بسمه تعالی



گزارش کار دوم آزمایشگاه معماری کامپیوتر

جمع/تفریق‌کننده ممیز شناور

استاد

دکتر سربازی آزاد

نویسندگان

امیررضا سلیمان‌بیگی 98109837

آرین احدی‌نیا 98103878

سایه جارالهی 98101339

دانشگاه صنعتی شریف

تابستان ۱۴۰۱

فهرست مطالب

[1- هدف آزمایش 3](#_Toc111219172)

[2- بررسی ورودی و خروجی‌های مدار 4](#_Toc111219173)

[3- الگوریتم 5](#_Toc111219174)

[4- ماژول‌های استفاده شده 7](#_Toc111219175)

[4-1 نمای کلی مدار 7](#_Toc111219176)

[4-2 Controller : تولید بیت‌های کنترلی مدار 8](#_Toc111219177)

[4-3 Mantissa\_reg :‌ ثبات نگه‌داری مقادیر مانتیس 12](#_Toc111219178)

[4-4 Exp\_reg: نگه‌داری توان عددها 13](#_Toc111219179)

[4-5 Exp\_comparator: مقایسه توان‌ها 15](#_Toc111219180)

[4-6 Sign\_magnitude\_adder: جمع/تفریق کننده علامتدار مانتیس 15](#_Toc111219181)

[4-7 24Adder: جمع/تفریق کننده ۲۴ بیتی 18](#_Toc111219182)

[4-8 Final\_result\_reg: ثبات نگه‌داری حاصل نهایی 18](#_Toc111219183)

[4-9 ماژول آماده 4013 19](#_Toc111219184)

[4-10 ماژول آماده 74198 19](#_Toc111219185)

[4-11 ماژول آماده 4512 19](#_Toc111219186)

[4-12 ماژول آماده 4516 19](#_Toc111219187)

[4-13 تمام صفر(Zero) 19](#_Toc111219188)

[4-14 شیفت رجیستر 25 بیتی(SHIFT REG) 20](#_Toc111219189)

[4-15 بالارونده و پایین‌رونده(INC\_DEC) 21](#_Toc111219190)

[4-16 واحد کنترل Normalizer(CONTROL) 22](#_Toc111219191)

[4-17 Normalizer: نرمال‌سازی خروجی نهایی 24](#_Toc111219192)

[5- ساختار مدار 25](#_Toc111219193)

[6- خروجی نهایی مدار 28](#_Toc111219194)

1. هدف آزمایش

در این آزمایش، واحد محاسباتی برای اعداد ممیز شناور طراحی و پیاده سازی شده است. این واحد محاسباتی قابلیت انجام عملیات جمع و تفریق بر روی دو عدد ورودی ممیز شناور را دارد. دو عدد ورودی 32 بیتی هستند که به ترتیب با A[31..0] و B[31..0] نمایش داده می‌شوند. درصورتی که ورودی addsub مقدار یک را داشته باشد، عملیات تفریق A-B و درغیر اینصورت عملیات جمع A+B انجام می‌شود.

دو ورودی A و B در فرمت IEEE754 هستند به صورتی که 23 بیت کوچکتر مقدار مانتیس و بیت های 23 تا 20 مقدار نما را نشان میدهند. همچنین پر ارزشترین بیت نیز بیانگر علامت عدد است که درصورتی که یک باشد عدد منفی و درغیر اینصورت مثبت است.

در انتها و پس از آنکه تمام عملیات‌های مورد نیاز انجام شود، خروجی در ثباتی ریخته می‌شود و به صورت یک عدد 32 بیتی در فرمت IEEE754 نمایش داده می‌شود. همچنین underflow و overflow نیز دارای بیت‌های جداگانه‌ای هستند. لازم به ذکر است که خروجی نهایی normalized است. همچنین تضمین شده است که ورودی ها نیز حالت نرمال دارند و درصورتی که ورودی نرمال نباشد، رفتار مدار معتبر نیست.

1. بررسی ورودی و خروجی‌های مدار

ورودی‌ها:

* A: این ورودی 32 بیتی است و نمایانگر عدد اول است که به صورت نرمال و در فرمت IEEE754 داده میشود.
* B: این عدد 32 بیتی دوم است که دقیقا شرایطی مشابه ورودی A دارد.
* Addsub: درصورتی که یک باشد حاصل A-B و در غیر اینصورت حاصل A+B محاسبه میشود.
* Clock: کلاک مدار است.
* Start: بیت مشخص کننده آماده بودن ورودی‌ها است.

خروجی‌ها: . (توجه داشته باشید که در این آزمایش، به جهت آسانتر شدن بررسی وضعیت مدار، خروجی‌های دیگری نیز علاوه بر خروجی های موجود در دستور کار در نظر گرفته شده است که حذف آن ها تاثیری در جواب ندارد.)

* C: خروجی 32 بیتی است که در فرمت IEEE754 است و نرمالایز شده است و جواب محاسبه را نمایش میدهد.
* Overflow: درصورتی که حاصل عملیات سرریز داشته باشد یک میشود.
* Underflow: درصورتی که حاصل عملیات underflow داشته باشد یک میشود.
* End: پس از پایان یافتن عملیات و درصورتی که جواب معتبر باشد (سیگنال start در میانه محاسبه یک نشده باشد) یک میشود و تا زمانی که ورودی بعدی آماده شود یک میماند.
* Calc\_end: این خروجی اضافه طراحی شده است و برای راحتتر چک کردن مدار قرار داده شده است. هنگامی که کار مدار تمام میشود (فارغ از آنکه جواب معتبر است یا خیر) مقدار آن یک میشود.
* Start\_status: این سیگنال نیز برای راحتی چک کردن مدار تعبیه شده است. هنگامی که start معتبر باشد و توسط مدار شناخته شود و مدار شروع به کار کند یک میشود. در واقع با استفاده از این سیگنال، اینکه بیت start به صورت پالس کار میکند را نمایش میدهیم. هنگامی که start به عنوان سیگنال شروع مدار شناخته میشود و مدار شروع به محاسبه میکند این بیت به مدت یک کلاک مقدار 1 را به خود میگیرد.

در این آزمایش با یک شدن مقدار بیت start عملیات شروع میشود. طبق گفته های دستیار آموزشی محترم، بیت start تنها برای یک کلاک لازم است که یک باشد. پس از یک کلاک مقادیر ورودی در ثبات‌ها ریخته شده و تا انتهای محاسبه تغییر نمی‌کند. در انتها نیز خروجی نهایی در ثباتی ریخته میشود و نمایش داده می‌شود.

درصورتی که قبل از اینکه محاسبات تمام شود، سیگنال start مجددا یک شود، خروجی مدار معتبر نخواهد بود و در نتیجه پس از پایان عملیات سیگنال end که برای اعتبارسنجی خروجی نیز استفاده میشود، یک نخواهد شد

اما همانطور که پیشتر گفته شد، سیگنال calc\_end یک میشود تا متوجه شویم که کار مدار تمام شده است. همچنین توجه داریم که خروجی های نهایی در ثباتی ریخته میشود و تا پایان عملیات بعدی مقدار آن تغییر نمی‌کند.

1. الگوریتم

برای انجام عملیات جمع و تفریق بین دو عدد ممیز شناور، مراحل زیر طی می‌شود.

* **برابر کردن توان دو عدد ورودی:** در ابتدا توان دو عدد ورودی باید برابر شود. برای آنکه خطای محاسبه کمتر شود، عددی که توان کوچکتری دارد را انتخاب کرده و توان آن را تاجایی افزایش میدهیم که دو عدد هم توان شوند. برای افزایش توان عدد کوچکتر، هربار یک واحد مانتیس را به راست شیفت میدهیم و توان را یک واحد زیاد میکنیم. اینکار چندین کلاک طول میکشد.
* **جمع/تفریق دو عدد با درنظر گرفتن علامت آن‌ها:** جمع و تفریق دو عدد به صورت sign magnitude انجام میشود. در این روش، علامت حاصل محاسبه به صورت جداگانه محاسبه میشود و مقدار عدد خروجی نیز جداگانه محاسبه می‌شود. برای انجام عملیات حالات جدول زیر درنظر گرفته می‌شود. در واقع میتوان گفت، درصورتی که پس از اثر دادن علامت عملیات، دو عدد هم علامت شدند، مقدار مانتیس ها باهم جمع میشود و در غیر اینصورت مانتیس بزرگتر منهای مانتیس کوچکتر میشود. این مقایسه نیز بدون در نظر گرفتن علامت است. همچنین علامت عدد خروجی نیز برابر علامت عددی است که مانتیس بزرگتری دارد.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| توضیحات | علامت خروجی | وضعیت مانتیسها | جمع/تفریق | علامت B | علامت A |
| |A|+|B| | + |  | + | B>0 | A>0 |
| |A|-|B| | + | |A|>|B| | - | B>0 | A> 0 |
| |B|-|A| | - | |A|<|B| | - | B>0 | A>0 |
| |A|+|B| | + |  | - | B<0 | A>0 |
| |A|-|B| | + | |A|>|B| | + | B<0 | A>0 |
| |B|-|A| | - | |A|<|B| | + | B<0 | A>0 |
| |A|+|B| | - |  | - | B>0 | A<0 |
| |A|-|B| | - | |A|>|B| | + | B>0 | A<0 |
| |B|-|A| | + | |A|<|B| | + | B>0 | A<0 |
| |A|+|B| | - |  | + | B<0 | A<0 |
| |A|-|B| | - | |A|>|B| | - | B<0 | A<0 |
| |B|-|A| | + | |A|<|B| | - | B<0 | A<0 |

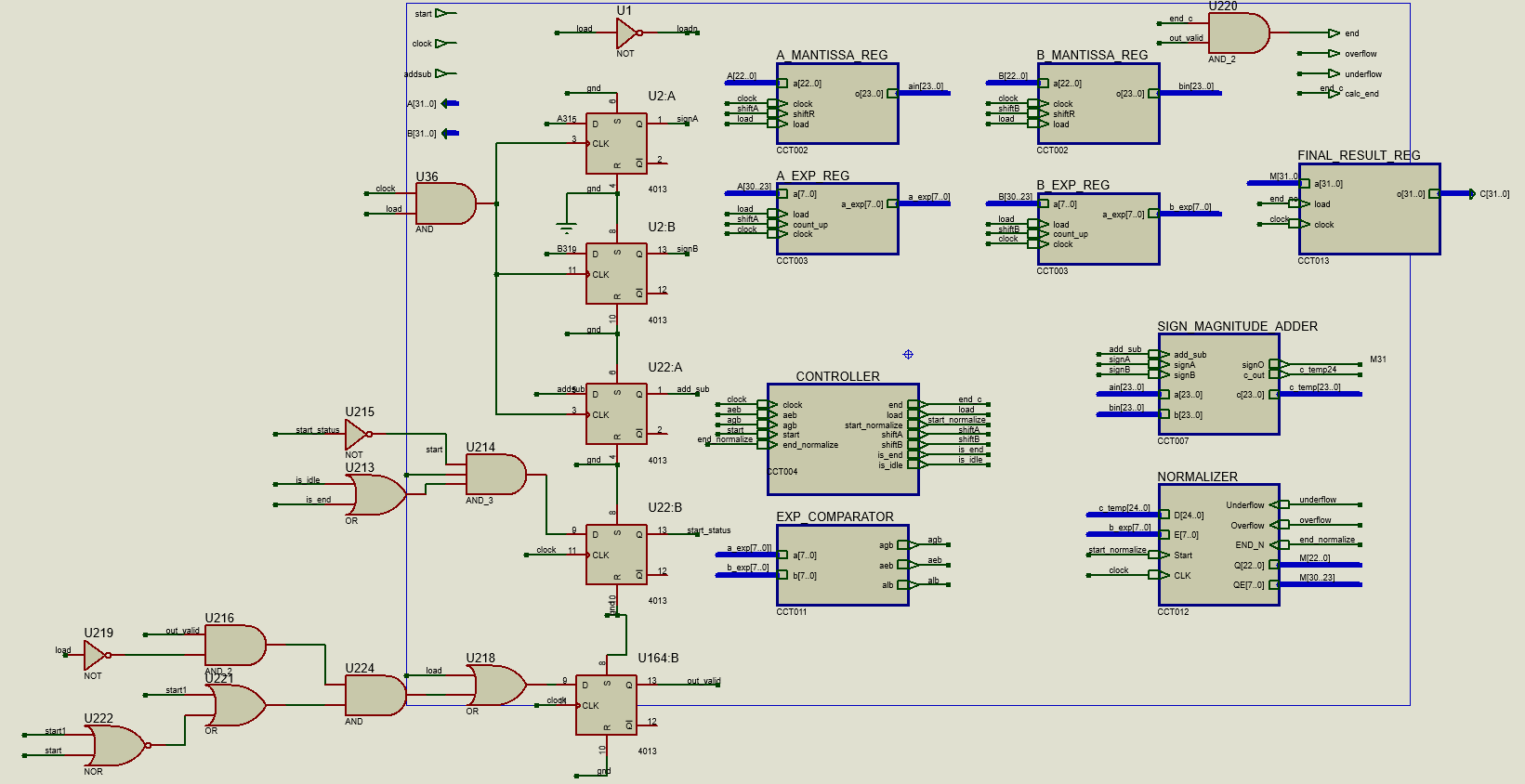
جدول 1

* **نرمال کردن پاسخ نهایی:** در این بخش پاسخ حاصل از دو بخش قبلی باید نرمال‌سازی شود. نرمال کردن عدد به این معناست که در بخش صحیح عدد تنها یک عدد 1 وجود داشته باشد. برای اینکار با شیفت دادن عدد به سمت راست و یا چپ، خروجی را نرمال میکنیم. پاسخ نهایی محاسبه، خروجی بخش نرمالایزر خواهد بود.

