

به نام خدا



## آزمایش شماره ۶

آزمایش شماره ۶  
از معماری - دکتر سربازی آزاد

دانشکده مهندسی کامپیوتر

دانشگاه صنعتی شریف

نیمسال تابستان ۱۴۰۳

اعضای گروه:

میترا قلی پور چناری - ۰۱۱۰۶۳۶۳

نیکا قادری - ۰۱۱۰۶۳۲۸

ملیکا علیزاده - ۰۱۱۰۶۲۵۵



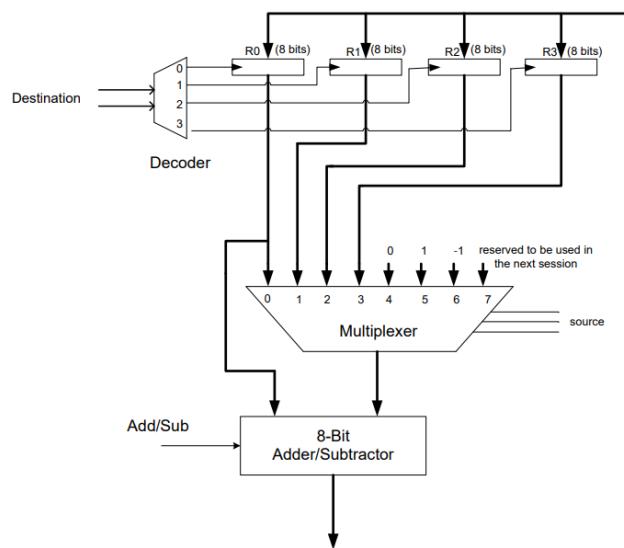
## ۱ هدف و ایده کلی

هدف از این آزمایش طراحی یک واحد محاسبات و منطق می باشد که اعمال جمع و تفریق را انجام می دهد. به این صورت که اطلاعات ورودی در قالب یک دستور ۶ بیتی وارد می شود و با توجه به آن، عملیات محاسباتی روی یک رجیستر مقصد انجام می شود. یکی از رجیستر های مبدأ به طور ثابت رجیستر شماره ۰ می باشد و دیگری می تواند به طور دلخواه یک رجیستر یا عدد ثابت باشد. جزئیات پیاده سازی به صورت زیر می باشد:

### شرح آزمایش

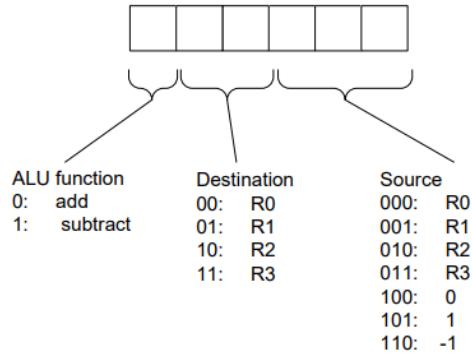
در این آزمایش، واحد محاسبات و مجموعه ثبات های عمومی ماشین را طراحی و پیاده سازی می کنیم. معماری مورد نظر در شکل ۸ نشان داده شده است. این معماری امکان انجام جمع و تفریق با انتخاب ثبات های مبدأ و ثبات نگهدارنده نتیجه (مقصد) را فراهم می کند. چهار ثبات عمومی R0, R1, R2 و R3 هشت بیتی هستند. همانطور که در شکل ۸ نمایش داده شده، یکی از عملوندهای ALU می تواند به صورت ثابت محتوای ثبات R0 و دیگری محتوای یکی از ثبات های R0 تا R3 یا مقادیر ثابت ۰, ۱ و -۱ باشد. حاصل تولید شده توسط ALU (جمع/تفرق) به یکی از ثبات های مقصد R0 تا R3 منتقل می شود.

این معماری را به گونه ای پیاده سازی کنید که قابلیت انجام فرمان های شش بیتی در شکل ۹ را داشته باشد.



