

بسمه تعالی



گزارش کار پنجم آزمایشگاه معماری

واحد محاسبه با امکان انتخاب ثبات مبدأ و مقصد

استاد:

دکتر حمید سربازی آزاد

نویسندها:

امیررضا آذری 99101087

بزرگمهر ضیا 99100422

غزل طحان 99106374

دانشگاه صنعتی شریف

تهران 1402

فهرست

3	هدف
3	بخش اول: طراحی مازول های مدار
8	بخش دوم: تست مدار
14	کار در کلاس
20	نتیجه گیری
20	منابع و مراجع:

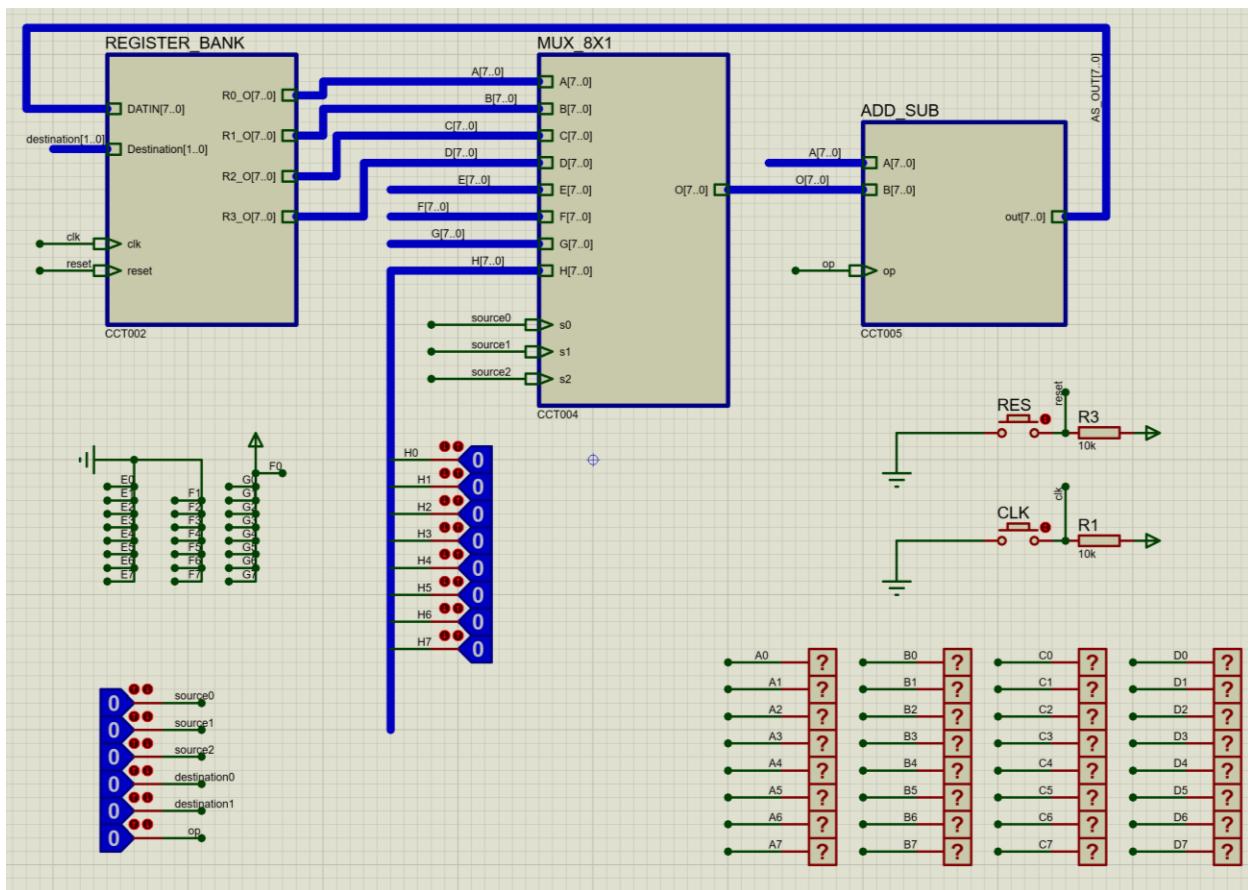
هدف

می خواهیم در این آزمایش، واحد محاسبات و مجموعه ثباتهای عمومی ماشین را طبق گزارش کار آزمایش، طراحی و پیاده سازی کنیم. این معماری امکان انجام جمع و تفریق با امکان انتخاب ثباتهای مبداء و ثبات نگهدارنده نتیجه (مقصد) را دارد. چهار ثبات عمومی R0, R1, R2, R3 هشت بیتی هستند. یکی از عملوندهای ALU به صورت ثابت محتوای ثبات R0 دیگری میتواند محتوای یکی از ثباتهای R0 تا R3 را مقادیر ثابت 0 و 1 و -1 باشد. حاصل تولید شده توسط (ALU جمع / تفریق) به یکی از ثباتهای مقصد R0 تا R3 منتقل میشود.

بخش اول: طراحی مازول های مدار

گزارش کار:

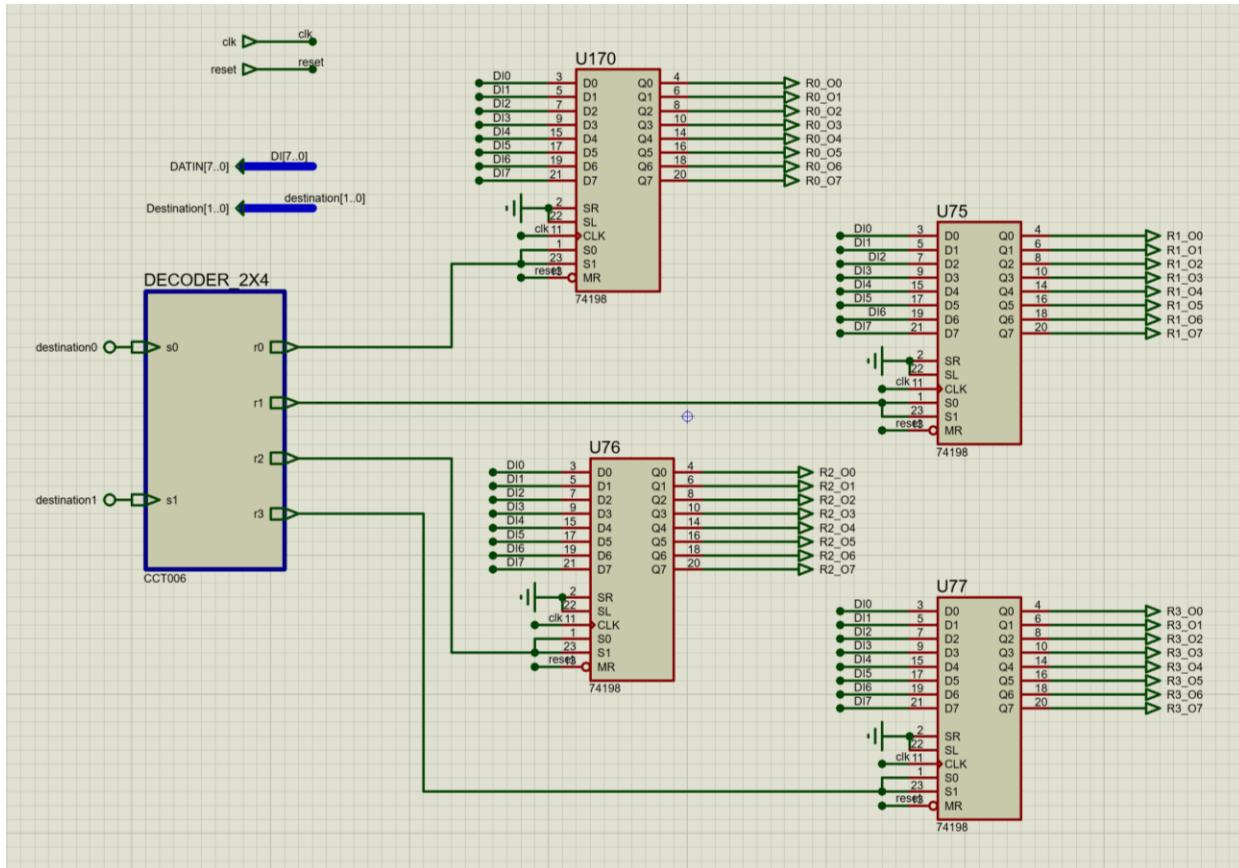
همان طور که در شکل 1 مشاهده میکنید، ساختار کلی مدار واحد محاسبه کننده نمایش داده شده است. به طور خلاصه مازول MUX_8X1 همان بخش بانک رجیستر است که داده های محاسبه شده را در رجیستر مربوطه ذخیره میکند. مازول ADD_SUB رجیستری که اپرند دوم جمع/تفریق است را انتخاب میکند و مازول push button نیز بسته به دستور ورودی، عملیات جمع یا تفریق را روی اپرند ها انجام داده و حاصل را خروجی میدهد. دو push button نیز برای ایجاد کلک مدار و ریست کردن آن طراحی شده است. در ادامه به توضیح مفصل هر یک از مازول ها میپردازیم.