1. ماژول‌های استفاده شده

به منظور سادگی در طراحی و همینطور اشکال‌زدایی،‌ این آزمایش را به صورت سلسله مراتبی انجام می‌دهیم. در ادامه تمامی بلوک‌ها را شرح می‌دهیم.

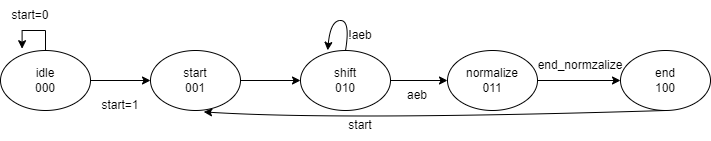
* 1. نمای کلی مدار

در شکل 1 نمای کلی مدار قابل مشاهده است. این نما صرفا جهت آشنایی اولیه با ماژولهای مدار داده شده است.

شکل 1

* 1. Controller : تولید بیت‌های کنترلی مدار

این ماژول وظیفه تولید بیت‌های کنترلی تمام مدار را برعهده دارد و مطابق با transition diagram شکل 2 کار میکند.



شکل 2

ورودی‌ها:

* Agb: این بیت حاصل مقایسه توان دو عدد ورودی است. در صورتی که توان A>B باشد مقدار آن یک و در غیر اینصورت صفر است.
* Aeb: درصورتی که توان A و B برابر باشد، این بیت برابر یک و در غیر اینصورت برابر صفر است.
* Clock: کلاک کل مدار است.
* Start: این بیت، متصل به start ورودی کلی مدار اصلی است.
* End\_normalize: این سیگنال از طرف ماژول نرمالایزر تولید میشود و یک بودن آن به این معنی است که محاسبه نرمالایزر تمام شده و خروجی نهایی آماده است.

خروجی‌ها:

* End: بیانگر تمام شدن محاسبات مدار است.
* Load: هنگامی که مدار باید شروع به کار کند، این سیگنال برای یک کلاک مقدار 1 را میگیرد تا تمامی رجیسترها مقادیر ورودی را ذخیره کنند تا درصورت تغییر ورودی در حین محاسبه، خروجی نهایی تغییر نکند.
* Start\_normalize: پس از آنکه عملیات برابر کردن توان دو عدد تمام شد، این سیگنال برای یک کلاک 1 میشود تا نرمالایزر فعالیت خود را شروع کند.
* shiftA: این بیت خروجی، در هنگام برابر کردن توان دو عدد و درصورتی که در استیت shift باشیم به کار می آید. درصورتی که توان A از B کوچکتر باشد این بیت فعال میشود.
* shiftB: مشابه shiftA است با این فرث که درصورتی که توان B از A کوچکتر باشد یک میشود.
* Is\_idle: درصورتی که در استیت idle باشیم و مدار شروع به فعالیت نکرده باشد این بیت یک است.
* Id\_end: در صورتی که در استیت end باشیم این بیت یک میشود.

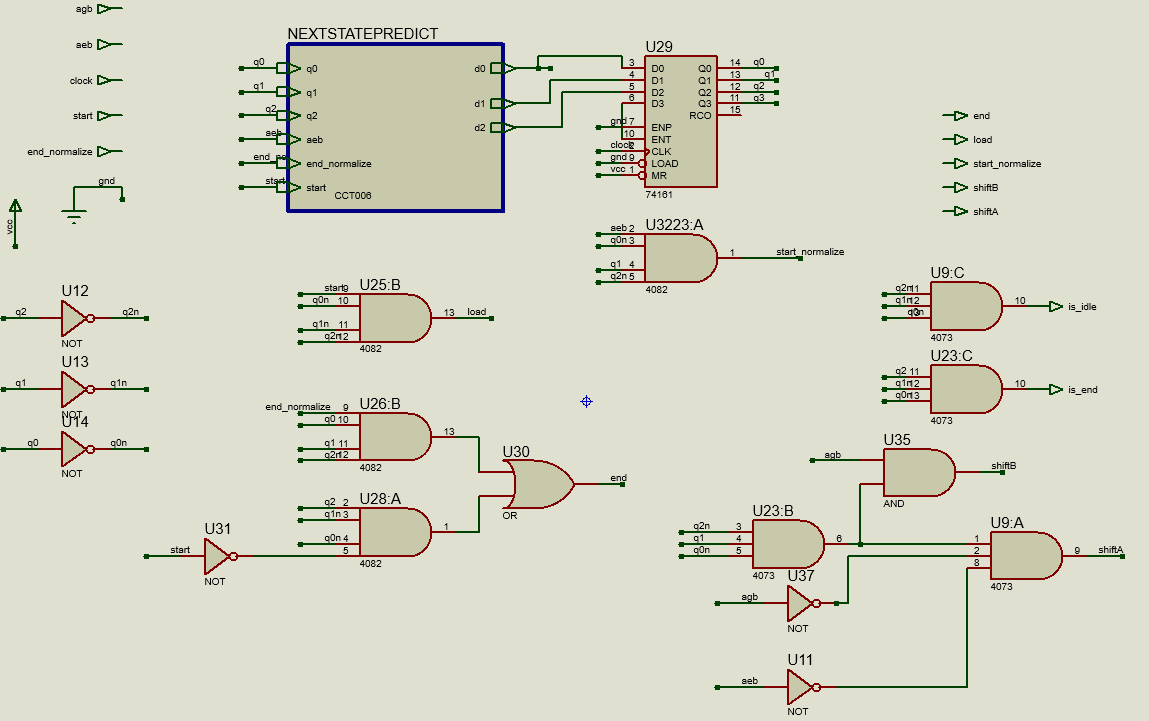
برای طراحی دیاگرام شکل 2 از یک ثبات 3 بیتی استفاده شده است. به اینصورت که شماره استیت را نگه میدارد و در هر کلاک بسته به ورودی ها و حالت فعلی، حالت بعدی را توسط NEXTSTATEPREDICT محاسبه میکند.

بیت های خروجی و حالات بعدی مطابق جدول زیر در نظر گرفته شده است.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | Next state | | |  |  |  |  | Present state | | |
| shiftB | shiftA | Start\_normalizer | end | load | Q0 | Q1 | Q2 | End\_normalize | start | agb | aeb | Q0 | Q1 | Q2 |
| X | X | 0 | 0 | 1 | 1 | 0 | 0 | X | 1 | X | X | 0 | 0 | 0 |
| X | X | 0 | 0 | 0 | 0 | 0 | 0 | x | 0 | X | X | 0 | 0 | 0 |
| X | X | 0 | 0 | 0 | 0 | 1 | 0 | X | X | X | X | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | X | X | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | X | X | 0 | 0 | 0 | 1 | 0 |
| X | x | 1 | 0 | 0 | 1 | 1 | 0 | X | X | X | 1 | 0 | 1 | 0 |
| X | X | 1 | 0 | 0 | 1 | 1 | 0 | 0 | X | X | X | 1 | 1 | 0 |
| X | X | 0 | 1 | 0 | 0 | 0 | 1 | 1 | X | X | X | 1 | 1 | 0 |
| X | X | 0 | 0 | 0 | 0 | 0 | 0 | X | 1 | X | X | 0 | 0 | 1 |
| x | X | 0 | 1 | 0 | 0 | 0 | 1 | X | 0 | x | x | 0 | 0 | 1 |

جدول 2

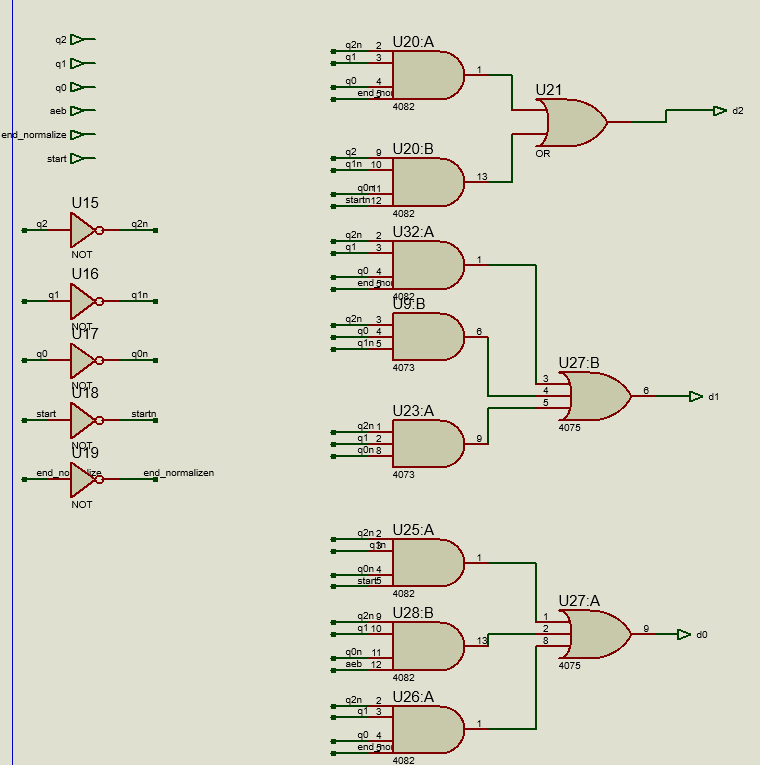
بخش محاسبه بیت‌های خروجی توسط خود این ماژول و بخش محاسبه حالت بعدی توسط ماژول NEXTSTATEPREDICT تولید میشود.



شکل 3

شکل 3 نمای اصلی این ماژول را نشان می‌دهد که مطابق با جدول 2 طراحی شده است.

همچنین ماژول NEXTSTATEPREDICT نیز که تنها از گیت‌های ساده استفاده کرده است در شکل 4 قابل مشاهده است.

