طراحی 4-bit ALU

محمد جواد باورصاد

استاد : دکتر سید وحید میر مقتدایی

نیمسال دوم سال تحصیلی ۹۶-۹۵

انتشار اینترنتی : تیر ماه ۹۶

فهرست

چکیده

واحد محاسبه و منطق (Arithmetic & Logic Unit) به اختصار (ALU)، مداری دیجیتالی است که عملیات حساب و منطق را انجام می‌دهد و یک قطعه اساسی از واحد پردازش مرکزی در کامپیوتر است. در این پروژه شماتیک و Layout یک ALU ساده که هشت عمل دیجیتالی مختلف را بر روی دو عدد چهار بیتی A و B انجام می‌دهد، طراحی شد و مورد آزمایش قرار گرفت. مدار طوری طراحی شده است که به صورت سنکرون و PipeLine عمل کند و در هر پالس ساعت یک داده‌ی خروجی داشته باشیم. حداکثر فرکانس کاری این ALU در بدترین حالت ممکن ۵۱۵ MHz به دست آمد و Layout این مدار مساحتی حدود mm^2 ۰/۱۱ اشغال کرد.

مقدمه

در طول سال‌های گذشته معمولا کاهش ابعاد مدارهای الکترونیکی و افزایش توان قابل حمل آن‌ها یکی از اهداف مهندسین الکترونیک بوده است. چنین آرمانی همواره عامل رشد سیستم‌هایی با تجمع بسیار زیاد یا به اختصار VLSI (Very Large Scale Integration) جهت پردازش سیگنال‌های دیجیتال خصوصاً در زمینه-هایی که سرعت بالا از مهم‌ترین عوامل می‌باشد، بوده است. در این راستا ترانزیستورهای تولید شده از اتصال فلز- اکسید- نیمه هادی MOS (Metal Oxide Semiconductor) از اهمیت به خصوصی برخوردار هستند. دو نوع ترانزیستور بسیار معروف NMOS وPMOS به صورت مجتمع در قالب کلی CMOS شناخته می‌شوند که در طراحی ICها کاربرد بسیاری دارند. تعداد بسیار زیادی از ترانزیستورها در یک مدار به صورت همزمان و در مساحت بسیار کم ساخته می‌شوند که برای انجام عمل دیجیتالی مشخصی طراحی شده اند. یکی از مداراتی که معمولا جزئی از یک مداربزرگ‌تر است و به صورت VLSI تولید می‌شود، ALU است. واحد محاسبه و منطق (Arithmetic & Logic Unit) به اختصار (ALU)، مداری دیجیتالی است که عملیات حساب و منطق را انجام می‌دهد و یک قطعه اساسی از واحد پردازش مرکزی در کامپیوتر است. و حتی ساده‌ترین میکروپردازنده‌ها نیز دارای یک واحد محاسبه و منطق برای کارهایی از قبیل نگهداری زمان هستند. پردازنده‌های موجود درپردازشگرهای (CPU) مدرن و و پردازنده‌های گرافیکی(GPU) داری واحد محاسبه و منطق قدرتمند ودر عین حال پیچیده‌ای هستند. ممکن است هر قطعه دارای بیش از یک واحد محاسبه و منطق باشد. جان فون نویمن ریاضی دان قرن بیستم مفهوم ALU را در سال ۱۹۴۵، هنگامی که در حال نوشتن گزارش برای کامپیوتر جدید خود به نام EDVAC بود، مطرح کرد. در این پروژه شماتیک و Layout یک ALU ساده که هشت عمل دیجیتالی مختلف را بر روی دو عدد چهار بیتی A و B انجام می‌دهد، طراحی شد و مورد آزمایش قرار گرفت.

طراحی

ALU مورد نظر ما باید بتواند هشت دستور زیر را انجام دهد:

۱) A

۲) A+۱

۳) A+B

۴) A-B

۵) $\overline{A}$

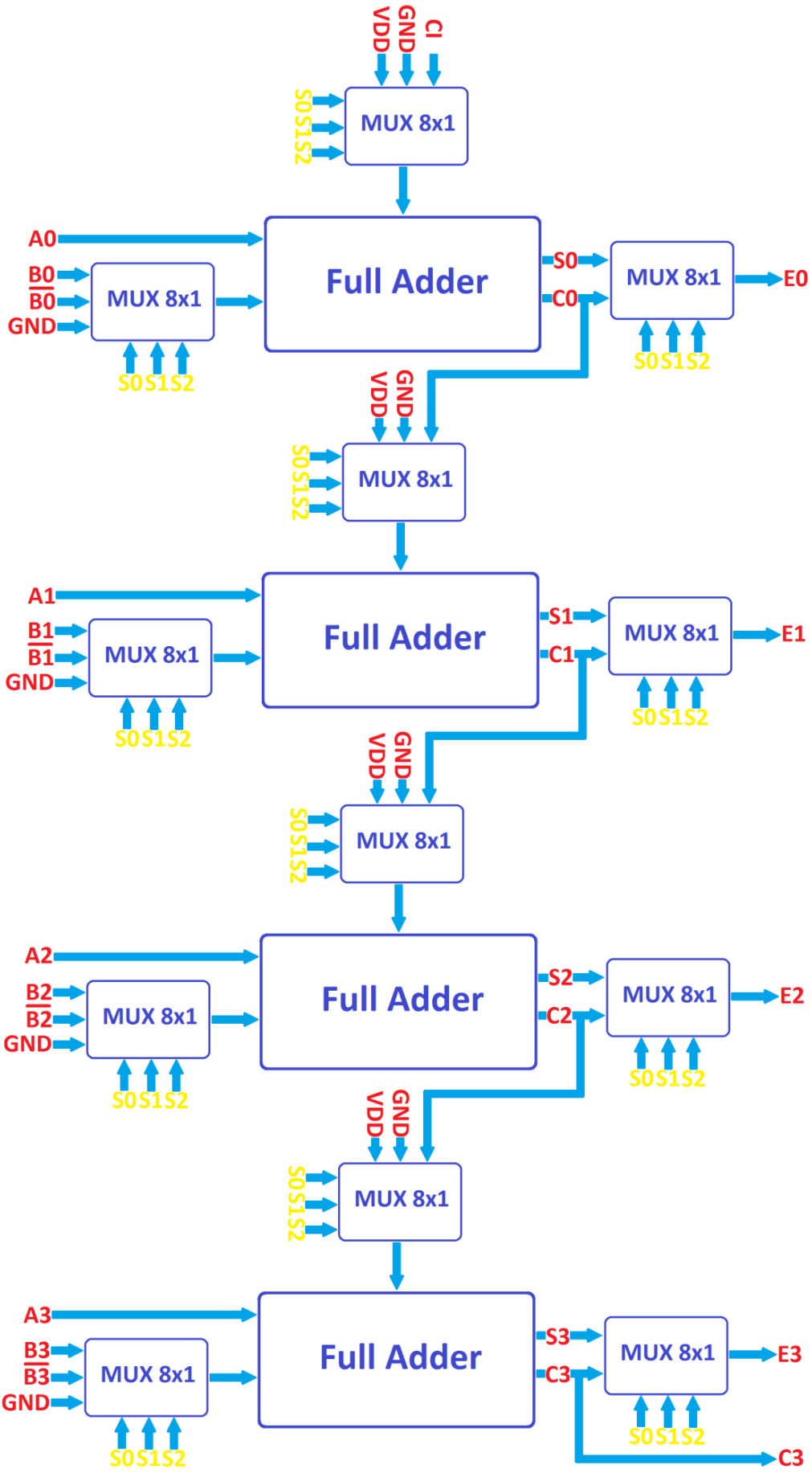
۶) A.B

۷) A$^{\vee}$B

۸) A XOR B

مدار ALU بسته به مقدار دستور وارد شده به آن باید یکی از دستورات بالا را انجام دهد. هر کدام از اعداد A و B در این طرح، ۴بیتی هستند اما مدار به سادگی قابلیت ارتقا به تعداد بیشتری بیت را دارد. در صورت افزایش تعداد ارقام اعداد A و B ، تاخیر تبدیل به مسئله‌ای مهم می‌شود زیرا هر جمع کننده نیاز به کری جمع کننده قبلی دارد و تا بلوک قبل محاسباتش را تمام نکند، محاسبات بلوک بعدی اعتباری ندارد. طرح پیشنهادی برای ساخت این ALU به این صورت است که ورودی‌ها (غیر از خود عدد A) ابتدا به یک مالتی پلکسر وارد شوند سپس مقدار دستور، مشخص می‌کند که کدام ورودی‌ها به جمع کننده وارد شوند و در واقع مشخص می‌کند که کدام عمل انجام شود. سه مالتی‌پلکسر برای تعیین ورودی B، CI وخروجی (S یا CO) در نظر گرفته شده است. بنابر این چون تعداد دستورات هشت عدد می‌باشد، ما به مالتی‌پلکسرهای هشت به یک نیاز داریم.به هر دستور بالا یک کد دودویی سه بیتی از صفر تا هفت به همان ترتیب بالا اختصاص داده شد که در جدول زیر آمده‌اند :

کد دستور به دودویی(S0 S1 S2)	کد دستور به ده دهی	دستور
۰۰۰	۰	A
۰۰۱	۱	A+1
۰۱۰	۲	A+B
۰۱۱	۳	A-B
۱۰۰	۴	$\overline{A}$
۱۰۱	۵	A.B
۱۱۰	۶	A$^{\vee}$B
۱۱۱	۷	A XOR B



شکل (۱) بلوک دیاگرام طرح اولیه‌ی ALU

ورودی‌های انتخاب هر مالتی‌پلکسر سه بیت به صورت S0 S1 S2 می‌باشند. ورودی های داده‌ی مالتی‌پلکسرها از D0 تا D7 طوری مقدار دهی می‌شوند که بسته به این که ورودی انتخاب مالتی‌پلکسرها که تعیین کننده نوع دستور می‌باشد چه مقداری باشد، عمل متناظر با آن دستور را اجرا کند. طرح اولیه‌ی این مدار به صورت بلوک-دیاگرام در شکل (۱) آمده است.

برای تعیین ورودی‌های مالتی‌پلکسرها بهتر است جدول درستی جمع‌کننده را رسم و با توجه به آن ورودی مالتی‌پلکسرها را مشخص کنیم. جدول درستی جمع کننده و تعدادی از دستوراتی که به راحتی نمی‌توان ورودی‌های مرتبط با آن‌ها را مشخص کرد در جدول زیر آمده است.

ورودی‌ها			خروجی		خروجی مورد انتظار برخی دستورات			
A	B	CI	S	CO	$\bar{A}$	A.B	A$^{\vee}$B	AXORB
0	0	0	0	0	1	0	0	0
0	0	1	1	0	1	0	0	0
0	1	0	1	0	1	0	1	1
0	1	1	0	1	1	0	1	1
1	0	0	1	0	0	0	1	1
1	0	1	0	1	0	0	1	1
1	1	0	0	1	0	1	1	0
1	1	1	1	1	0	1	1	0

۱)دستور شماره صفر (A)

در حالتی که $S_0S_1S_2$ برابر 000 باشد، باید خود عدد A به خروجی برود برای این کار کافی است ورودی D0 مالتی‌پلکسر اول و دوم، صفر (زمین) و خروجی مالتی‌پلکسر سوم از S گرفته شود.یعنی D0 ورودی مالتی-پلکسر سوم به خروجی S از Adder متصل باشد.

۲)دستور شماره یک (A+1)

در حالتی که $S_0S_1S_2$ برابر 001 باشد، عدد A باید با یک جمع شده و در خروجی ظاهر شود. برای این کار ورودی B را صفر و کری ورودی CI را برابر یک قرار می‌دهیم. بنابر این باید ورودی D1 مالتی‌پلکسر اول به زمین، و ورودی D1 مالتی‌پلکسر دوم و سوم به ترتیب به VDD و S متصل شوند.