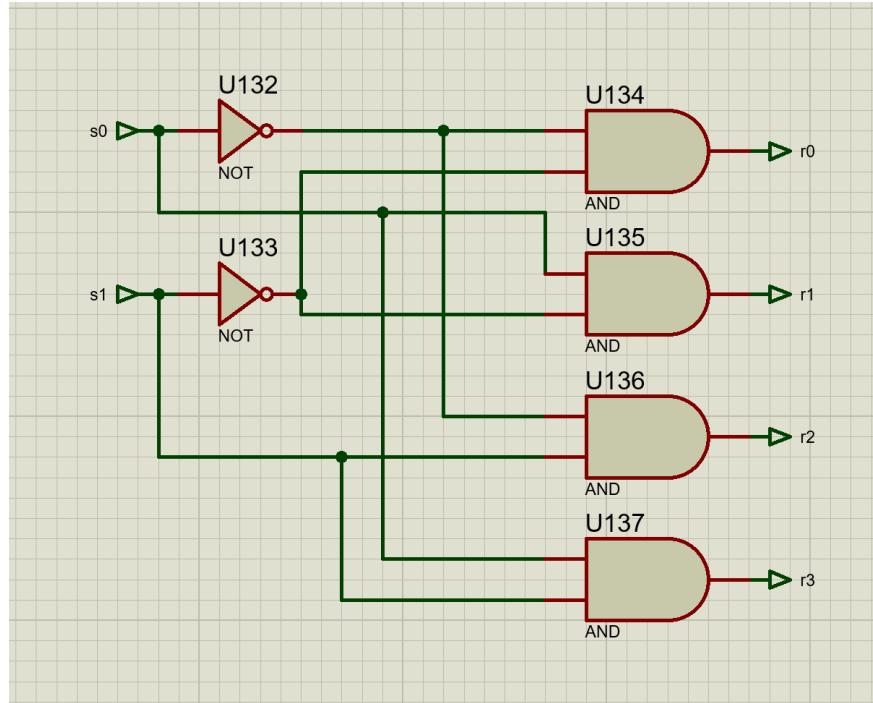
شکل 1. ساختار مدار کلی

ماژول : RESIGTER_BANK

همان طور که در شکل 2 مشاهده میکنید این ماژول از چهار ماژول شیفت رجیستر 74198 و یک دیکودر 2 به 4 تشکیل شده است. این ماژول دو بیت destination که نشان دهنده رجیستر مقصد است، هشت بیت DATAIN که خروجی واحد محاسبه کننده adder/subtractor است و سیگنال های کلک و ریست (برای کار با رجیستر ها) به عنوان ورودی دریافت میکند؛ ابتدا دو بیت destination را دیکود میکند تا مشخص شود کدام یک از چهار رجیستر باید حاصل محاسبه را در خود لود کند (و محتوای مابقی رجیستر ها دست نخورده باقی میماند) و درنهایت محتوای هر چهار رجیستر خروجی داده میشود. در شکل 3 نیز مدار دیکودر را مشاهده میکند که به علت سادگی، از توضیح آن صرف نظر میکنیم.