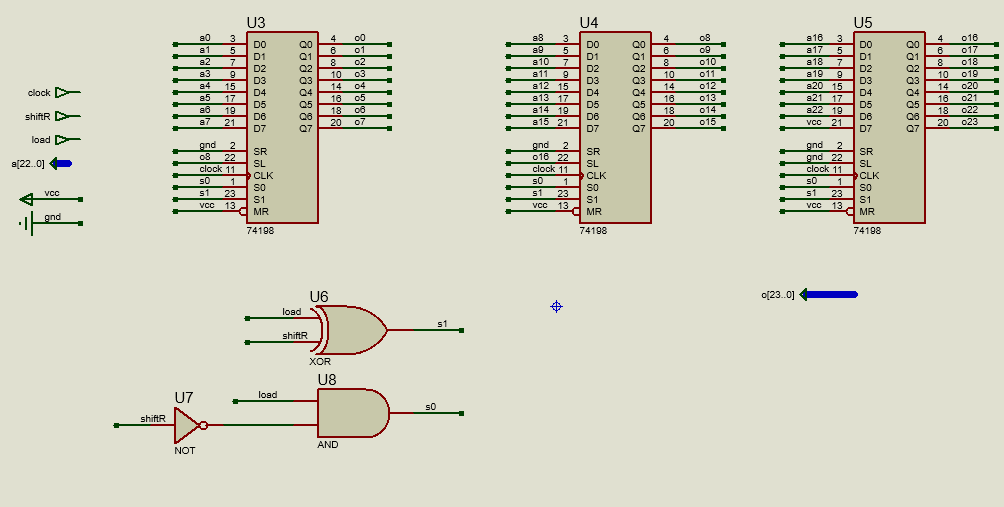


شکل 4

* 1. Mantissa\_reg :‌ ثبات نگه‌داری مقادیر مانتیس

دو ثبات برای نگه‌داری بخش مانتیس عدد ورودی طراحی شده است که به ترتیب A\_MANTISSA\_REG و B\_MANTISSA\_REG نام دارند و مطابق شکل 5 طراحی شده اند. این دو ماژول ورودی 23 بیتی را میگیرند و خروجی آن‌ها 24 بیتی است. درصورتی که ورودی load=1 باشد، بارگذاری موازی انجام میشود و درصورتی که shiftR فعال باشد، مقدار قبلی رجیسترها یک واحد به راست شیفت میخورد. همچنین درصورتی که هردو مقدار load و shiftR صفر باشند، مقدار ثباتها هیچ تغییری نمیکند.

درحالت بارگذاری موازی، بیت 24 ام که همان msb است، مقدار 1 را به خود میگیرد. این یک به دلیل آن اضافه میشود که اعداد ورودی نرمال هستند و در سمت چپ آن ها یک بخش صحیح که مقدار آن یک است وجود دارد.



شکل 5

در این ماژول از ثبات‌های 8 بیتی استفاده شده است. این دو ثبات توسط بیت‌های ورودی s1 و s0 کنترل میشوند و در لبه بالارونده کلاک، عملیات مشخص شده را به صورت زیر انجام میدهند:

* S1=S0=0: حفظ مقدار قبلی رجیستر
* S1=S0=0: بارگذاری موازی
* S1=1,S0=0: شیفت به راست

این دو بیت کنترلی توسط گیتهای پایه تولید میشود.

* 1. Exp\_reg: نگه‌داری توان عددها

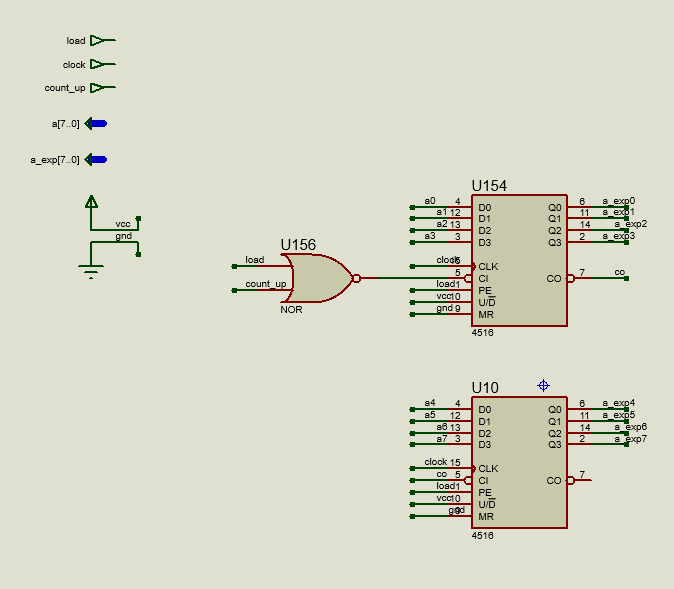
دو ثبات برای نگه‌داری و شیفت توان‌های دو ورودی طراحی شده است. این دو ماژول به نام‌های A\_EXP\_REG و B\_EXP\_REG در ماژول اصلی قرار دارند.

ورودی‌ها:

* A: 8 بیت است که همان توان های دو عدد است.
* Load: مقدار آن توسط کنترلر داده میشود و درصورتی که یک باشد، مقادیر ورودی در ثباتها ذخیره میشوند.
* Count\_up: مقدار آن توسط کنترلر تولید میشود و درصورتی که یک باشد، مقادیر ثبات‌ها یک واحد افزایش می‌یابد. همانطور که پیشتر نیز ذکر شد،‌ در مرحله shift ممکن است این بیت یک شود و برای ثباتی یک میشود که مقدار تون کوچکتری دارد.
* Clock: کلاک اصلی مدار است.

خروجی‌ها:

* A\_exp: خروجی ۸ بیتی که بیانگر مقدار فعلی ثبات‌ها است.



شکل 6

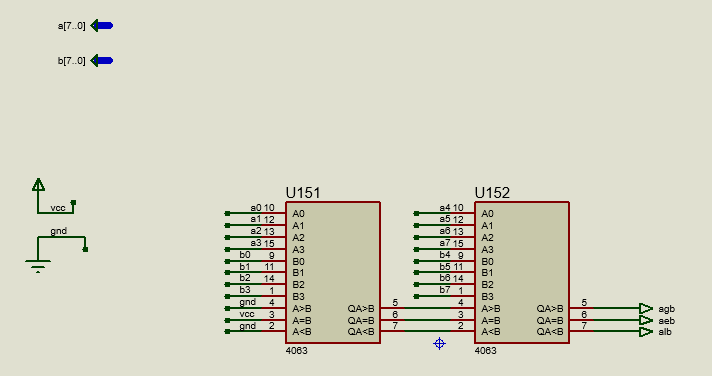
همانطور که در شکل ۶ نیز قابل مشاهده است‌، برای نگه‌داری ۸ بیت توان از دو ثبات با شماره ماژول 4516 استفاده شده است که به صورت سری به هم وصل شده اند. این ثبات‌ها قابلیت شمارش رو به بالا و پایین دارند. درصورتی که PE=0 باشد و CI=0 باشد شمارش رو به بالا (یک واحد) انجام میشود.

مقادیر این ماژول در دو استیت shift و start از کنترلر تغییر میکند.

* 1. Exp\_comparator: مقایسه توان‌ها

این ماژول یک ماژول ترکیبی است و با گرفتن دو ورودی ۸ بیتی a و b مشخص میکند که کدام یک از این دو بزرگتر هستند. مقایسه این دو عدد بدون در نظر گرفتن علامت آن ها است و دو عدد unsigned درنظر گرفته میشوند. درصورتی که a>b باشد خروجی agb فعال میشود. در غیر اینصورت اگر دو عدد برابر باشند، aeb یک شده و در غیر اینصورت alb یک میشود.

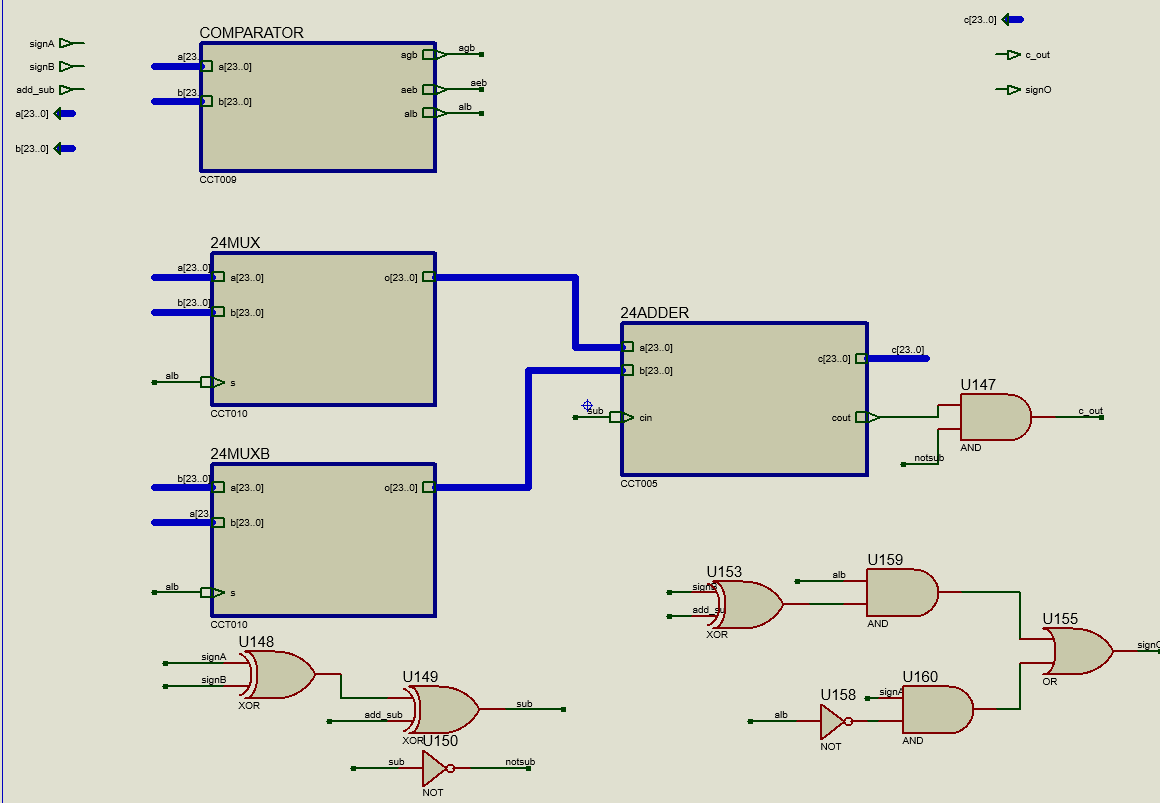
خروجی‌های این مدار، به عنوان ورودی به کنترلر داده میشود و همان بیت‌های aeb و agb را تشکیل میدهد که در بخش کنترلر به آن ها اشاره شده بود.



شکل 7

* 1. Sign\_magnitude\_adder: جمع/تفریق کننده علامتدار مانتیس

این ماژول بخش اصلی محاسبات جمع و تفریق مدار را بر عهده دارد و به‌صورت ترکیبی کار میکند.



شکل 8

ورودی‌ها:

* A: ورودی ۲۴ بیتی اول
* B: ورودی ۲۴ بیتی دوم
* Add\_sub: درصورتی که مقدار آن یک باشد a-b و درغیراینصورت a+b محاسبه میشود.
* signA: علامت ورودی اول
* signB: علامت ورودی دوم

خروجی‌ها:

* C: خروجی حاصل نهایی است.
* C\_out کری تولید شده است.
* signO: علامت نهایی خروجی است.

در این ماژول دو عدد ورودی a و b گرفته میشود و باتوجه به علامت آن ها و نوع عملگر، خروجی محاسبه می‌شود. همانطور که در بخش الگوریتم توضیح داده شد، محاسبات طبق جدول ۲ انجام میشود. به اینصورت که ابتدا یک مقایسه بدون علامت بین دو عدد انجام میشود. همچنین مشخص میشود که در انتها باید بین a و b تفریق انجام شود یا جمع. درصورتی که علامت دو عدد برابر باشد، نوع عملگر برابر با همان ورودی add\_sub است. اما در صورتی که علامت ها یکی نباشد، نقیض add\_sub باید عملگر نهایی باشد.