۳)دستور شماره دو (A+B)

در حالتی که $S_0S_1S_2$ برابر 010 باشد، عدد A باید با عدد B جمع شده و در خروجی ظاهر شود. بنابر این باید ورودی D2 مالتی‌پلکسر اول به خود عدد B، و ورودی D2 مالتی‌پلکسر دوم و سوم به ترتیب به CI و S متصل شوند.

۴)دستور شماره سه (A-B)

در حالتی که $S_0S_1S_2$ برابر 011 باشد، عددB باید از A کم شده و در خروجی ظاهر شود. در دیجیتال برای انجام عمل تفریق ، مقدار B را متمم۲ کرده و با عدد A جمع می‌کنیم. بنابر این ورودی D3 مالتی‌پلکسر اول $\overline{B}$، مالتی‌پلکسر دوم VDD و مالتی‌پلکسر سوم S خواهد بود. به این ترتیب عدد B ابتدا متمم۱ (NOT) می‌شود و با یک جمع می‌شود که تبدیل به متمم۲ عدد B می‌شود سپس با عدد A جمع می‌شود. حاصل این عمل تفریق B از A را به دست می‌دهد. پس باید ورودی D3 مالتی‌پلکسرهای اول، دوم و سوم به ترتیب به $\overline{B}$، VDD و S متصل شوند.

۵)دستور شماره چهار ($\overline{A}$)

در حالتی که $S_0S_1S_2$ برابر 100 باشد، متمم عدد A باید در خروجی ایجاد شود. بنابر این مشخص است که نیازی به عدد B نداریم پس ورودی D4 مالتی‌پلکسر اول باید زمین شود. همان‌طور که از جدول درستی که قبلا آمده است بر می‌آید، در حالتی که CI یک و خروجی از S گرفته شود، $\overline{A}$ ایجاد می‌شود. پس ورودی D4 مالتی‌پلکسر دوم و سوم را به ترتیب VDD وS قرار می‌دهیم.

۶)دستور شماره پنج (A.B)

در حالتی که $S_0S_1S_2$ برابر 101 باشد، حاصل ضرب منطقی دو عدد A و B باید در خروجی ایجاد شود. واضح است که ورودی D5 مالتی‌پلکسر اول باید خود عدد B باشد. از بررسی جدول درستی جمع‌کننده در می‌یابیم که اگر CI را برابر صفر قرار دهیم و خروجی را از CO بگیریم نتیجه، حاصل ضرب منطقی A و B خواهد بود. بنابراین ورودی D5 مالتی‌پلکسر دوم و سوم را به ترتیب به GND و CO متصل می‌کنیم.

۷)دستور شماره شش (A$^{\vee}$B)

در حالتی که $S_0S_1S_2$ برابر 110 باشد، حاصل جمع منطقی دو عدد A و B باید در خروجی ایجاد شود. مشخص است که ورودی D6 مالتی‌پلکسر اول باید خود عدد B باشد. از بررسی جدول درستی جمع‌کننده در می‌یابیم که اگر CI را برابر یک قرار دهیم و خروجی را از CO بگیریم نتیجه، حاصل جمع منطقی A و B خواهد بود. بنابراین ورودی D6 مالتی‌پلکسر دوم و سوم را به ترتیب به VDD و CO متصل می‌کنیم.

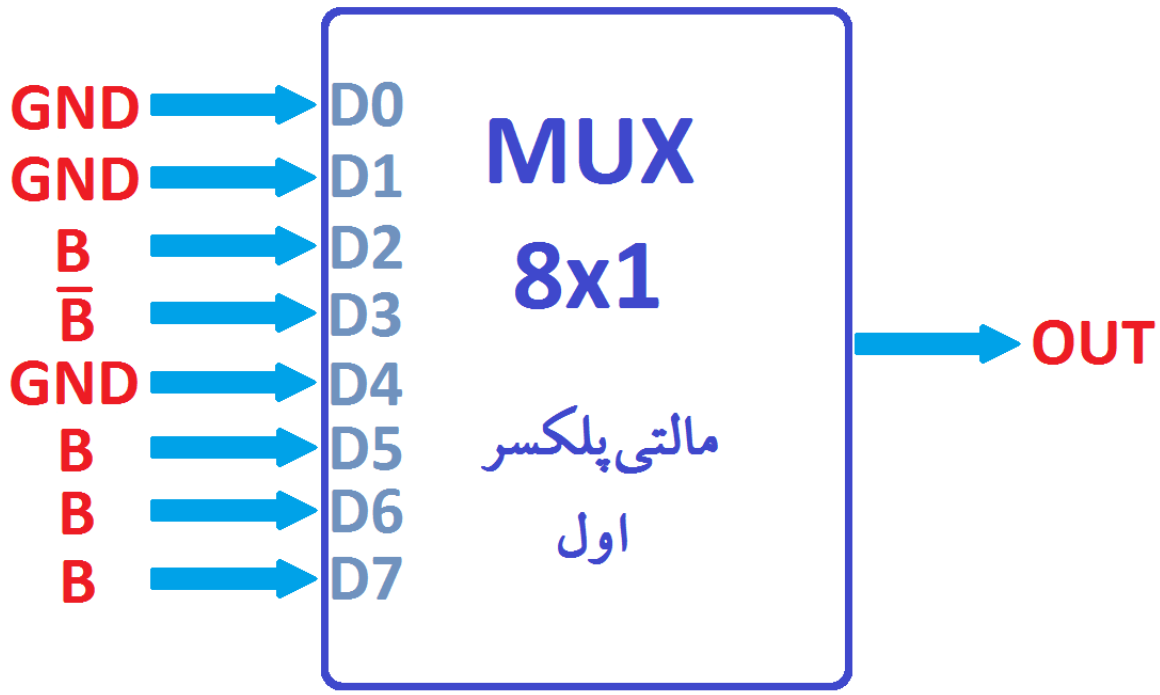
۸)دستور شماره هفت (A XOR B)

در حالتی که $S_0S_1S_2$ برابر 111 باشد، حاصل جمع انحصاری دو عدد A و B باید در خروجی ایجاد شود. روشن است که ورودی D7 مالتی‌پلکسر اول باید خود عدد B باشد. از بررسی جدول درستی جمع‌کننده در می‌یابیم که اگر CI را برابر صفر قرار دهیم و خروجی را از S بگیریم نتیجه، حاصل جمع انحصاری منطقی A و B خواهد بود. بنابراین ورودی D7 مالتی‌پلکسر دوم و سوم را به ترتیب به GND و S متصل می‌کنیم.

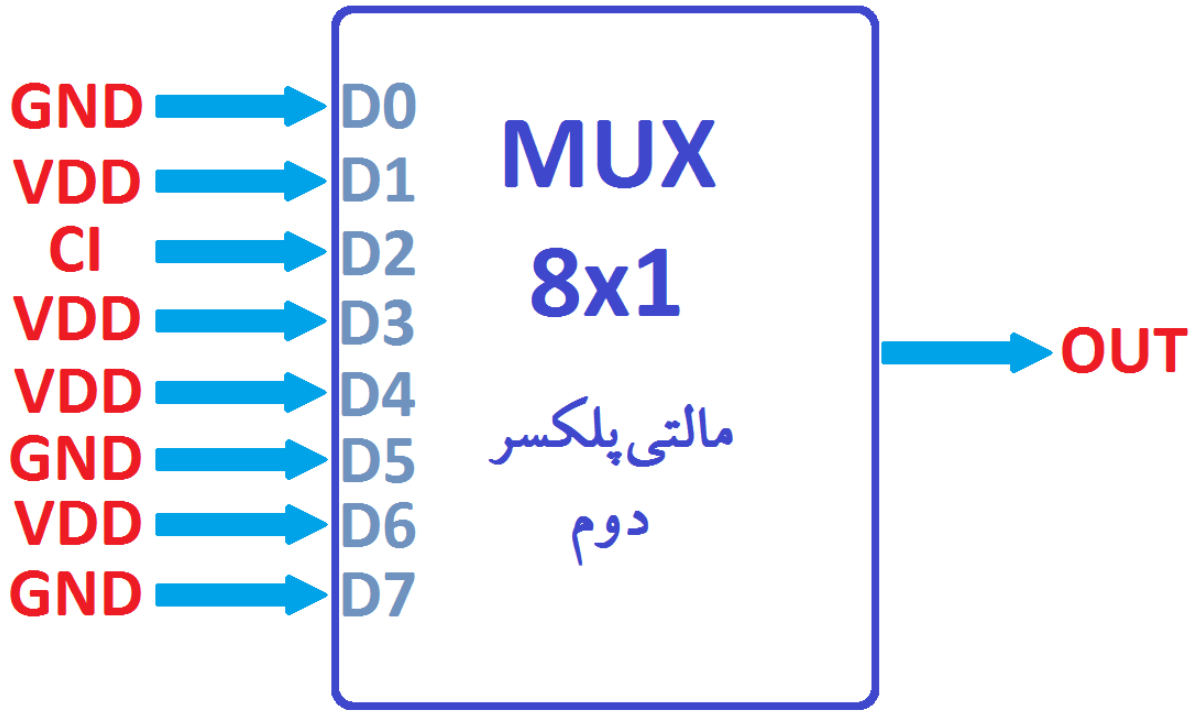
تفاوت

نکته‌ای که لازم است ذکر شود این است که اتصالاتی که بیان شد مربوط به بلوک جمع کننده‌ی اول هستند و در بلوک‌های بعدی تفاوتی جزئی ایجاد می‌شود. تفاوت آن است که در دستورهای شماره یک، دو و سه ما در بلوک‌های بعدی نیاز به کری تولید شده در بلوک قبلی داریم بنابر این ورودی‌های D1، D2 و D3 مالتی‌پلکسر دوم در بلوک‌های بعدی، هر سه باید به کری بلوک قبل متصل شوند. بنابر این بلوک اول در ورودی‌های مالتی-پلکسر دوم با سایر بلوک ها تفاوت دارد.

در شکل‌های زیرخلاصه اتصالات هر کدام از مالتی‌پلکسرها آورده شده است.



شکل(۲) ورودی‌های مالتی‌پلکسر اول



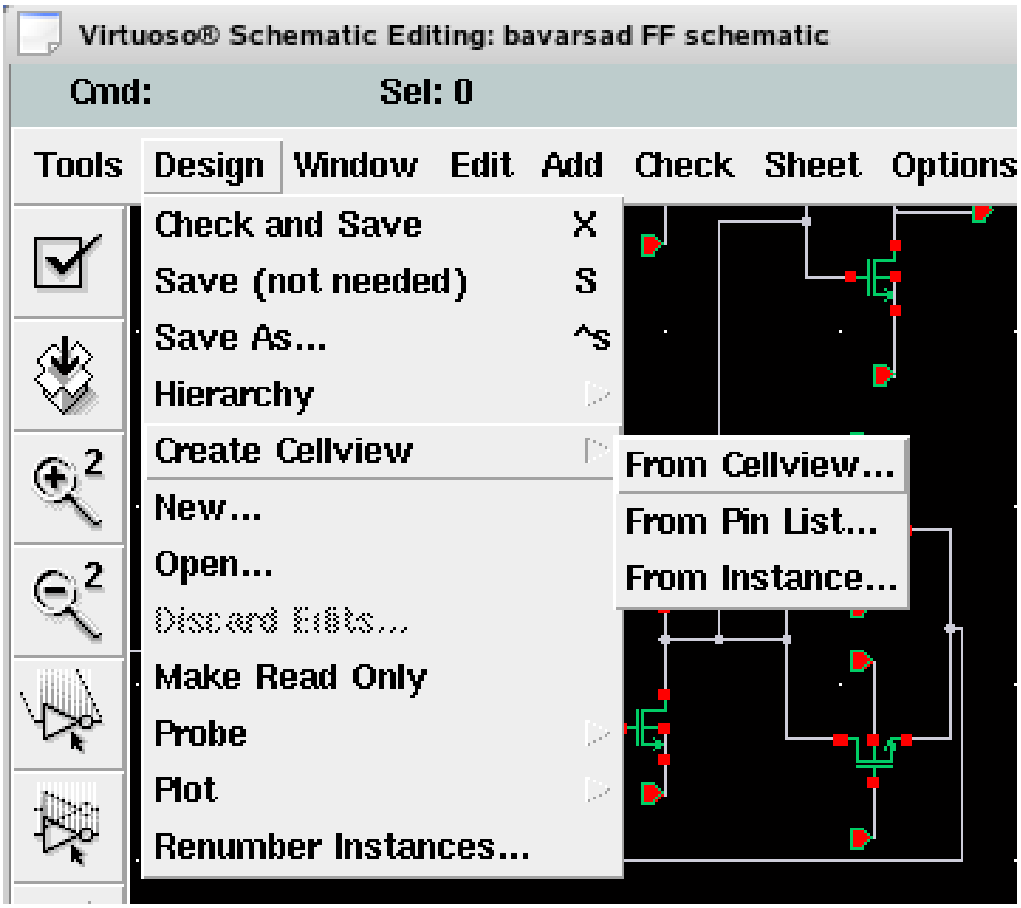
شکل (۳)ورودی‌های مالتی‌پلکسر دوم مربوط به بلوک اول