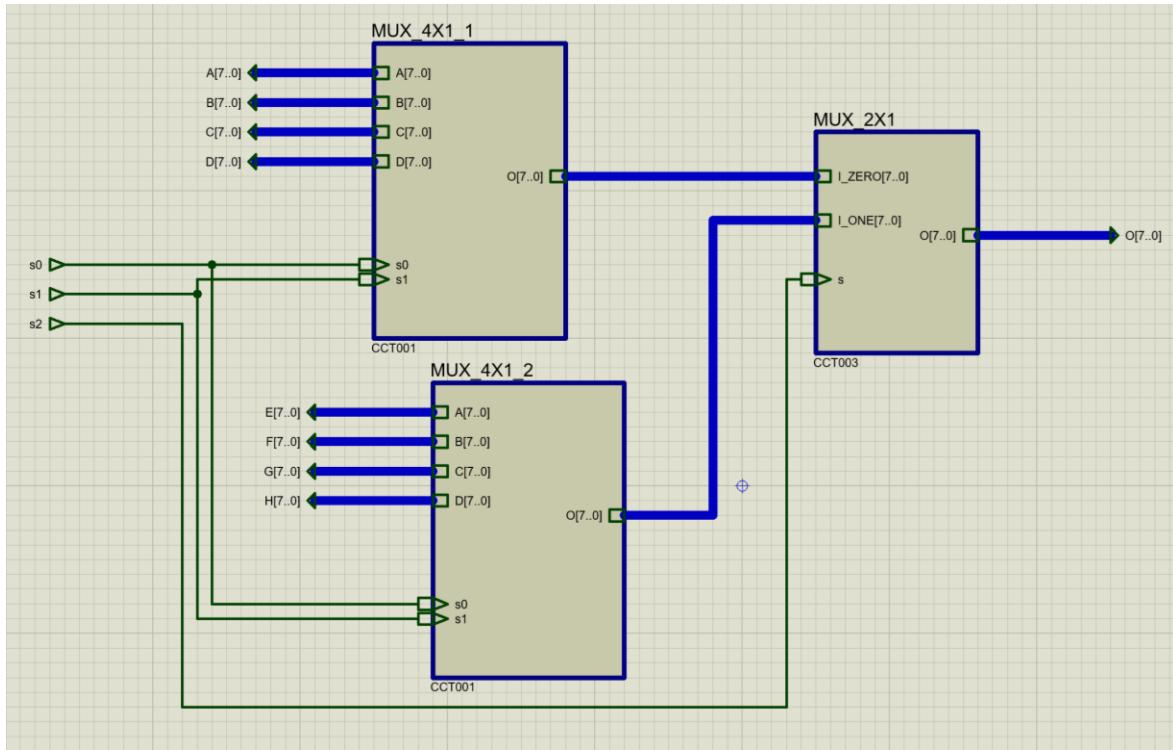
شکل 2. ساختار مدار رجیستر بانک



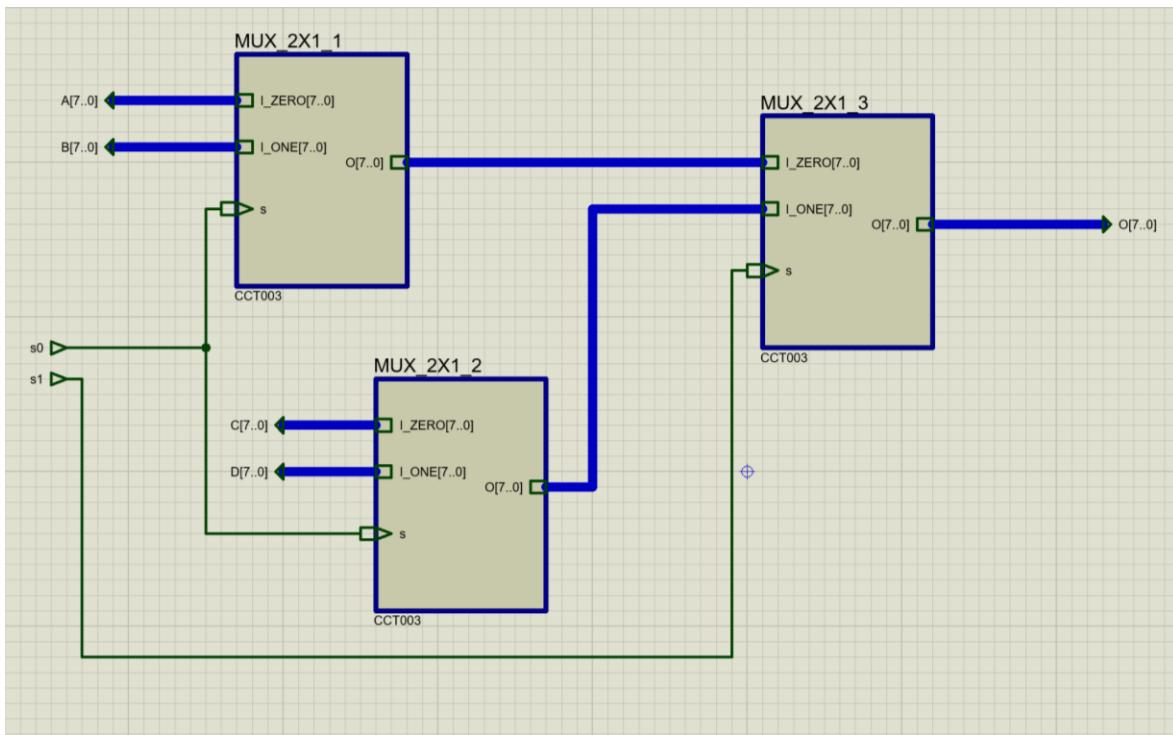
شکل ۳. ساختار مدار دیکودر ۲ به ۴

MUX_8X1 : مازول

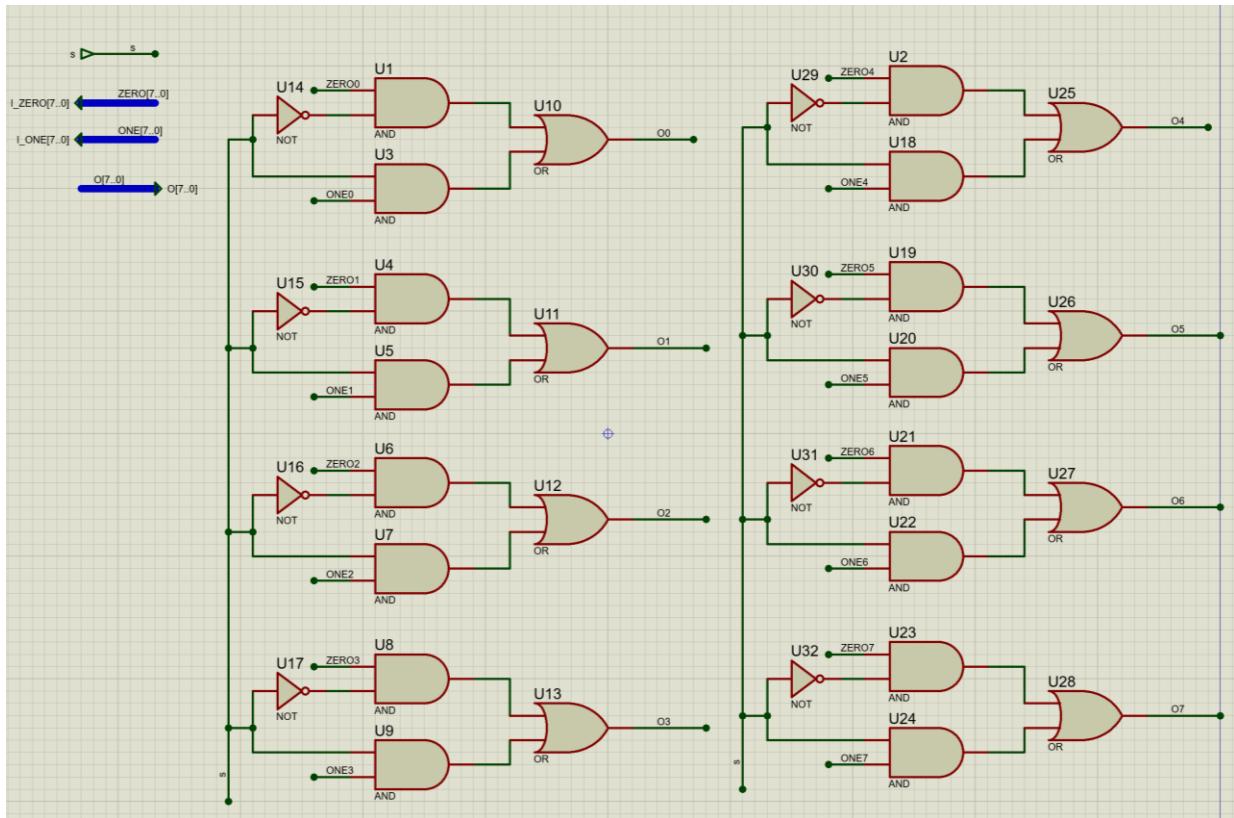
همان طور که در شکل ۴ مشاهده میکنید، این مالتی پلکسراز دو مالتی پلکسرا ۴ به ۱ و یک مالتی پلکسرا ۲ به ۱ برای به دست آوردن خروجی نهایی تشکیل شده است و مطابق شکل ۵، هر کدام از مالتی پلکسراهای ۴ به ۱ نیز با سه مالتی پلکسرا ۲ به ۱ طراحی شده اند (برای راحتی طراحی مدار آن و بدليل اينكه هر کدام از ورودی ها هشت بیتی هستند). این مازول محتوای رجیسترها و مقادیر مجاز immediate (صفر، یک و منفی یک) و یک ورودی هشت بیتی رزرو شده را ورودی میگیرد و با سه بیت نشان دهنده *source* که از شش بیت دستور گرفته شده، از میان آنها انتخاب میکند و آن را خروجی میدهد. ساختار مدار مالتی پلکسرا ۲ به ۱ با پهنهای ورودی هشت بیت نیز در شکل ۶ آمده که به دلیل سادگی منطق، از توضیح آن گذر میکنیم.



شکل ۴. ساختار مدار مالتی پلکسrer ۸ به ۱ با پهنهای ورودی هشت بیت



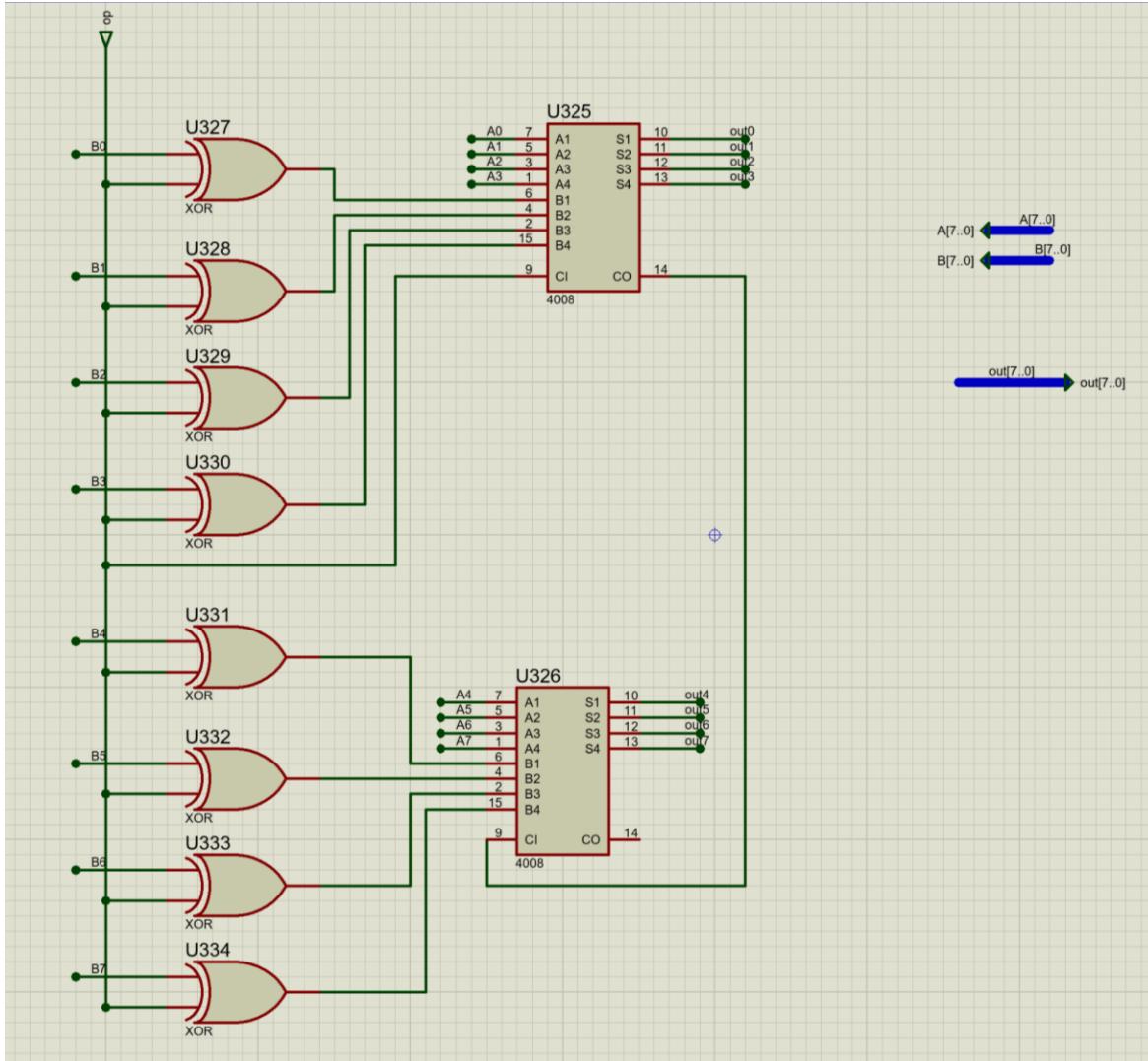
شکل ۵. ساختار مدار مالتی پلکسrer ۴ به ۱ با چهنهای ورودی هشت بیت



شکل 6. ساختار مدار مالتی پلکسر 2 به 1 هشت بیتی

ماژول ADD_SUB

همان طور که در شکل 7 پیداست، این ماژول بیت op که نشان دهنده ی جمع یا تفریق بودن عملیات است را از شش بیت دستور گرفته و این عملیات را روی اپرندی که از ماژول مالتی پلکسر آمده و R0 (اپرند ثابت این محاسبه کننده است) اعمال میکند. در این مدار از دو ماژول جمع کننده چهاربیتی 4008 استفاده شده و Cout اولی به Cin دومی، ورودی داده شده تا بتوازد جمع یا تفریق را روی هشت بیتی انجام دهد (بیت op را با اپرند دوم xor میکنیم و هم چنین به عنوان Cin به ماژول جمع کننده اول، ورودی میدهیم تا در صورتی که این بیت یک شد، مکمل دو اپرند دوم به ماژول های جمع کننده ورودی داده شود تا عملیات تفریق صورت بگیرد). و درنهایت حاصل محاسبه خروجی داده میشود تا به بانک رجیستر رود و در رجیستر مربوطه ذخیره شود.



