دانشگاه صنعتی شریف
دانشکده مهندسی کامپیوتر

عنوان:

آزمایش دوم: ضرب کننده

اعضای گروه

امیرحسین علمدار_۴۰۰۱۰۵۱۴۴

پیام تائبی_۴۰۰۱۰۴۸۶۷

ماهان بیهقی_۴۰۰۱۰۴۸۳۴

نام درس

آزمایشگاه معماری کامپیوتر

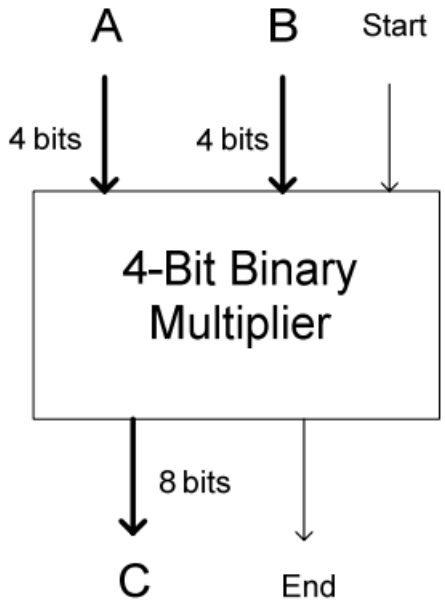
تابستان ۱۴۰۲

نام استاد درس

حمید سربازی آزاد

چکیده: در این آزمایش قصد طراحی و پیاده سازی یک ضرب کننده دودویی ۴ بیتی را داریم. مدار مورد نظر، باید مشخصات زیر را داشته باشد:

مضروب (ورودی) : A
مضروب فیه (ورودی) : B
حاصل ضرب (خروجی) : C
شروع ضرب (ورودی) : Start
پیان ضرب (خروجی) : End

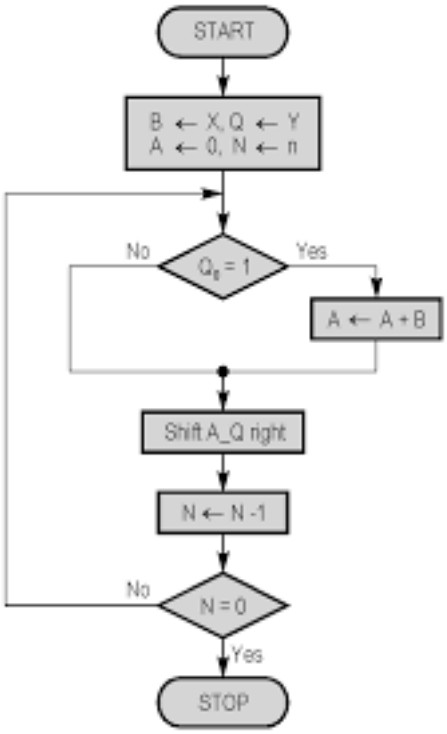


شکل ۱: مشخصات مدار ضرب کننده ممیز ثابت

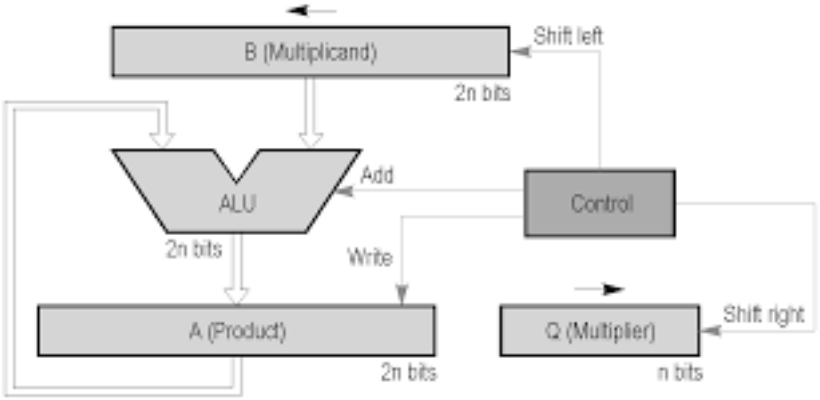
ابتدا مقادیر A و B را مشخص میکنیم، سپس باید سیگنال start را یک کنیم و منتظر بمانیم تا پس از چند پالس کلاک نتیجه با فعال شدن سیگنال end در C آماده شود.
همچنین دستیار آموزشی یک نکته مهم راجه به سیگنال start ذکر کرد که این سیگنال پس از یک شدن باید صفر شود و اگر برای کار کرد مدار نیاز به یک ماندن آن باشد، مدار طراحی درستی نداشته است. به همین دلیل در ادامه سیگنال start را به همین صورت طراحی میکنیم.

