یک جمع کننده میدهیم که این جمع کننده ۷۴۸۳ است که دو عدد ۴ بیتی را جمع میزند ورودی ۷۴۸۳ یکی ۴ بیت خروجی رجیستر است و یکی عدد ۱۱۰۱ (چون باید منهای ۳ شود و ۳ معادل ۰۰۱۱ است میتوانیم با مکمل دوی آن جمع کنیم و مکمل دوی عدد ۳ میشود ۱۱۰۱) پس خروجی این جمع کننده منها شده ی عدد ۳ از مقدار فعلی است حال باید بین این مقدار ، نقدار قبلی خود رجیستر و مقدار ورودی جدید انتخاب کنیم که کدام دوباره در رجیستر نوشته شود برای این کار از مالتی پلکسر ها استفاده میکنیم به این شکل که فرض کنید مالتی پلکسری داریم که یک بیت کنترلی دارد و بین ۲ گذرگاه ۴ بیتی انتخاب میکند این مالتی پلکسر ۷۴۱۵۷ هست که بین دو عدد ۴ بیتی انتخاب میکند ؛ حال باید ۲ تا از این مالتی ملکسر ها را پشت سر هم قرار دهیم تا انتخاب انجام شود، در این مدار ابتدا بین مقدار منهای ۳ شده و ورودی IN تصمیم گرفته میشود با طوری که اگر در استیت INIT باشیم IN از مالتی پلکسر خارج میشود و در غیر این صورت مقدار منهای سه شده خارج میشود پس به یک بیت INIT هم نیاز داریم که از کنترل یونیت بگیریم ، در مالتی پلکسر بعدی بین همین مقداری خروجی این مالتی پلکسر و مقدار اولیه رجیستر باید تصمیم بگیریم که با یک گیت NOR میتوان این کار را انجام داد به این شکل که اگر در INIT نباشیم و بیت آخر رجیستر هم صفر باشد باید به این معنی است که در استیت اپدیت هستیم ولی چون بیت اخر صفر هست نباید تغیری انجام شود پس در این حالت باید مقدار اولیه انتخاب شود و در هر حالتی غیر از این باید مقدار مالتیپلکسر اول انتخاب شود پس از گیت NOR اینجا استفاده میکنیم که ۷۴۰۲ نام دارد

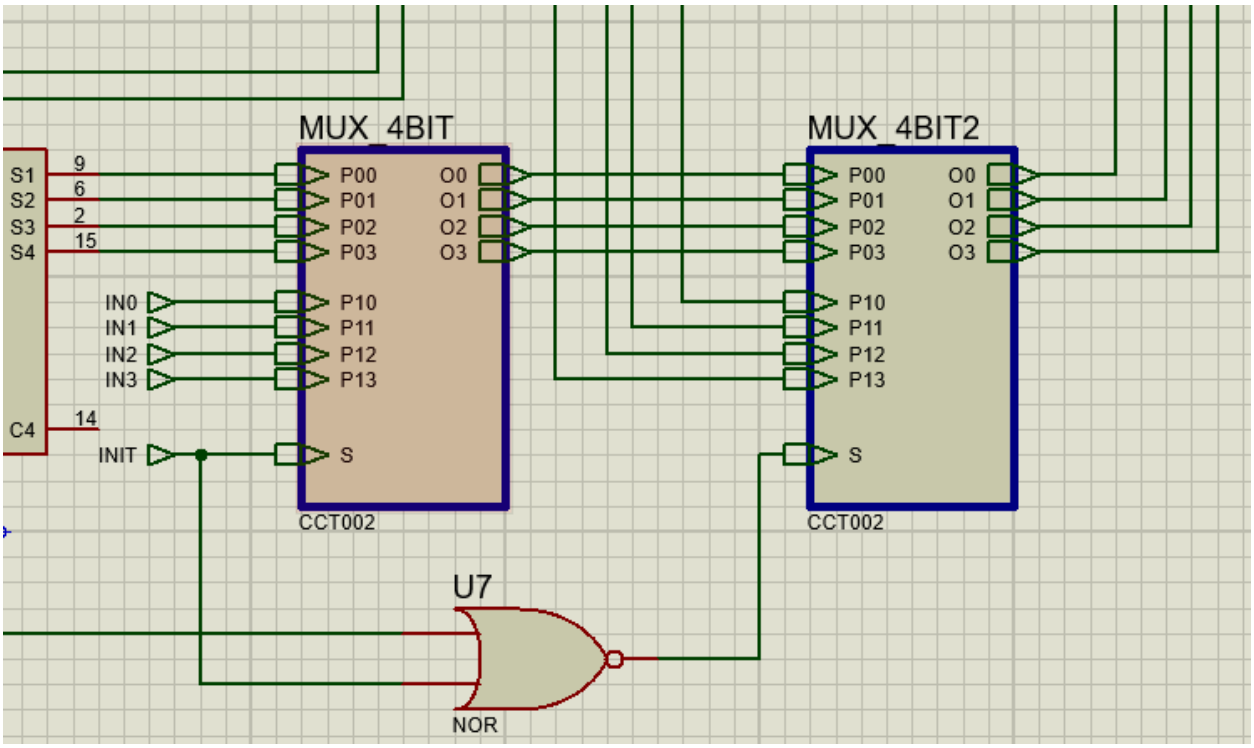
رجیستر ۴ بیتی:



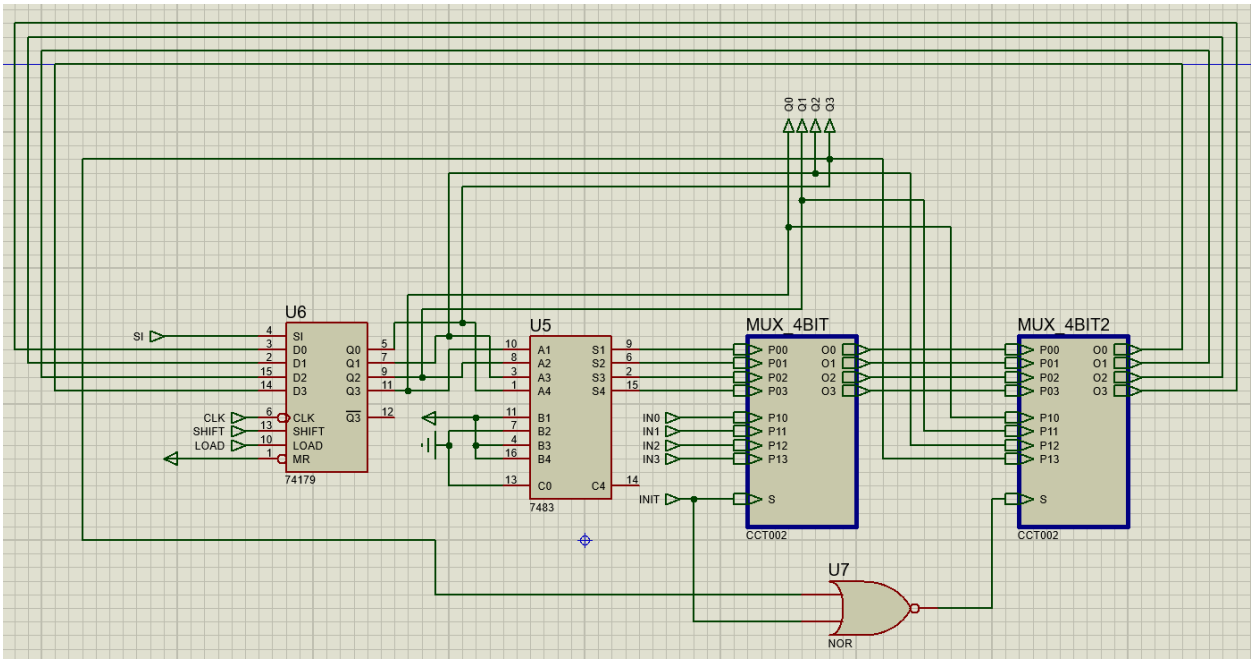
منهای ۳ کردن :



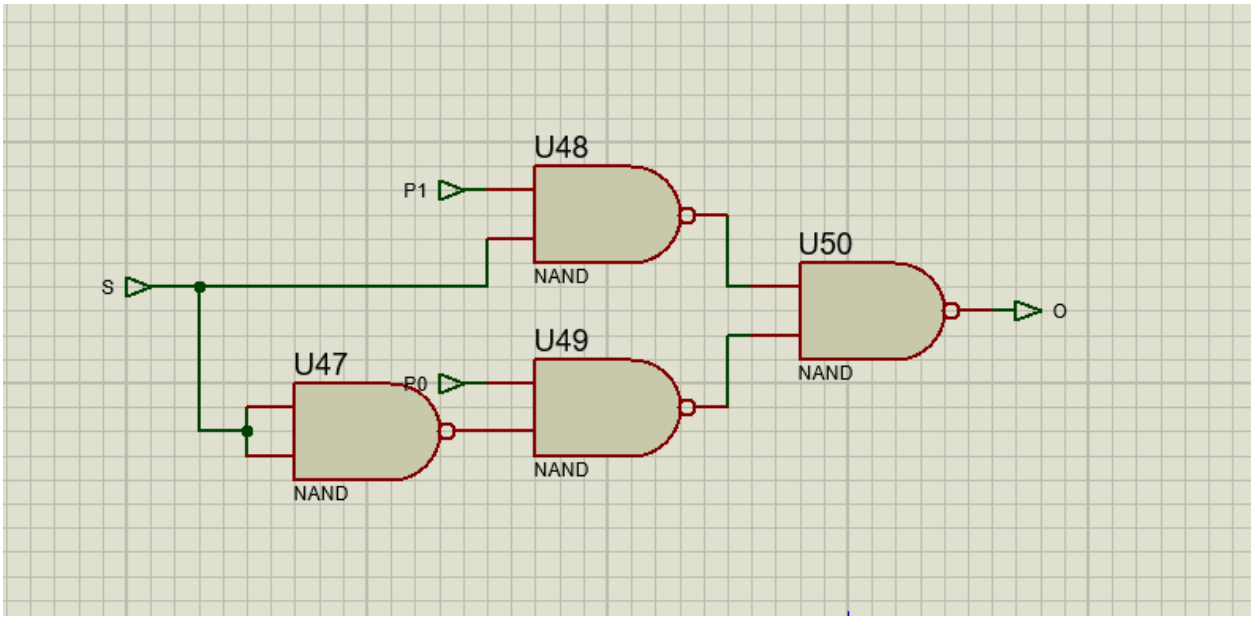
مالتی پلکسرها :