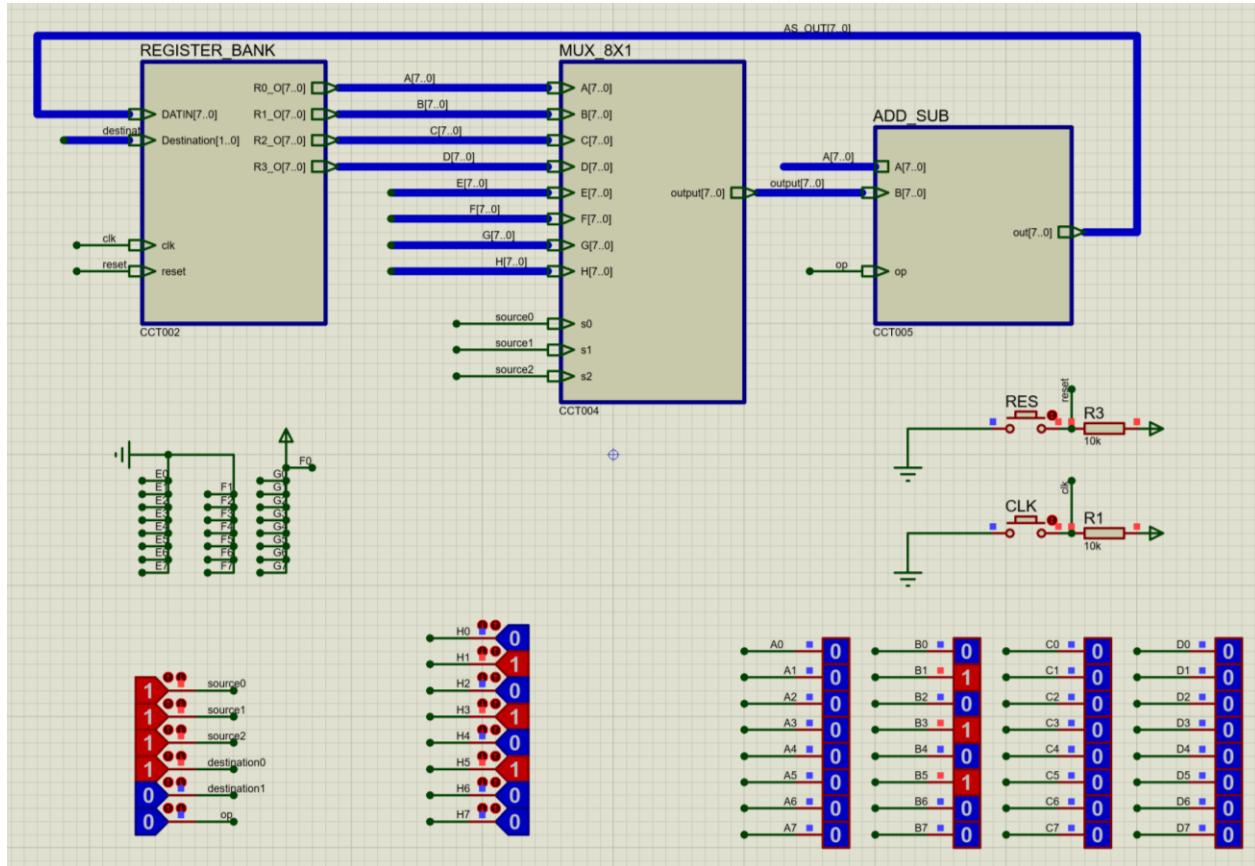
شکل 7. ساختار مازول جمع/تفریق کننده

بخش دوم: تست مدار

گزارش کار:

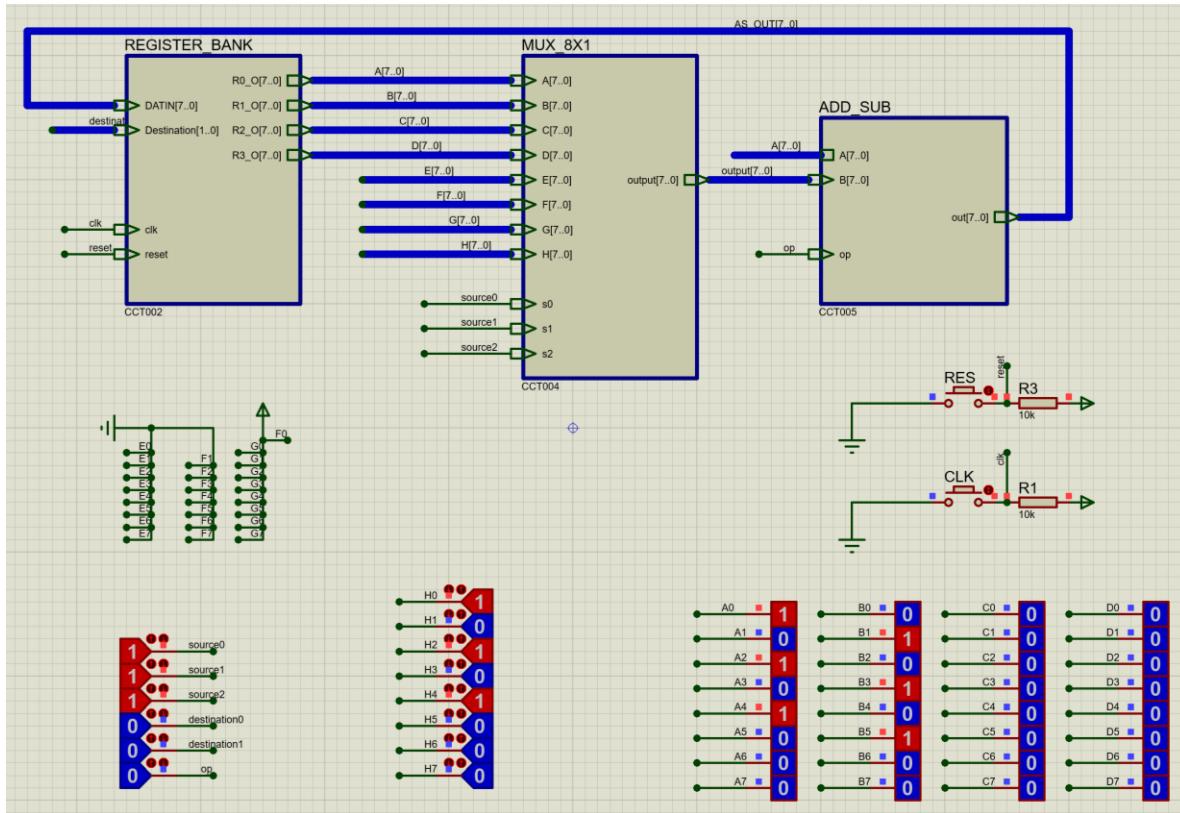
برای تست این مدار، ابتدا مقادیری را در رجیستر رزرو ورودی میدهیم و سپس این مقدار را با رجیستر R0 (با مقدار ابتدایی صفر) جمع و در آن رجیستر ها لود میکنیم تا هر کدام از رجیسترها مقادیر دلخواه ما را بگیرند و سپس عملیات مختلف جمع یا تفریق را روی آنها انجام میدهیم.