۱ الگوریتم ضرب

در این آزمایش برای ضرب دو عدد دودویی ۴ بیتی از الگوریتم shift and add بهره می بریم. همانگونه که در شکل ۲ و ۳ مشاهده می کنید، به علت اینکه حاصلضرب دو عدد ۴ بیتی، ۸ بیتی می شود، برای مضروب از شیفت رجیستر ۸ بیتی استفاده میکنیم، و هربار مضروب فیه را یک بیت به راست شیفت می دهیم، در صورت یک بودن lsb ، مقدار ۸ بیتی شیفت رجیستر مربوط به مضروب را با حاصل فعلی جمع میکنیم. اگر lsb صفر بود نیز، حاصل فعلی را با صفر جمع می کنیم.(در مدار ما این اتفاق می افتد) در نهایت نیز مقدار شیفت رجیستر ۸ بیتی را یک واحد به چپ شیفت می دهیم تا ارزش آن ها در هنگامی که با توجه به بیت های پرارزش تر مضروب فیه جمع می شوند، حفظ شود.



شکل ۲: فلوچار الگوریتم add and shift



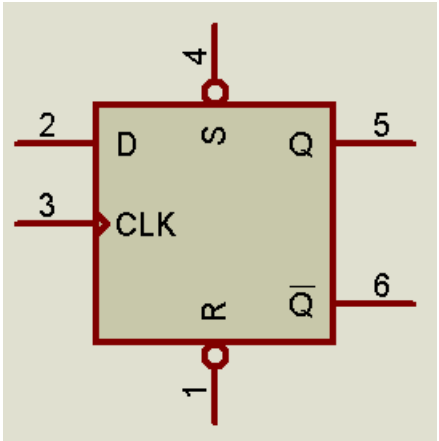
شکل ۳: چگونگی اجرای الگوریتم در مدار

۲ قطعات مورد نیاز

این مدار را با استفاده از قطعات زیر که می دانیم در آزمایشگاه موجودند طراحی میکنیم تا مجبور به تغییر طراحی نشویم. قطعات اصلی در ادامه آمده اند. برخی قطعات مانند ۷ display segment برای آسانی شبیه سازی اند و در پیاده سازی واقعی استفاده نشده اند. همچنین برخی قطعات مانند and ، nor ، or و not هم استفاده شدند ولی از آنجایی که در آزمایش گذشته کارکرد آن ها توضیح داده شد، دیگر به توضیح آن ها نمی پردازیم.

۷۴۷۴ ۱-۲

فلیپ فلاپ(DFF) تک بیتی عادی شامل set و reset آسنکرون.(شکل ۴)



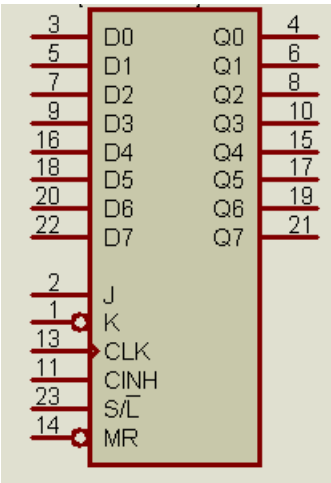
شکل ۴: ۷۴۷۴

۷۴۱۹۹ ۲-۲

$8\ bit\ universal\ shift\ register$ که شامل سیگنال های زیر است:

MR در واقع حکم enable را دارد و اگر آن را صفر بدهیم خروجی صفر خواهد بود.(در واقع فلیپ فلاپ ها را کلییر می کند)

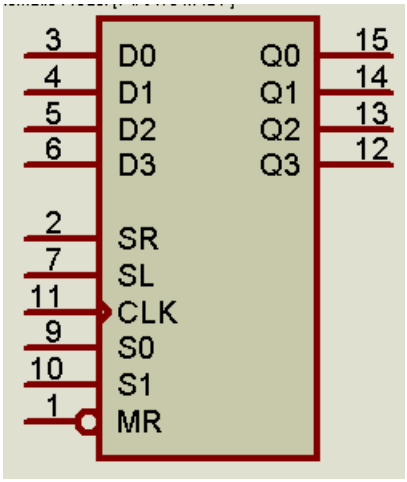
S/L نیز اگر یک باشد این قطعه مقادیر درونی را شیفت می دهد و در غیراینصورت مقادیر ورودی را لود می کند. همانگونه که در الگوریتم توضیح دادیم، حاصلضرب ۴ بیت در ۴ بیت، ۸ بیتی است به همین دلیل به این قطعه برای مضروب نیاز داریم. J و K که برای تنظیم مقدار ورودی از چپ به راست هنگام شیفت دادن استفاده می شود، مانند فلیپ فلاپ، آن ها را صفر میدهیم تا مقدار صفر بعد از شیفت دادن اضافه شود.(K اکتیو لو است)(شکل ۵)