همچنین طبق توضیحات ابتدایی، درصورتی که قرار باشد تفریق انجام شود،‌ عدد بزرگتر باید منهای عدد کوچکتر شود. در نتیجه از دو مالتی پلکسر ۲ به ۱ استفاده میکنیم که ورودی های جمع کننده/تفریق کننده بدون علامت را تشکیل میدهند و عددی را به عنوان ورودی اول به جمع کننده میدهند که بدون در نظر گرفتن علامت بزرگتر باشد. همچنین نوع عملیات مورد نظر نیز به جمع کننده 24ADDER داده میشود و این ماژول حاصل جمع یا تفریق را محاسبه میکند. این ماژول در بخش بعدی به طور دقیق مورد بررسی قرار میگیرد.

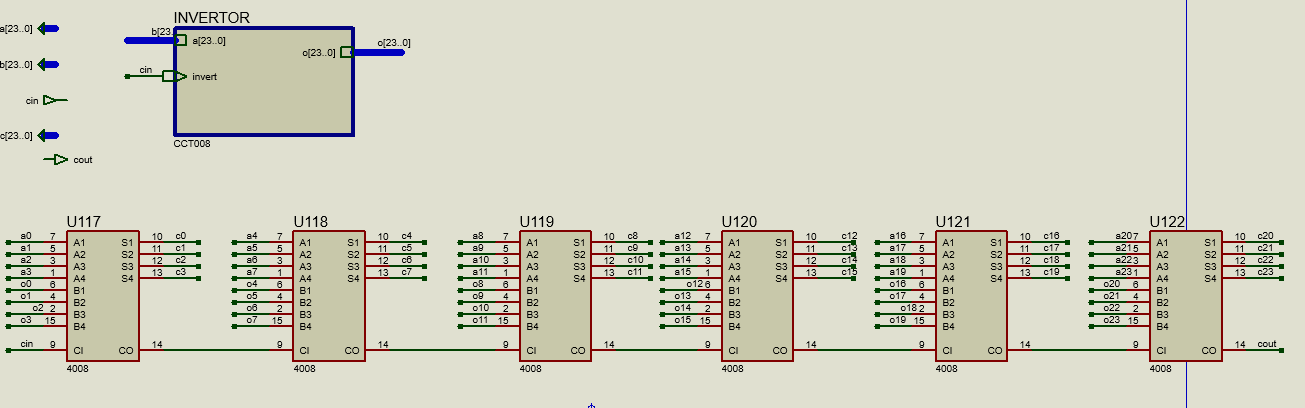
همچنین علامت نهایی خروجی طبق جدول ۲ از رابطه زیر به دست می آید.

((signB XOR add\_sub).alb) +NOT(alb)signA = signO

لازم به ذکر است که ماژول COMPARATOR که در این بخش طراحی شده است دقیقا مشابه با مقایسه کننده توان‌ها است که در بخش ۵-۴ توصیف شد. همچنین 24MUX یک مالتی پلکسر ساده است که از قرار دادن ۲۴ مالتی پلکسر ۲ به ۱ تک بیتی درست شده است و پیچیدگی ندارد.

* 1. 24Adder: جمع/تفریق کننده ۲۴ بیتی

این ماژول دو ورودی ۲۴ بیتی a و b را میگیرد. همچنین cin را نیز دریافت میکند. درصورتی که cin یک باشد، عملیات a-b را انجام می‌دهد و در غیر اینصورت عملیات a+b را انجام می‌دهد. این جمع کننده از اتصال سری ۴ جمع کننده ۸ بیتی آماده 4008 طراحی شده است.



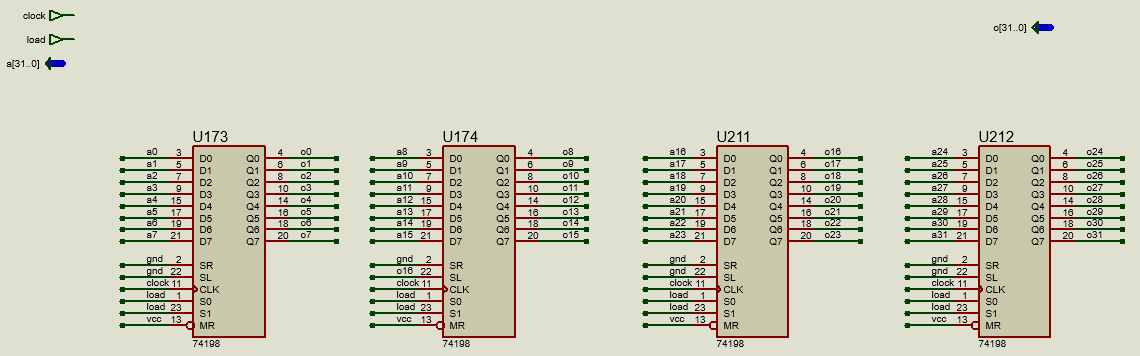
شکل 9

برای آنکه تفریق انجام شود، ابتدا تمام بیت های ورودی b باید نقیض شود. در نتیجه در ماژول INVERTOR تمام بیت‌ها با cin ایکس‌اور شده و درصورتی که عملیت تفریق باشد، نقیض میشوند. سپس مقدار cin به عنوان کری ورودی جمع کننده بیتهای ۷ تا ۰ داده میشود تا درصورتی که عملیات تفریق بود، مکمل ۲ عدد دوم با a جمع شود و به این صورت تفریق انجام شود.

* 1. Final\_result\_reg: ثبات نگه‌داری حاصل نهایی

این ماژول، درصورتی مقدار ۳۲ بیتی ورودی را در ثبات‌ها لود میکند که کار ماژول normalizer تمام شده باشد و سیگنال end\_normalize را تولید کرده باشد. در غیر اینصورت مقدار قبلی خود را نگه میدارد. خروجی ۳۲ بیتی این ماژول همواره جواب درست محاسبه را نمایش میدهد.

باتوجه به آنکه خواسته شده بود که درصورتی که start در میانه محاسبه نیز صفر شد، خروجی محاسبه در انتها نمایش داده شود،‌ این ماژول حتی در صورتی که ورودی ها invalid باشد (start در وسط کار ۱ شده باشد) نیز خروجی را نمایش میدهد.



شکل 10

شکل ۱۰،‌ طراحی این ماژول را نشان میدهد که پیچیدگی خاصی ندارد و دقیقا مشابه ثبات‌های قبلی است.

* 1. ماژول آماده 4013

این ماژول یک D-flipflop است. دیتاشیت این ماژول در این [لینک](https://www.ti.com/lit/ds/symlink/cd4013b.pdf) قابل مشاهده است.

* 1. ماژول آماده 74198

این ماژول یک Universal Shift register می‌باشد. دیتاشیت این ماژول در این [لینک](https://susta.cz/fel/74/pdf/sn_74198.pdf) قابل مشاهده است.

* 1. ماژول آماده 4512

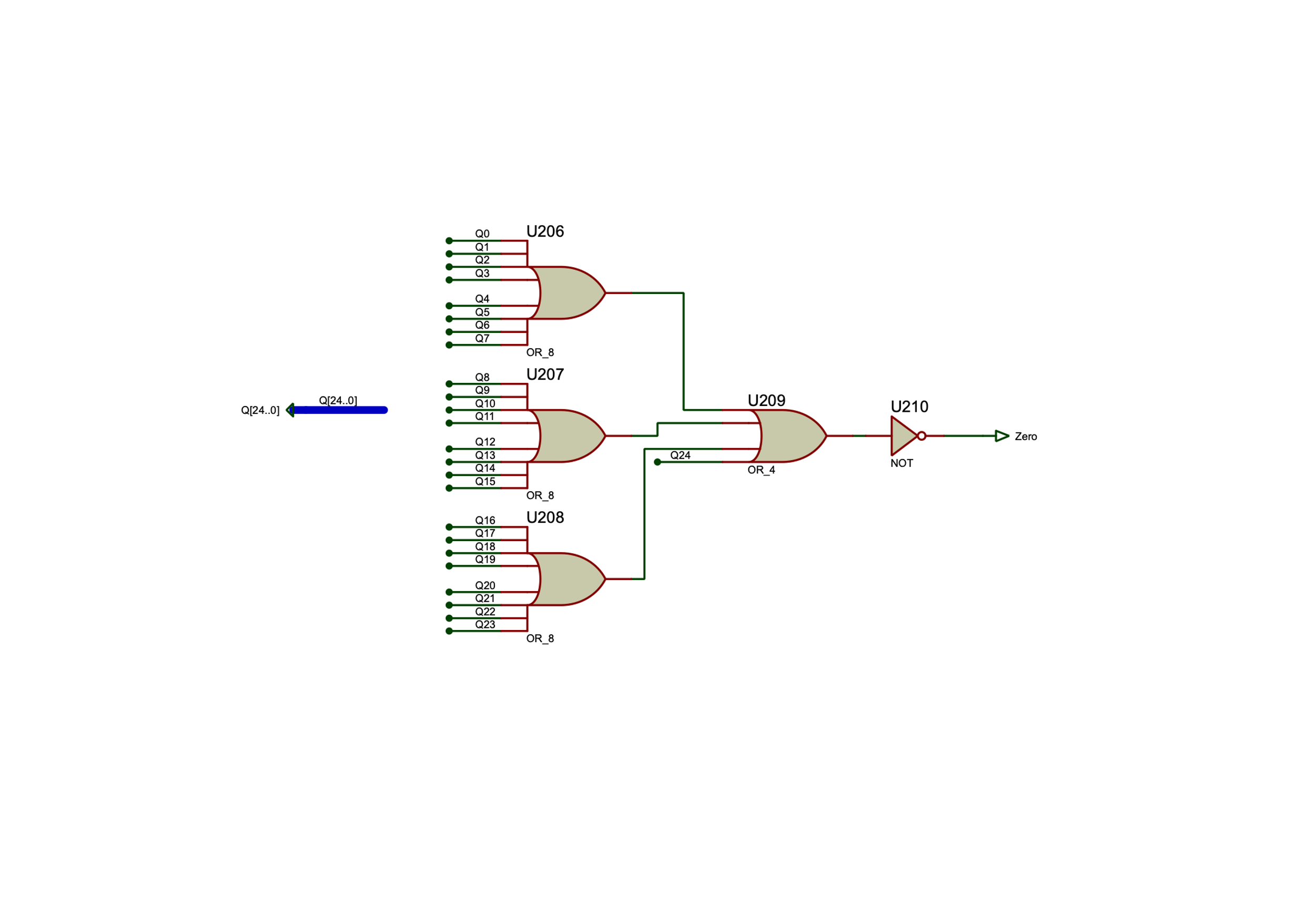
این ماژول یک مالتی‌پلکسر 8 ورودی می‌باشد. دیتاشیت این ماژول در این [لینک](https://www.ti.com/lit/ds/symlink/cd4512b.pdf) قابل مشاهده است.

* 1. ماژول آماده 4516

این ماژول یک بالارونده و پایین‌رونده 4 بیتی با قابلیت بارگذاری موازی می‌باشد. دیتاشیت این ماژول در این [لینک](https://www.ti.com/lit/ds/symlink/cd4516b.pdf) قابل مشاهده است.