شکل ۸: معماری واحد محاسبات

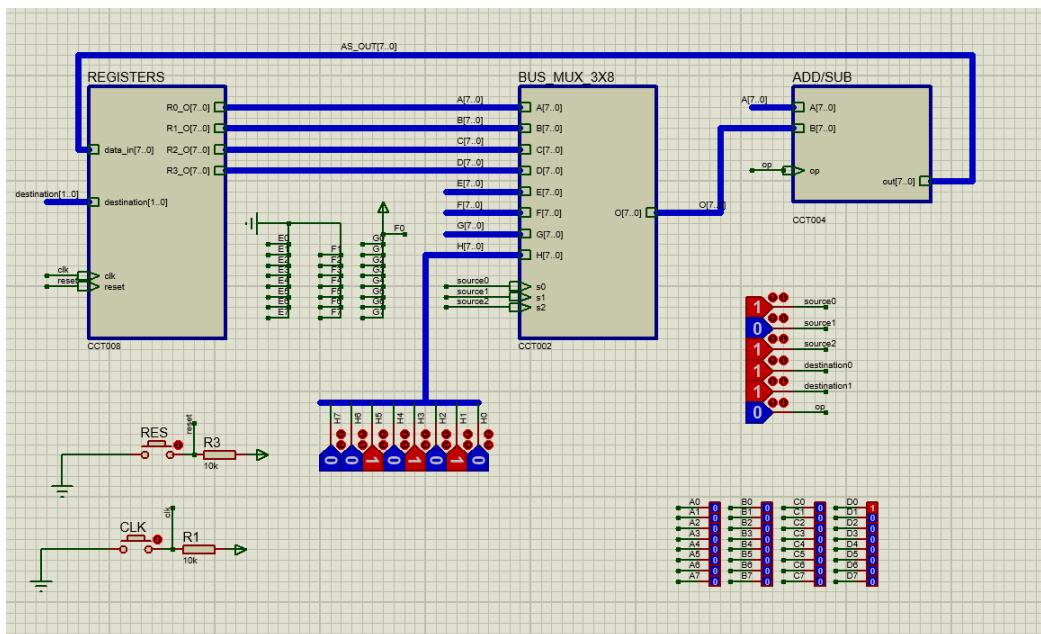
فرمت نمایش اعداد به صورت زیر می باشد. یعنی بیت اول، مشخص کننده عملیات، ۲ بیت بعدی نمایانگر رجیستر مقصد و ۳ بیت بعدی، رجیستر مبدأ می باشند:



شکل ۹: قالب فرمان‌های شش بیتی

## ۲ جزئیات پیاده‌سازی و توضیحات

نمای کلی به صورت زیر می‌باشد:



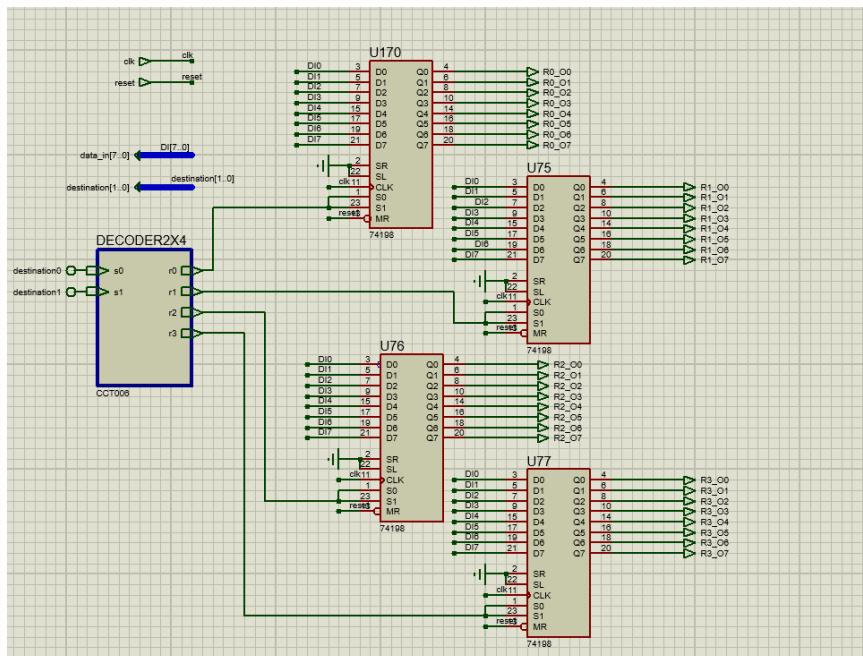
در اینجا دکمه‌های استارت و کلک به صورت پوش بازن پیاده‌سازی شده‌اند و ورودی و خروجی‌ها همانند دستور کار قابل مقداردهی و مشاهده می‌باشند.



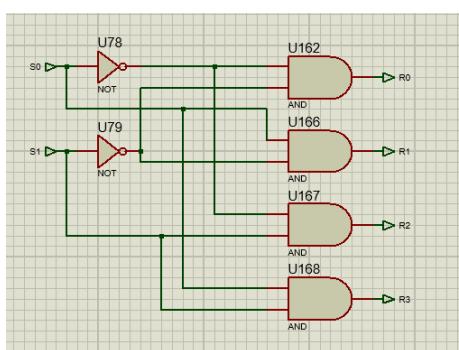
این مدل، متشکل از بخش‌های مختلفی است که در کنار هم عملیات مورد نظر را انجام می‌دهند: هر کدام از این مأژول‌ها با دقت بیشتری مورد بررسی قرار خواهد گرفت.