شکل ۵: ۷۴۱۹۹

۷۴۱۹۴ ۳-۲

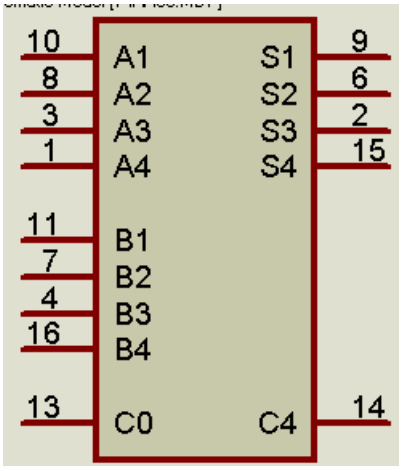
این قطعه یک شیفت رجیستر ۴ بیتی است که با قطعه بالا تفاوت هایی دارد. سیگنال MR همانند گذشته معادل enable است. همچنین کاربرد کلاک نیز باری شیفت یا لود است. اما سیگنال ها SR و SL به ترتیب برای شیفت ورودی به چپ یا راست هستند و در مدار ما ازین سیگنال ها استفاده نمی شود زیرا ما میخواهیم مقدار درونی رجیستر ها را شیفت بدهیم. برای کاربرد ما از سیگنال های S_0 و S_1 استفاده می شود. همانگونه که در دیتا شیت نوشته است، برای شیفت مقدار درونی به راست، S_1 را یک و S_0 را صفر می دهیم. اگر این دو سیگنال ۱ باشند، لود انجام می شود که جلوتر استفاده خواهیم کرد.(شکل ۶)



شکل ۶: ۷۴۱۹۴

۷۴۸۳ ۴-۲

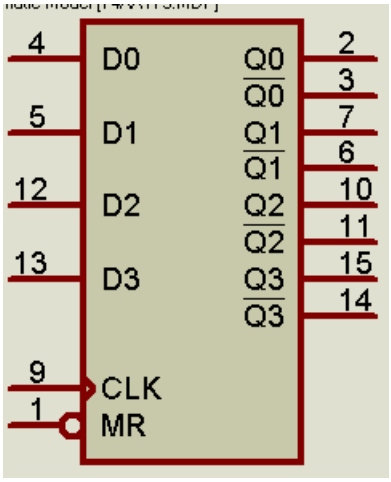
این قطعه یک جمع کننده ۴ بیتی است که یک کری ورودی و یک کری خروجی دارد.(شکل ۷)



شکل ۷: ۷۴۸۳

۷۴۱۷۵ ۵-۲

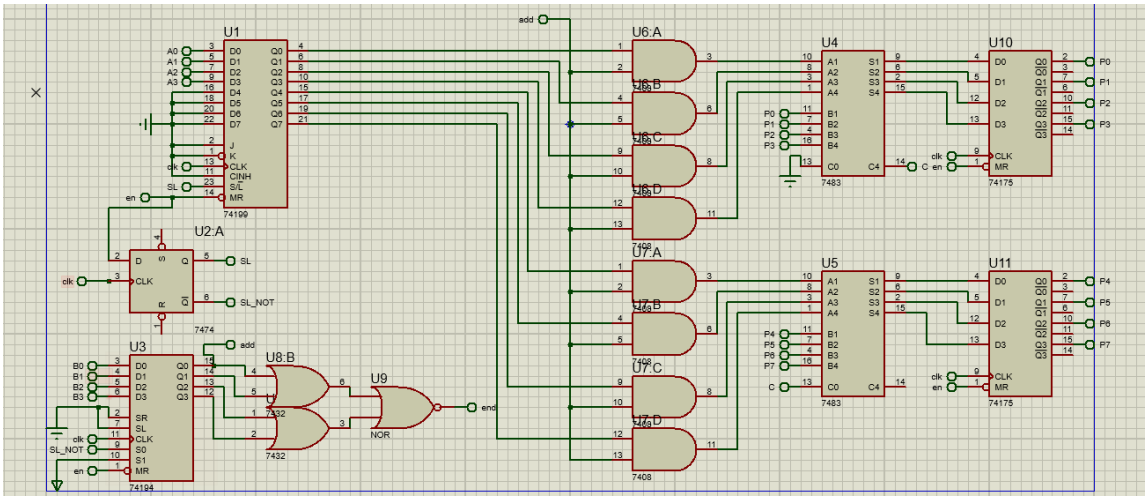
این قطعه یک رجیستر ۴ بیتی است که مقدار نهایی را درون خود نگاه خواهد داشت. این قطعه نیز یک MR دارد که با فعال شدن آن، رجیستر ها بصورت آسنکرون کلییر می شوند. سیگنال کلاک نیز برای مقدار دادن به رجیستر ها است.(شکل ۸)