لود کردن عدد 42 در R1 :



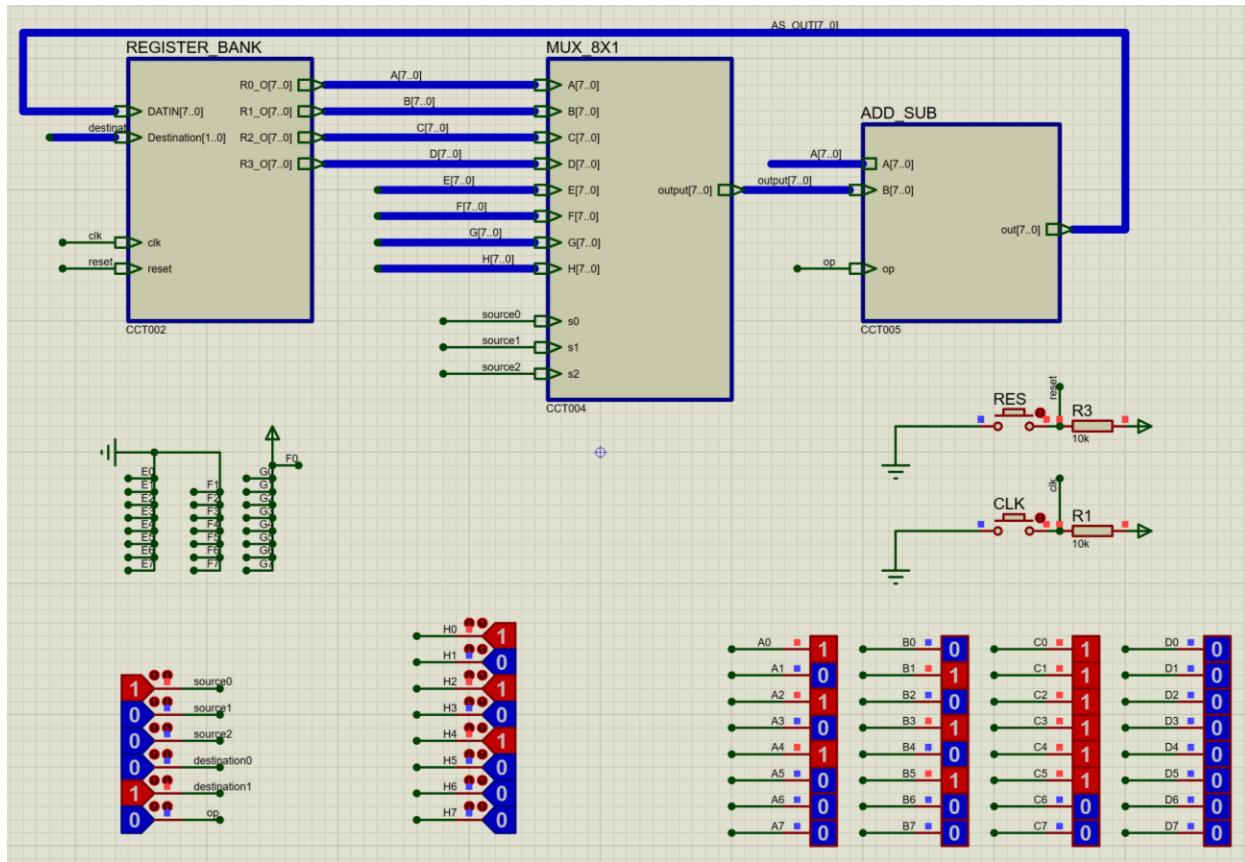
شکل 8 . < R2

و سپس لود کردن عدد 21 در R0 :

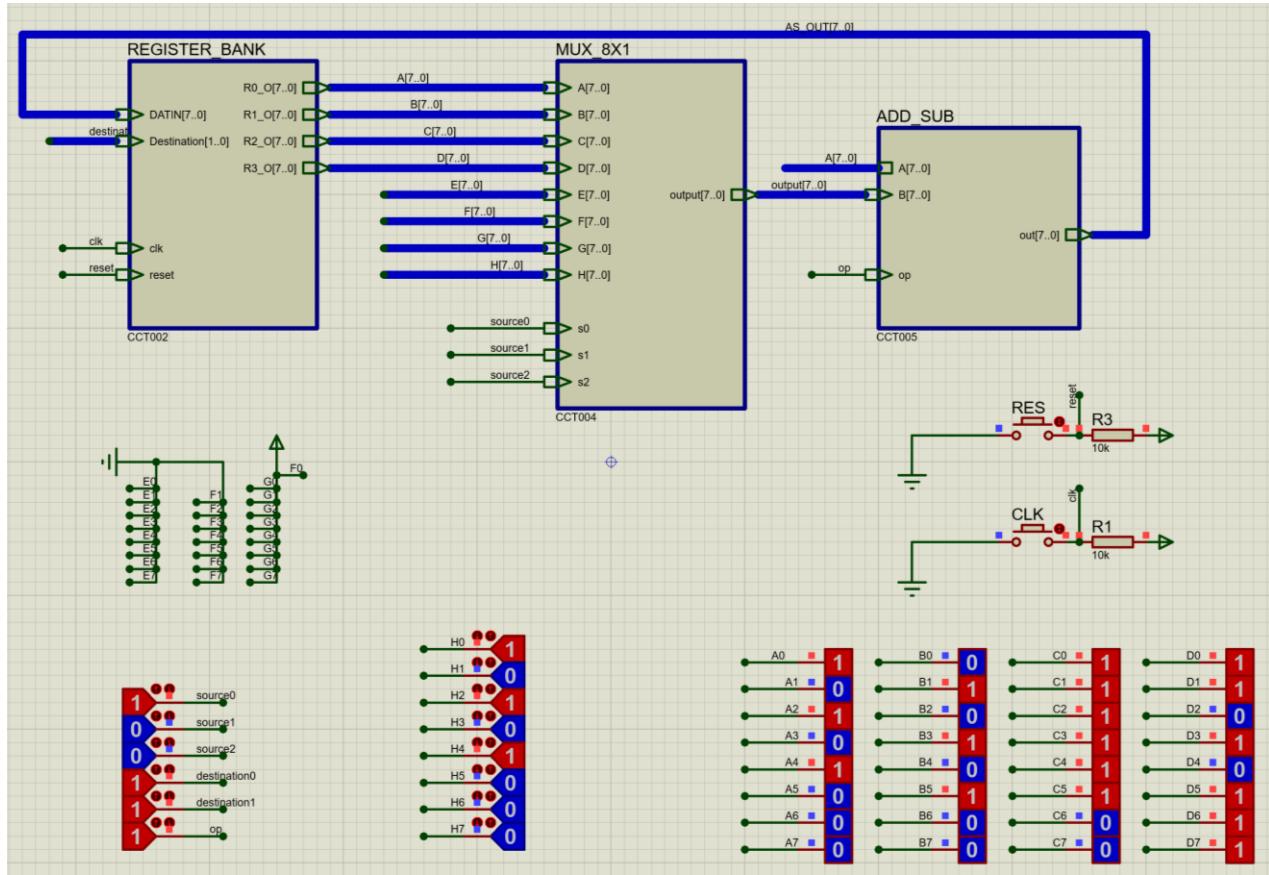


شكل .9 $R1 < 21$

و سپس جمع کردن R0 و R1 و ذخیره در R3 (حاصل 63) :

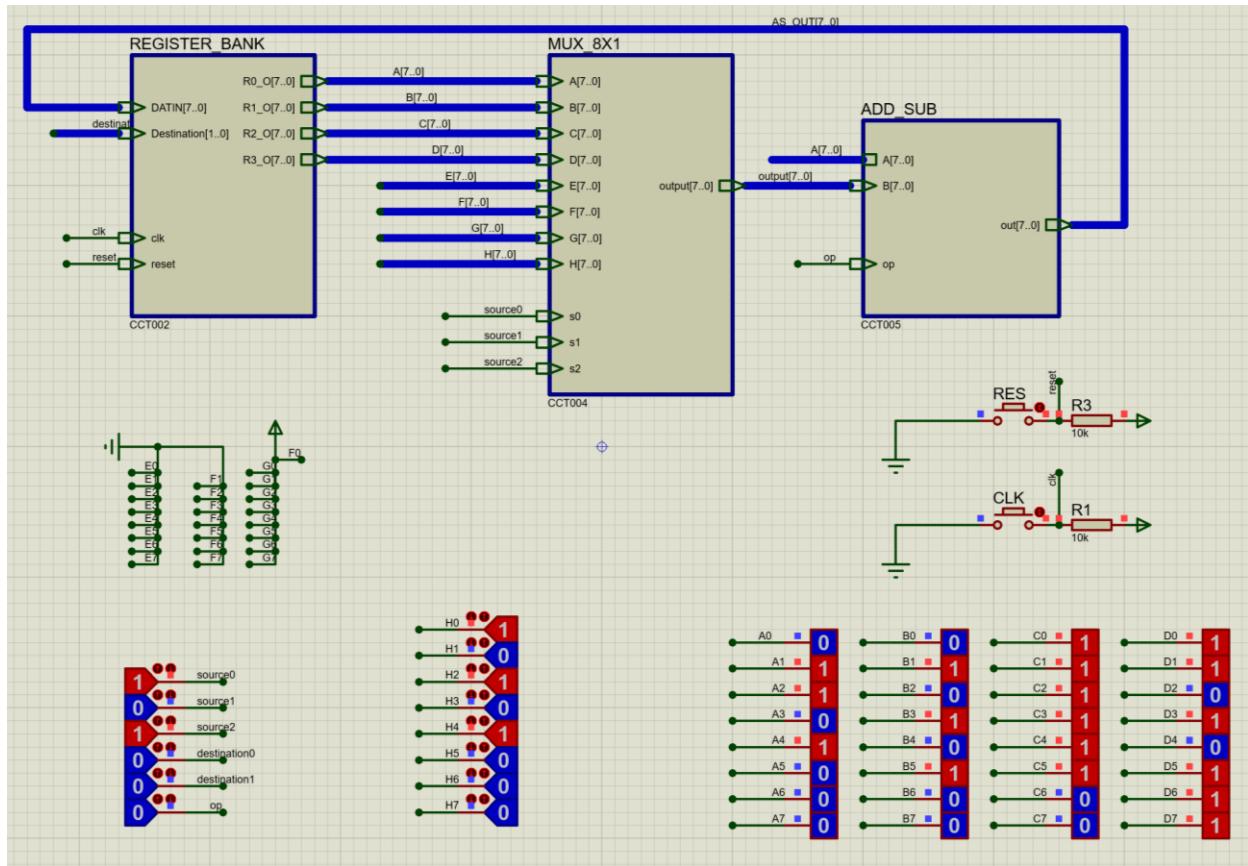


و سپس کم کردن R0 از R1 و ذخیره در R4 (حاصل 21-) :