## Registers ۱.۲

این واحد ثبات‌ها را پیاده سازی می‌کند و ورودی لازم را در رجیستر انتخاب شده لود می‌کند.



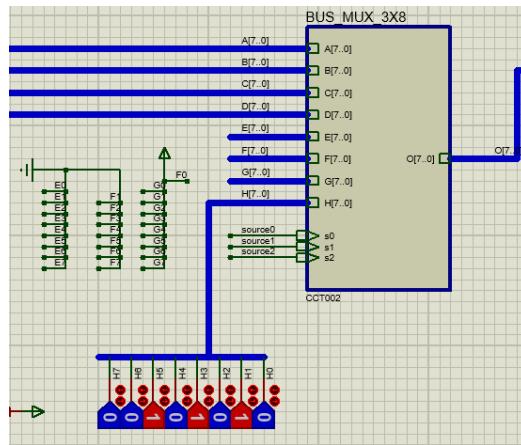
برای پیاده سازی رجیسترها از ۴ قطعه ۷۴۱۹۸ استفاده می‌کنیم. دو بیت کنترل هر یک را به خروجی دیکودر متصل کردیم تا با فعال شدن خروجی متناظر دیکودر، داده در رجیستر لود شود و در غیر این صورت، هیچ کاری انجام نشود. همچنین برای انتخاب این که داده ورودی در کدام یک از رجیسترها باید لود شود، از یک دیکودر ۴ به یک استفاده می‌کنیم که به صورت زیر پیاده سازی شده است:



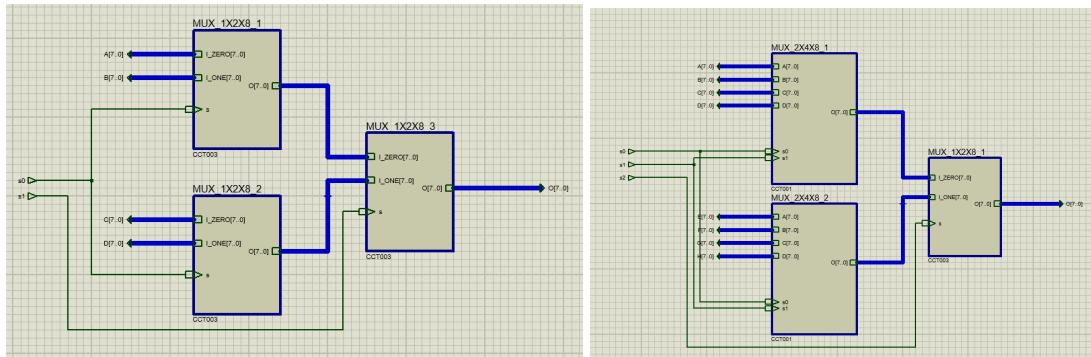


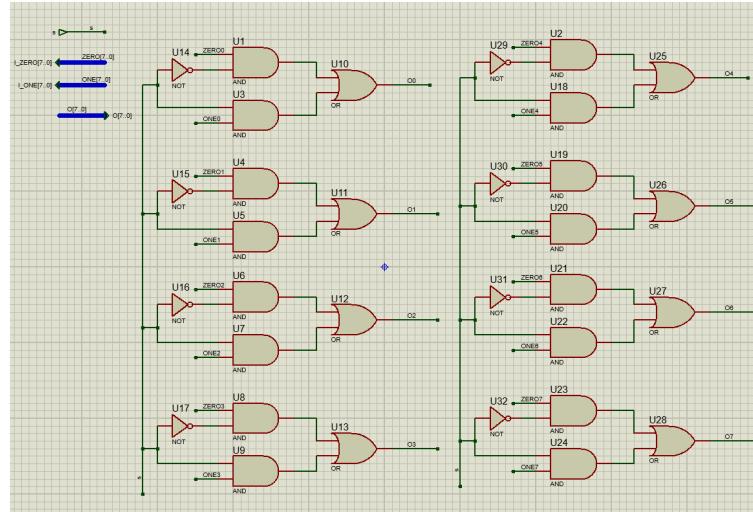
## Bus mux ۲.۲

این بخش از خروجی‌های register استفاده می‌کند تا عملوند دوم واحد جمع کننده را تولید کند. این واحد در واقع یک مالتی پلکسر ۸ به ۱ با ورودی‌های هشت بیتی می‌باشد. چهار ورودی اول آن مستقیماً از رجیسترها می‌آید و ۴ ورودی بعدی آن به ترتیب ۰، ۱، ۱- و یک عدد ورودی هستند. برای ۱-، از مکمل دوی عدد یک استفاده کردیم.