شکل ۸: ۷۴۱۷۵

۳ نحوه پیاده سازی در Proteus

با توجه به الگوریتم add and shift ، ابتدا باید مضروب و مضروب فیه را در شیفت رجیستر ۸ و ۴ بیتی که در بالا ذکر شد لود کنیم. سپس در هر کلاک، اگر lsb مضروب فیه، یک بود، مقدار مضروب با حاصل جمع زده خواهد شد و در غیر اینصورت مقدار صفر با آن جمع می شود، پس از ۸ اند استفاده می کنیم تا خروجی های شیفت رجیستر با بیت کوچک مضروب فیه اند شوند. و این سیگنال ها وارد یک ادر می شوند و با حاصل کنونی رجیستر های ۴ بیتی جمع خواهند شد. با زده شدن کلاک، هم زمان جمع رخ می دهد و هم چنین مضروب یک بیت به چپ و مضروت فیه یک بیت به راست شیفت داده می شود. در ابتدای کار سیگنال S/L باید صفر باشد و از کلاک بعدی یک شود تا اول لود و سپس عملیات شیف انجام شود. برای این امر یک فلیپ فلاپ قرار می دهیم تا که در کلاک اول صفر است و اگر قطعات enable بودند، در کلاک بعدی خروجی آن یک می شود. از خروجی نات این فلیپ فلاپ نیز برای ورودی S_0 مضروب فیه استفاده می کنیم تا ابتدا مقدار در آن لود شود و از کلاک بعد که این سیگنال ۰ است، شیفت به راست انجام شود. برای سیگنال end نیز، هرگاه تمام بیت های مضروب فیه صفر شد، یعنی نتیجه محاسبه شده است. پس کافیست برای بدست آوردن آن،، بیت های مضروب فیه را nor کنیم. (شکل ۹) (همچنین چگونگی سیگنال ها ورودی قطعات شیفت رجیستر نیز قبلا توضیح داده شد.)



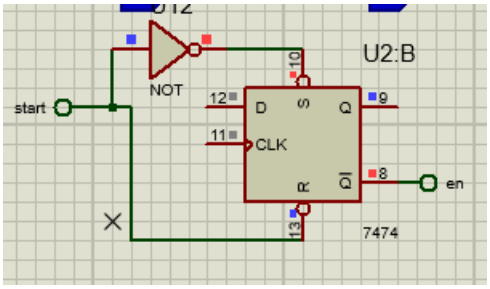
شکل ۹: شماتیک مدار

۳-۱ نکات پیاده سازی

برای ورودی MR قطعات بالا، یک سیگنال جدا گانه می سازیم، به این صورت که اگر start یک شود، این سیگنال صفر می شود و تمام مقادیر مدار صفر خواهتد شد. همچنین فلیپ فلاپ در نظر گرفته شده برای S/L با خوردن کلاک صفر خواهد شد. حال تا وقتی start یک باشد، تمام مقادیر صفر خواهند ماند و تغییری در مدار حاصل نمی شود. به محض صفر شدن ،start این سیگنال صفر یک می ماند و از کلاک بعدی مقادیر لود و عملیات از سر گرفته خواهند شد. بدین صورت برای شروع کار مدار نیاز به یک شدن و سپس صفر شدن استارت است و وقتی نتیجه آماده شود و end روشن شود، start خاموش خواهد بود. بدین ترتیب مدار دقیقا طوری که خواسته شده عمل خواهد کرد.(شکل ۱۰)

نکته مهم این است که سیگنال start حداقل به اندازه یک لبه بالا رونده کلاک یک بماند تا تغییرات ایجاد شوند.