شکل (۴)ورودی‌های مالتی‌پلکسر سوم

شکل (۵) مالتی‌پلکسر دوم مربوط به بلوک‌های بعدی

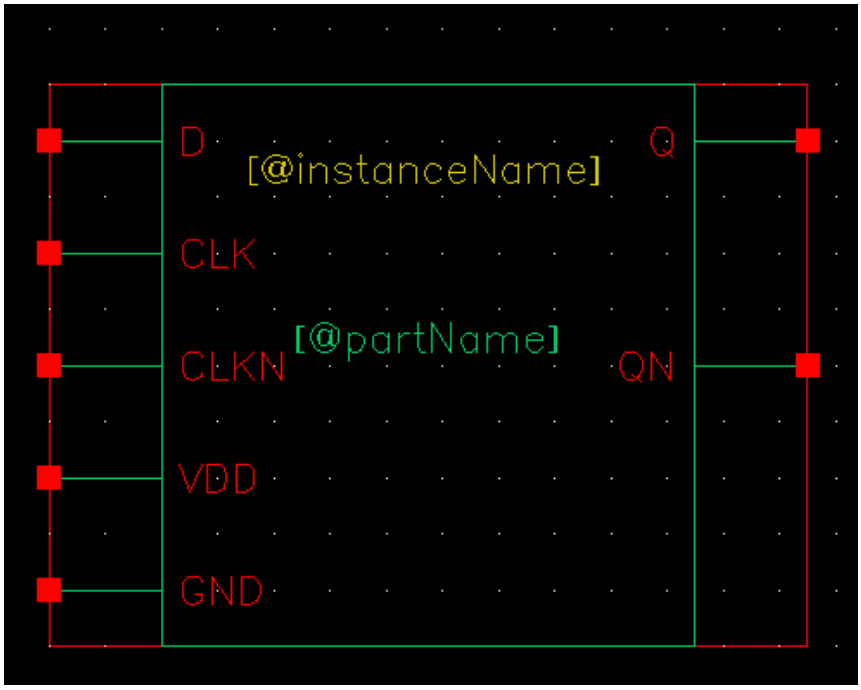
رسم شماتیک

پس از طراحی اولیه‌ی ALU نوبت به رسم شماتیک آن در نرم‌افزار Cadence می‌رسد. در رسم شماتیک اصلی ALU از سه جزء مختلف استفاده شده است که شامل Adder، مالتی‌پلکسر و فلیپ‌فلاپ می‌باشد. شماتیک هر کدام از این اجزاء به صورت جداگانه با استفاده از ترانزیستوهای NMOS و PMOS در تکنولوژی ۱۸۰nm در یک فایل مجزا رسم شد سپس layout شان نیز رسم شده، تست‌های DRC و LVS گرفته شد و در نهایت سیمبول مربوط به هر کدام ایجاد گردید. برای جلوگیری از پیچیدگی مدار در رسم ALU نهایی که شامل تعدادی Adder، فلیپ‌فلاپ و مالتی‌پلکسر است از این سیمبول‌ها استفاده شد. برای ساخت سیمبول هر جزء پس از رسم شماتیک آن از منوی Design گزینه‌ی From Cellview » Create Cellview را انتخاب می‌کنیم.



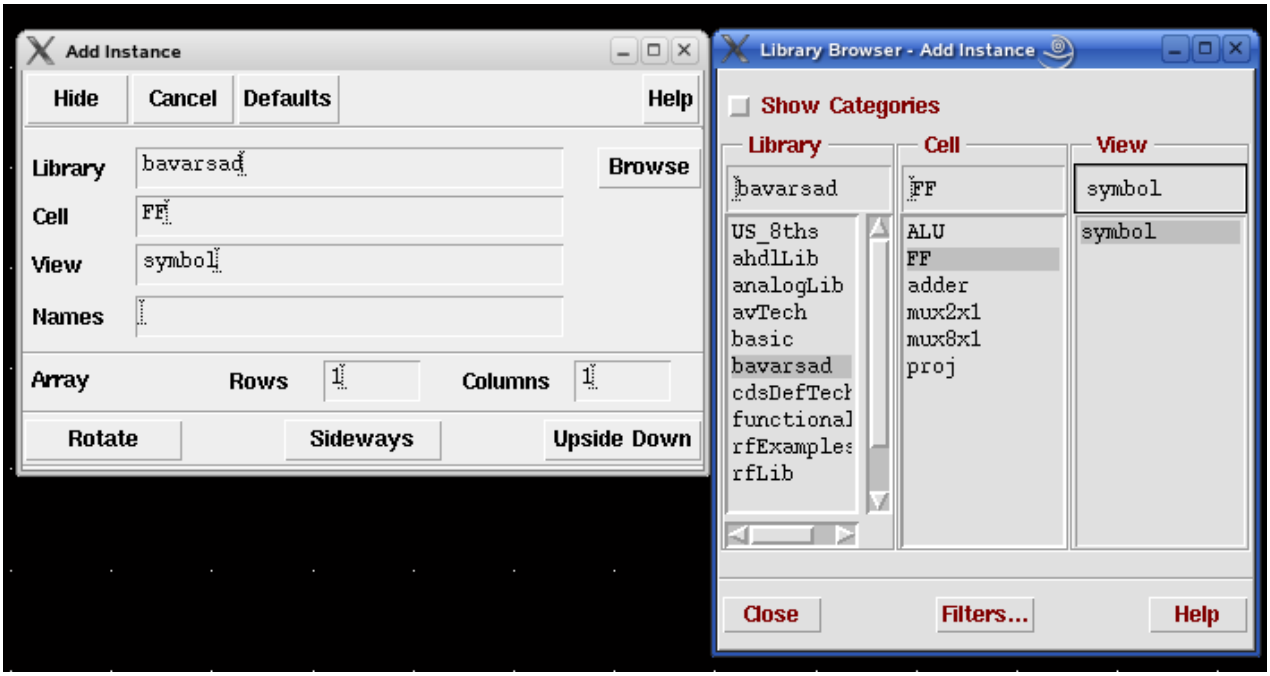
شکل (۶) مسیر ایجاد سیمبول

پس از تایید پیغام‌ها، وارد صفحه‌ی سمبول می‌شویم. در این صفحه می‌توان جای پورت‌ها، اندازه و شکل سیمبول را به دلخواه تغییر داد. پس از ویرایش نهایی از منوی Design گزینه‌ی Check and Save را انتخاب کرده و پنجره را می‌بندیم.



شکل (۷) سیمبول ساخته شده برای فلیپ‌فلاپ

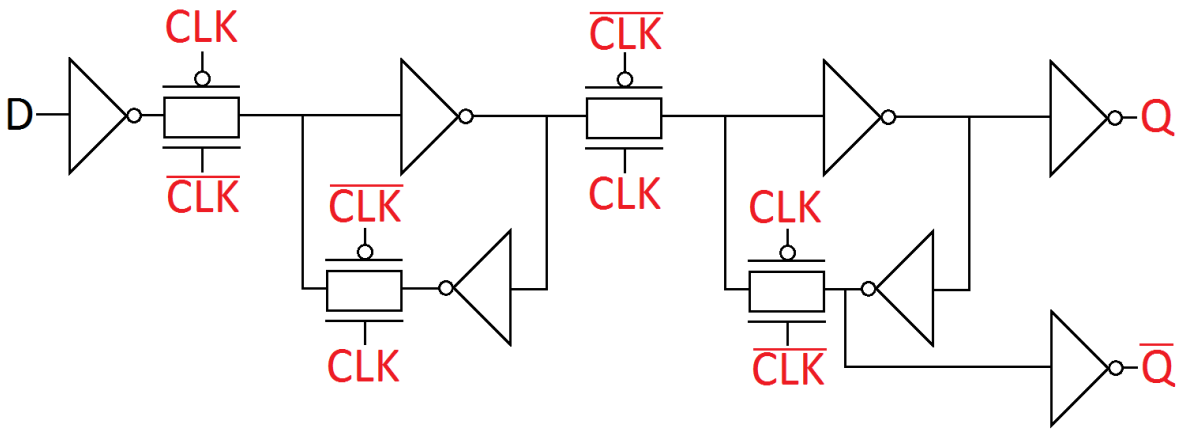
از این پس برای استفاده از این مدار در طراحی‌های بعدی نیاز به رسم دوباره‌ی آن نیست و میتوان از مسیر Instance»Brows»Bavarsad»FF»symbol از سیمبول ایجاد شده استفاده کرد. شکل (۸) نحوه استفاده از سیمبول ساخته شده را نشان می‌دهد. در اینجا Bavarsad نام Library و FF نام Cell مدار من بوده است.



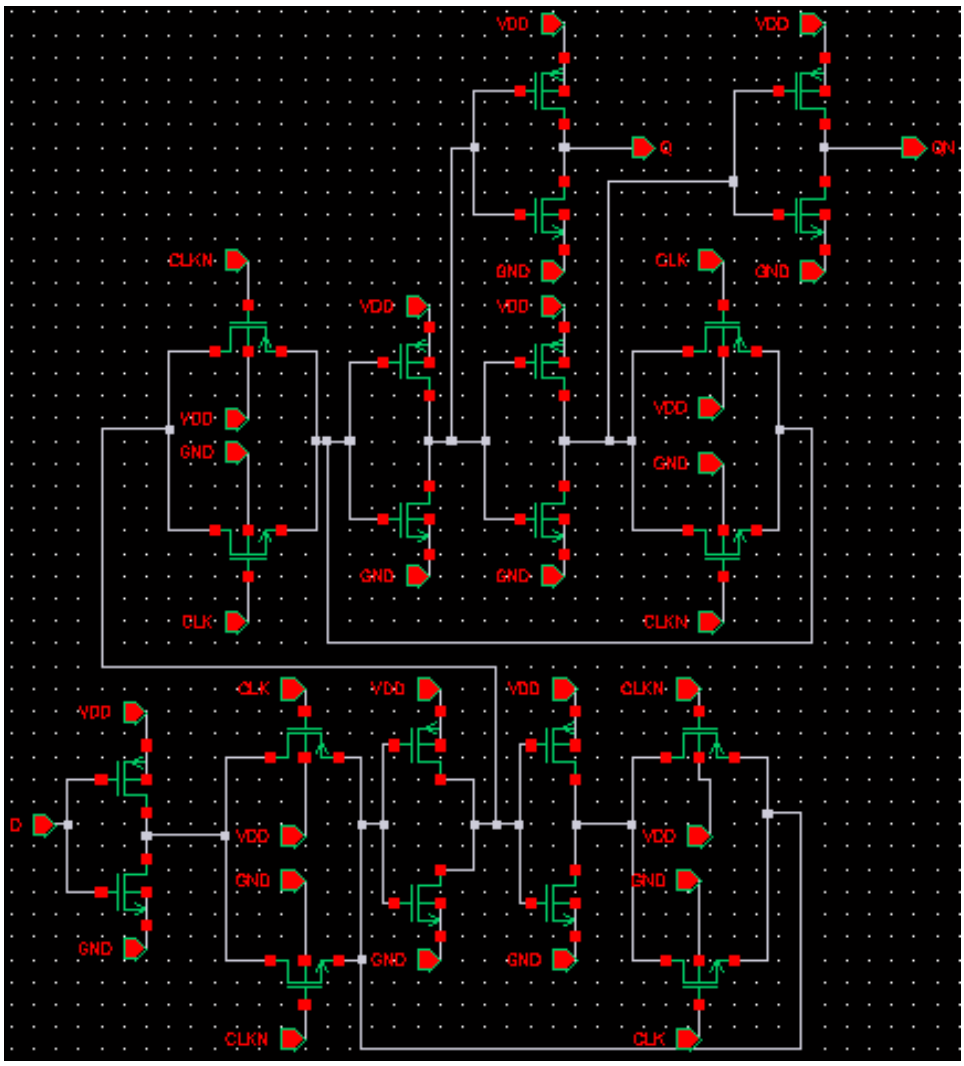
شکل (۸) استفاده از سیمبول ساخته شده در رسم شماتیک

مدار فلیپ فلاپ

برای ساخت فلیپ فلاپ از مدار شکل زیر استفاده شده است. این فلیپ فلاپ حساس به لبه‌ی بالا رونده است و Q و $\overline{Q}$ را تولید می‌کند. شماتیک رسم شده‌ی آن در نرم‌افزار نیز در شکل (۱۰) آورده شده است.