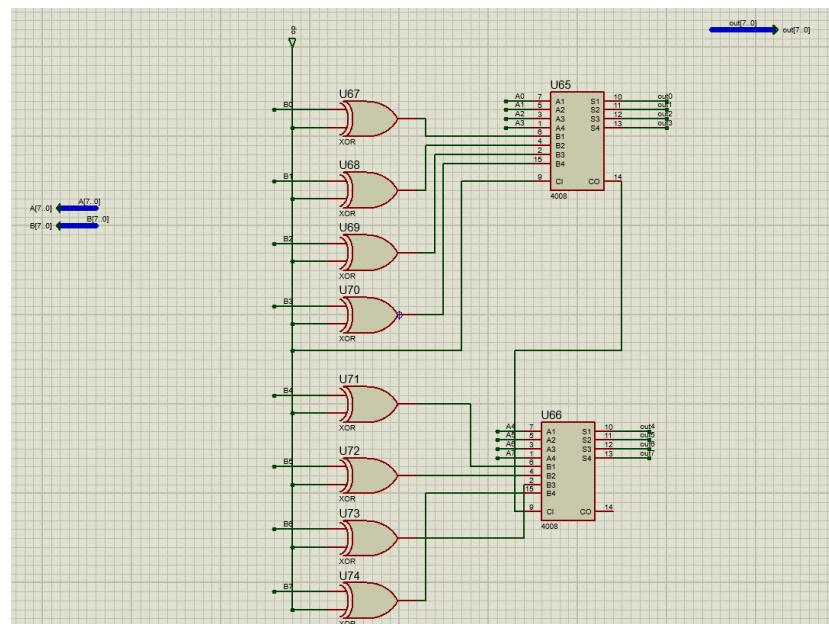
برای ساخت آن نیز از تعدادی مالتی پلکسرهای کوچک‌تر استفاده شده است:





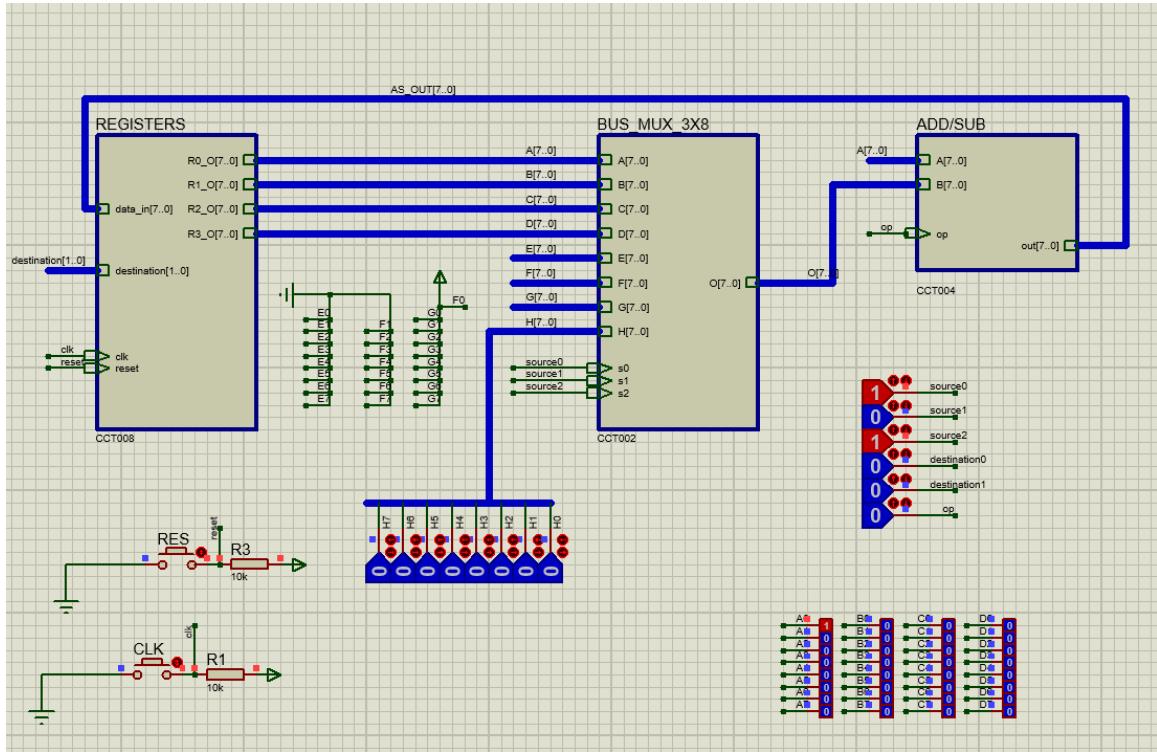
## Add/Sub ۳.۲

این یک واحد جمع/تفريق کننده هشت بیتی می باشد که یکی از ورودی های آن به صورت ثابت رجیستر است و ورودی دیگر آن از Bus mux تامین می شود. برای پیاده سازی آن از دو واحد جمع کننده چهار بیتی استفاده شده است. همچنین برای پیاده سازی تفريقي، تمام بیت های عدد دوم با  $xor$  شده و همچنین به عنوان  $Cin$  به جمع کننده اول داده شده است تا وقتی که مقدارش یک باشد، عمل تفريقي انجام شود.