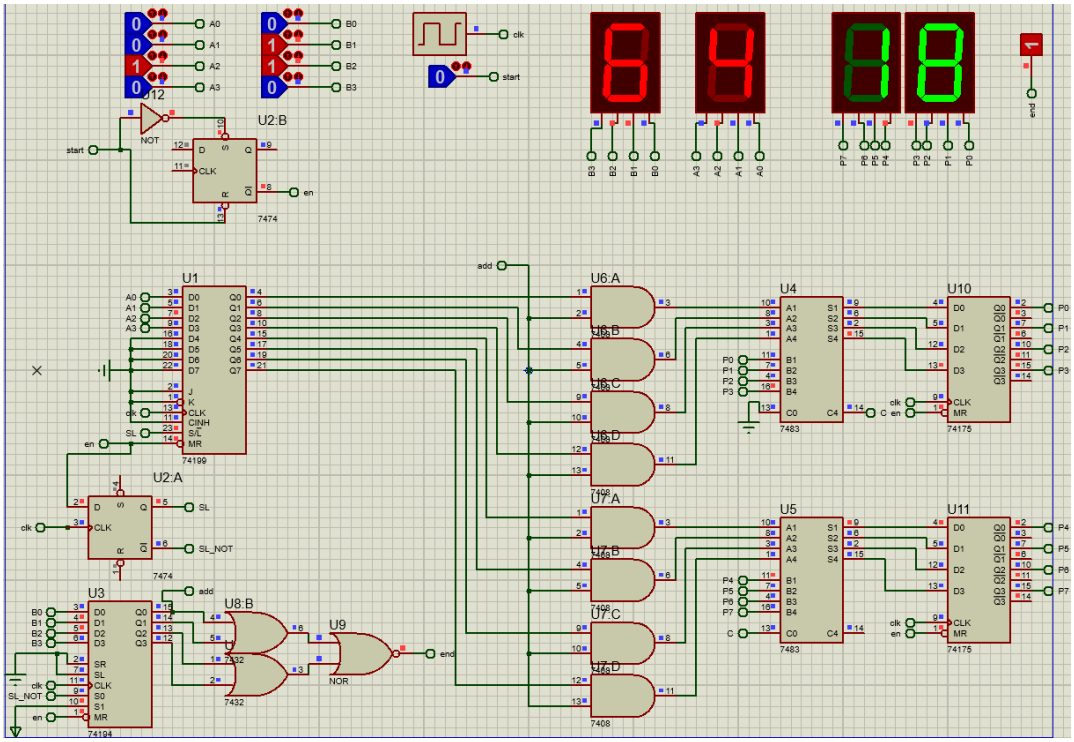
همچنین همانطور که در شکل ۹ دیدید، از دو جمع کننده ۴ بیتی برای جمع دو عدد ۸ بیتی استفاده کردیم.



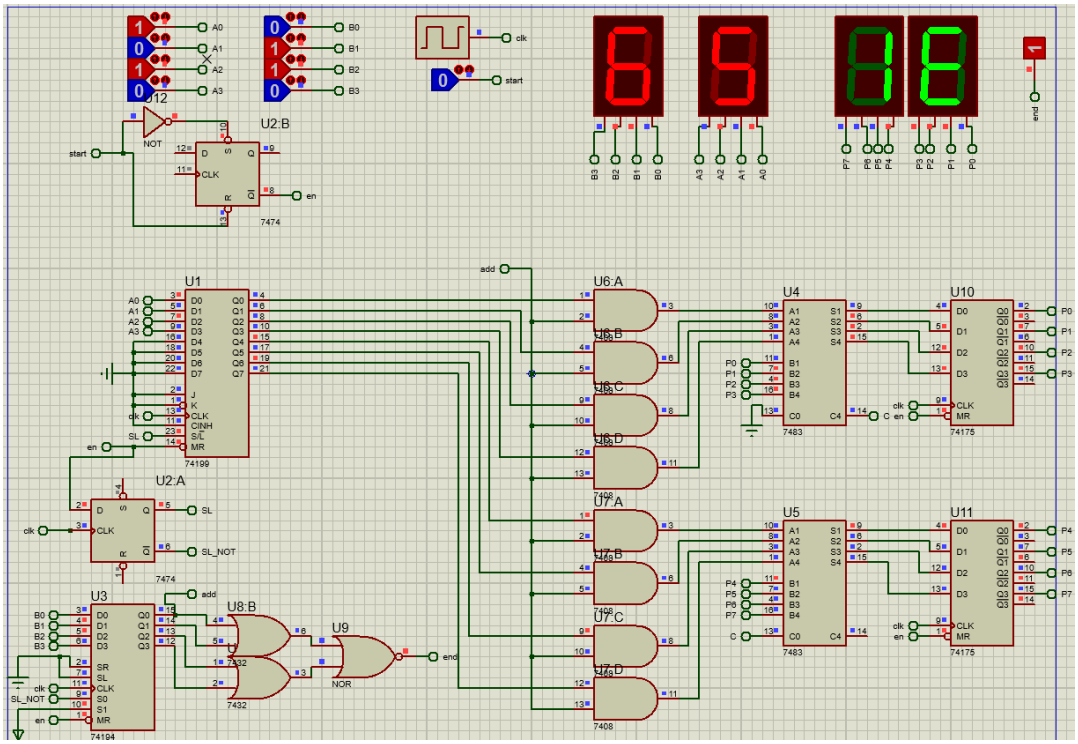
شکل ۱۰: پیاده سازی سیگنال enable

۳-۲ نتیجه شبیه سازی

برای نشان دادن کارکرد درست مدار، ابتدا تست شکل ۱۱ را انجام می دهیم و سپس ورودی ها را تغییر می دهیم و با روشن و خاموش کردن start ، منتظر می مانیم تا نتیجه جدید آماده شود.(شکل ۱۲)