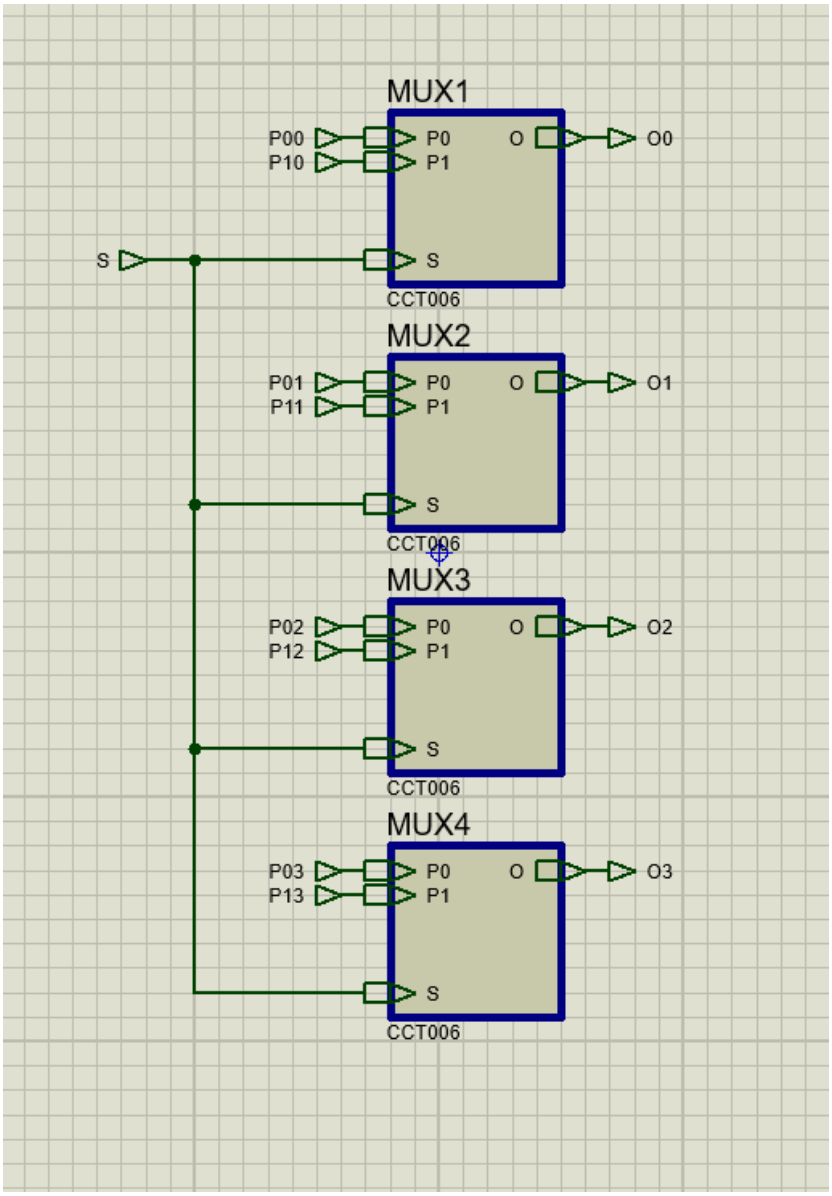
مدار نهایی:



اما نکته مهمی که باید به آن اشاره کنم این است که در طراحی پروتئوس از مالتی پلکسر به شکل مستقیم استفاده نکردیم(۷۴۱۵۷) و با توجه به قطعات داخل آزمایشگاه ، فرض کردیم که مالتی پلکسری نداریم و باید خودمان طراحی کنیم پس برای طراحی این مالتیپکلسر ابتدا یک مالتی پلکسر ۲ به ۱ که یک مدار ترکیبی خیلی ساده دارد به این شکل ساختیم :

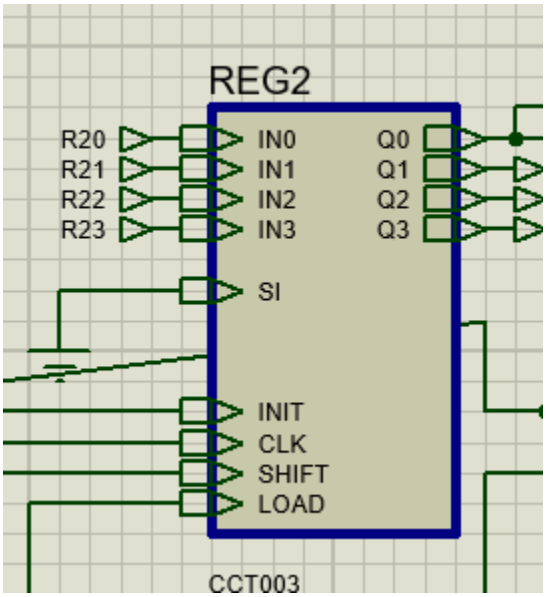


که اگر S صفر باشد P0 و اگر یک باشد P1 از مدار خارج میشود و سپس با کنار هم قرار دادن ۴ تا از این مدار میتوان به راحتی این مالتی پلکسر را ساخت به این شکل:



البته در پیاده سازی نهایی متوجه شدیم که ۷۴۱۵۷ در آزمایشگاه وجود دارد و این طراحی بیهوده و اضافی است و در آخر از آن استفاده نکردیم و مستقیما از قطعه ۷۴۱۵۷ استفاده کردیم که دقیقا همین کار را میکند.

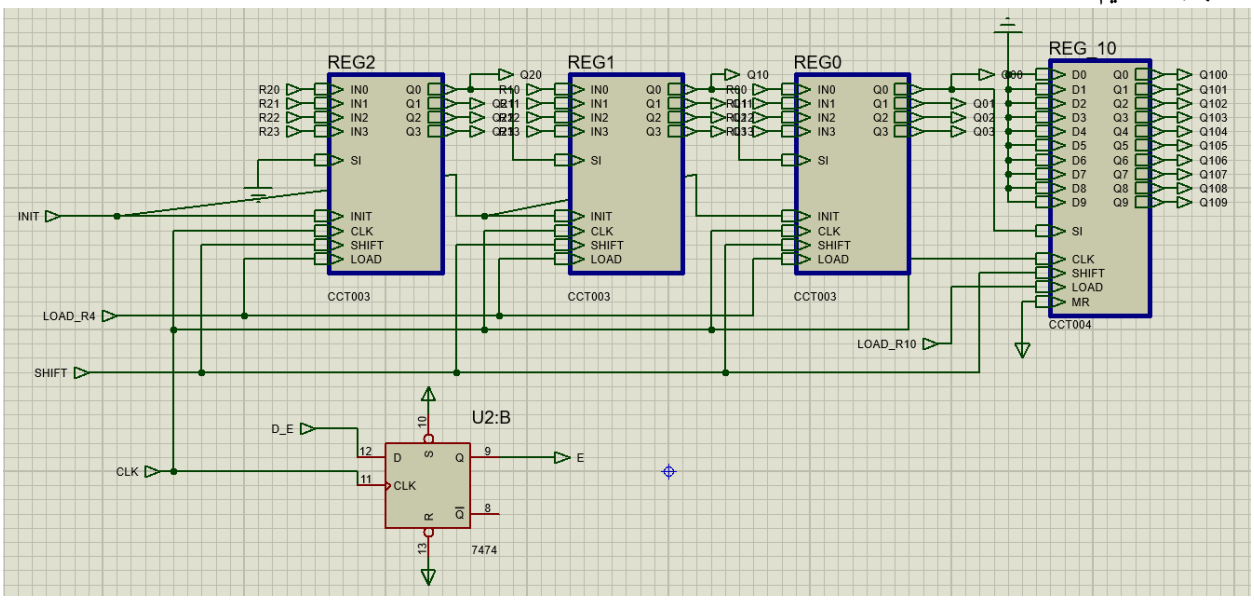
پس در نهایت همچین قطعه ای داریم که برای این ۳ رجیستر از این ها استفاده میکنیم:



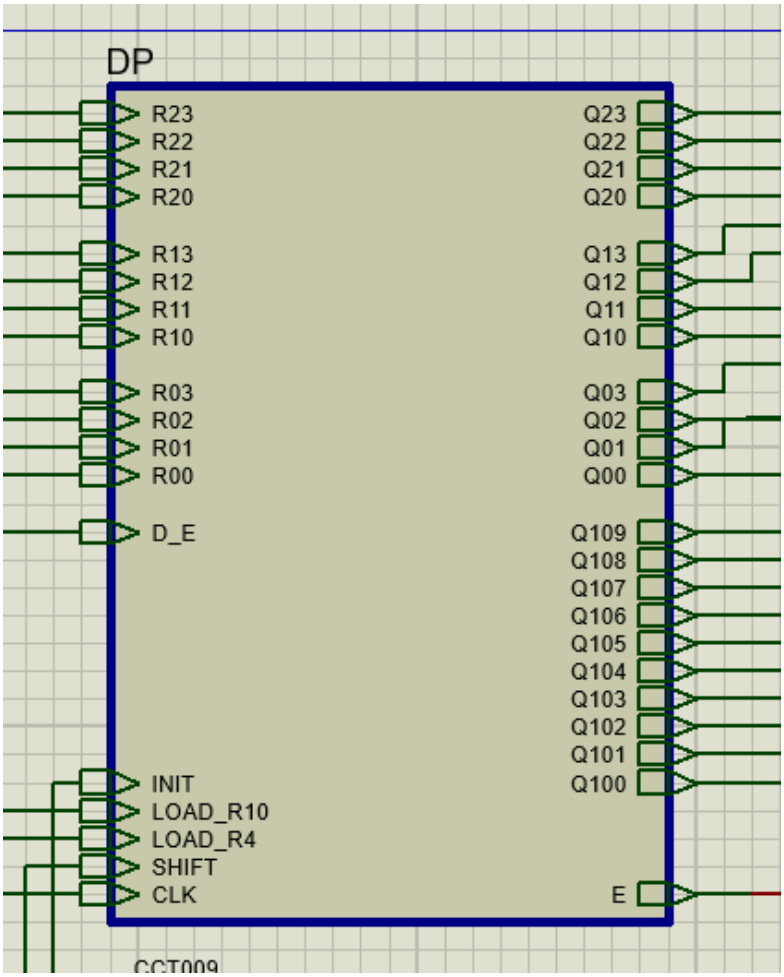
که ۴ بیت ورودی اولیه دارد و ۴ بیت کنترلی که یکی سریال اینپوت است و یکی INIT که باتوجه به اینکه در استیت INIT هستیم یا نه هنگامی که لود مدار فعال باشد عملکرد مدار متفاوت است و یک بیت هم برای شیفت که هر وقت فعال باشد یعنی ما در استیت شیفت هستیم و رجیستر شیفت میدهد، لود هم وقتی فعال است که یا در استیت UPDATE هستیم یا در INIT که برای اینکه بفهمد در کدام هستیم بیت ورودی INIT این را به رجیستر میفهماند (دقت کنید کنترل یونیت باید طوری باشد که لود این رجیستر فقط در این ۲ استیت فعال شود و جای دیگری فعال نشود)

حال باید ۳ تا ازین رجیستر ها را بمشت هم ببندیم و بیت کم ارزش هر کدام را به سریال اینپوت بقلی وصل کنیم و بیت کم ارزش اخری به سریال اینپوت R_Out که قبل تر طراحی کردیم وصل میشود و سریال اینپوت رجیستر اول هم باید صفر باشد.