* 1. تمام صفر(Zero)

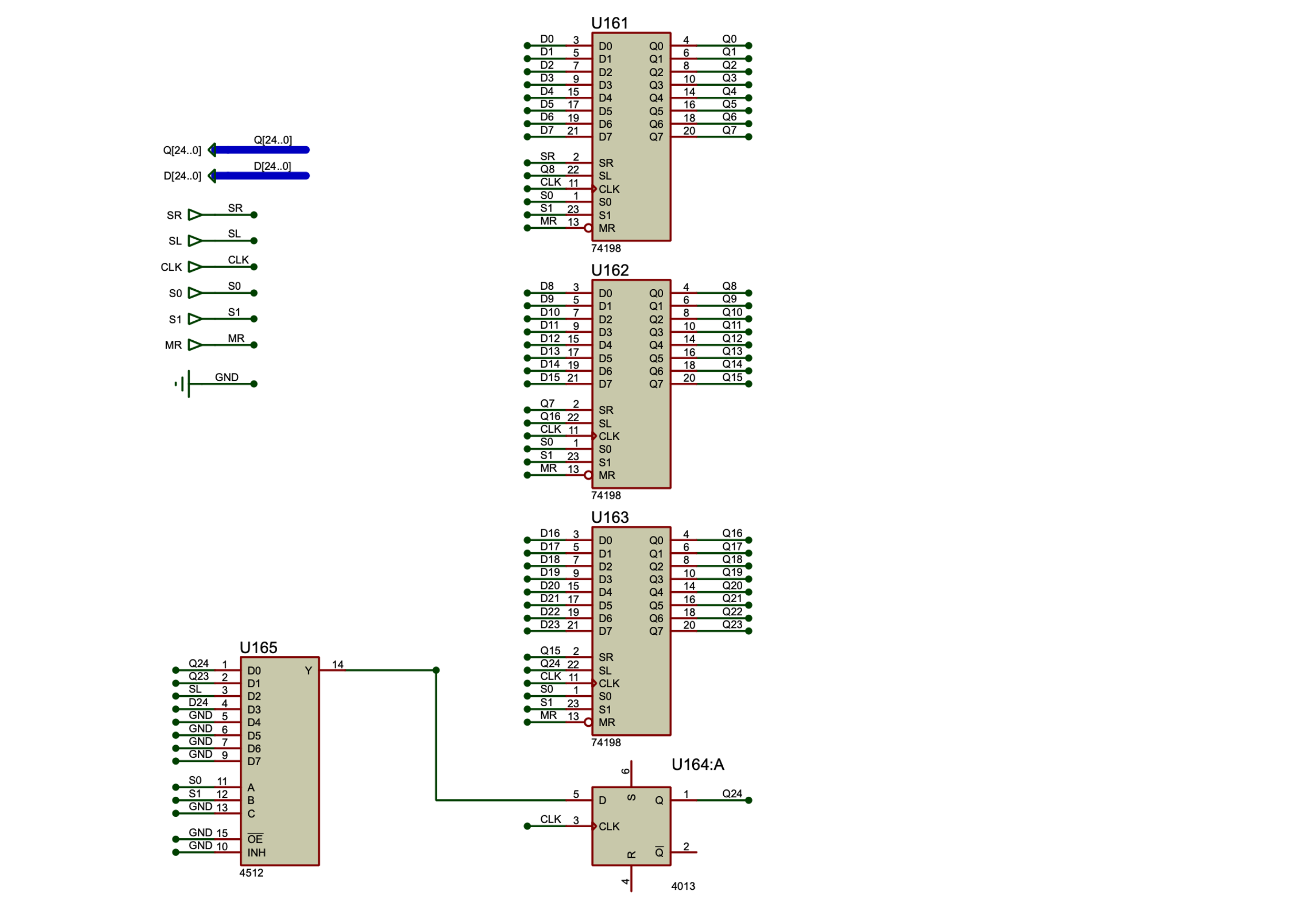
این بلوک مطابق شکل 11 طراحی شده است. این بلوک زمانی که تمامی بیت‌های ورودی 25 بیتی آن برابر با 0 باشد، خروجی 1 می‌دهد و در بقیه حالت‌ها خروجی آن برابر با 0 می‌باشد.



شکل 11

* 1. شیفت رجیستر 25 بیتی(SHIFT REG)

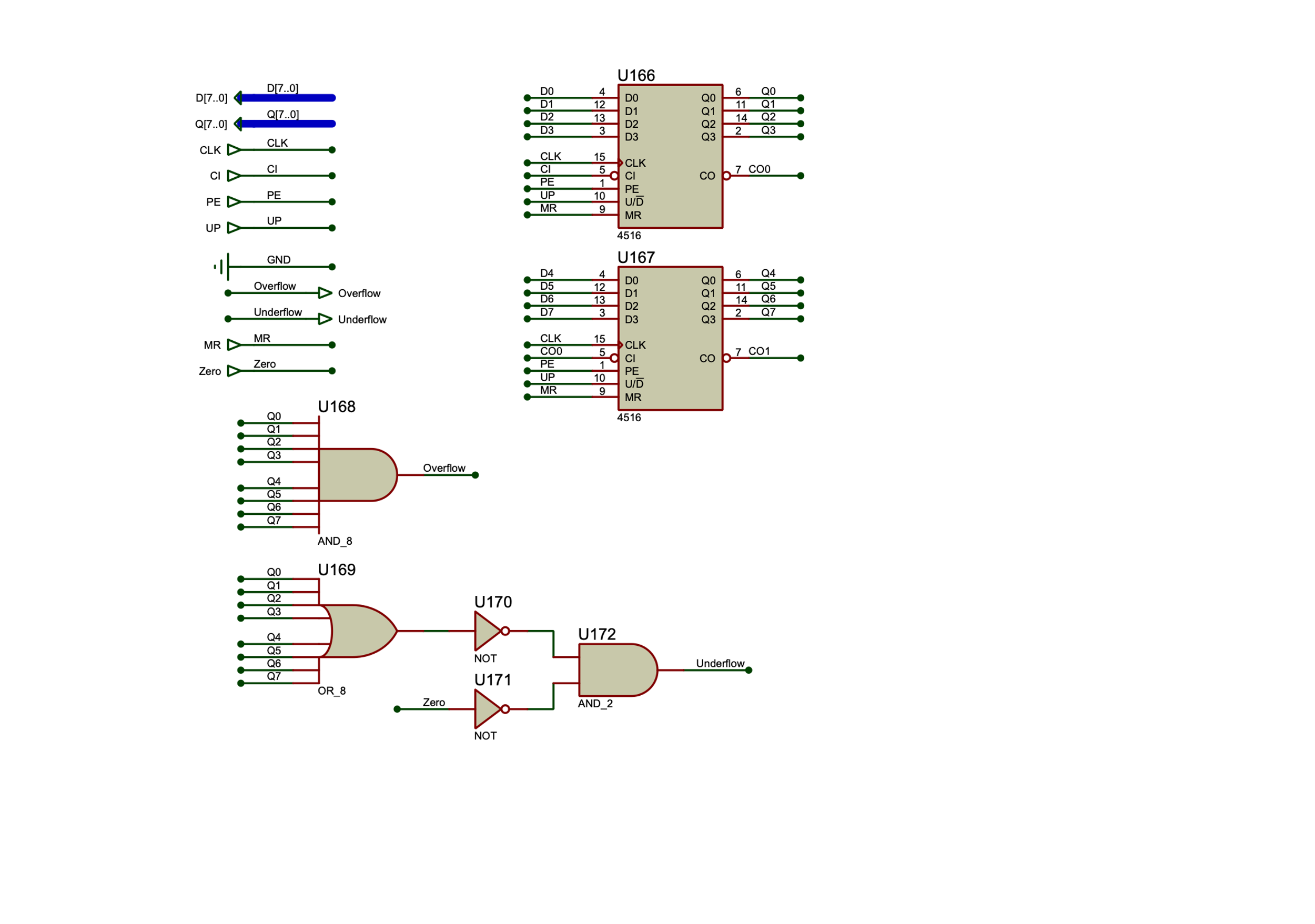
بلوک شیفت رجیستر مطابق شکل 12 طراحی شده است. ورودی‌های این بلوک SR، SL، S0، S1، CLK، MR و D[24..0] و خروجی این بلوک Q[24..0] است. ورودی‌های این بلوک مطابق با ماژول آماده 74198 طراحی شده است. تنها تفاوت این بلوک و ماژول آماده 74198 در ورودی موازی 25 بیتی است که در ماژول آماده 74198، این ورودی 8 بیتی می‌باشد همچنین در خروجی نیز، خروجی این بلوک 25 بیتی می‌باشد و در ماژول آماده 74198 این خروجی 8 بیتی می‌باشد. این بلوک از cascade کردن 3 ماژول آماده 74198 و یک ماژول آماده 4013 تشکیل شده است. ورودی‌ این ماژول آماده 4013 از خروجی یک ماژول آماده 4512 گرفته شده است.



شکل 12

* 1. بالارونده و پایین‌رونده(INC\_DEC)

این بلوک مطابق شکل 13 طراحی شده است. ورودی‌های این بلوک E[7..0]، CLK، CI، PE،‌ UP، MR و Zero می‌باشد. خروجی‌های این بلوک نیز Q[7..0]، Underflow و Overflow می‌باشد. ورودی‌های CLK، CI، PE، UP و MR این بلوک به ترتیب مانند ورودی‌های CLK، CI، PE، U/D و MR ماژول آماده 4516 عمل می‌کند. ورودی E[7..0] مقدار exponent حاصل را به عنوان ورودی می‌گیرد. ورودی Zero نیز در زمانی 1 است که نتیجه حاصل 0 شده باشد و در بقیه حالات 0 می‌باشد. خروجی Q[7..0] مقدار نهایی exponent را نشان می‌دهد. خروجی Overflow زمانی 1 می باشد که تمامی بیت‌های Q برابر با یک باشد و در بقیه حالت‌ها 0 می‌باشد. خروجی Underflow نیز زمانی 1 می‌باشد که تمام بیت‌های Q برابر با 0 باشد و مقدار Zero هم برابر با 1 باشد و در بقیه حالت‌ها 0 می‌باشد.



شکل 13

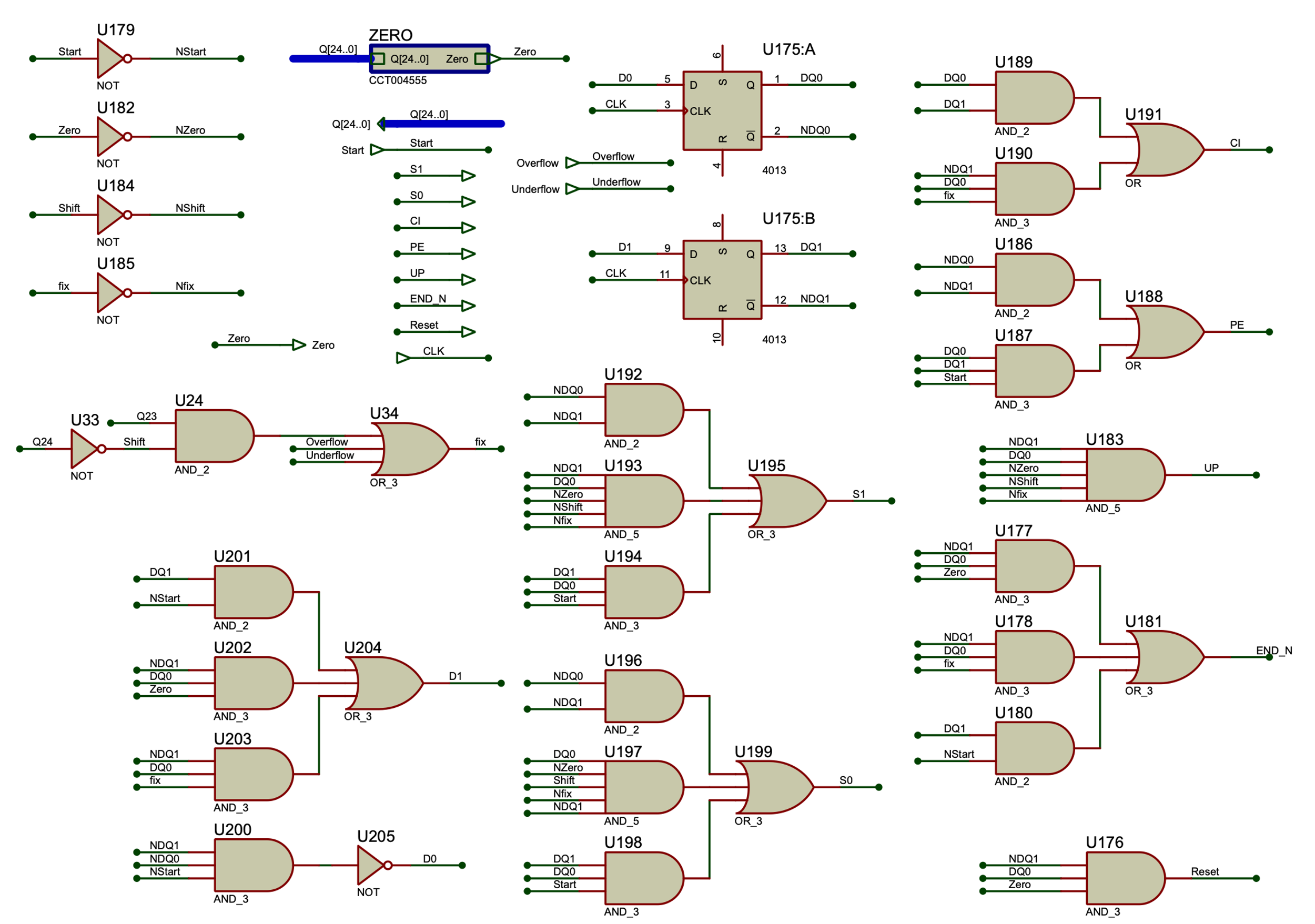
* 1. واحد کنترل Normalizer(CONTROL)