شکل ۱۱: تست ۱

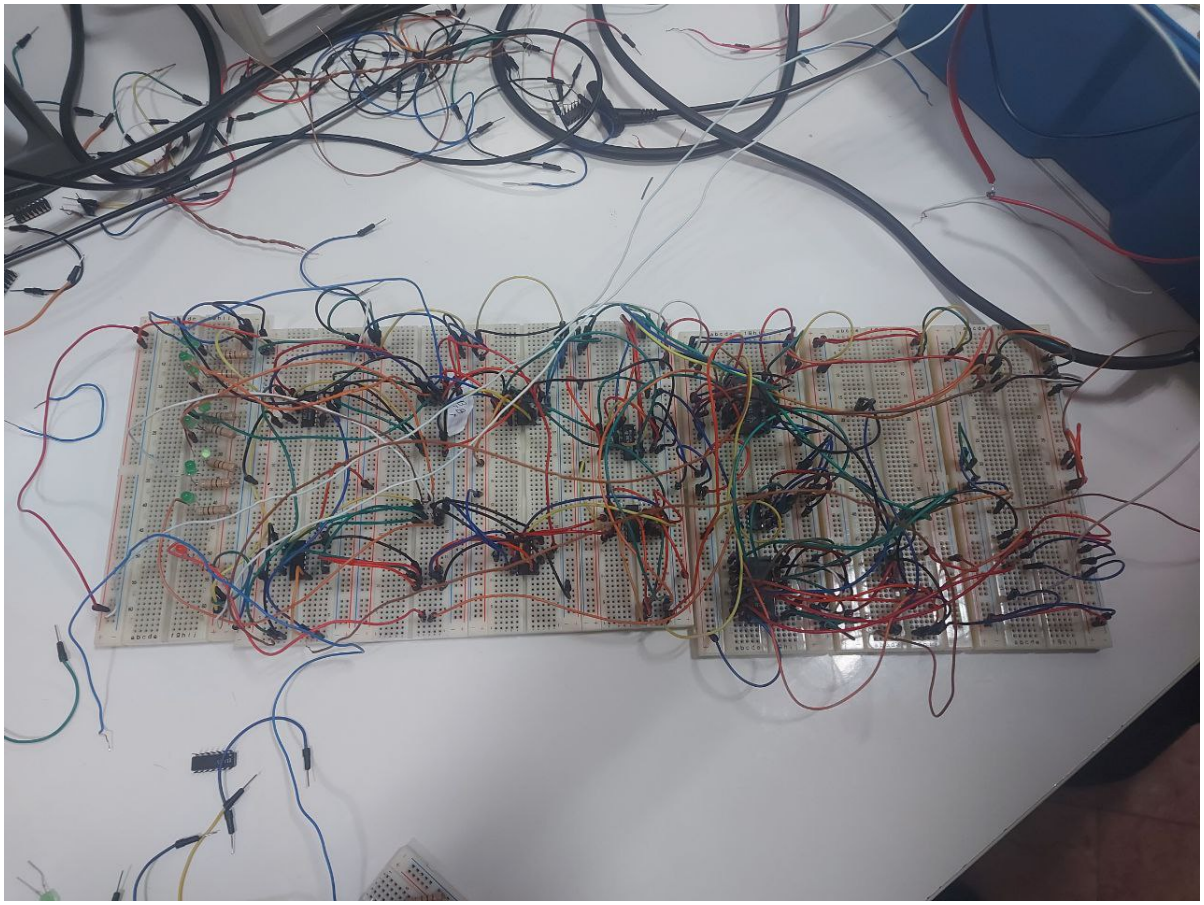


شکل ۱۲: تست ۲، نحوه ورودی های جدید با خاموش و روشن کرد سیگنال start و بدون نیاز به شبیه سازی دوباره

۴ نحوه پیاده سازی در آزمایشگاه

تمام قطعات استفاده شده در Proteus با توجه به وجود آنها در آزمایشگاه انتخاب شده بود پس صرفا قطعات مورد نظر را برداشتیم و شروع به بستن مدار کردیم. در اینجا دو نکته وجود دارد، یک اینکه در آزمایشگاه برخلاف Proteus ، سیگنال end را با nor کردن بیت های مضروب فیه ساختیم در صورتی که در شبیه سازی ابتدا آن ها را دو به دو or و سپس حاصل را nor کردیم ولی در آزمایشگاه nor ۴ پایه موجود بود و کارمان آسان تر شد. نکته دوم اینکه باکس قطعه ۷۴۱۹۴ که همان شیفت رجیستر ۴ بیتی است، خالی بود و تصمیم گرفتیم از قطعه ۷۴۱۹۸ استفاده کنیم، این قطعه قابلیت شیفت به چپ و راست را داشت اما تنها ۳ قطعه از آن باقی مانده بود که بعد تست متوجه معیوب بودنشان شدیم. در نهایت تصمیم گرفتیم برای شیفت رجیستر ۴ بیتی از قطعه ۷۴۱۹۹ استفاده کنیم ولی همانطور که میدانید این قطعه فقط قابلیت شیفت به چپ دارد، پس تصمیم گرفتیم مضروب فیه را بصورت وارون به آن ورودی دهیم، به اینصورت که بیت کم ارزش مضروب فیه به پر ارزش ترین پایه قطعه داده می شود و ۴ بیت کم ارزش قطعه نیز صفر داده می شود، با این ترفند طوری به ۷۴۱۹۹ ورودی دادیم تا عملا کار شیفت به راست را انجام دهد. سیگنال های کنترلی این قطعه نیز مانند ۷۴۱۹۹ مربوط به مضروب داده شدند . همچنین یک برد بورد را به نمایش خروجی