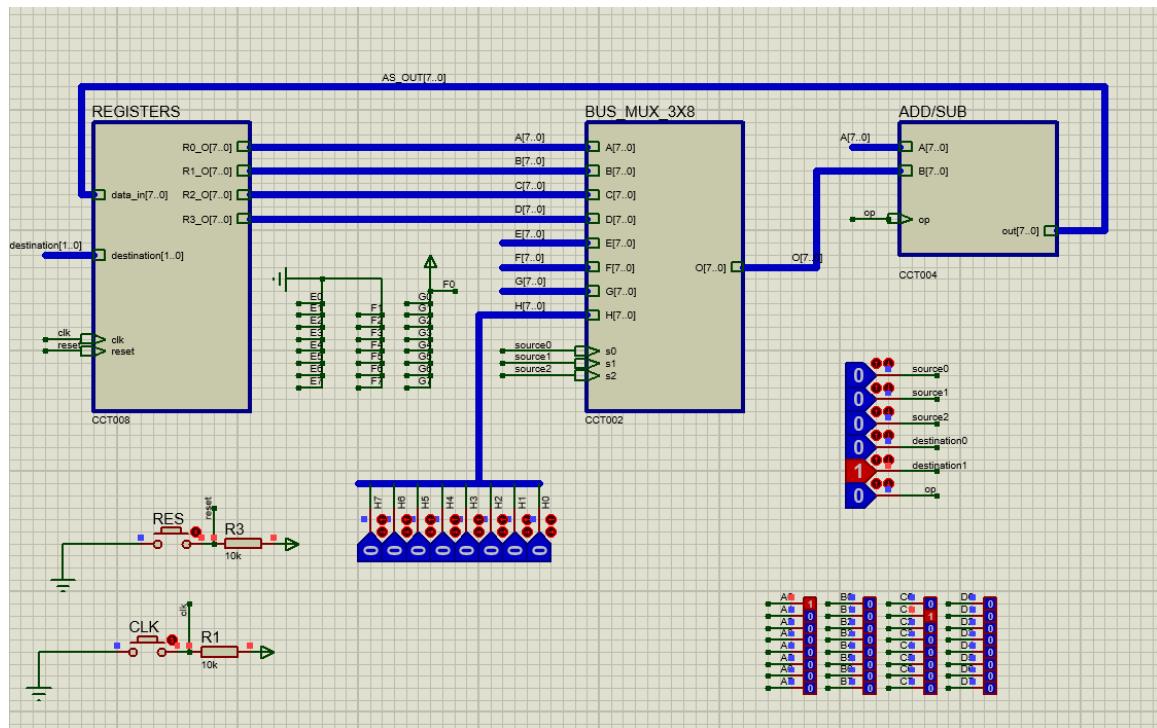




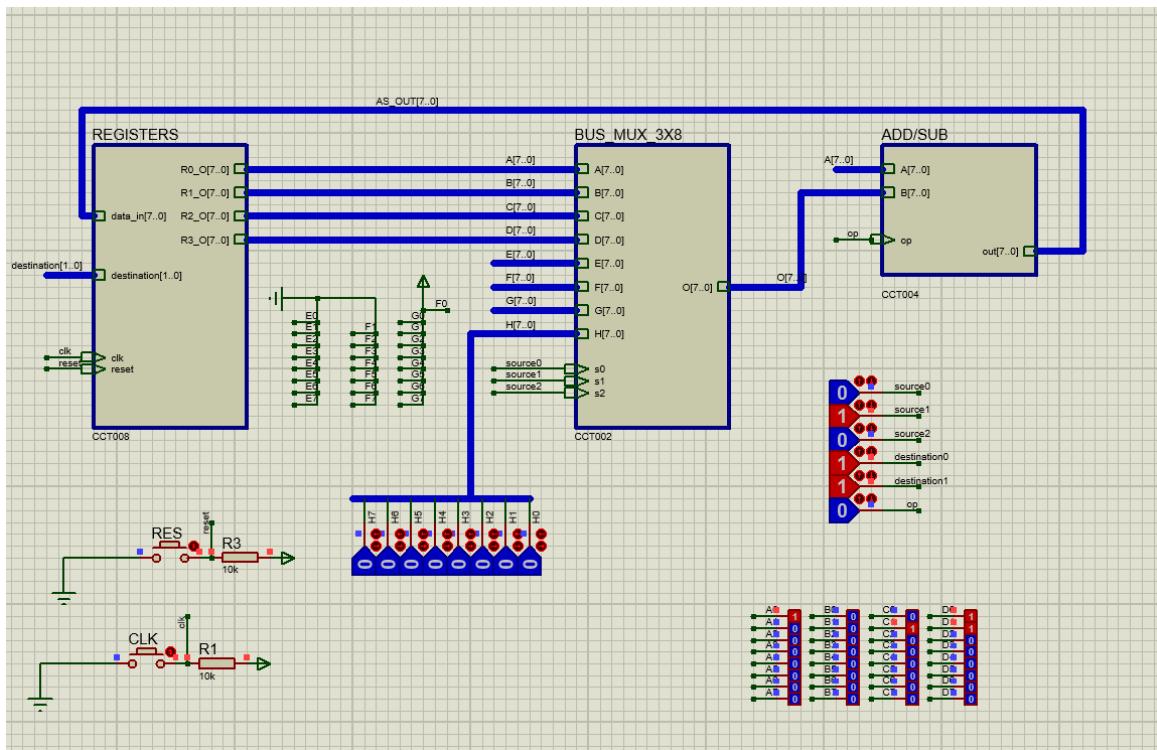
## ۳ شبیه‌سازی و تست