در نهایت داریم :



که شکل مجمتع آن به این شکل است :

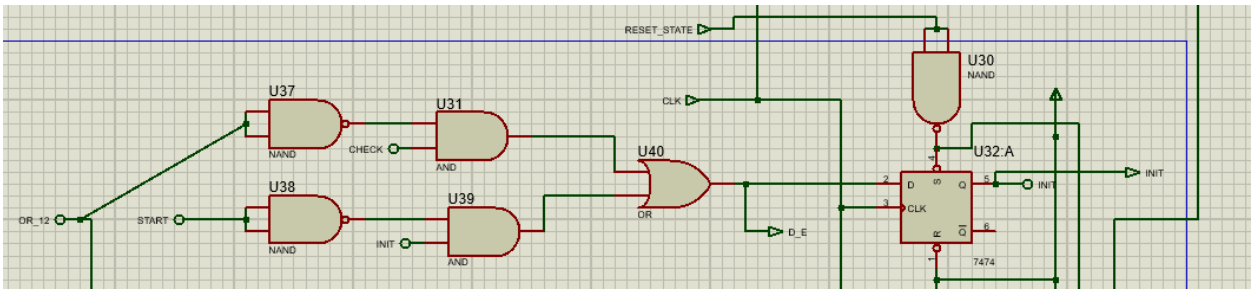


۱۲ بیت که ورودی مستقیم از کاربر می آید ، یک بیت D_E و ۴ بیت کنترلی پایین سمت چپ قطعه باید از کنترل یونیت بیاید، کلاک مدار که برای همه قطعات یکی است و خروجی ها نیز ۱۰ بیت خروجی اصلی مدار است، ۱۲ بیت رجیستر های R1,2,3 را نیز جهت دیباگ و نشان دادن عملکرد مدار قراره داده ام که میتوان قرار نداد و تاثیری در عملکرد مدار ندارد، در آخر یک بیت E هم خروجی است که میگوید کار تمام شده یا خیر حال با توجه به این ۵ بیت کنترلی ورودی باید به طراحی کنترل یونیت بپردازیم که هم ترتیب استیت ها را رعایت کند و هم این ۵ بیت را به دیتا پف بدهد.

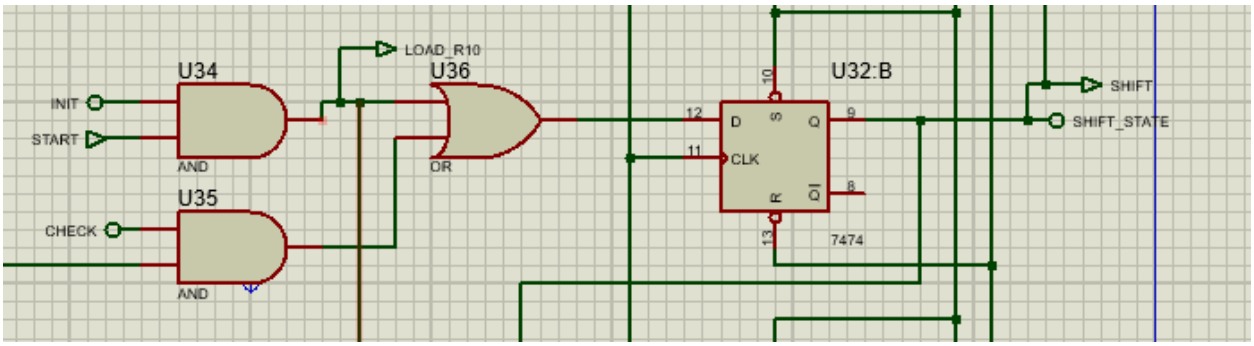
حال دیتاپف آماده است و به سراغ کنترل رونیت میرویم.

CONTROL UNIT :

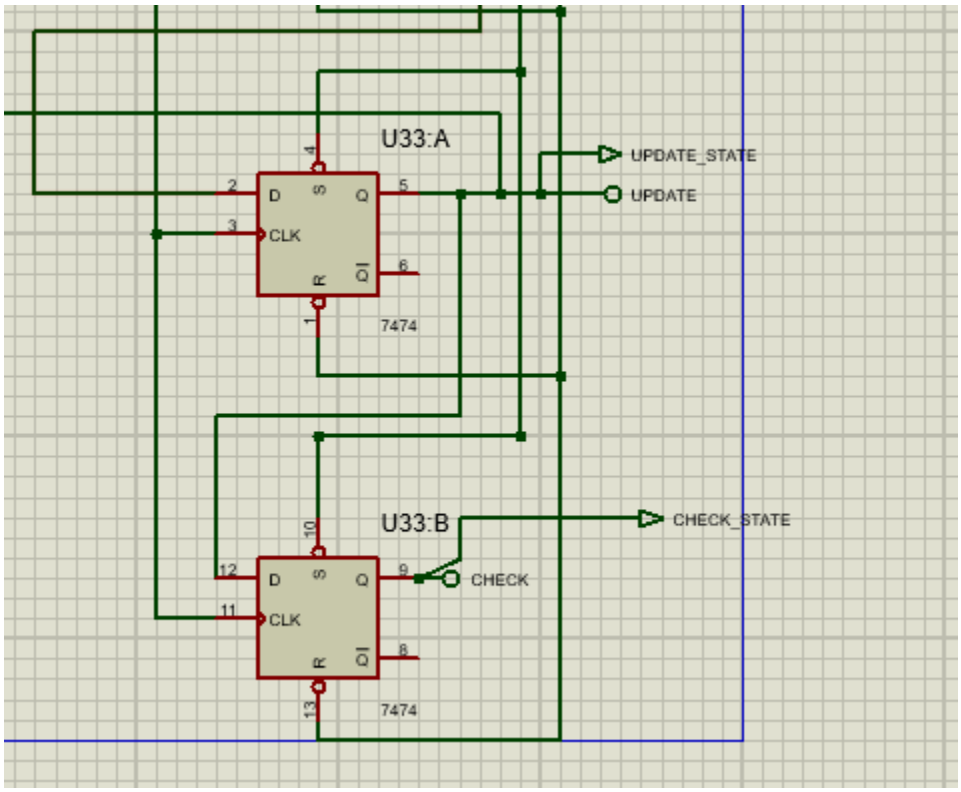
ابتدا باید به شکل one hot برای هر استیت یک فلیپ فلاپ از نوع ۷۴۷۴ قرار دهیم سپس برای هر یک ، یک مدار ترکیبی میسازیم که مشخص میکند در چه حالاتی وارد این استیت خواهیم شد که مطابق شکل هنگامی که در اینیت هستیم و هنوز کار مدار شروع نشده و یا وقتی که مدار در استیت چک هست و کارش تمام شده باز به اینیت بر میگردد پس مدار مربوط به این استیت به این شکل هست:



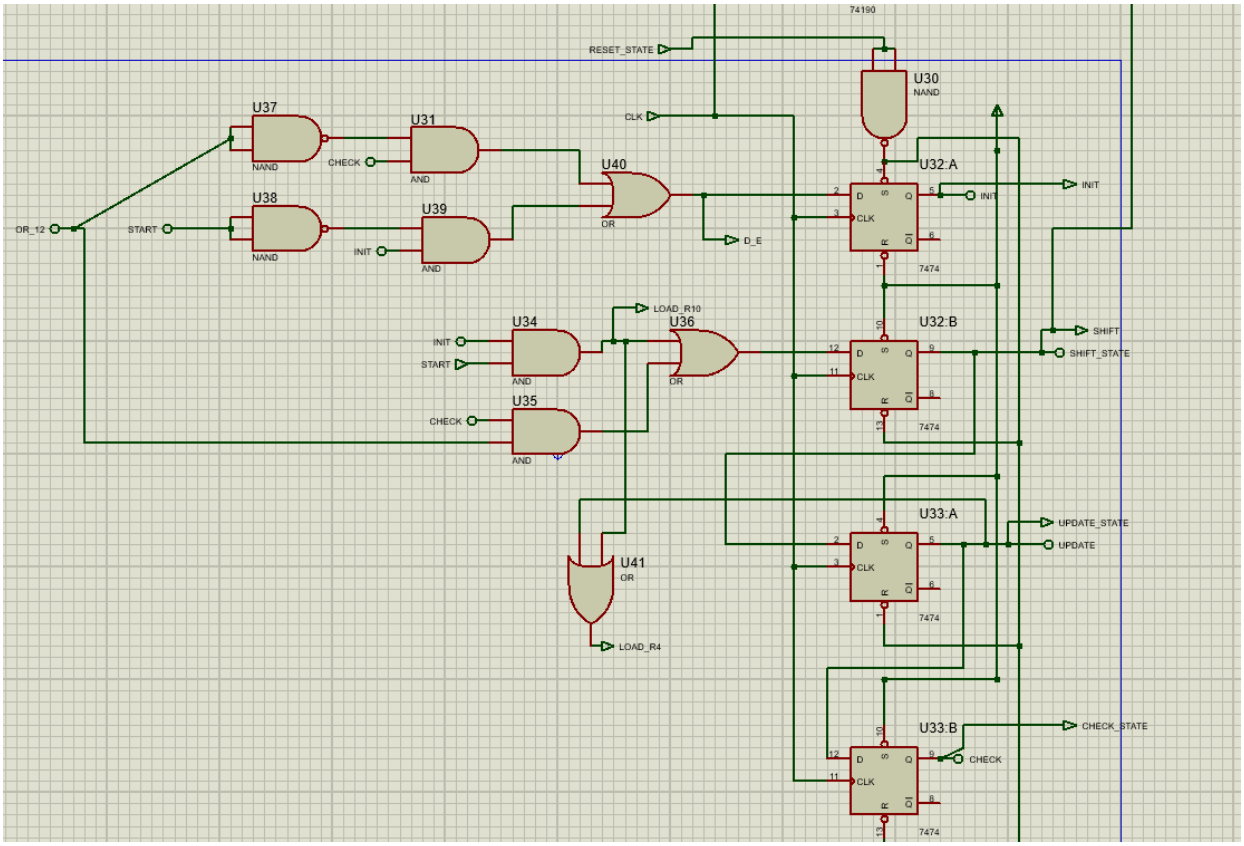
استیت بعدی استیت شیفت است که طبق همین طراحی در ۲ حالت وارد آن میشویم یکی هنگام چک به شرط یک بودن OR_12 و یکی هم به شرط اینکه در اینیت باشیم و s یک باشد پس مدار این استیت نیز به این شکل است:



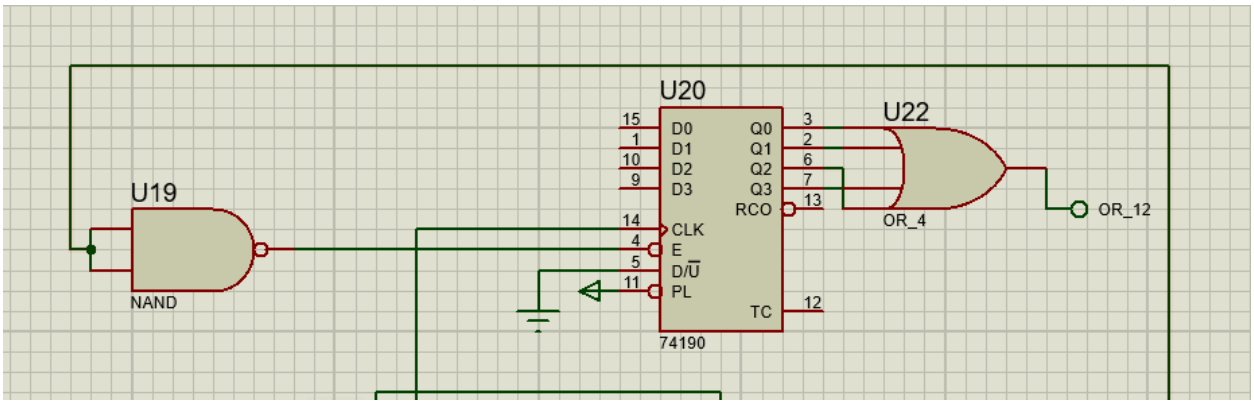
استیت اپدیت و چک نیز به راحتی پیاده میشوند چون ورودی اپدیت فقط بعد از استیت شیفت است و ورودی چک نیز دقیقا استیا اپدیت است پس این ۲ به مدار ترکیبی این نیاز ندارند و به این شکل میشود:



دقت کنید که هنگام شروع مدار باید یک سیم اسنکرون داشته باشیم که وقتی یک میشود ما را به استیت INIT ببرد چون هنگام شروع کار همه فلیم فلام ها صفر اند و انگار در ایچ استیتی نیستیم! پس یک سیم به اسم RESET_STATE قرار دادم و به ست اینیت و به ریست ۳ فلیپ فلاپ دیگر وصل کردم تا در نهایت مدار این شکلی شود:



دقت کنید OR_12 که قرار دادم بیتی است که وقتی یک است یعنی مدار هنوز کار دارد و باید شیفت بخورد و وقتی صفر میشود یعنی کار ندار تموم شده و ۱۰ بیت شیفت انجام شده اسم گذاری نسبتا نا مربوطی دارد چون ابتدا مدار رو طوری ساخته بودم که این سیم واقعا ۱۲OR بیت ورودی بود اما خب الگوریتم دستور گزارش غلط بود و مدار به طوری تغیر کرد که یک شمارنده مشخص میکند که این سیم صفر باشد یا یک ; حال برای پیاده سازی این شمارنده از قطعه ی ۷۴۱۹۰ استفاده میکنیم که یک شنارنده BCD است که از ۰ تا ۹ میشمارد و مجددا صفر میشود و هنگامی که این خروجی ۰ را دارد باید سیم گبته شدا صفر شود و در غیر این صورت باید یک باشد که با یک OR به سادگی حل میشود پس مدار این بخش شمارش به این شکل است:



مدار بالا که مشاهده می کنید ورودی D0,1,2,3 ندارد چون ما از صفر میشماریم نه از عدد دیگری، باید به بالا بشمرد پس Du صفر است E نیز برابر نات شیفت است به این معنی که هر وقت کلاک میخورد اگر در استیت شیفت باشیم یک دونه میشمارد ولی اگر در استیت دیگری باشیم شمارشی انجام نمیشود و در آخر pl مربوط به ریست کردن است که به آن نیازی نداریم و آن را غیر فعال میکنیم.

برای ۴ بیت کنترلی خروجی نیز به راحتی با مداز های ترکیبی میتوان ساخت دقت کنید ما ۴ فلیپ فلاپ را خروجی میدهیم تا جهت راحت تر نشون دادن عملکرد مدار بتوان دید در هر کلاک در چه استیتی هستیم.

INIT که مشخص است و مداری نیاز ندارد

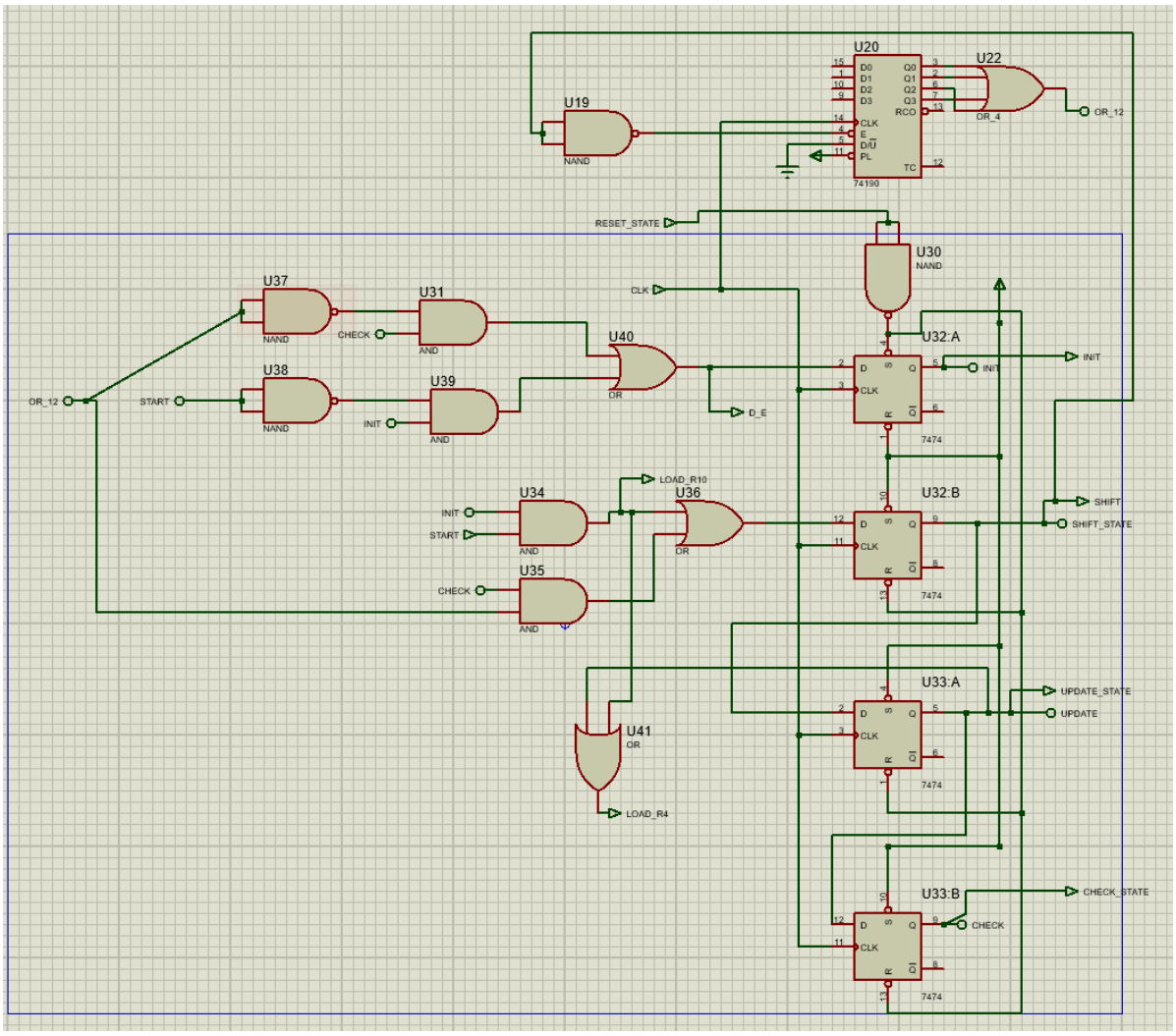
SHIFT هم همینطور

D_E را وقتی یک میکنیم که بخواهیم وارد استیت اینیت بشویم پس به ورودی فلیپ فلام مربوط به اینیت وصل میشود

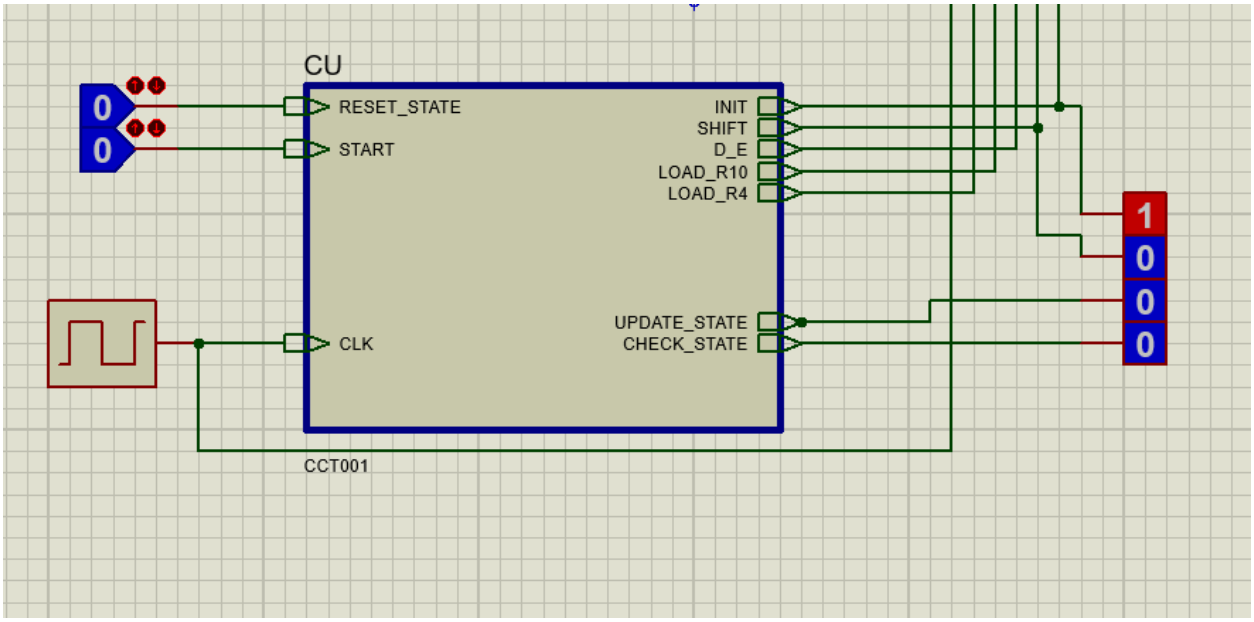
LOAD_R10 باید وقتی که در اینیت هستیم و استارت میشویم یک شود پس به آنجا وصل میکنیم

LOAD_R3 نیز هم در حالت بالا هم در اپدیت باید یک شود پس با حالت اپدیت اور میکنیم و خروجی میدهیم

در نهایت مدار کنترل یونیت ما به این شکل است:



که به شکل مجتمع داریم :



دقت کنید که کلاک را با کلاک جنریتور و با هر فرکانس دلخواهی میتوانیم بدهیم ۴ خروجی راست به ترتیب از بالا به مایین نشان دهنده استیت حال حاضر هستند که به ترتیب یعنی:

INIT

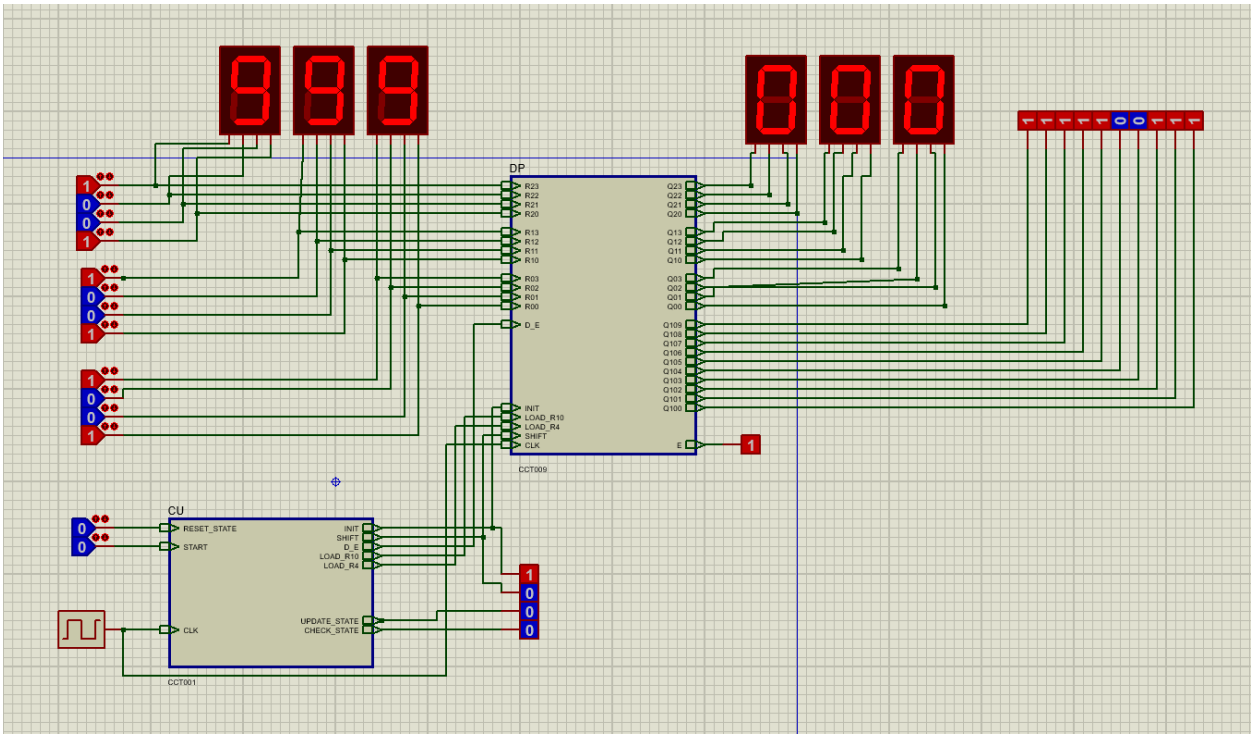
SHIFT

UPDATE

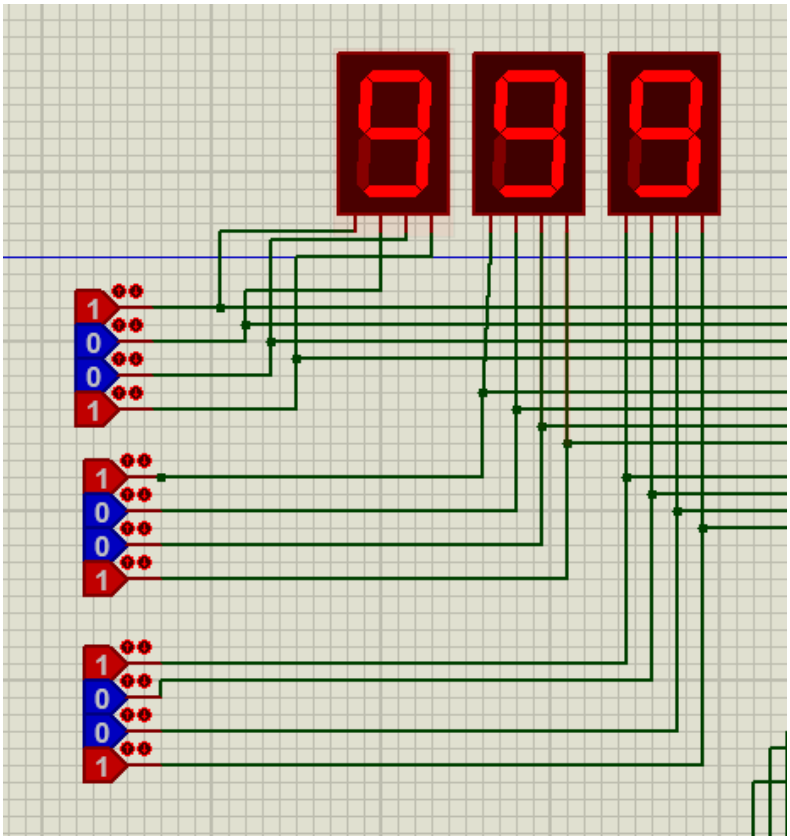
CHECK

اتصال کنترل یونیت و دیتا پف:

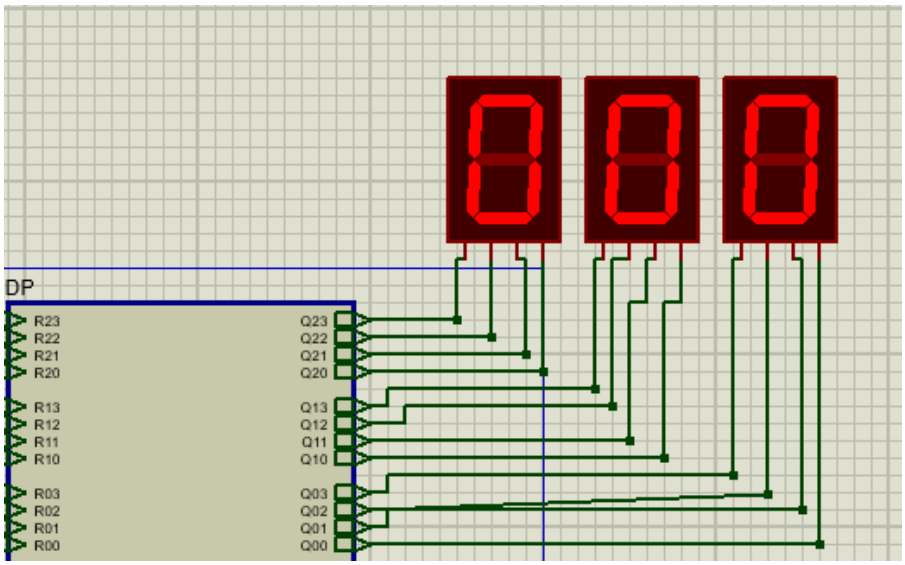
بعد از اتصال این ۲ قطعه و وصل کردن کلاک به و اتصال تمام سون سگمنت ها ندار نهایی به این شکل میشود:



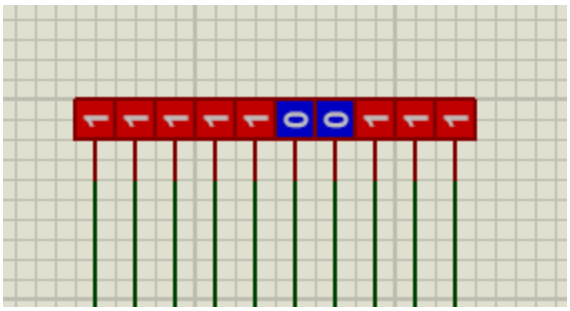
شرح مدار و طرز استفاده از آن :



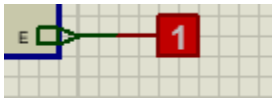
در این بخش ورودی ها قرار میگیرند ، طبیعتا باید ۴ بیت ها را طوری ورودی داد که BCD باشند



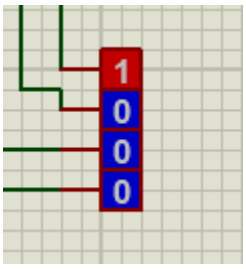
و در اینجا میتوانیم خروجی هر کدام از رجیستر های مربوط به ورودی را ببینیم که جهت راحت تر فهمیدن مدار قرار داده شده