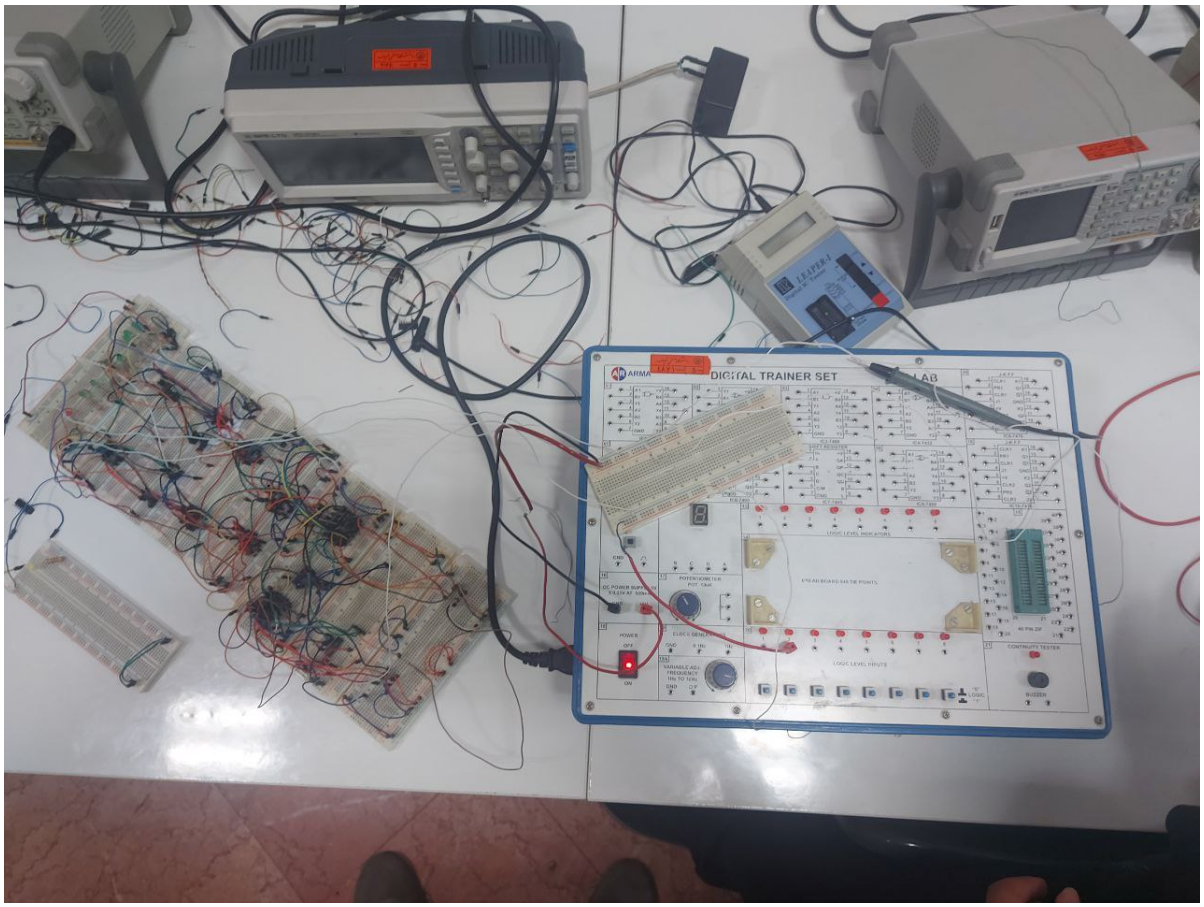
ها اختصاص دادیم و روی آن فقط ۹ LED و تعدادی مقاومت بسته شده بود. (شکل ۱۳)