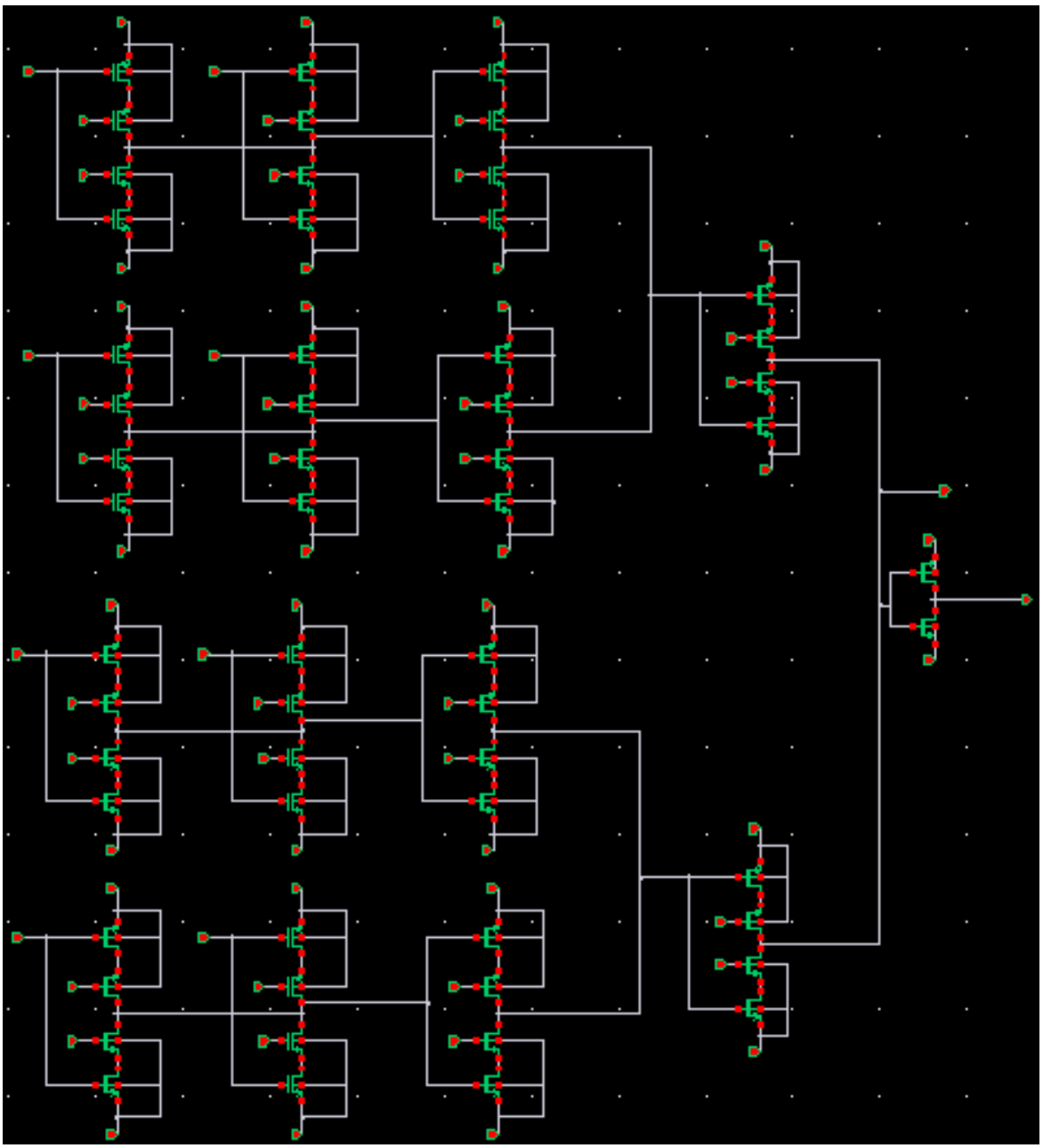
شکل (۹) مدار فلیپ فلاپ



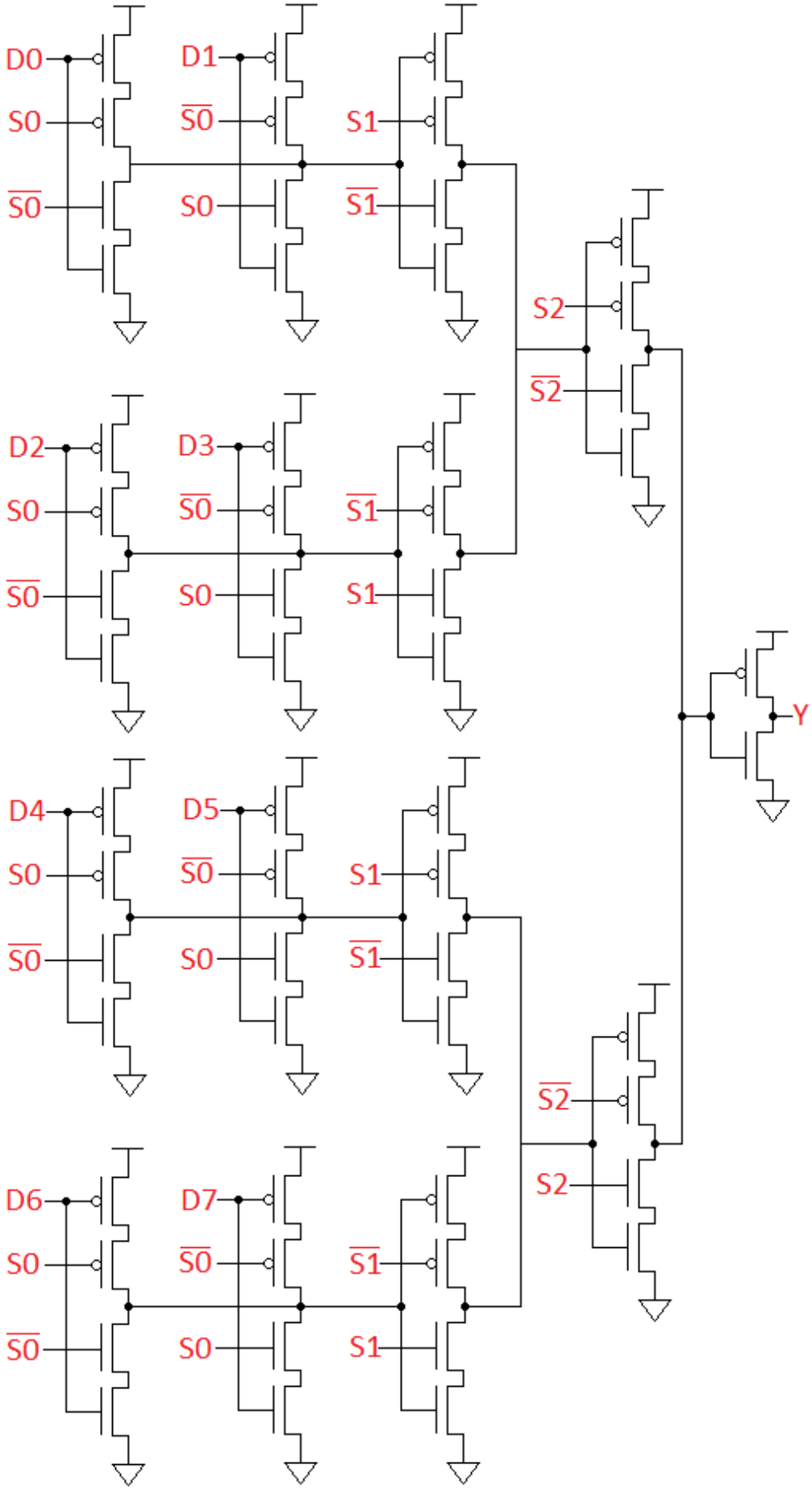
شکل (۱۰) شماتیک فلیپ فلاپ

مدار مالتی‌پلکسر

برای ساخت مالتی‌پلکسر از مالتی‌پلکسرهای دو به یک ریستورینگ استفاده شد. این روش ساخت مالتی‌پلکسر هشت به یک، علاوه بر تحمیل خازن کمتر دارای ولتاژ Low و High مناسب تری هست. اگر از ماکس چهار به یک استفاده می‌شد طبقات بیشتری از ترانزیستورها نیاز بود و بنابراین باید دارای سایز بزرگتری می‌شدند (که به معنی خازن بیشتر می‌باشد). همچنین طبقات سری ترانزیستور باعث افت ولتاژ High و افزایش ولتاژ Lowمی‌شود. به دلیل این که این نوع از مالتی‌پلکسرها ورودی را معکوس می‌کنند و ما تعداد فردی از آن‌ها را به صورت سری استفاده کرده‌ایم، در انتهای مدار نیاز به یک NOT داریم تا علامت خروجی و ورودی یکسان شوند. مدار استفاده شده برای ساخت مالتی‌پلکسر و شماتیک رسم شده در نرم افزار در شکل‌های زیر آمده اند.



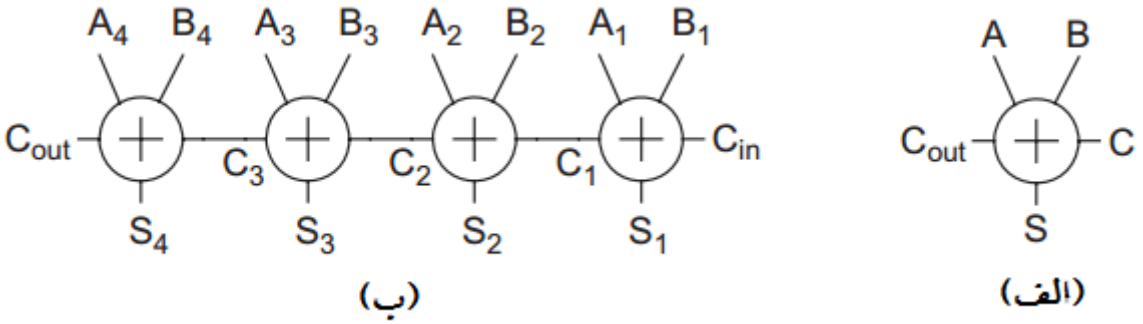
شکل (۱۱) شماتیک مالتی‌پلکسر هشت به یک



شکل(۱۲) مدار مالتی پلکسر هشت به یک

مدار جمع کننده (Adder)

نمودار بلوکی یک جمع کننده‌ی چهار بیتی که با استفاده از چهار Full Adder ساخته شده است، به صورت زیر است.