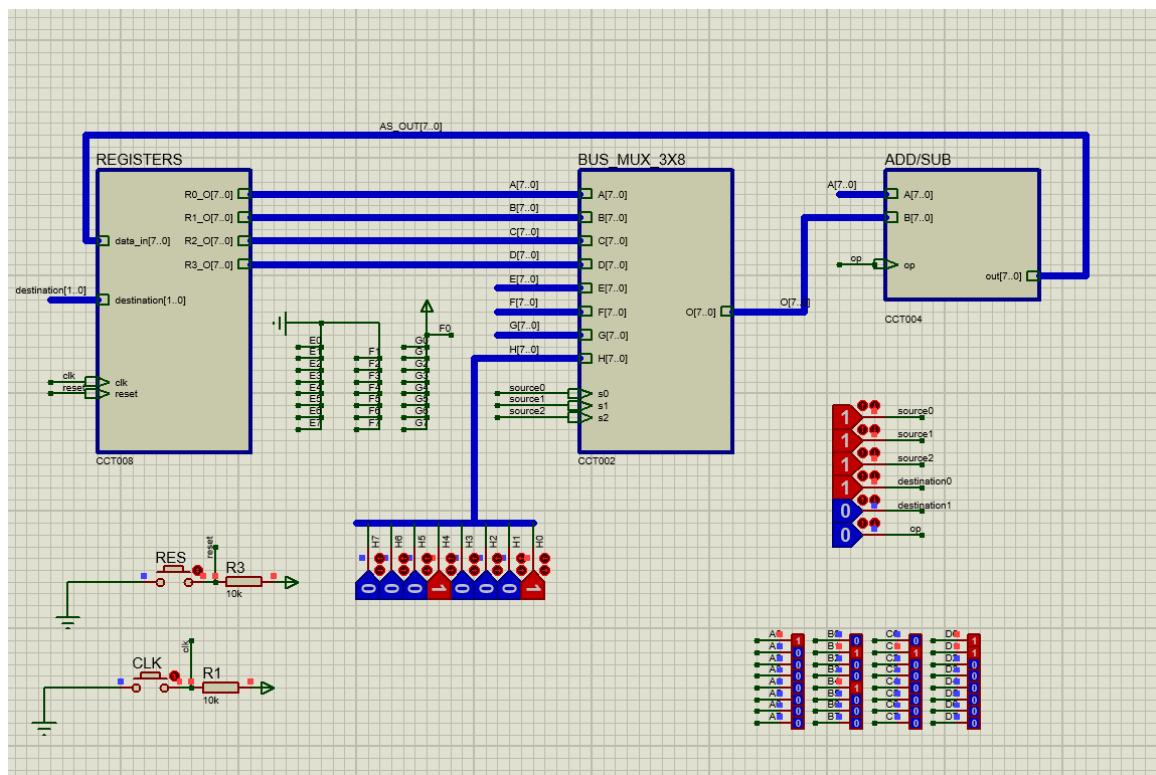
شکل ۱: رجیستر ۰ که در ابتدا مقدار ۰ را دارد با یک جمع شده است