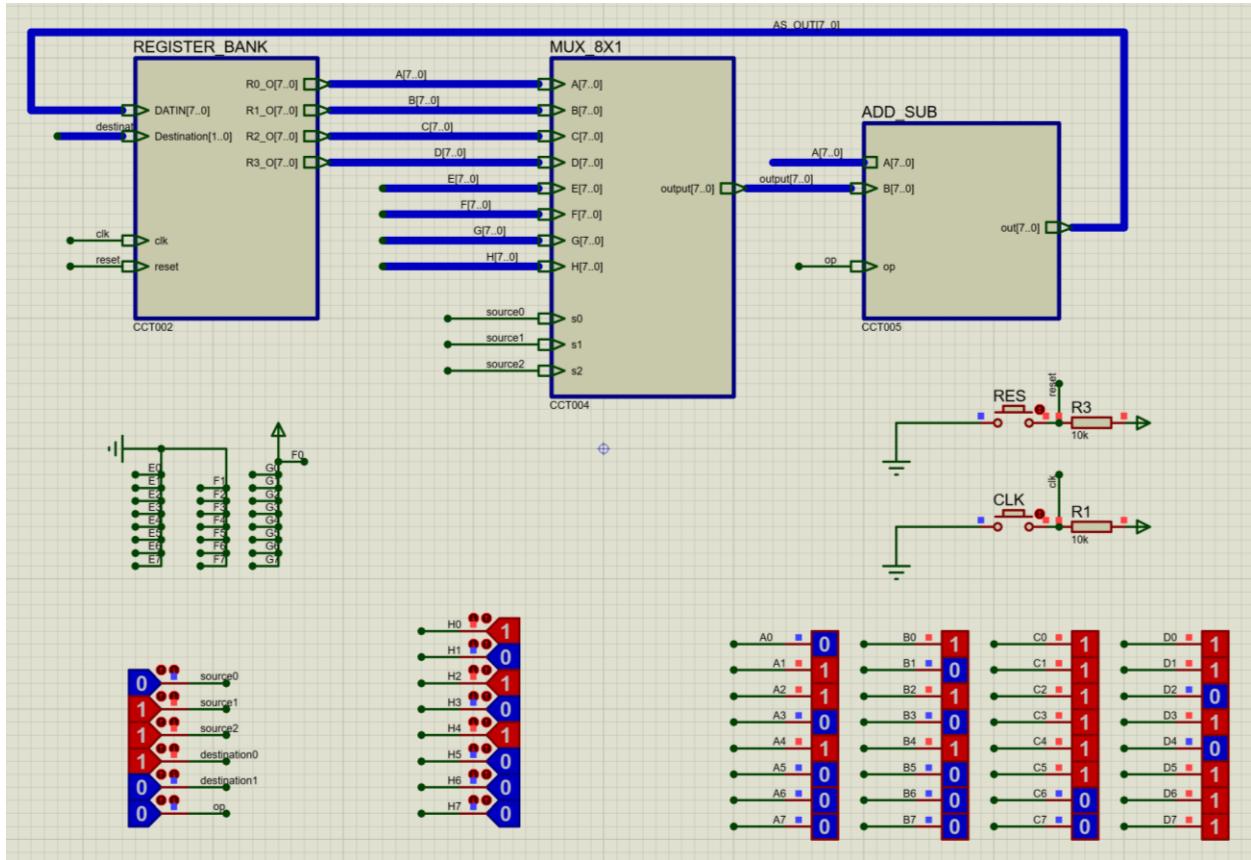
$R4 \leftarrow R1 - R2 .11$ کش

و سپس یک واحد اضافه کردن به R0 (حاصل 22) و ذخیره در R0 :



$$R1 \leftarrow R1 + 1 .12 \text{ کل}$$

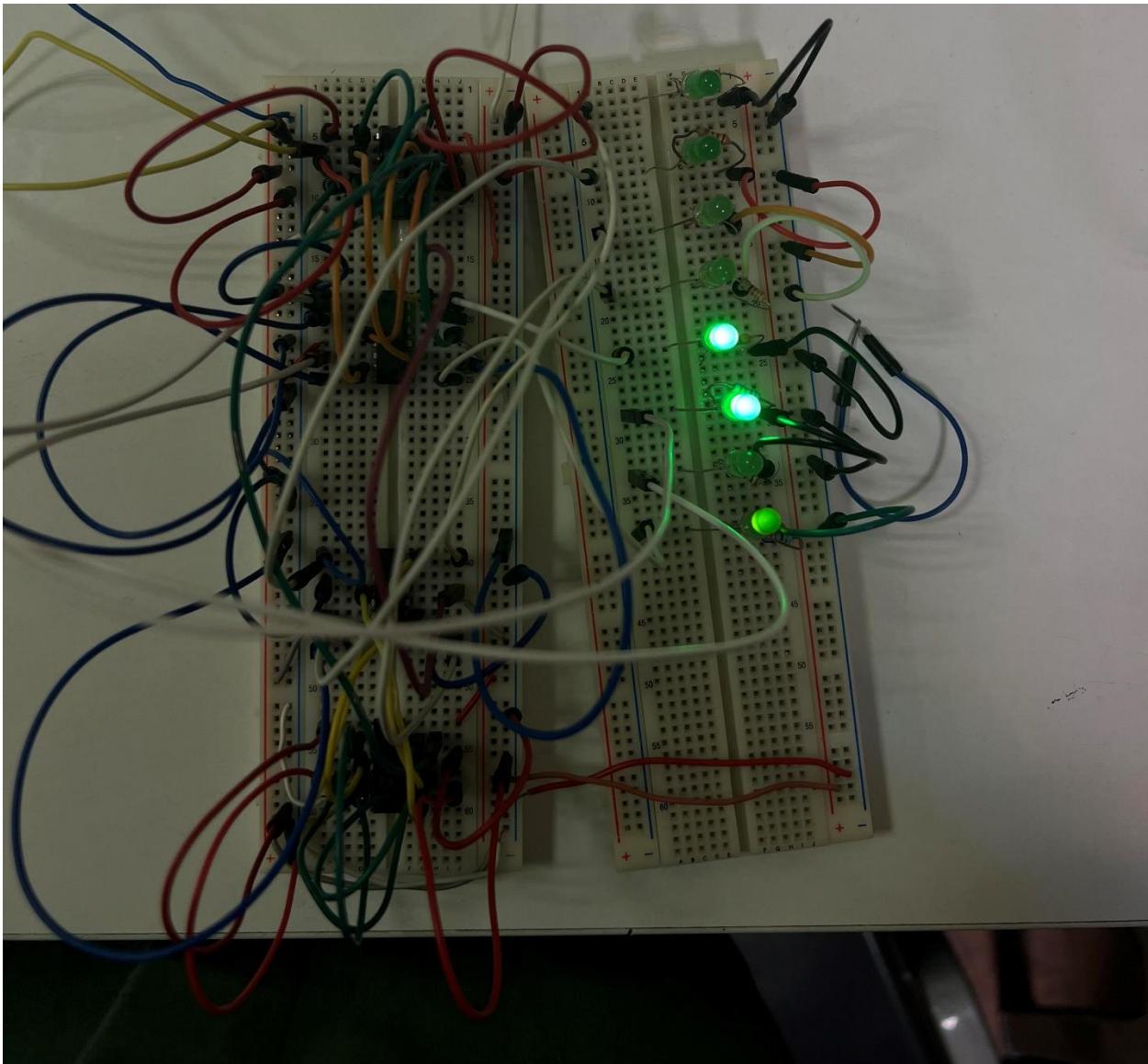
و سپس یک واحد کم کردن از R0 (مقدار 21) و ذخیره در R1 :