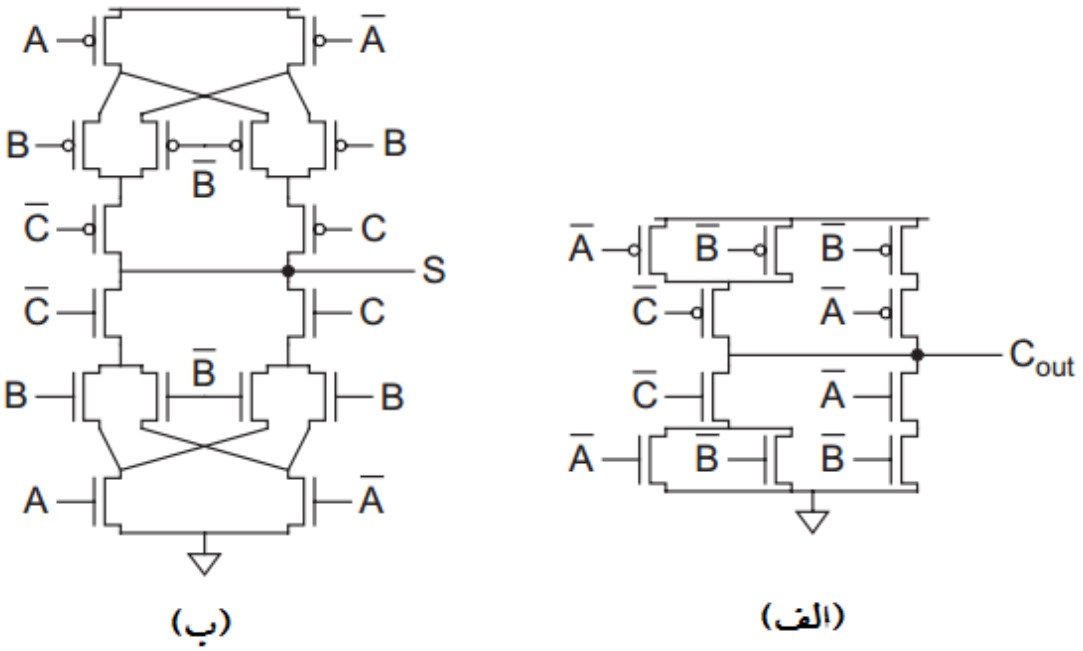
شکل (۱۳) الف) Full Adder ب) جمع کننده چهار بیتی

برای ساخت یک Full Adder جدول درستی را ایجاد کرده، روابط S و C_{out} را استخراج کرده و پس از ساده سازی به روابط زیر می‌رسیم.

$$S = A\bar{B}\bar{C} + \bar{A}B\bar{C} + \bar{A}\bar{B}C + ABC = \left(A \oplus B\right) \oplus C$$

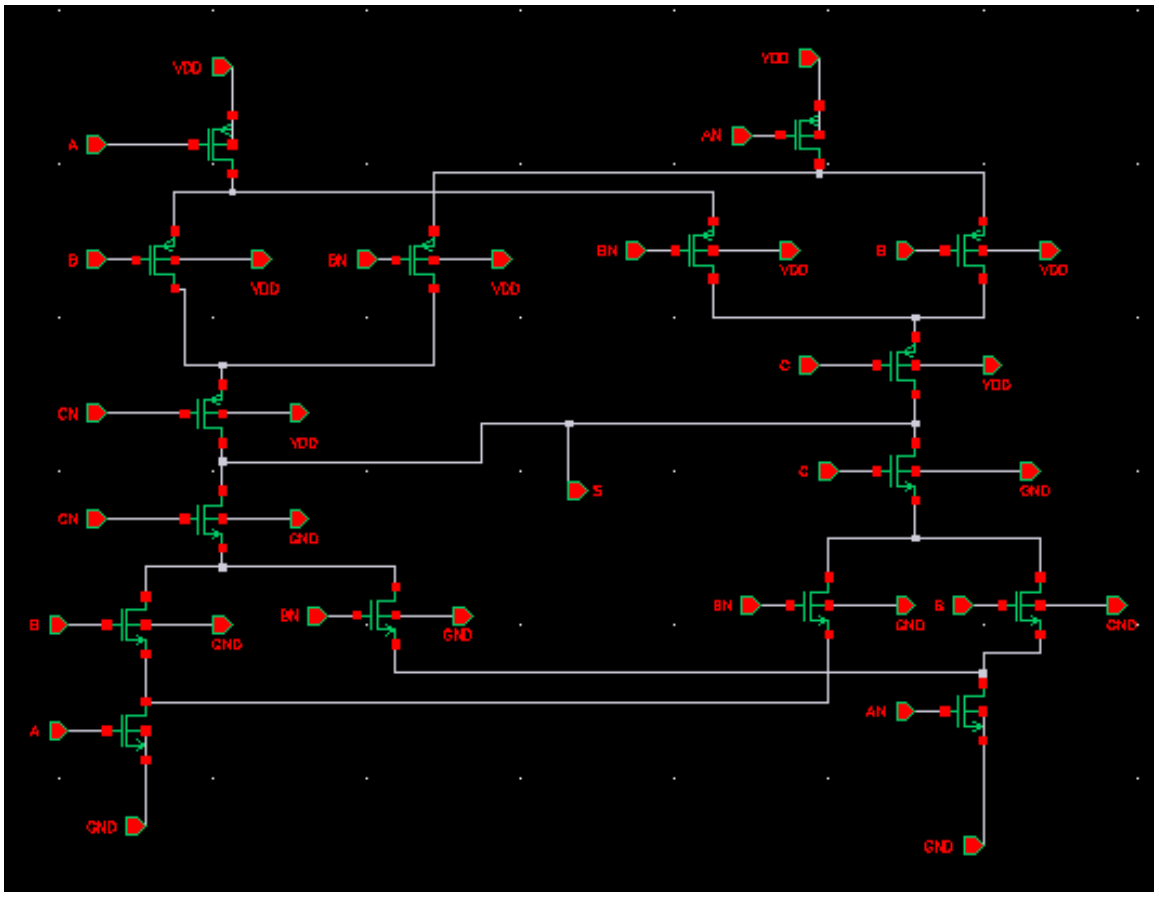
$$C_{out} = AB + AC + BC = AB + C\left(A + B\right) = \overline{\bar{A}\bar{B} + \bar{C}\left(\bar{A} + \bar{B}\right)}$$

در نهایت با استفاده از این توابع مدار جمع کننده به صورت زیر ساخته می‌شود.

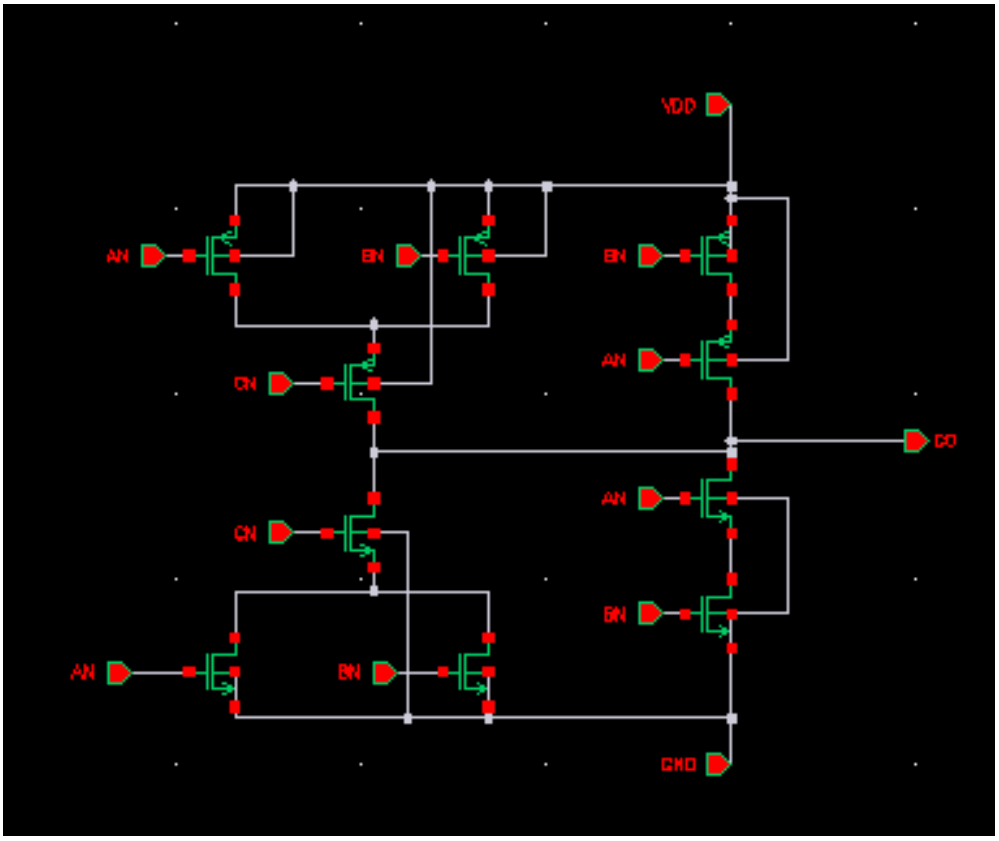


شکل (۱۴) مدار جمع کننده (Full Adder)

مدار رسم شده برای جمع کننده در نرم‌افزار Cadence در شکل‌های زیر آمده است.