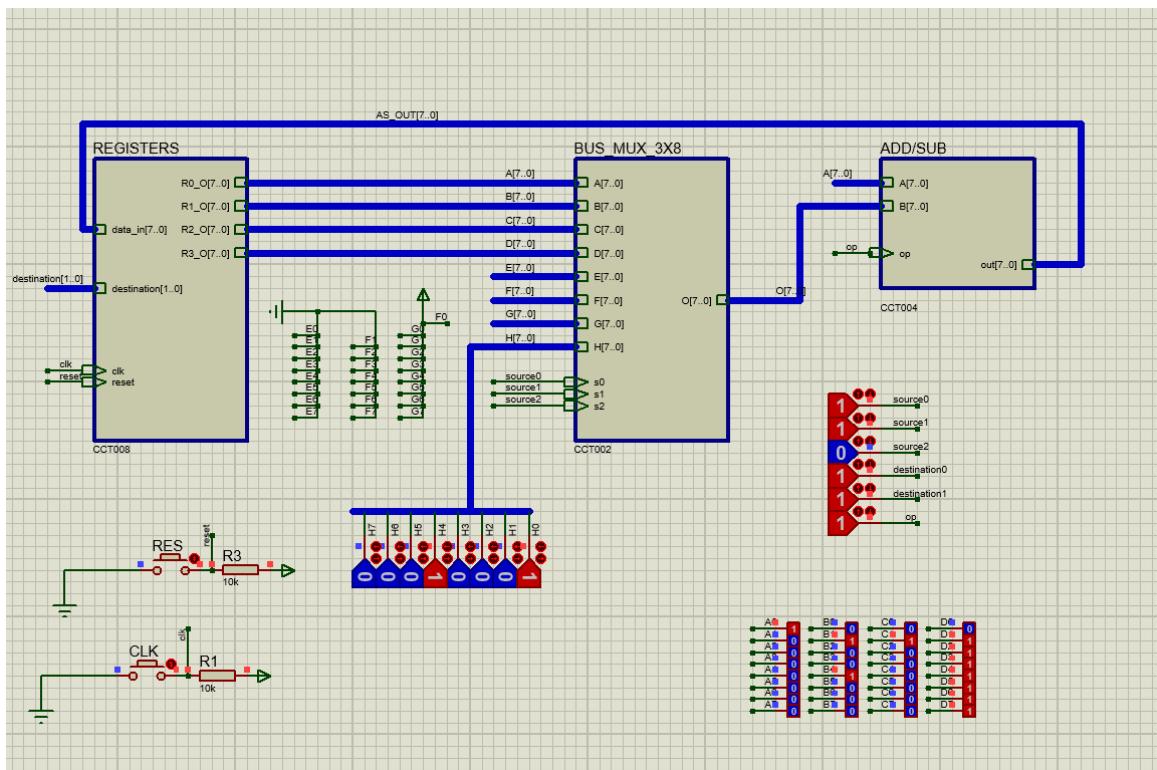
شکل ۲: رجیستر ۰ با خودش جمع شد و در ۲ لود شد