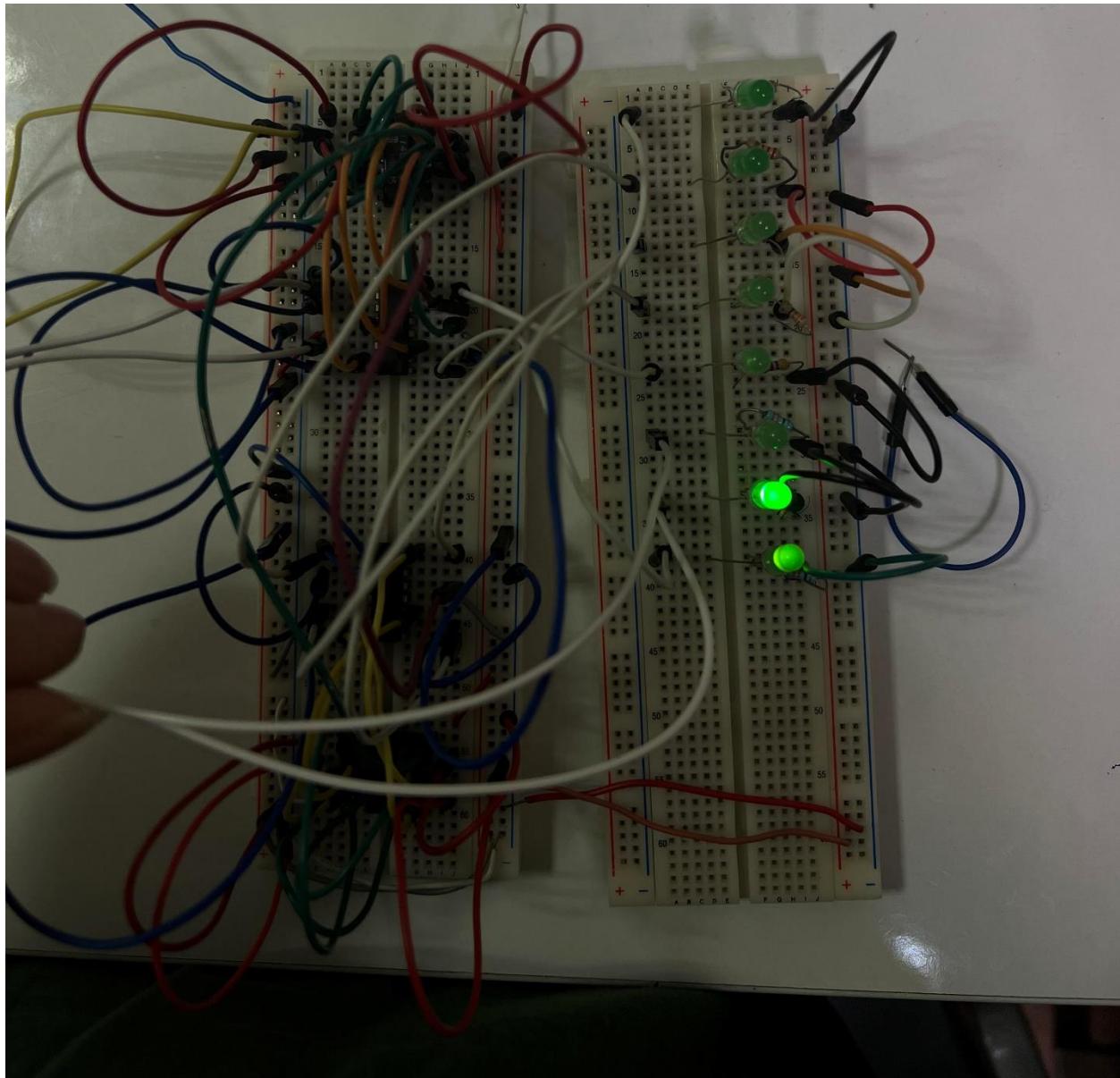
$R2 <- R1 - 1$. 13

کار در کلاس

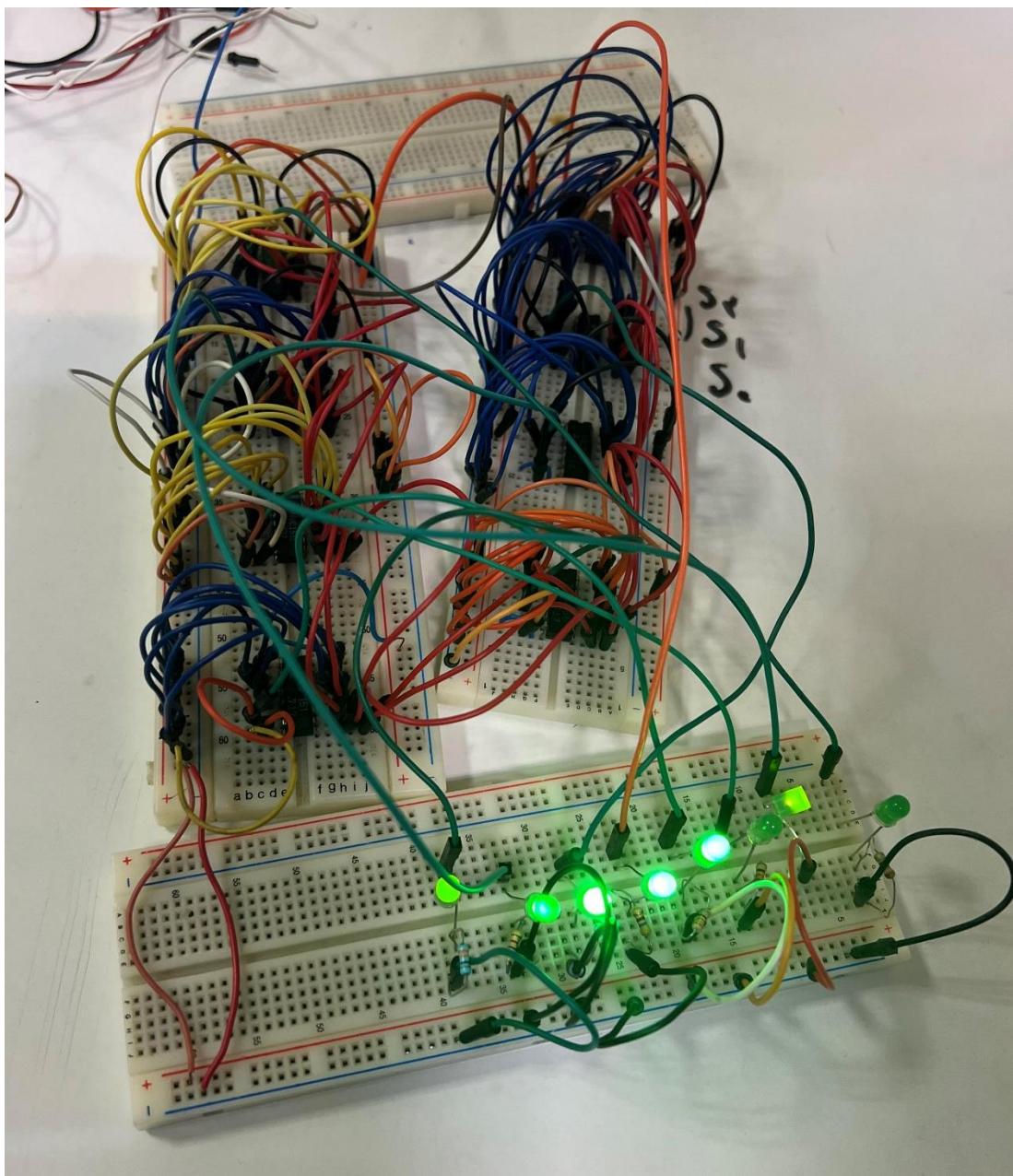
در روز سه شنبه 24 ام مرداد، این مدار را در آزمایشگاه معماری، روی بردبورد پیاده سازی کردیم. همان طور که در شکل های زیر مشاهده میکنید ابتدا مازول های جمع/تفريق کننده، مالتی پلکسر و بانک رجیستر را روی بورد ها پیاده سازی کردیم و در ادامه با ایجاد اتصالات بین آنها، مدار کلی را تشکیل دادیم.