این بلوک مطابق شکل 14 طراحی شده است. ورودی‌های این بلوک Q[24..0]، Overflow، Underflow، Start و CLK می‌باشد. خروجی‌های این بلوک S1، S0، CI، PE، UP، Reset، END\_N و Zero است. Q[24..0] مقدار Significand حاصل را نگه می‌دارد. مقدار Zero زمانی برابر با 1 می‌باشد که تمامی بیت‌های Q برابر با 0 باشند و در بقیه حالت‌ها برابر با 0 می‌باشد. خروجی‌های S0 و S1 ورودی‌های حالت S0 و S1 بلوک SHIFT REG می‌باشند. خروجی‌های CI، PE، UP و Reset نیز به ترتیب ورودی‌های Zero، CI، PE، UP و MR بلوک INC\_DEC می‌باشند. خروجی END\_N نیز زمانی 1 می‌شود که عملیات Normalize کردن به پایان برسد. این بلوک واحد کنترل‌کننده بلوک Normalizer می‌باشد. این بلوک را بر اساس جدول حالت طراحی می‌کنیم. این جدول حالت در جدول 3 نشان داده شده است. در ورودی‌های این جدول از Shift و fix استفاده شده است. زمانی که Significand حاصل نیاز به شیفت به راست دارد مقدار Shift=0 است و زمانی که نیاز به شیفت به چپ دارد مقدار Shift=1 است. همینطور زمانی که Significand نیاز به هیچ شیفتی ندارد مقدار fix=1 می‌باشد و در بقیه حالت‌ها مقدار fix=0 می‌باشد. با توجه به این نکات مقدار Shift=Q24’ و fix=Q24’.Q23 می‌باشد. برای این بلوک 3 حالت مختلف در نظر می‌گیریم. این حالت‌ها به شرح زیر است:

* Q1Q0=00: این حالت زمانی است که تا به حال از Normalizer استفاده نشده و منتظر حالتی است که Start=1 بشود.
* Q1Q0=01: این حالت زمانی است که ورودی را گرفته و در حال Shift دادن Significand است.
* Q1Q0=11: این حالت زمانی است که ورودی نرمالایز شده است و منتظر دریافت Start=1 برای نرمالایز کردن ورودی جدید است.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Present State | | Inputs | | | | Next State | | Outputs | | | | | | |
| Q1 | Q0 | Zero | Shift | fix | Start | Q1 | Q0 | S1 | S0 | CI | PE | UP | END | RESET |
| 0 | 0 | X | X | X | 0 | 0 | 0 | X | X | X | X | X | 0 | 0 |
| 0 | 0 | X | X | X | 1 | 0 | 1 | 1 | 1 | X | 1 | X | 0 | 0 |
| 0 | 1 | 1 | X | X | X | 1 | 1 | 0 | 0 | X | X | X | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | X | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | X | 1 | X | 1 | 1 | 0 | 0 | 1 | 0 | X | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | X | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | X | X | X | X | X | X | X | X | X | X | X | X | X |
| 1 | 1 | X | X | X | 0 | 1 | 1 | 0 | 0 | 1 | 0 | X | 1 | 0 |
| 1 | 1 | X | X | X | 1 | 0 | 1 | 1 | 1 | X | 1 | X | 0 | 0 |

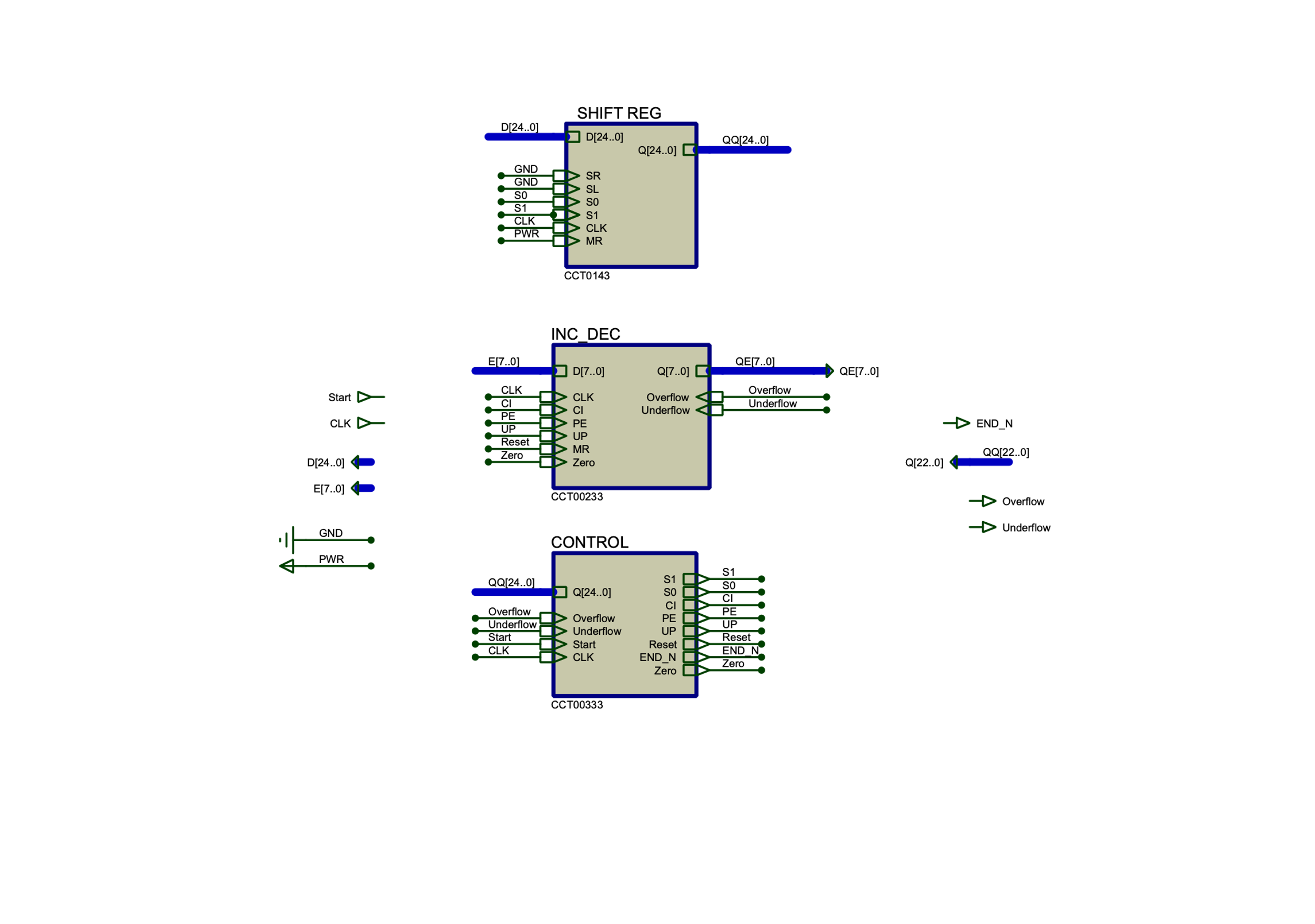
جدول 3



شکل 14

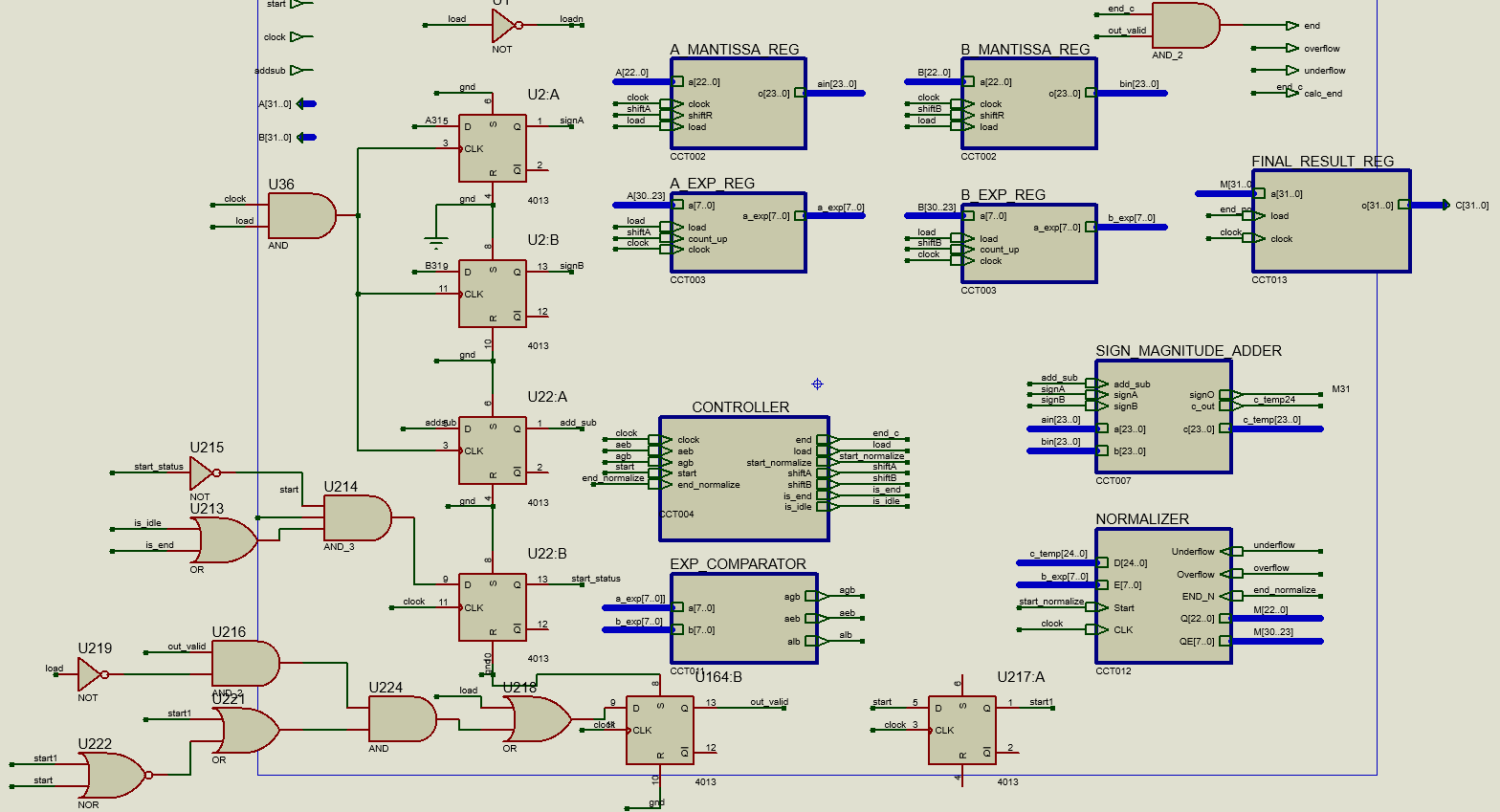
* 1. Normalizer: نرمال‌سازی خروجی نهایی

این بلوک مطابق شکل 15 طراحی شده است. ورودی‌های بلوک CLK، Start، D[24..0]، E[7..0] و خروجی‌های این بلوک END\_N، QE[7..0]، Q[22..0]، Underflow و Overflow می‌باشد. این بلوک Start را از واحد کنترل دریافت و مقادیر D[24..0] که مقدار حاصل Significand و E[7..0] که مقدار Exponent می‌باشد را ذخیره می‌کند. سپس آن را نرمالایز می‌کند و مقدار Fraction که برابر با Q[22..0] می‌باشد را به عنوان Fraction نرمالایز شده و QE[7..0] را به عنوان Exponent نرمالایز شده خروجی می‌دهد. زمانی که نرمالایز شده باشد مقدار END\_N یک می‌شود و به خروجی فرستاده می‌شود.