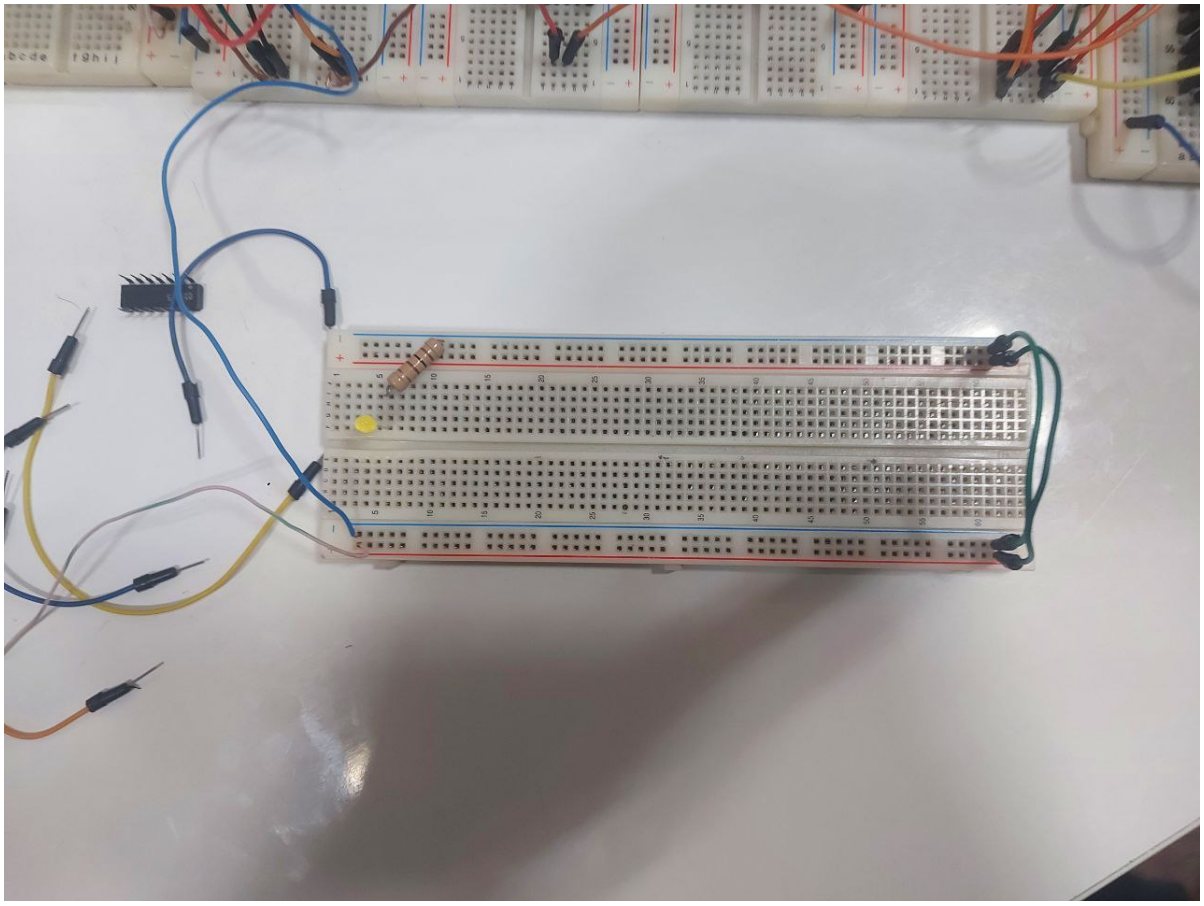
شکل ۱۳: مدار نهایی

۱-۴ مشکلات پیاده سازی

یکی از باگ هایی که خوردیم، لود نشدن شیفت رجیستر ها بود که در زمان کلاس با توجه به تمیز بودن مدار بازهم نتوانستیم مشکل آن را پیدا کنیم. از چالش های دیگر می توان به نبود باتن و سختی های ساخت کلاک اشاره کرد، همچنین برای سیگنال start نیز نیاز به یک دکمه داشتیم تا ابتدا یک و سپس صفر شود، به همین دلیل برای ورودی دادن این سیگنال هم مشکلات زیادی ایجاد شده که در عکس ها مشاهده خواهید کرد.(شکل ۱۴) حتی در انتهای کار تصمیم به کندن فیلپ فلاپ مربوط به enable کردیم تا آن را مستقیم از start بگیریم و با یک ماندن start ، نتیجه محاسبه شود و نیاز به تغییر آن نباشد تا بتوانیم راحت تر مدار را تست کنیم.اما باز هم علت باگ قبلی که لود نشدن شیفت رجیستر ها بود یافت نشد.(شکل ۱۵)



شکل ۱۴: دستگاه استفاده شده برای ایجاد کلاک و مشکلات به اشتراک گذاری آن با دیگر تیم ها



شکل ۱۵: بورد استفاده شده برای تست سیگنال های درونی مداری و رفع اشکال ها (با یک بودن سیگنال مورد نظر ، دیود روشن خواهد شد)