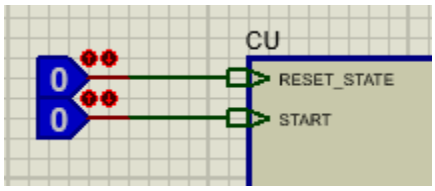
این ۱۰ بیت نیز خروجی نهایی است



این بیت به معنی آماده بودن یا نبودن مدار است



این ۴ بیت نشان میشدهد در کدام استیت هستیم



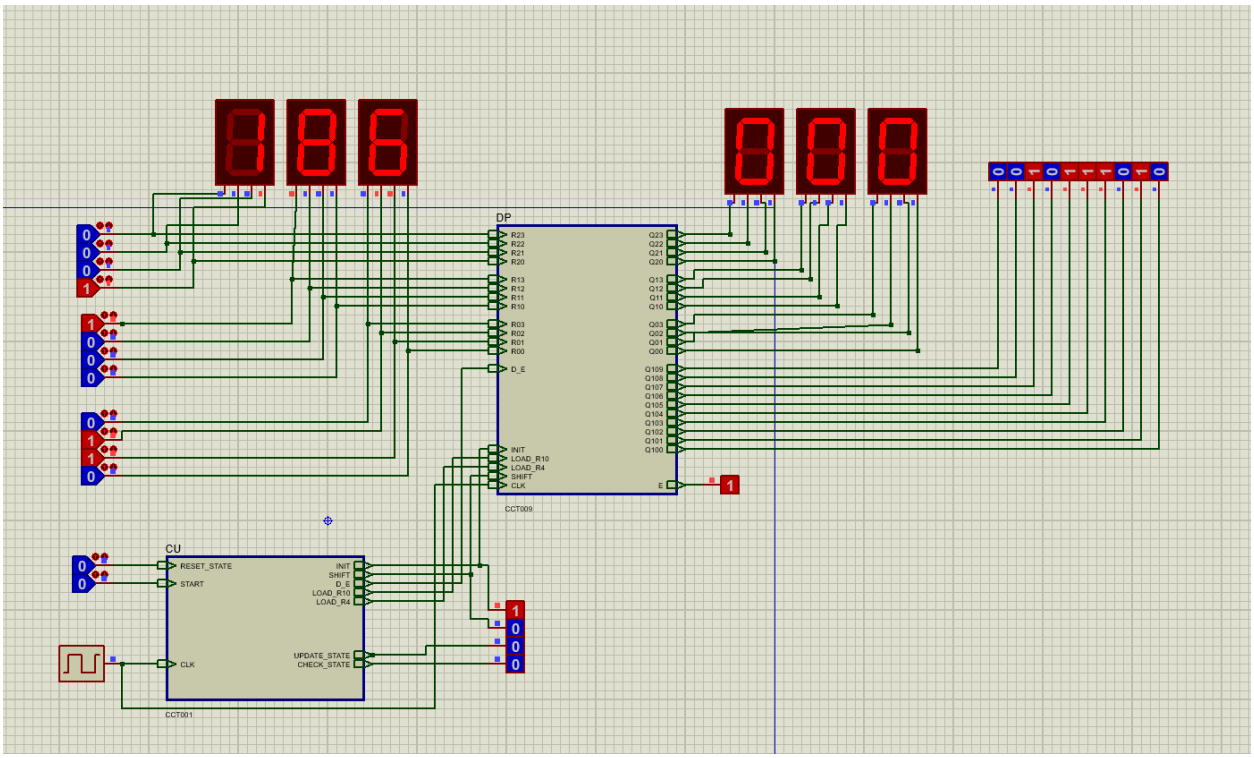
و این ۲ بیت ریست و استارت هستند

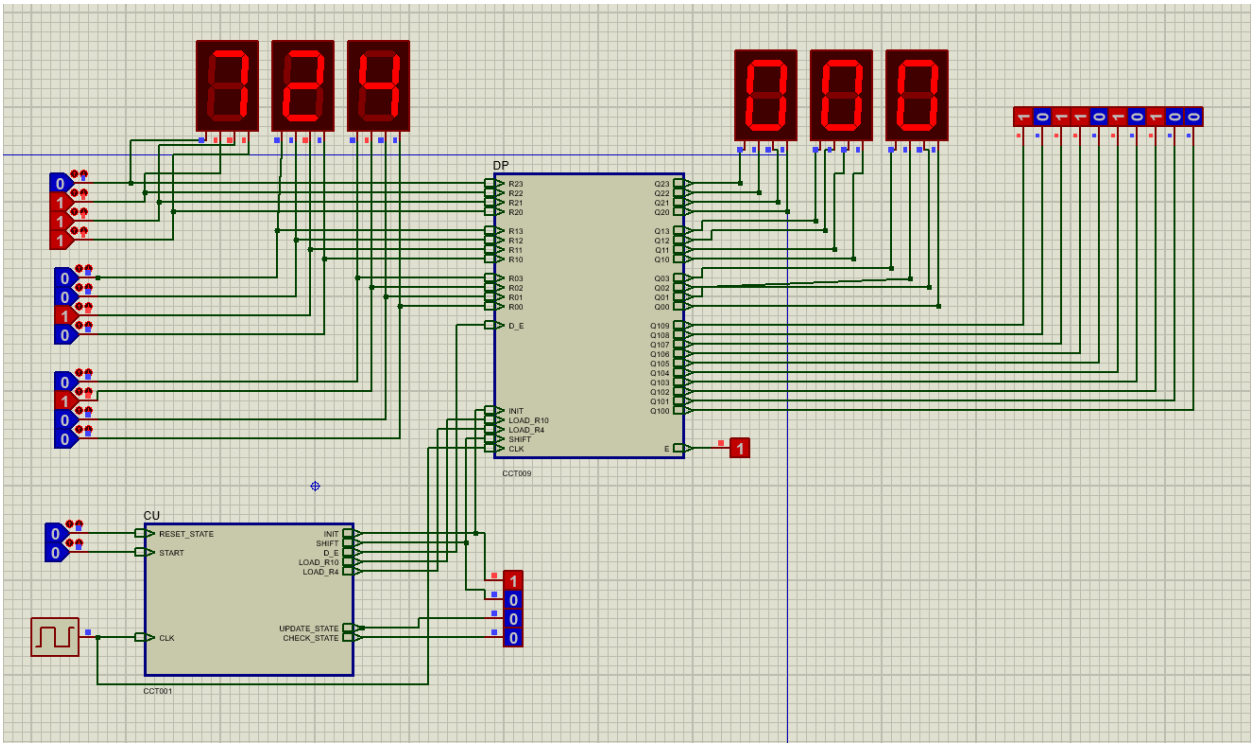
۲ نکته مهم:

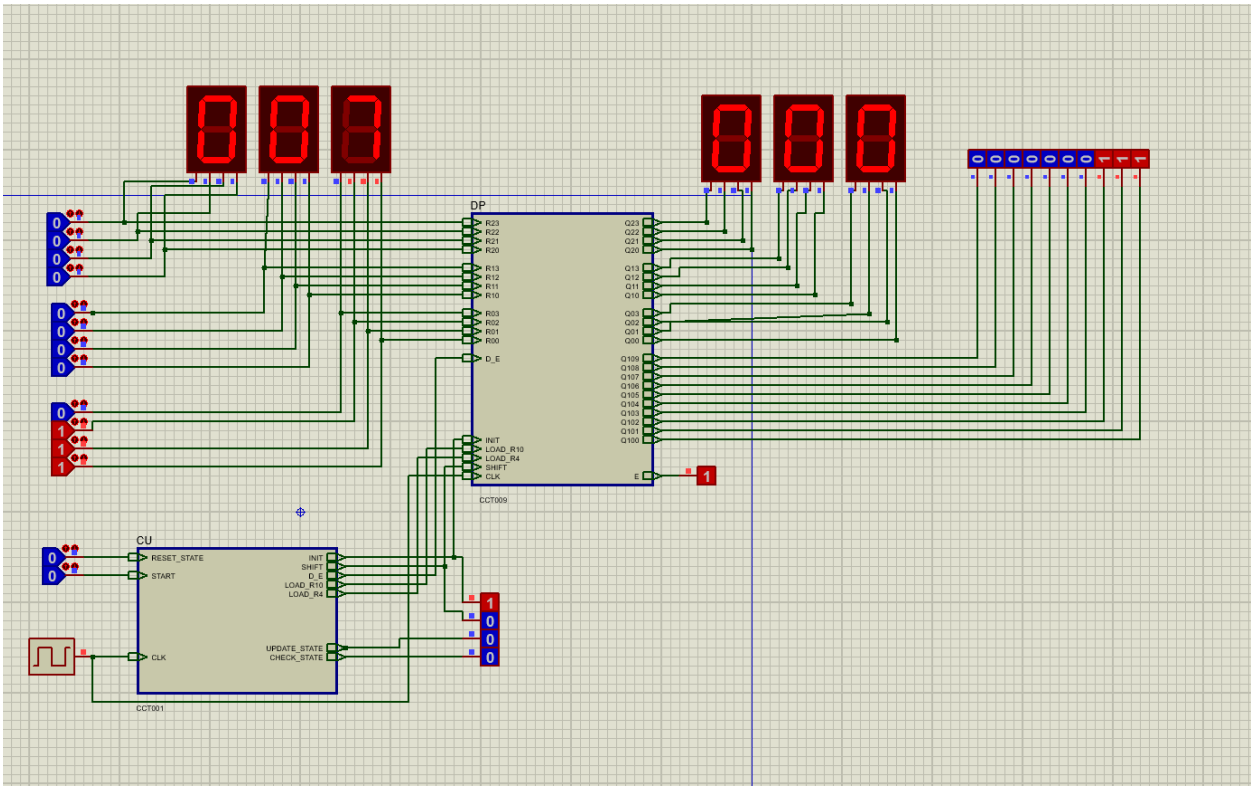
۱- موقع استارت کردن در هیچ استیتی نیستیم و باید یک بار RESET_STATE را یک کنیم تا به استیت INIT برویم سپس باید این سیگنال را صفر کنیم و دیگر به آن نیازی نداریم

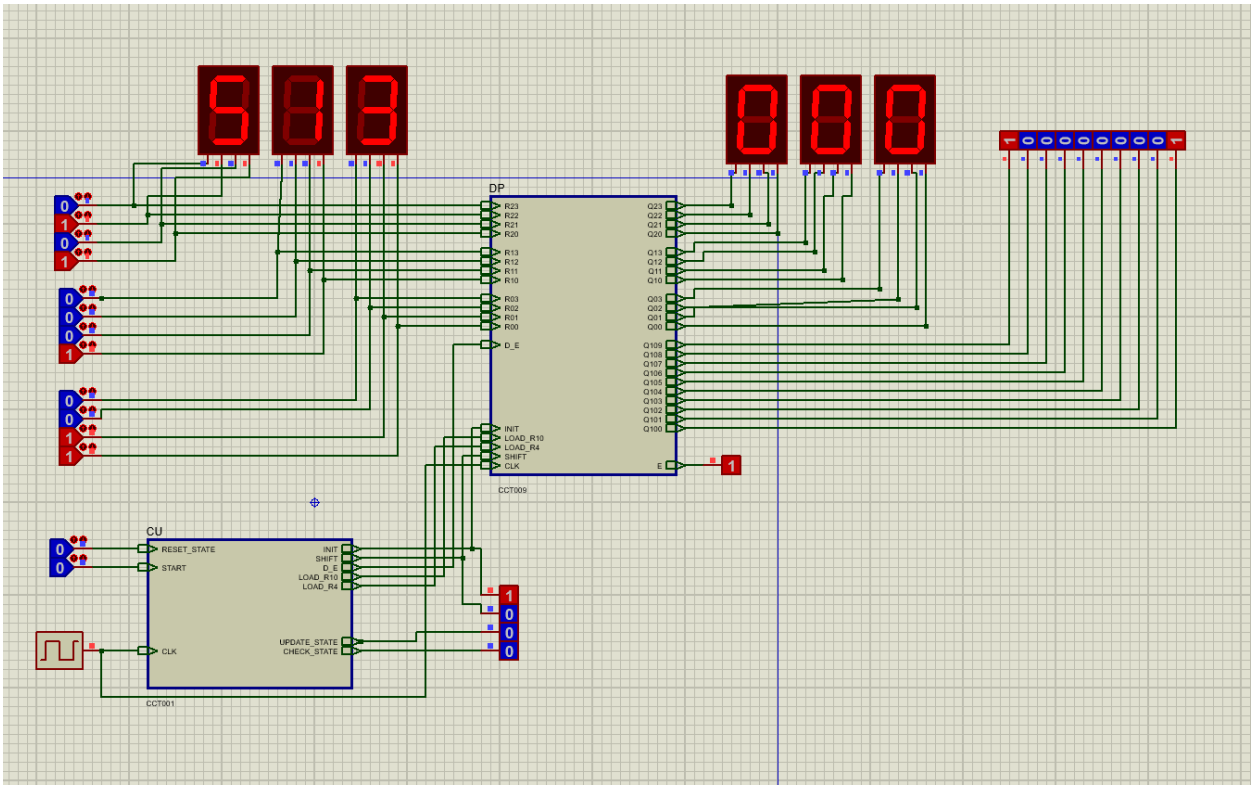
۲- سیگنال START اگر یک باشد و آن را صفر نکنیم بعد از محاسبه و نهایی شدن جواب مدار دوباره شروع به حساب کردن آن میکند و این لوپ تکرار میشود پس بعد از اینکه استارت را یک کردیم و کلاک اول زده شد باید استارت را صفر کنیم تا بعد از بازگشت به INIT دوباره شروع به محاسبه نکند.