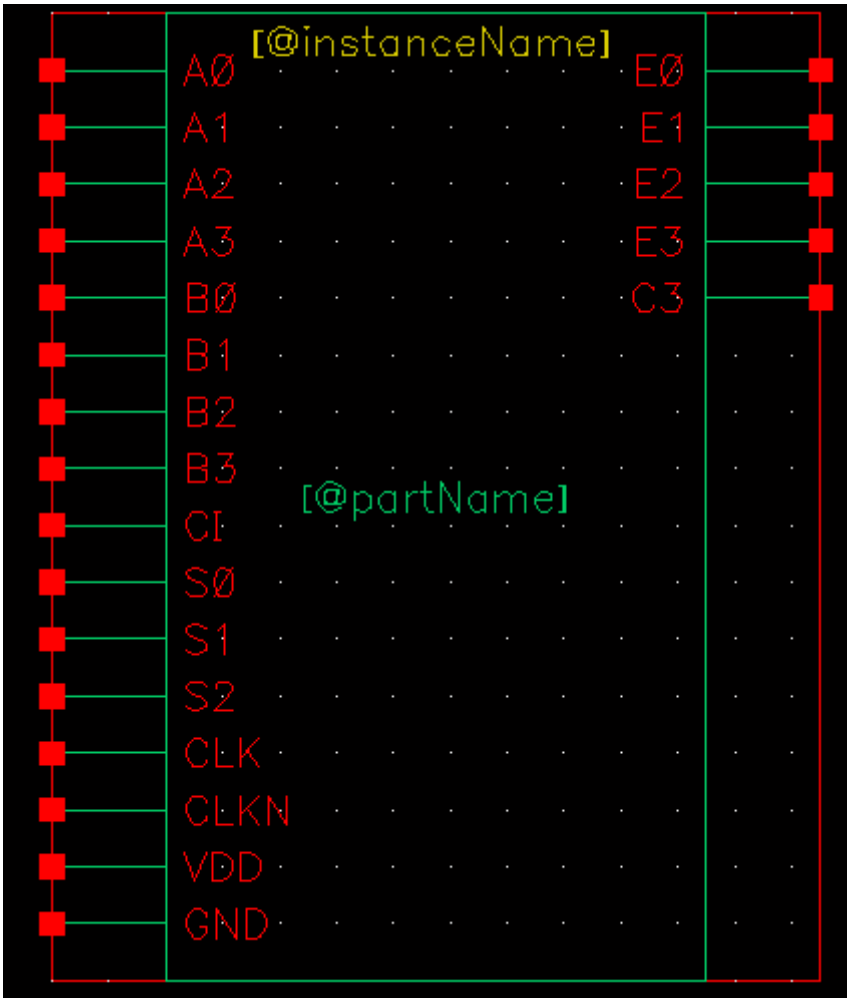
شکل (۱۵) شماتیک تولید S جمع کننده



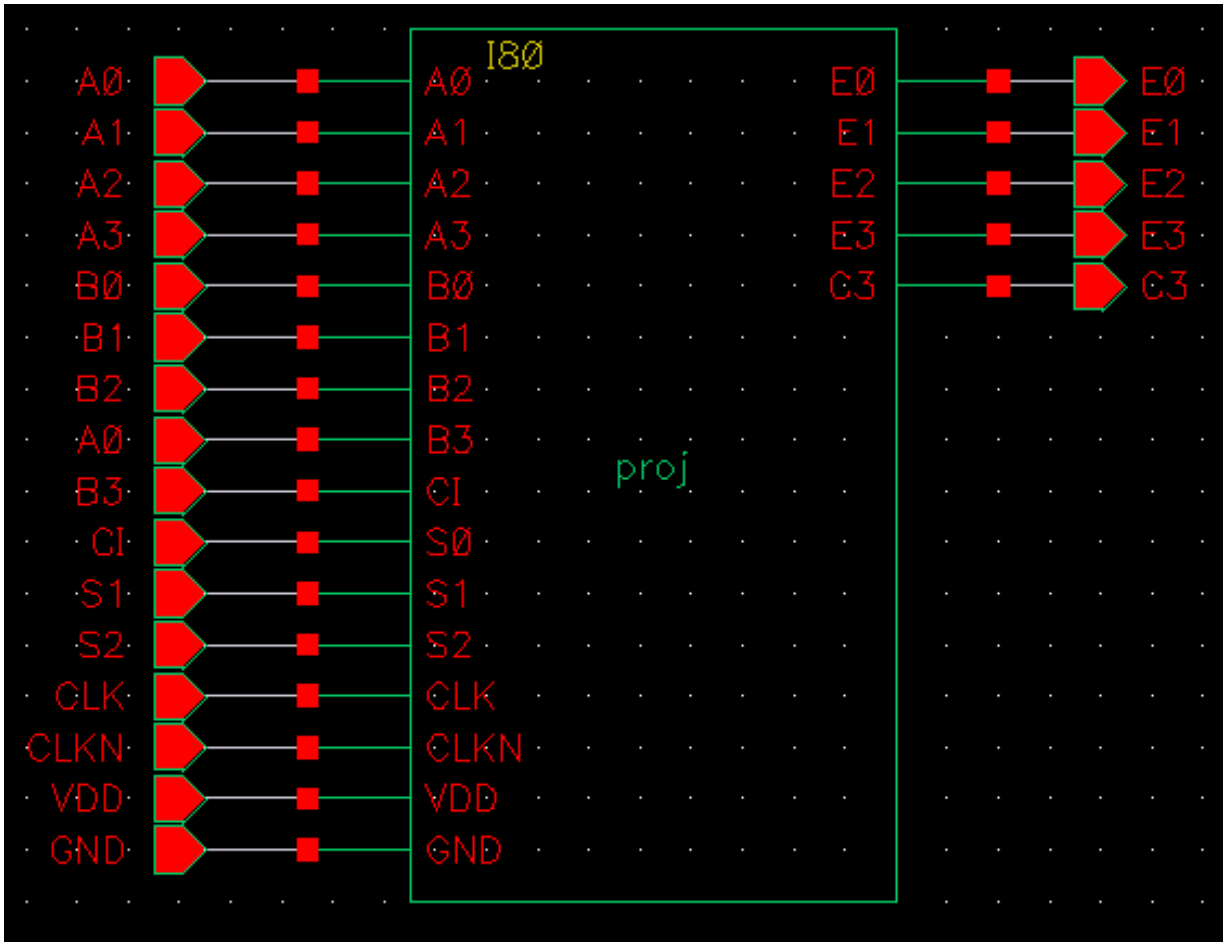
شکل (۱۶) مدار تولید C_{OUT} جمع کننده

مدار ALU

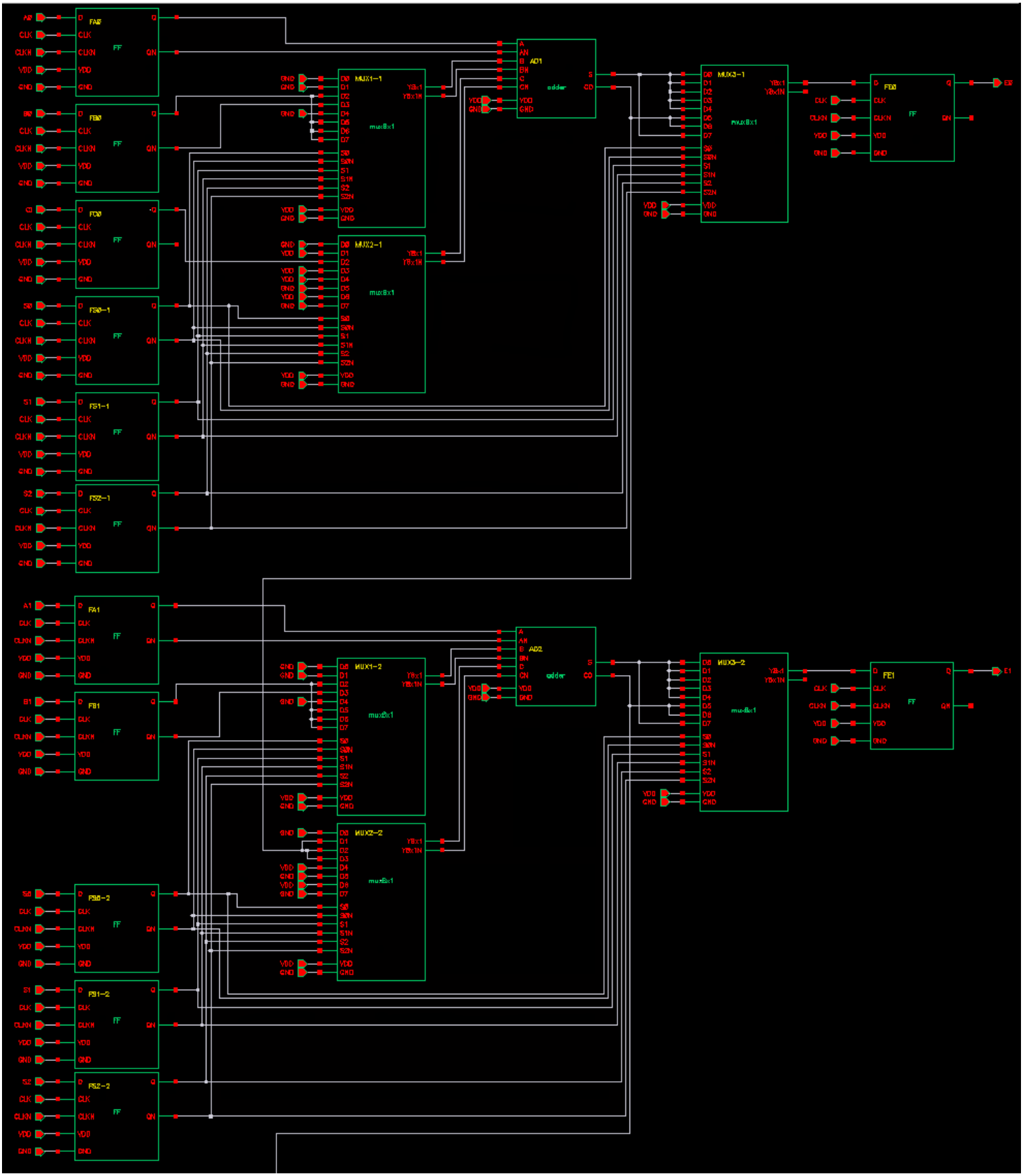
شماتیک رسم شده‌ی دو بلوک اول مدار ALU در نرم‌افزار با استفاده از سیمبول‌ها، در شکل(۱۹) آورده شده است. پس از رسم شماتیک کل مدار برای خود این مدار نیز یک سیمبول ساختیم که شبیه‌سازی و کار با آن ساده‌تر گردد. در یک Cellview جداگانه به نام ALU از این سیمبول استفاده شد. ورودی ها و خروجی ها مشخص شدند و با اعمال داده‌های مختلف مدار شبیه‌سازی شد و مورد آزمایش قرار گرفت.برای این این که مدار به صورت سنکرون کار کند باید قادر به استفاده از پالس کلاک ساعت باشد که خود مستلزم استفاده از فلیپ‌فلاپ است. همان طور که در شکل نیز مشخص است بر سر هر کدام از ورودی‌های A0 تا A3 همچنین B0 تا B3 و نیز CI،S0،S1 و S2 یک فلیپ‌فلاپ قرار گرفته است. همچنین بر سر خروجی‌های E0 تا E3 و C3 نیز فلیپ‌فلاپ قرار گرفته است. بنابر این سیستم به صورت PipeLine عمل می‌کند و با هر پالس ساعت یک خروجی داریم. به این ترتیب با هر پالس ساعت یک سری از داده‌ها برای عملیات وارد ALU می‌شوند و همزمان داده‌هایی که قبلا پردازش شده بودند به خروجی می‌روند. ورود داده‌های خام، پردازش و خروج داده‌های خروجی به اندازه‌ی دو پالس ساعت زمان می‌برد.



شکل (۱۷) سیمبول ساخته شده برای کل مدار ALU



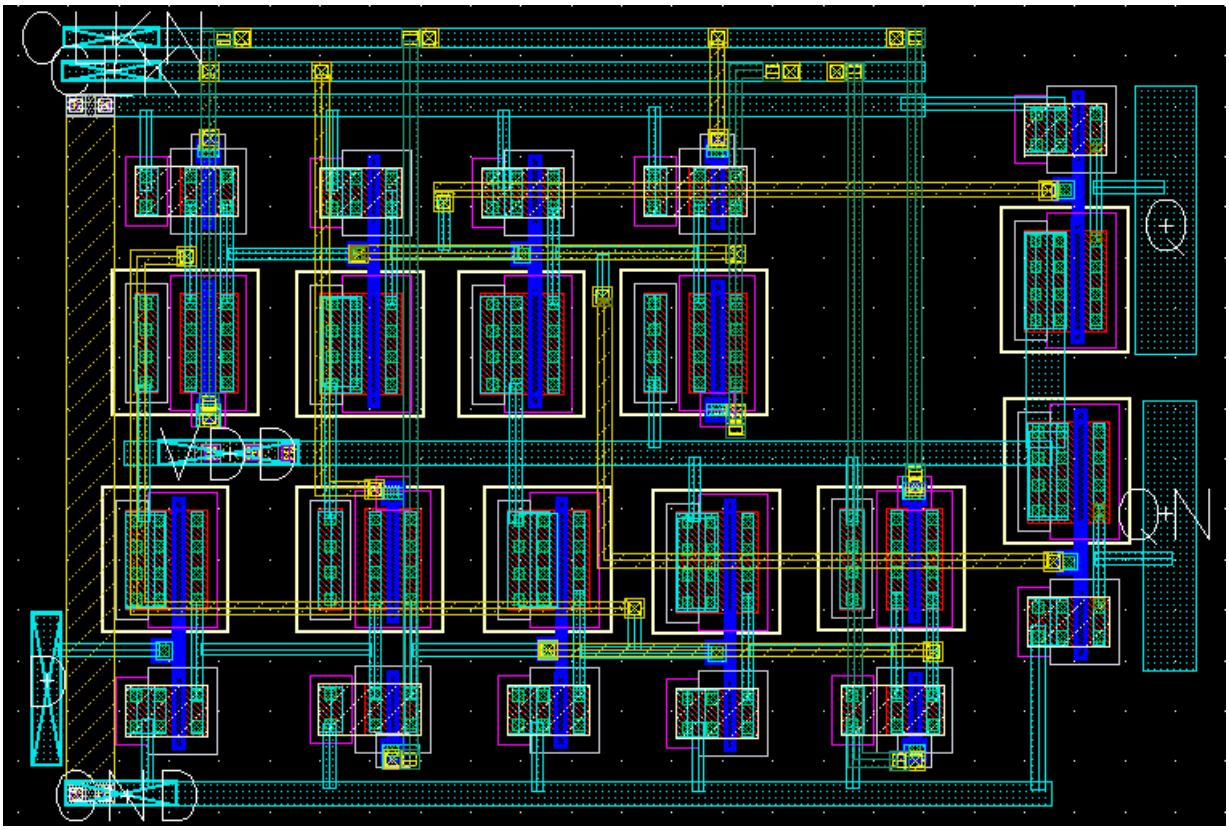
شکل (۱۸) استفاده از سیمبول ALU در صفحه‌ی شماتیک برای اجرای شبیه سازی