شکل 15

1. ساختار مدار



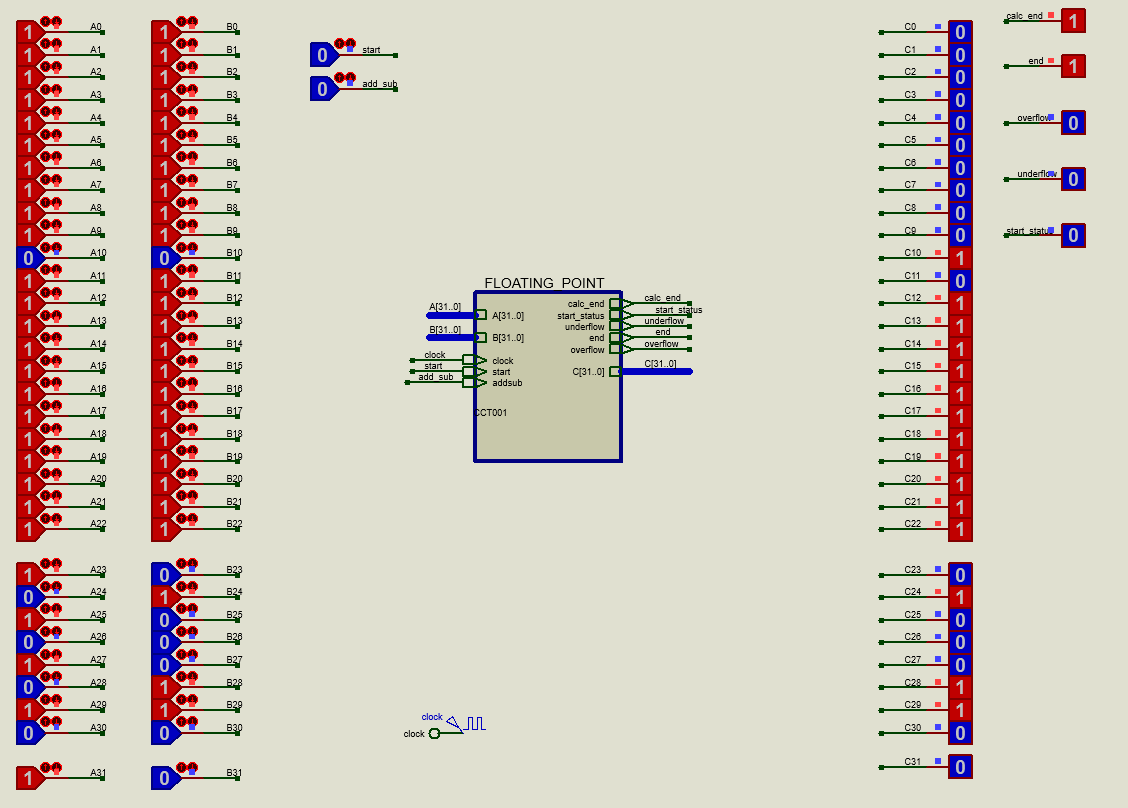
شکل 16

ساختار مدار مطابق تصویر بالا است. پس از اولین کلاک که سیگنال start=1 شود، مدار شروع به کار میکند. پس از سیگنال start مقدار صفر را میگیرد. مقدار موثر سیگنال start در خروجی start\_status نمایش داده میشود. در واقع تنهای start در حالتی در روند مدار موثر است که مدار در آن لحظه در حال انجام محاسبه نباشد. همچنین برای مشخص کردن اینکه خروجی معتبر است باید در نظر داشته باشیم که اگر سیگنال start در بین محاسبه مجددا یک شود، اعتبار خروجی از بین میرود. در نتیجه،‌ با سیگنال out\_valid معتبر بودن خروجی نهایی سنجیده میشود. به اینصورت که اولین کلاکی که در حین محاسبه سیگنال start دوباره یک شود، از ولید بودن در می آید. در نتیجه رجیستر تک بیتی U217:A قرار داده شده است که در هر کلاک، مقدار فعلی و قبلی start را داشته باشیم. در ابتدا نیز out\_valid را همزمان با اینکه سیگنال load فعال میشود یک میکنیم. پس از آن درصورتی که start یکبار صفر شود و سپس یک شود، out\_valid=0 شده و خروجی نهایی دیگر مورد قبول نیست و پس از پایان محاسبه تنها calc\_end مقدار یک را میگیرد و end=1 نمیشود.

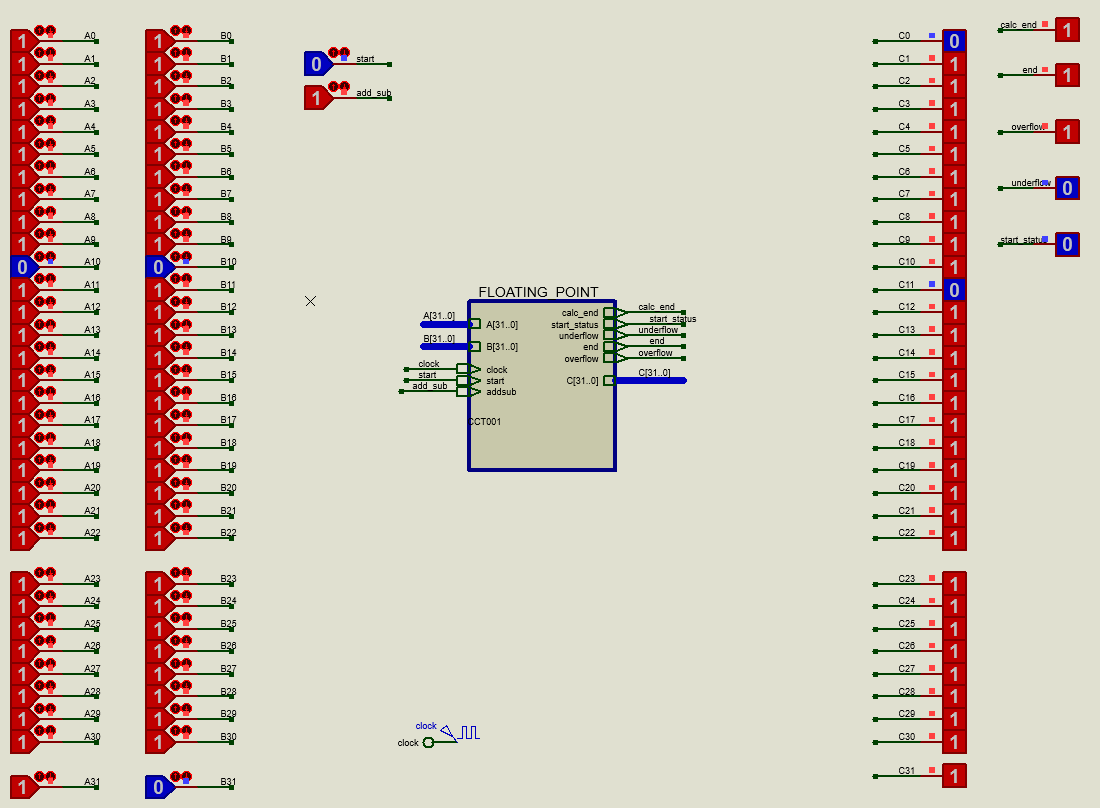
دررابطه با روند کلی مدار، میتوان گفت که در ابتدا به اندازه یک کلاک load فعال شده و تمامی ورودیها در ثباتها ریخته میشود. (برای علامت دو عدد و add\_sub نیز دو قبات تک بیتی در نظر گرفته شده است). سپس در مرحله محاسبه، ابتدا دو عدد تا جایی شیفت میخورند که بیت aeb مدار یک شود.(در این مرحله به صورت ترکیبی حاصل جمع مانتیس ها نیز محاسبه شده است)‌. پس از آن بیت start\_normalize از سمت کنترلر به normalizer میرود. نرمالایزر عدد را نرمالایز میکند و با استفاده از سیگنال end\_normalize به کنترلر اعلام میکند که عدد نهایی محاسبه شده است. در گام نهایی و کلاک آخر خروجی نرمالایزر در ثبات ریخته شده و درصورتی که out\_valid یک باشد، end نیز ۱ میشود.

1. خروجی نهایی مدار

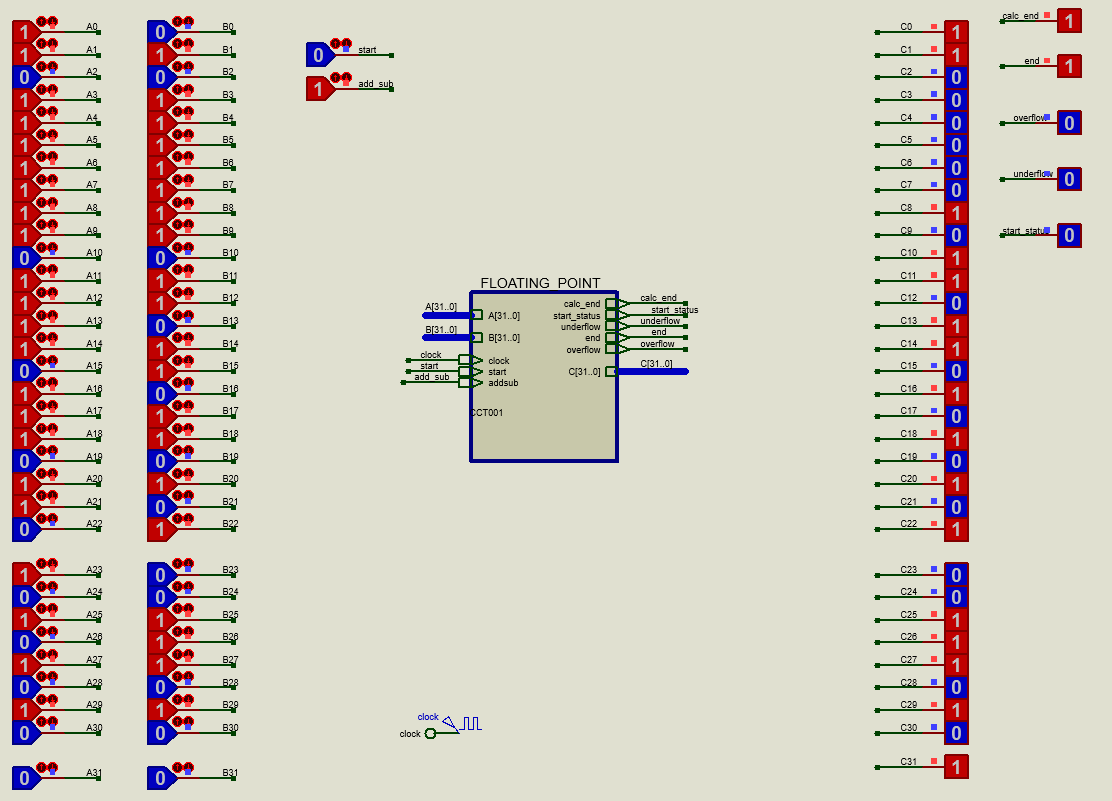
در این بخش خروجی مدار برای حالات مختلف ورودی بررسی شده است.