چند نمونه از جواب مدار :



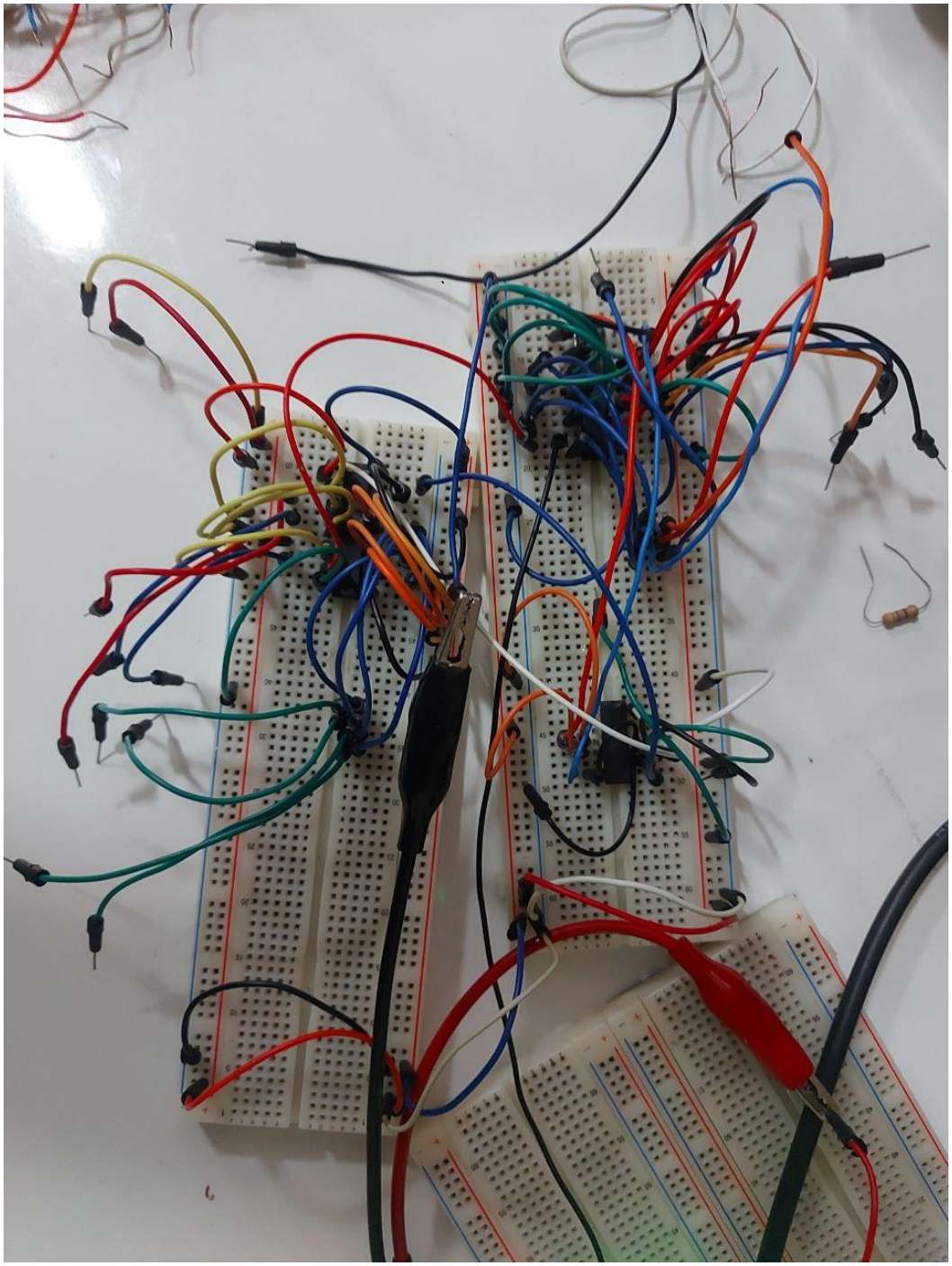



DP
CCT009
CU
RESET_STATE
START
CLK
INIT
SHIFT
D_E
LOAD_R10
LOAD_R4
UPDATE_STATE
CHECK_STATE
CCT001


DP
CCT009
CU
RESET_STATE
START
CLK
INIT
SHIFT
D_E
LOAD_R10
LOAD_R4
UPDATE_STATE
CHECK_STATE
CCT001

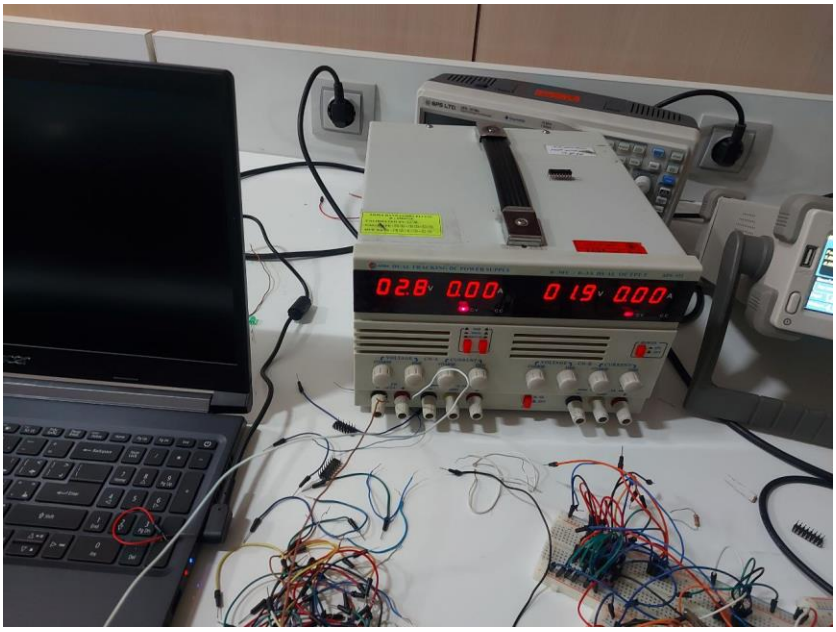
روز آزمایش :

برای پیاده سازی روی برد بورد حالتی که ۲ رقم bcd را به یک رجیستر ۸ بیتی تبدیل میکنیم پیاده کردیم به طوری که از ۲ رجیستر ۸ بیتی ۷۴۱۹۸ استفاده کردیم یکی برای خروجی و یکی برای دو رجیستری که باید ورودی را نگه دارند و این ۲ را پشت سر هم بستیم که میتوانید در عکس زیر مشاهده کنید:

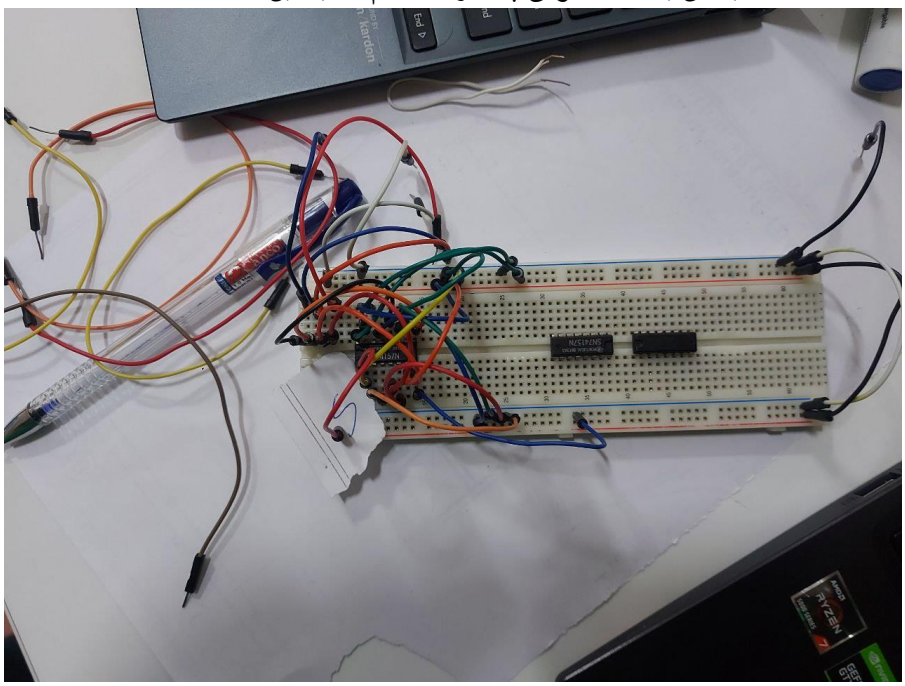


در ادامه برای وصل کردن کلاک از این دستگاه استفاده کردیم:

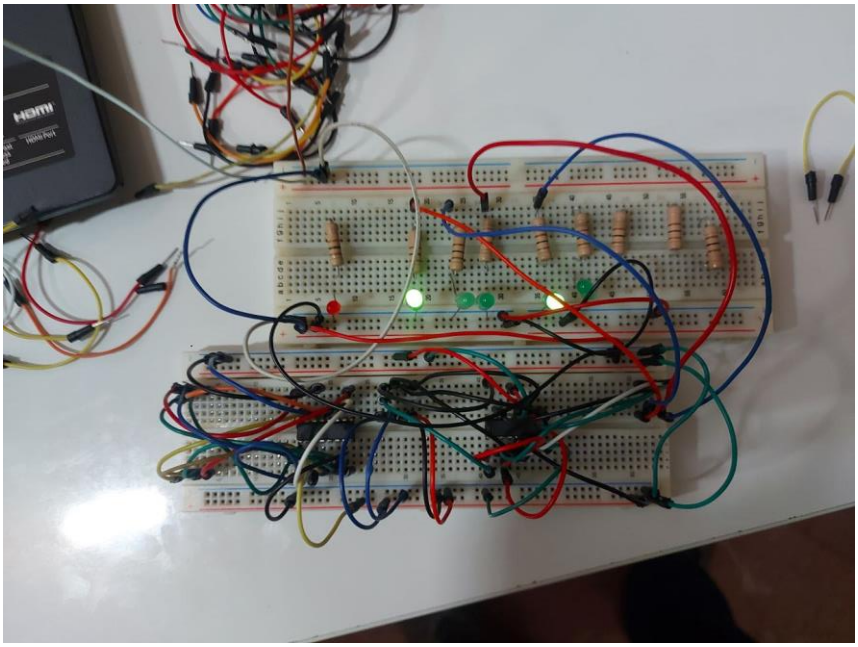
و برای تولید VCC ثابت نیز از این دستگاه استفاده کردیم :



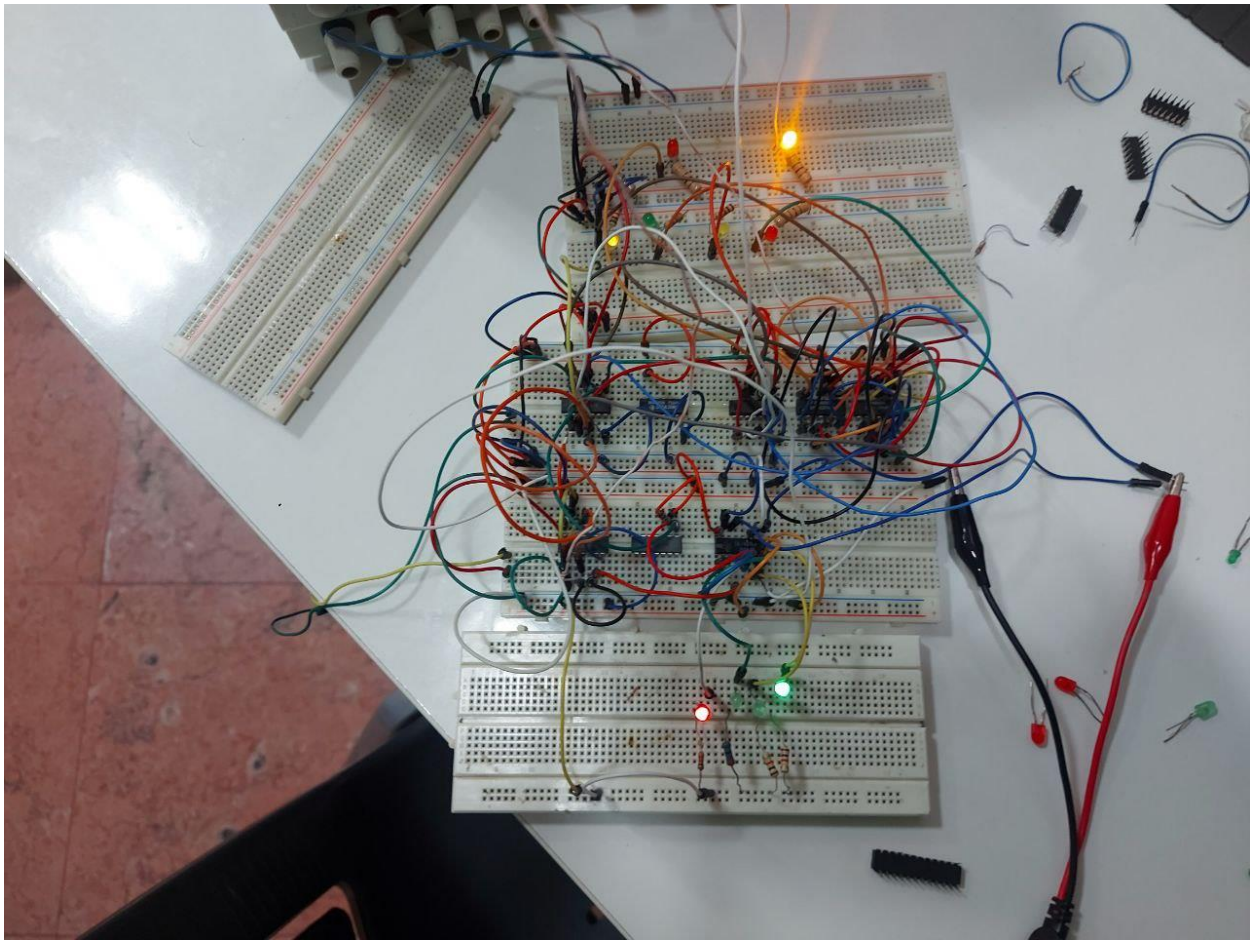
در بخش بعدی ۲ مولتی پلکسر ساختیم که به این شکل است :