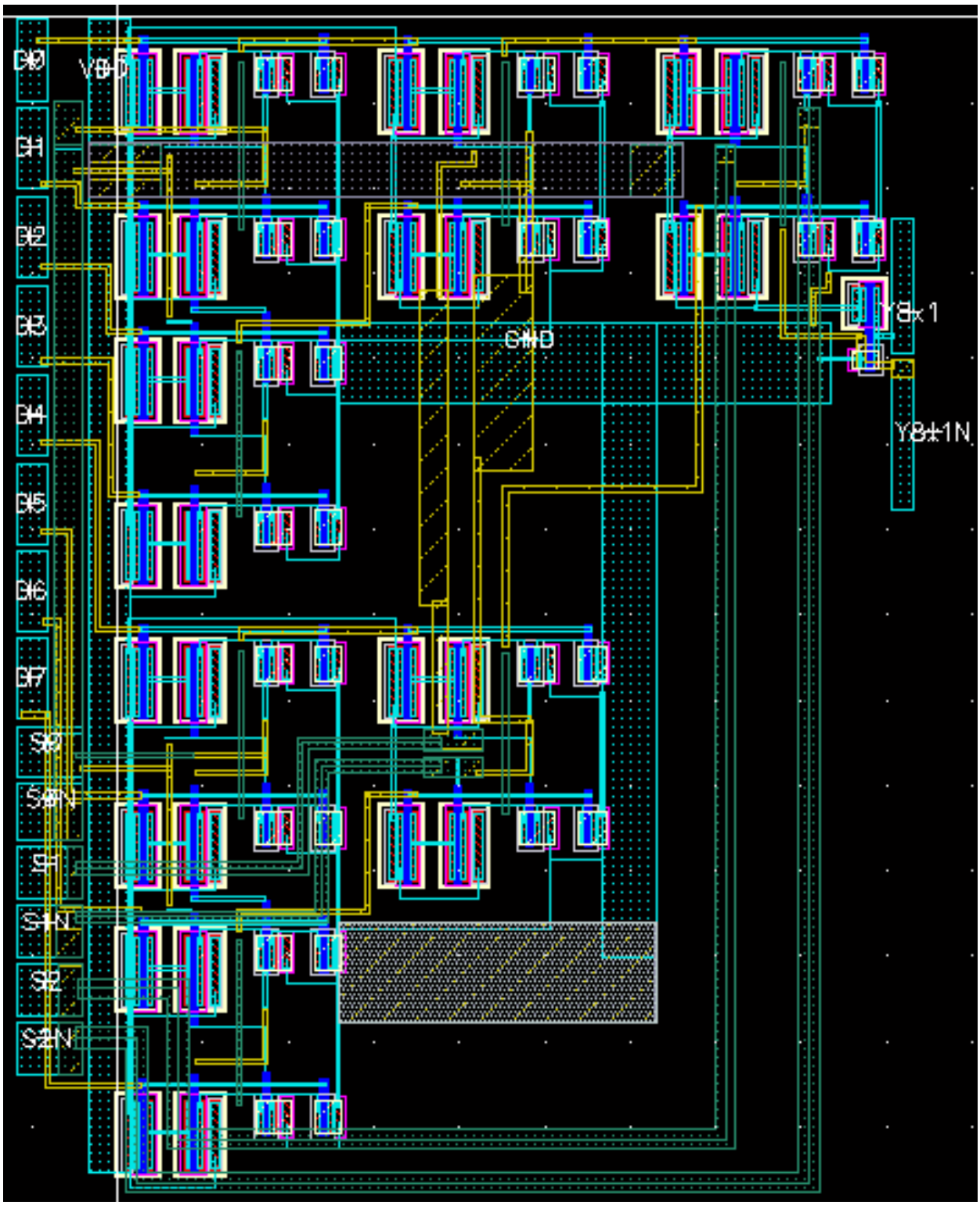
شکل (۱۹) شماتیک مدار ALU در نرم افزار Cadence

رسم Layout

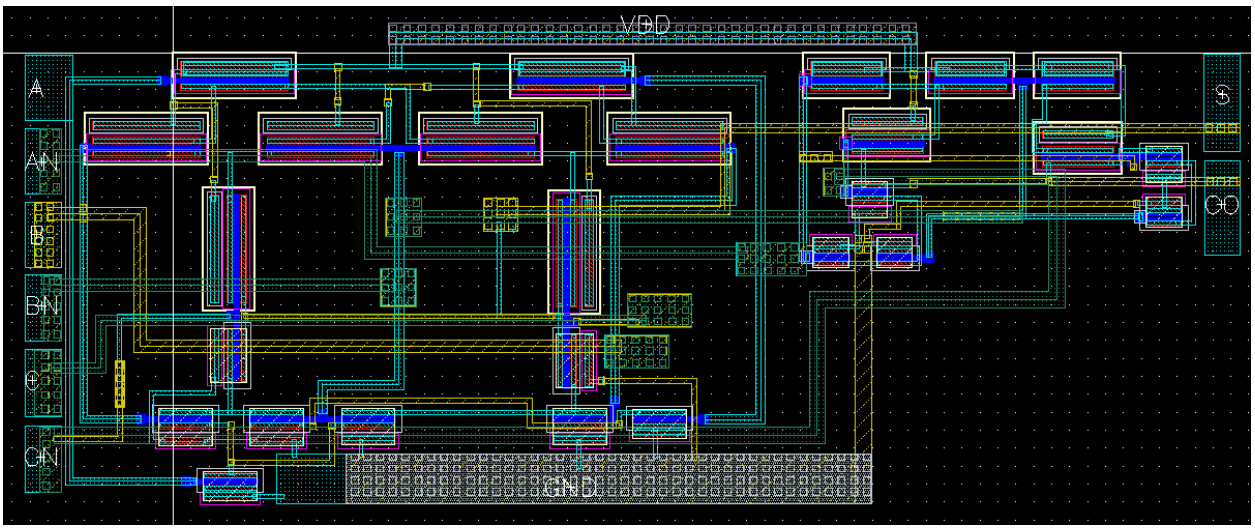
پس از رسم شماتیک و انجام شبیه سازی و اطمینان از صحت عملکرد مدار، نوبت به رسم Layout مدارها می‌رسد. برای رسم Layout، در صفحه‌ی شماتیک از منوی Tools ، Lauout XL«Design Synthesis را انتخاب می‌کنیم. پس از تایید پیغام‌های داده شده یک صفحه‌ی Layout باز می‌شود. حال در این صفحه Design » Gen From Source را اجرا می‌کنیم. پس از این کار همه‌ی ترانزیستورهای استفاده شده در شماتیک با همان اندازه‌ها و نام‌های مشابه وارد صفحه‌ی Layout می‌شوند. اکنون می‌توان Layout مدار را ترسیم کرد. لی‌اوت‌های رسم شده‌ی فلیپ‌فلاپ، مالتی‌پلکسر، Adder و ALU نهایی در شکل‌های زیر آورده شده‌اند.



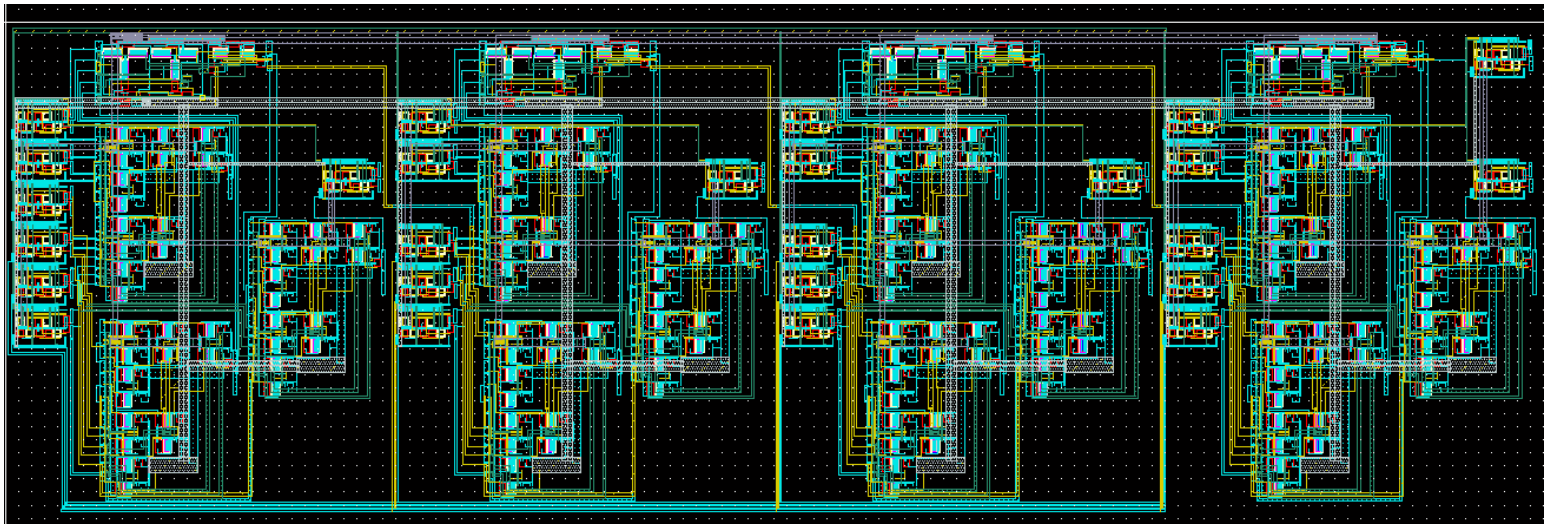
شکل (۲۰) Layout فلیپ فلاپ



شکل(۲۱) Layout مالتی پلکسر



شکل (۲۲) Layout جمع کننده



شکل (۲۳) Layout کلی

مشخصات و اندازه‌ها

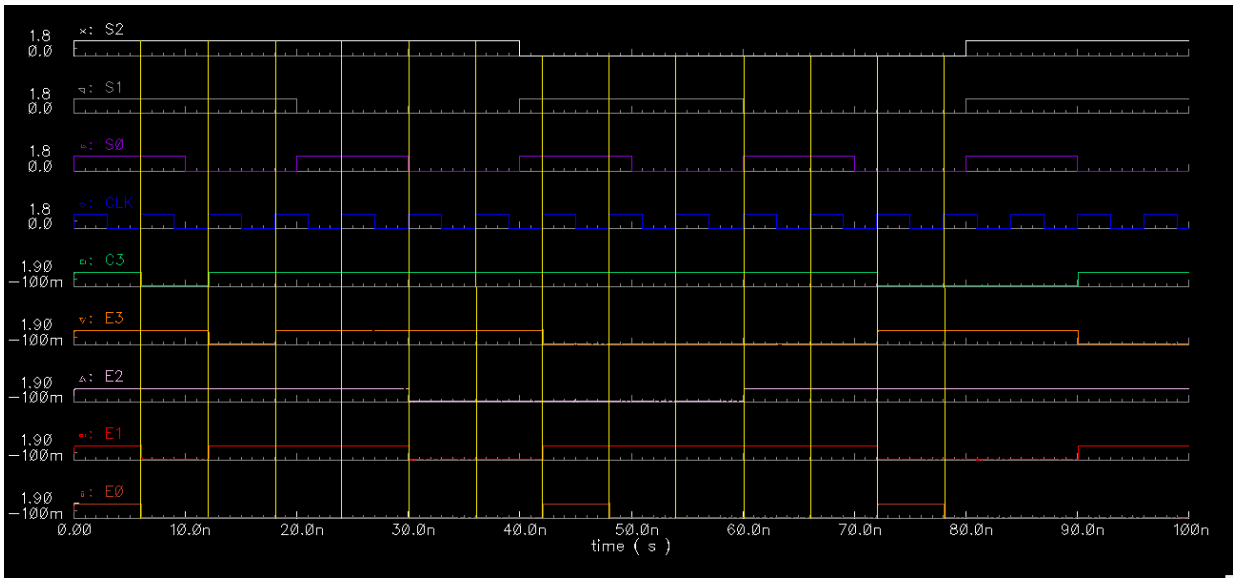
در رسم Layout از سه متال یک، دو و سه برای سیگنال، متال چهار برای VDDو متال پنج برای GND استفاده شد. طول و عرض مدار نهایی μm ۱۸۵ ×۵۸۹ به دست آمد. بنابراین مساحت کل مدار μm^2 ۱۰۸۹۶۵ یا mm^2 ۰/۱۰۸ محاسبه می‌شود.