* 

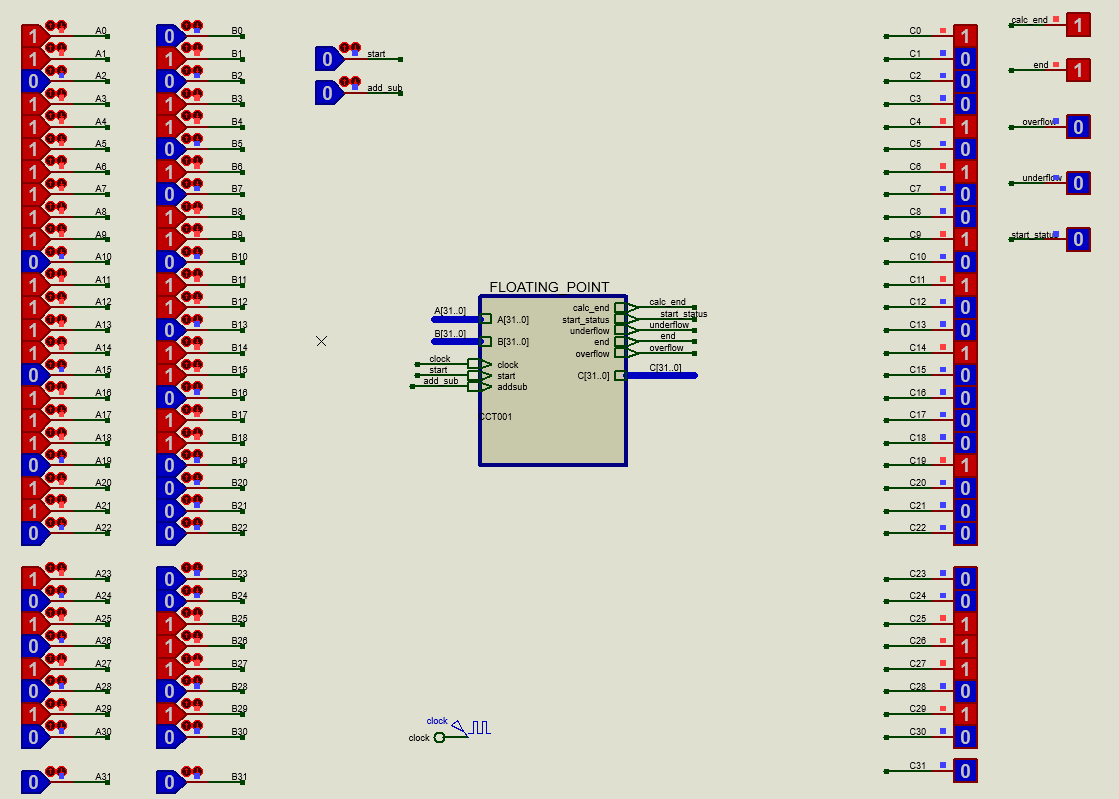
شکل 17

* 

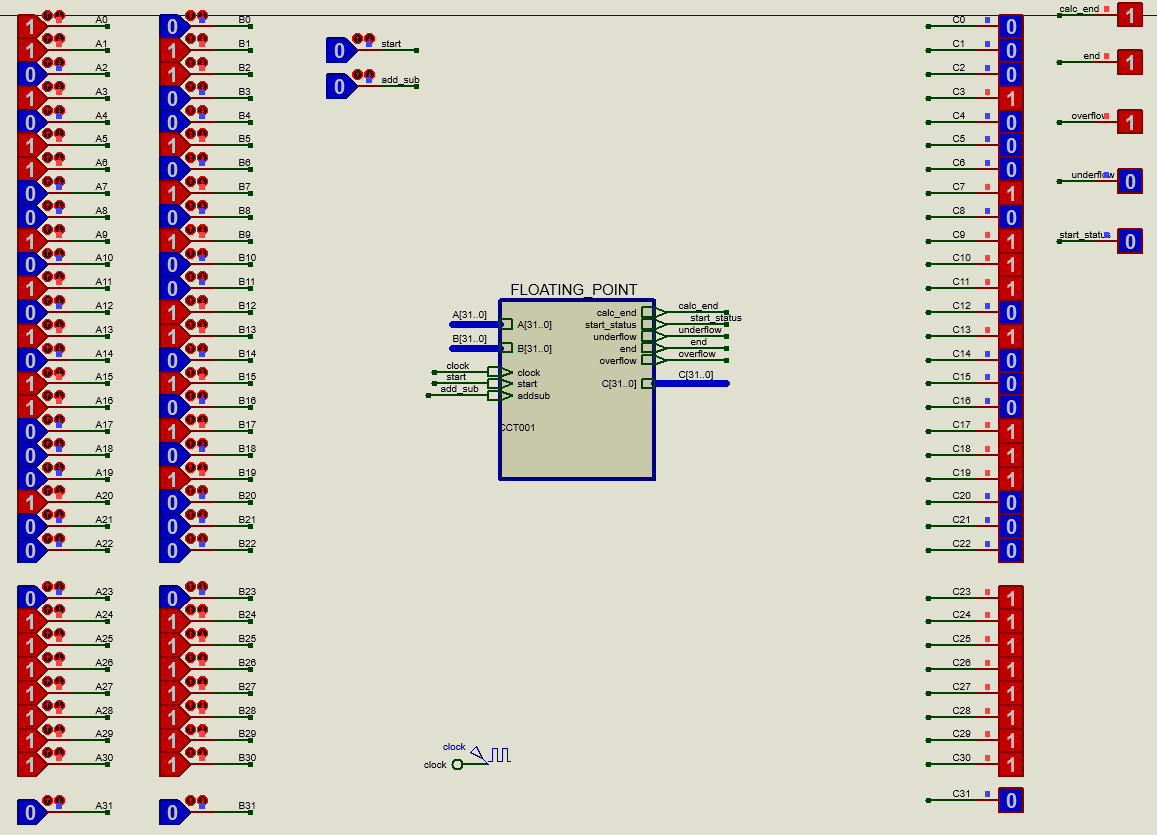
شکل 18

* 

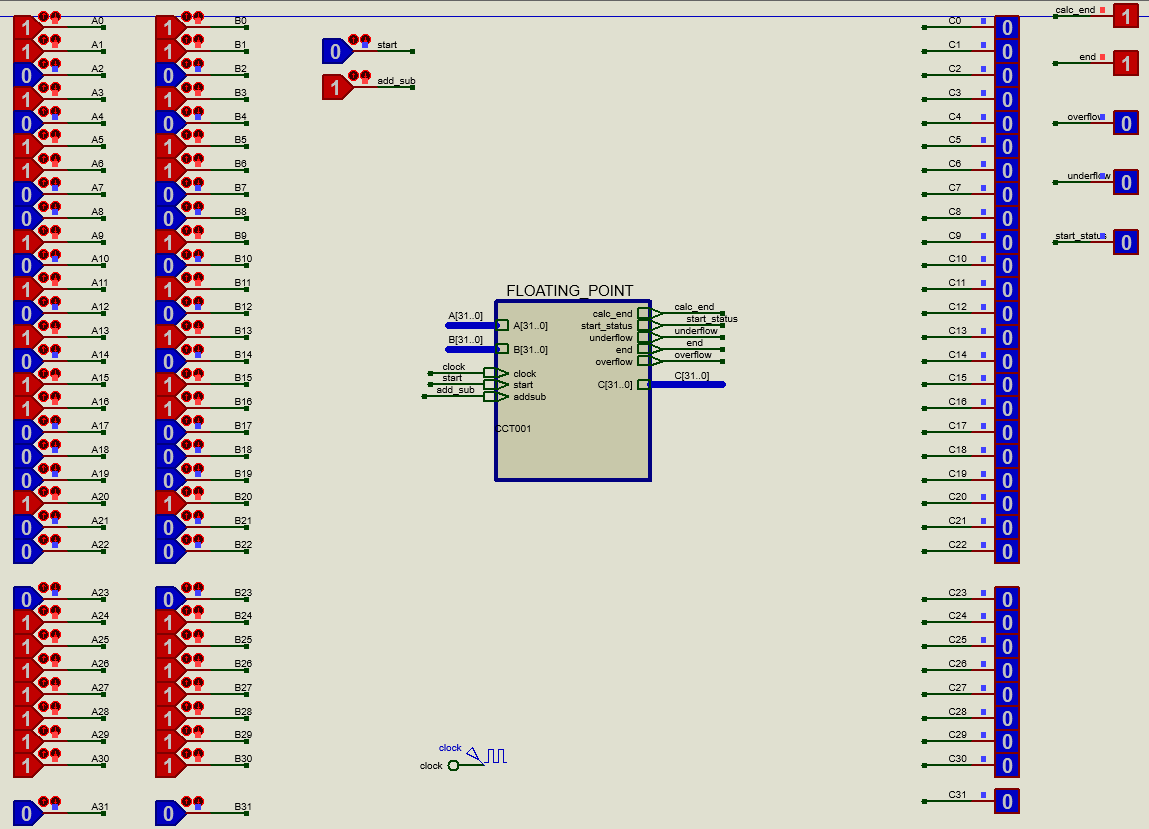
شکل 19

* 

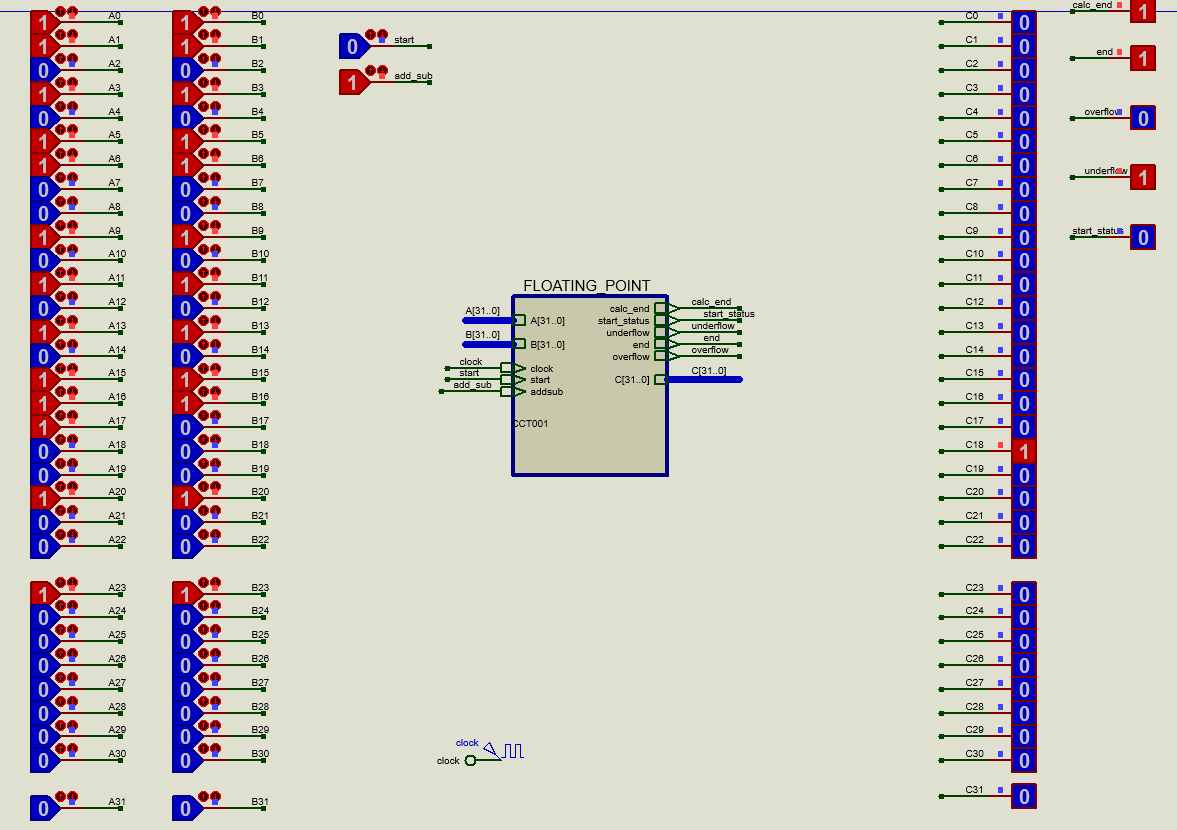
شکل 20

* حالت overflow

شکل 21

* صفر شدن خروجی

شکل 22

* Underflow

شکل 23