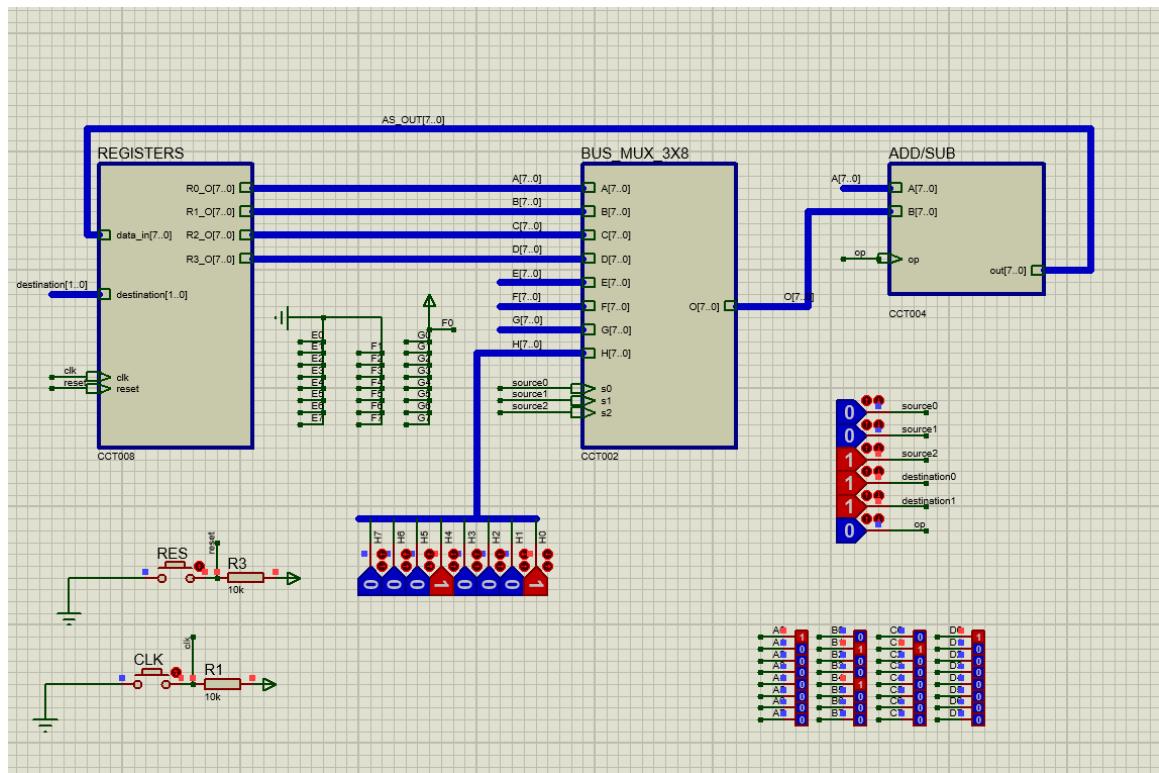
شکل ۳: ۲ با ۰ جمع شد و در ۳ لود شد



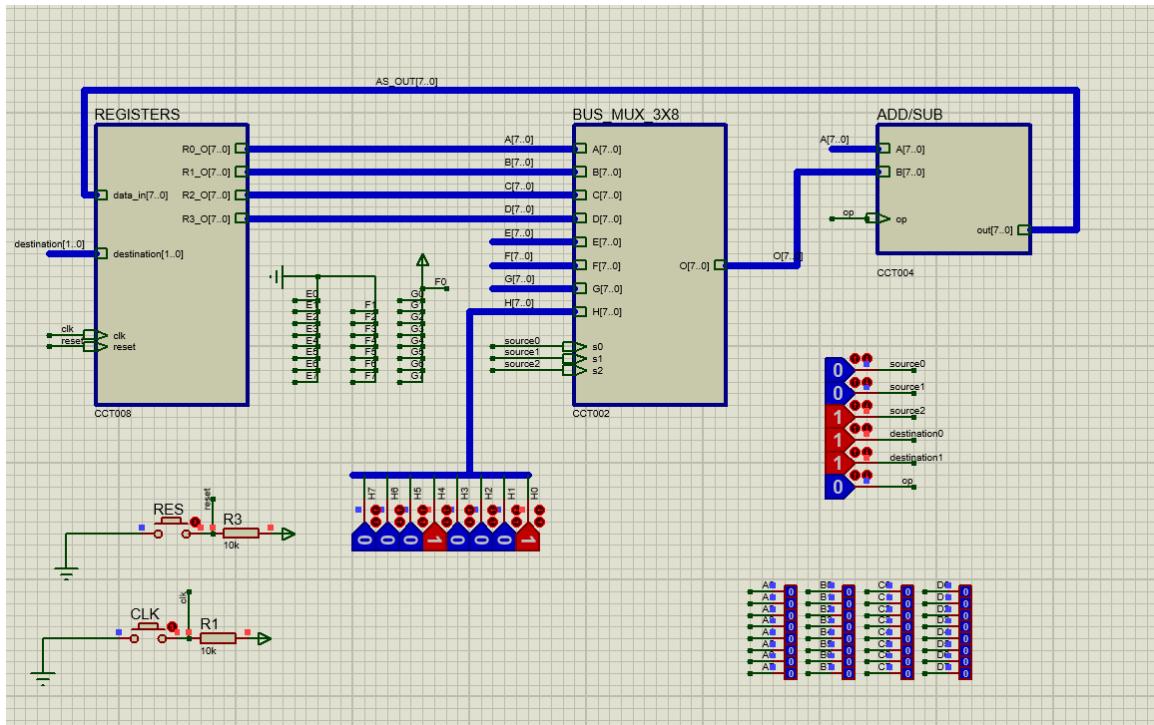
شکل ۴: ۰ با عدد ۱۷ جمع شد و نتیجه - ۱۸ - در رجیستر ۱ لود شد



شکل ۵: ۳ از ۱ کم شد و نتیجه -2 - در ۳ لود شد



شکل ۶: ۰ با عدد ۰ جمع شد و در ۳ لود شد



شکل ۷: در آخر، ریست فشرده شد



### ۱.۳ پیاده سازی فیزیکی

جهت ساده سازی هنگام ساخت مدار، چهار رجیستر را به دو رجیستر خلاصه کردیم. تعداد بیت های سورس را هم به دو کاهش دادیم که باعث شد کلا ۴ حالت برای انتخاب عملوند اول داشته باشیم: رجیستر ۰ و ۱، عدد صفر و عدد یک.

همان طور که توضیح داده شد از قطعات ۷۴۰۸، ۷۴۱۹۸، ۷۴۰۴، (not gate) ۷۴۳۲، (or gate) ۷۴۰۸ و (xor gate) ۷۴۸۶ استفاده شده است.