در گام بعدی ۲ فول ادر قرار دادیم که عدد ورودی را با MM0M جمع میکند که M بیت پر ارزش عدد ورودی است این مدار خود به خود کار آپدیت کردن رجیستر را انجام میدهد و یکی از مولتی پلکسر های استفاده شده ما در فاز قبل را کمتر میکند مطابق شکل:

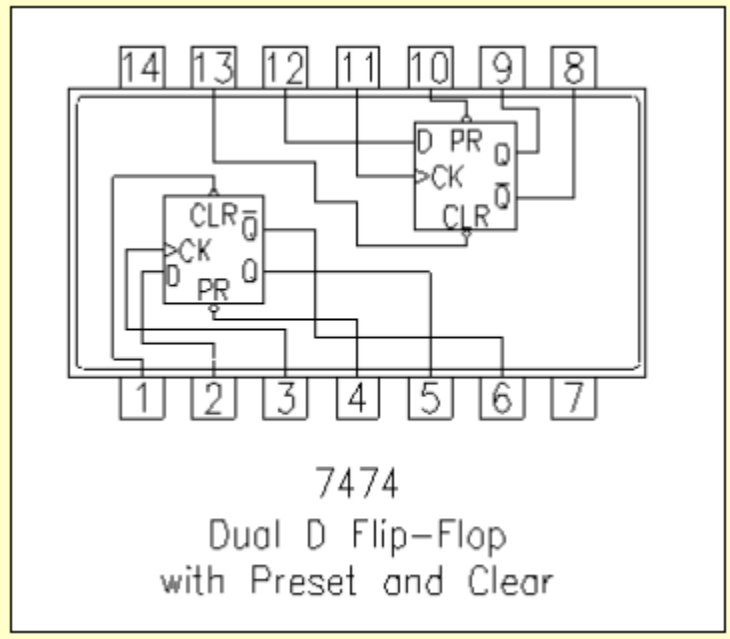


در آخر پس از اتمام دیتاپف باید ونتر یونیت را میساختیم که ندار خروجی را در زیر مشاهده میکنید(ال ای دی های پایینی مربوط به تست کردن بخش شمارش مدار بود که مدار درست از ۰ تا ۸ بشمرد و ال ای دی های بالایی مربوط به بیت های کنترلی خروجی مدار هستند):

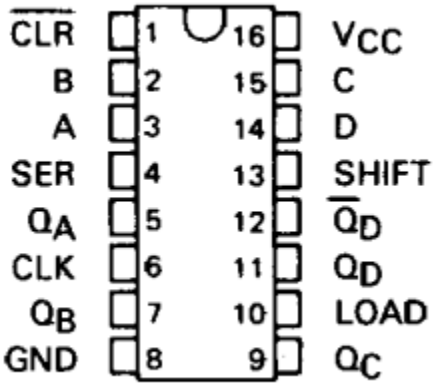


دیتا شیت قطعات استفاده شده :

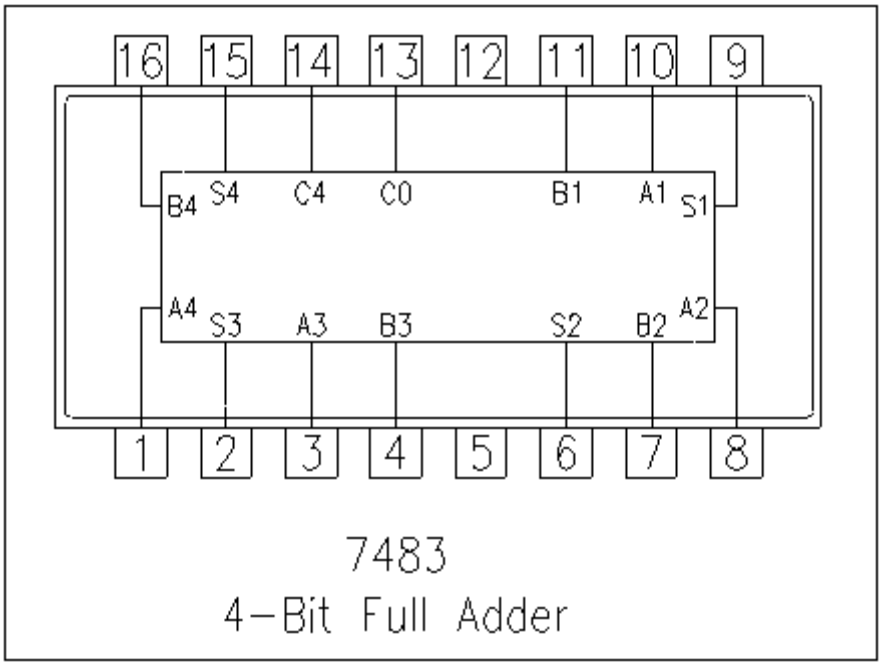
7474



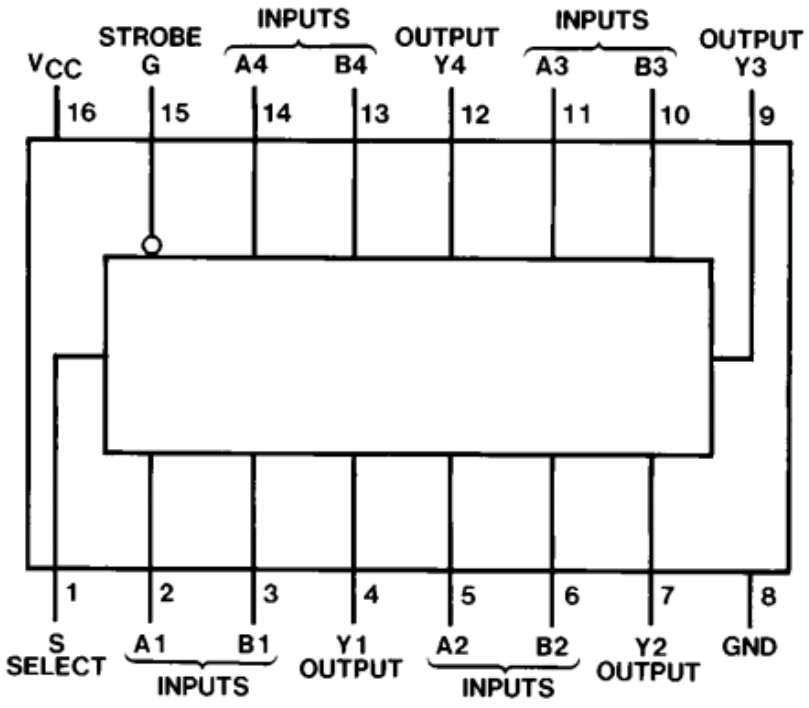
74179



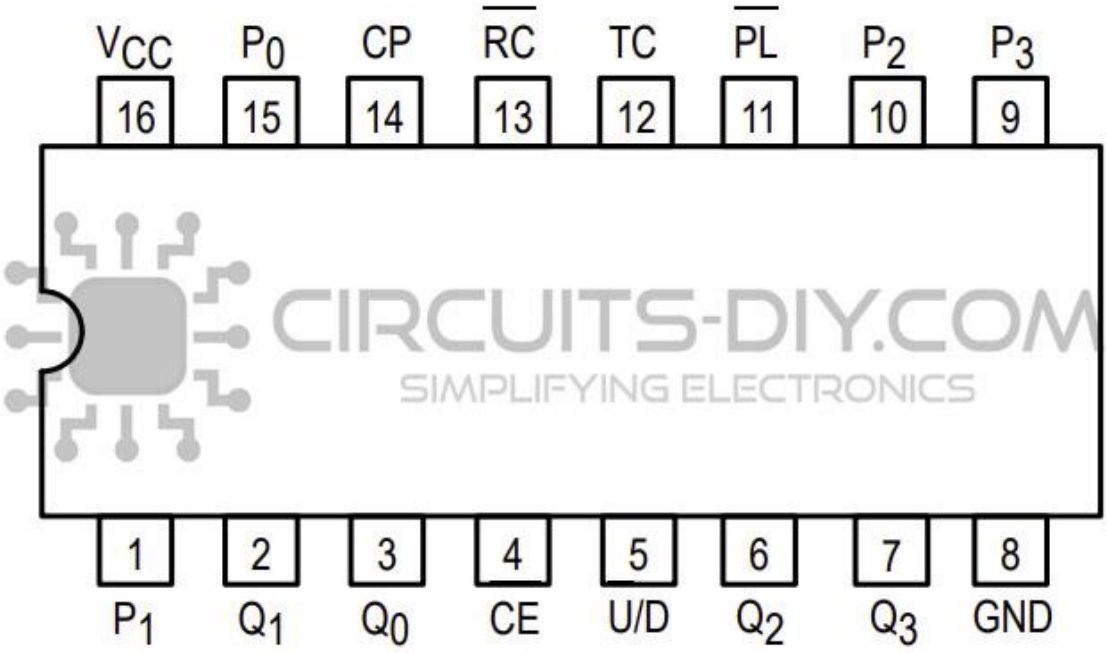
7483


16
15
14
13
12
11
10
9
B4
S4
C4
C0
B1
A1
S1
A4
S3
A3
B3
S2
B2
A2
1
2
3
4
5
6
7
8
7483
4–Bit Full Adder

74157


VCC
STROBE
G
INPUTS
A4
B4
OUTPUT
Y4
INPUTS
A3
B3
OUTPUT
Y3
16
15
14
13
12
11
10
9
1
2
3
4
5
6
7
8
S
SELECT
A1
B1
INPUTS
Y1
OUTPUT
A2
B2
INPUTS
Y2
OUTPUT
GND

74190



OR_NAND_NOR