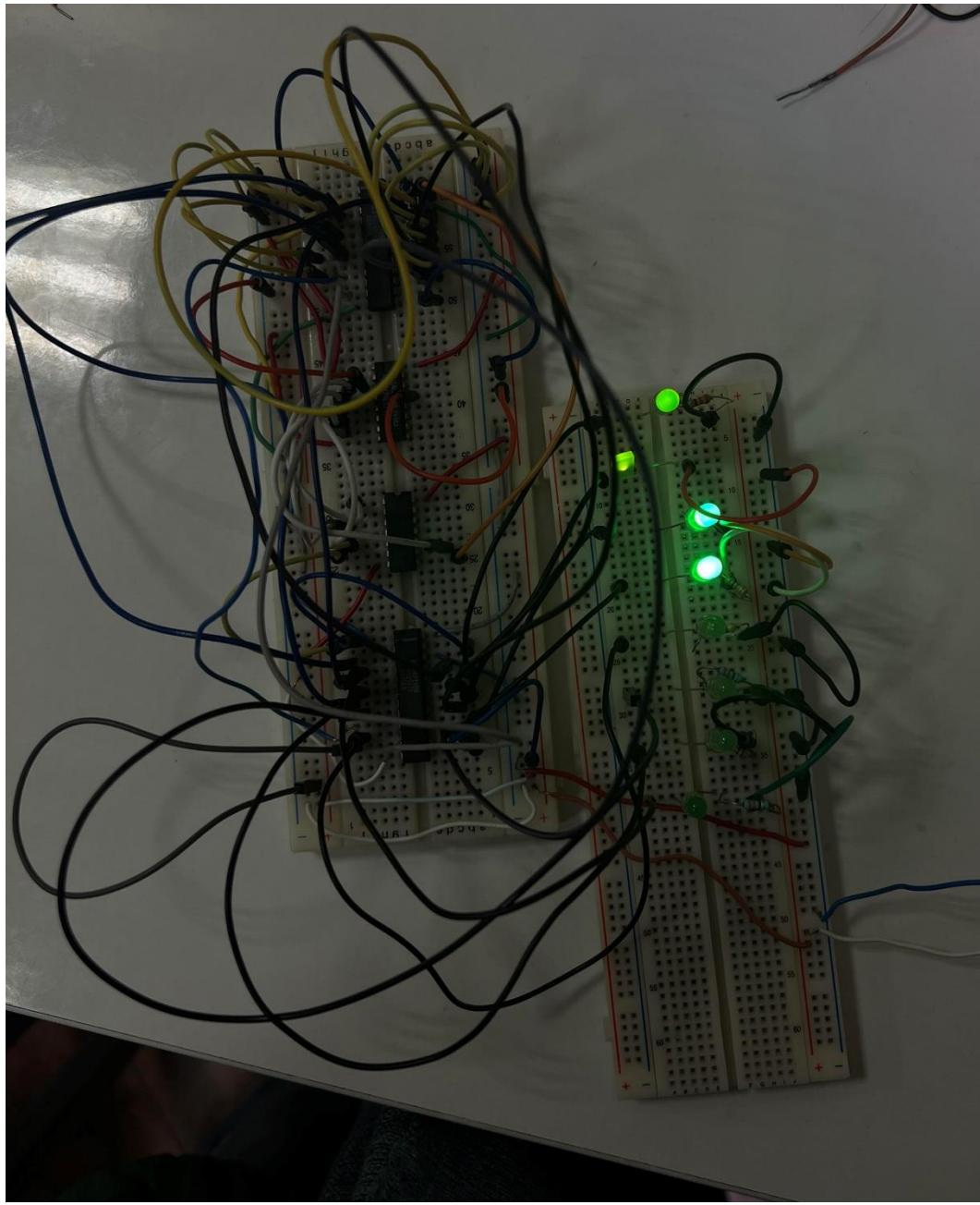
شکل 14. مازول جمع/تفریق کننده (مازول در وضعیت جمع کننده و با ورودی های $A=8$ و $B=5$ نشان داده شده است)



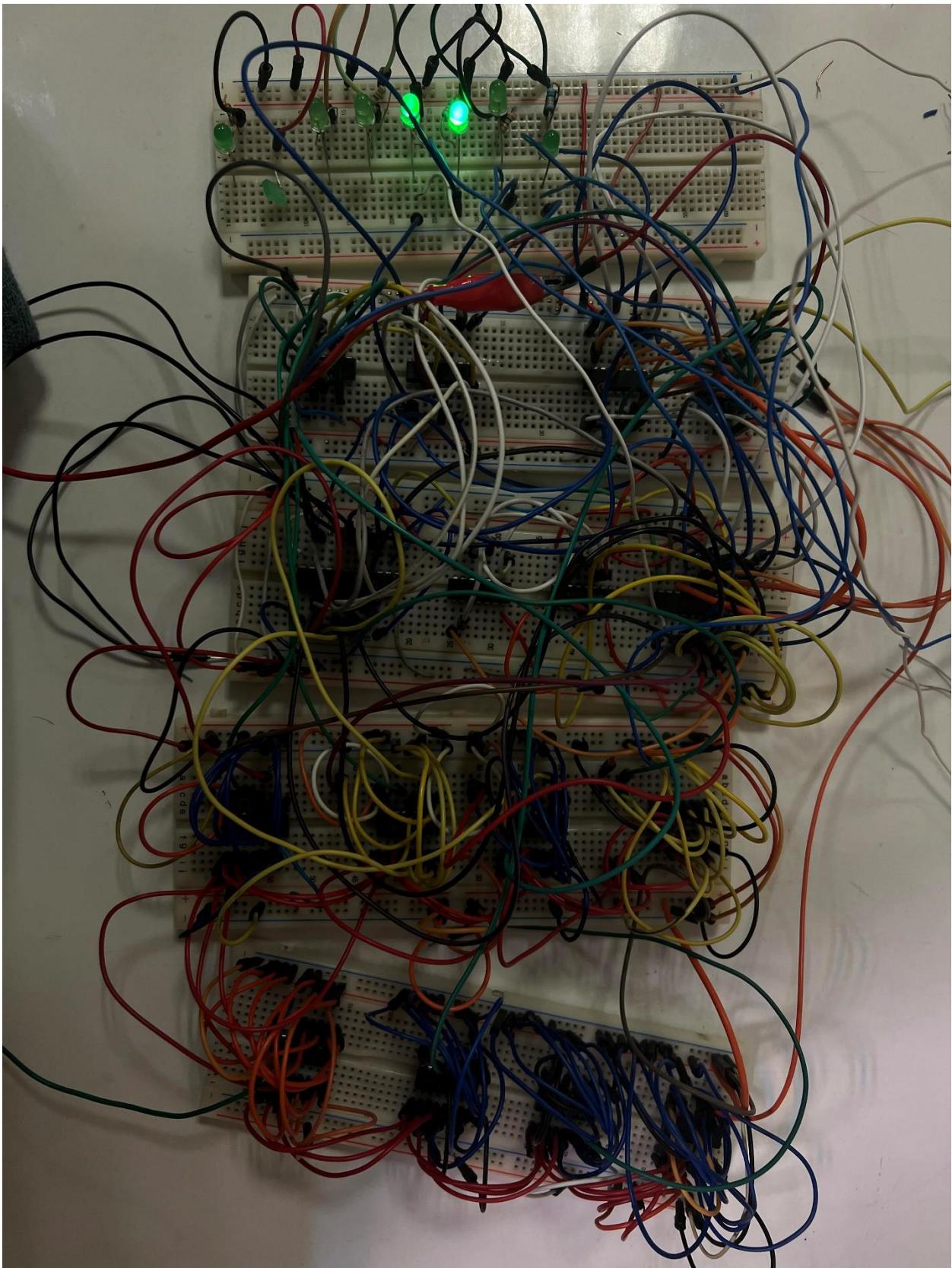
شکل 15. مازول جمع/تفرقی کننده (مازول در وضعیت تفرقی کننده و با ورودی های A<-8 و B<-5 نشان داده شده)



شکل 16. مازول مالتی پلکسی



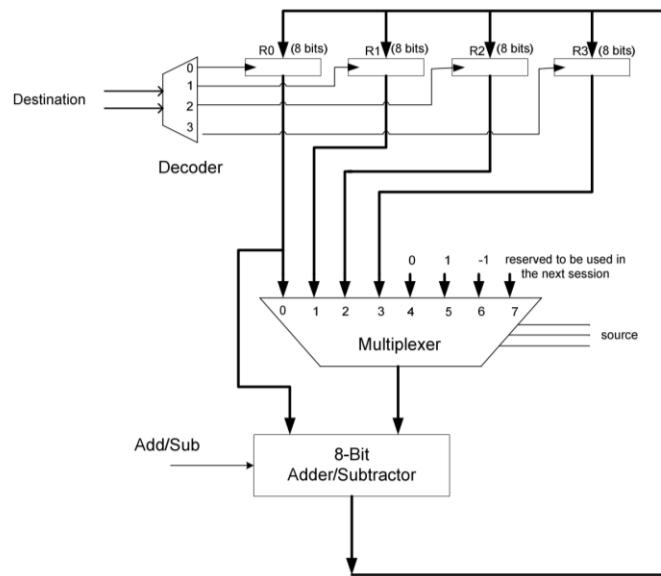
شکل ۱۷. مارژون پانک رجیستر (به دلیل وقت محدود جلسه آزمایشگاه، دو رجیستر را پیاده سازی کردیم)



شکل 18. مدار کلی آزمایش (که از متصل کردن مازول های بالا به دست آمده)

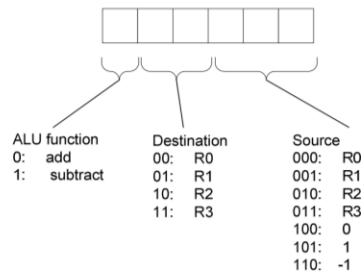
نتیجه‌گیری

در این آزمایش توانستیم واحد محاسبه کننده زیر را طراحی و روی بورد پیاده سازی کنیم.



شکل ۶: معماری واحد محاسبات

این معماری را طوری پیاده سازی کنید که قابلیت انجام فرمانهای شش بیتی زیر را داشته باشد:



شکل ۷: قالب فرمانهای شش بیتی

منابع و مراجع:

- Mano, M. Morris. Computer system architecture. Prentice-Hall of India, 2003.
- Computer Organization & Design, The Hardware / Software Interface”, D. Patterson and J. L. Hennessy, Morgan Kaufmann Publishing, 2005.