شبیه‌سازی

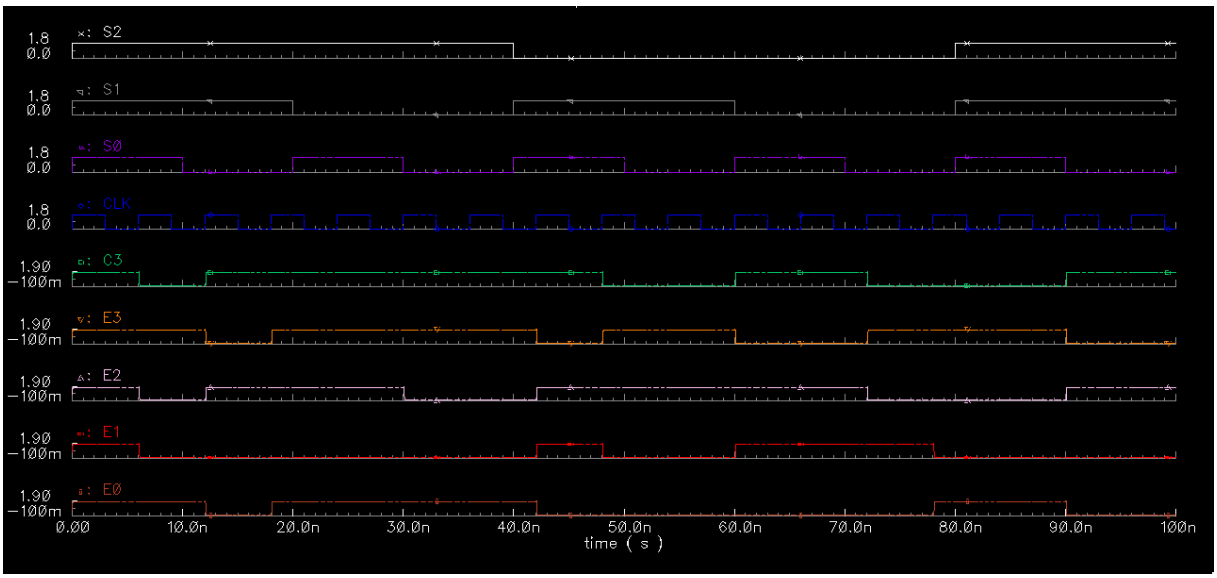
هر کدام از اجزا مدار یعنی فلیپ‌فلاپ،مالتی‌پلکسر و Adder به طور جداگانه مورد آزمایش قرار گرفتند که در اینجا فقط به ارائه‌ی تصاویری از شبیه سازی آن‌ها بسنده می کنیم.

برای شبیه سازی ALU دو حالت مختلف را در نظر گرفتیم. در هر حالت دو مقدار متفاوت به ورودی‌های A و B داده شد و نتایج با مقادیر از قبل محاسبه شده مقایسه گردید. در هر حالت مدار به خوبی عمل می کرد. مقادیر مورد انتظار و نتایج شبیه سازی در جداول و شکل‌های زیر آورده شده است.

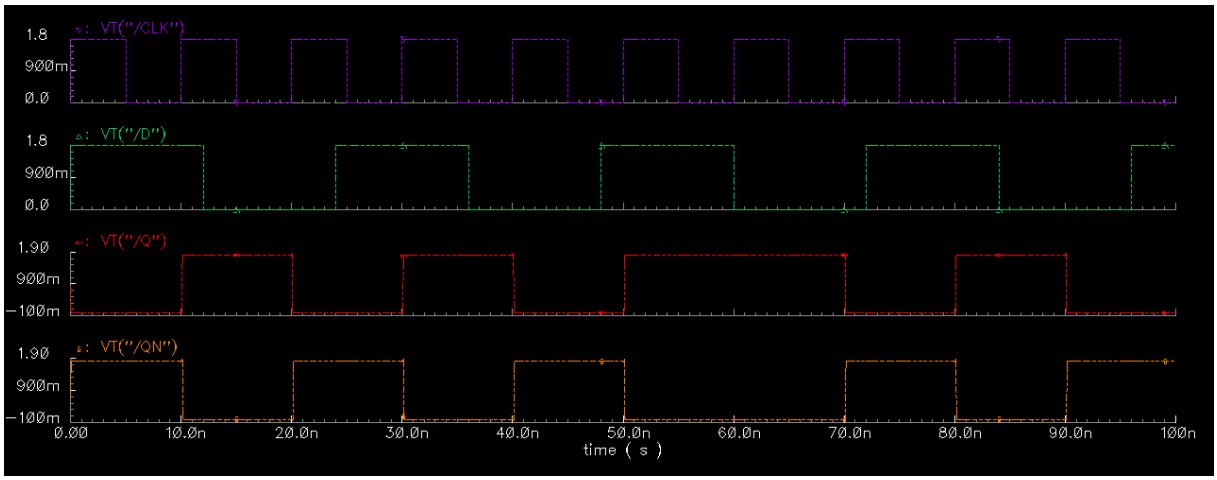
حالت	ورودی		$S_0S_1S_2$							
	A	B	000	001	010	011	100	101	110	111
حالت۱	1100	1010	1100	1101	10110	10010	0011	1000	1110	0110
حالت۲	1001	1101	1001	1010	10110	00110	0110	1001	1101	0100



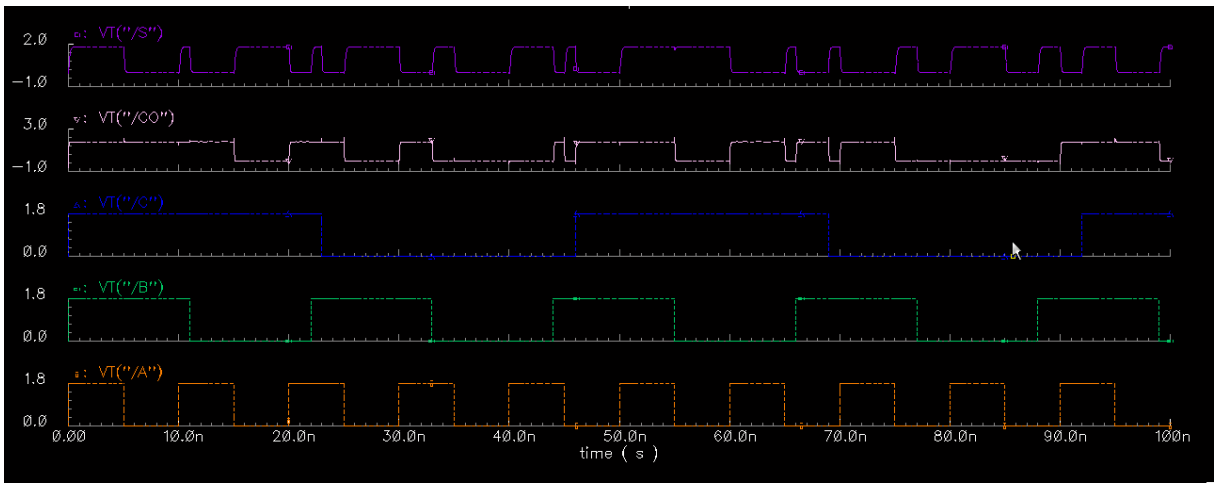
شکل (۲۴) شبیه سازی ALU در حالت اول



شکل (۲۵) شبیه سازی ALU در حالت دوم



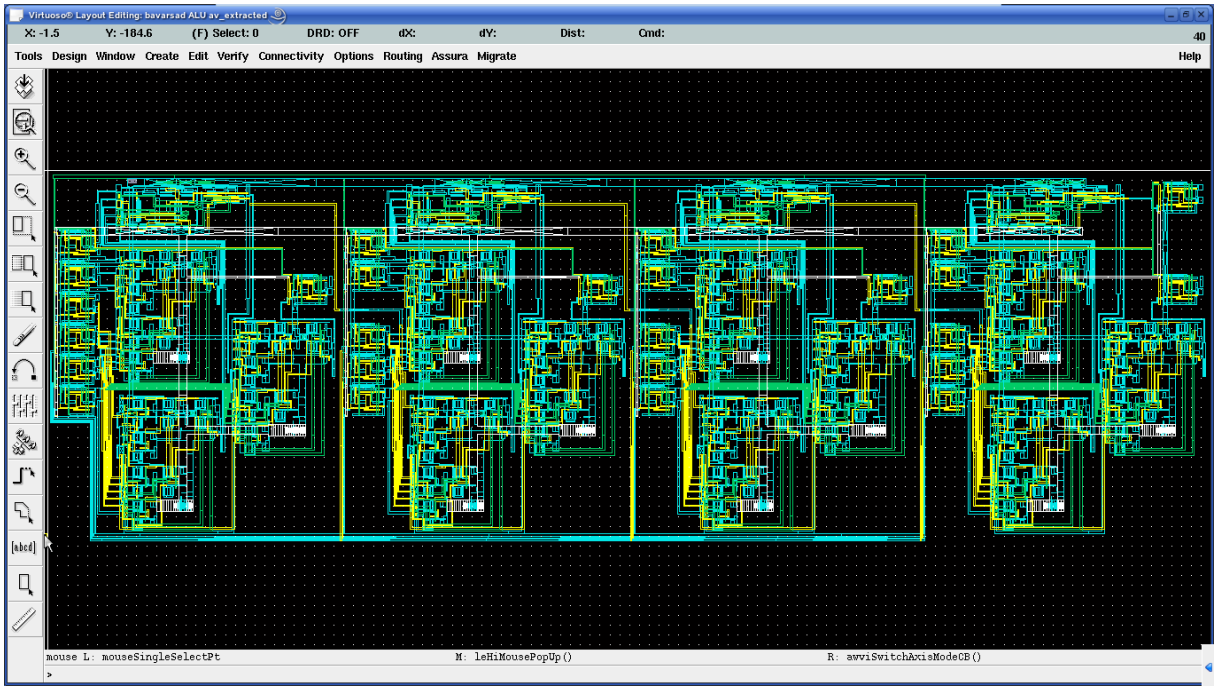
شکل (۲۶) شبیه سازی فلیپ فلاپ



شکل (۲۷) شبیه سازی Adder

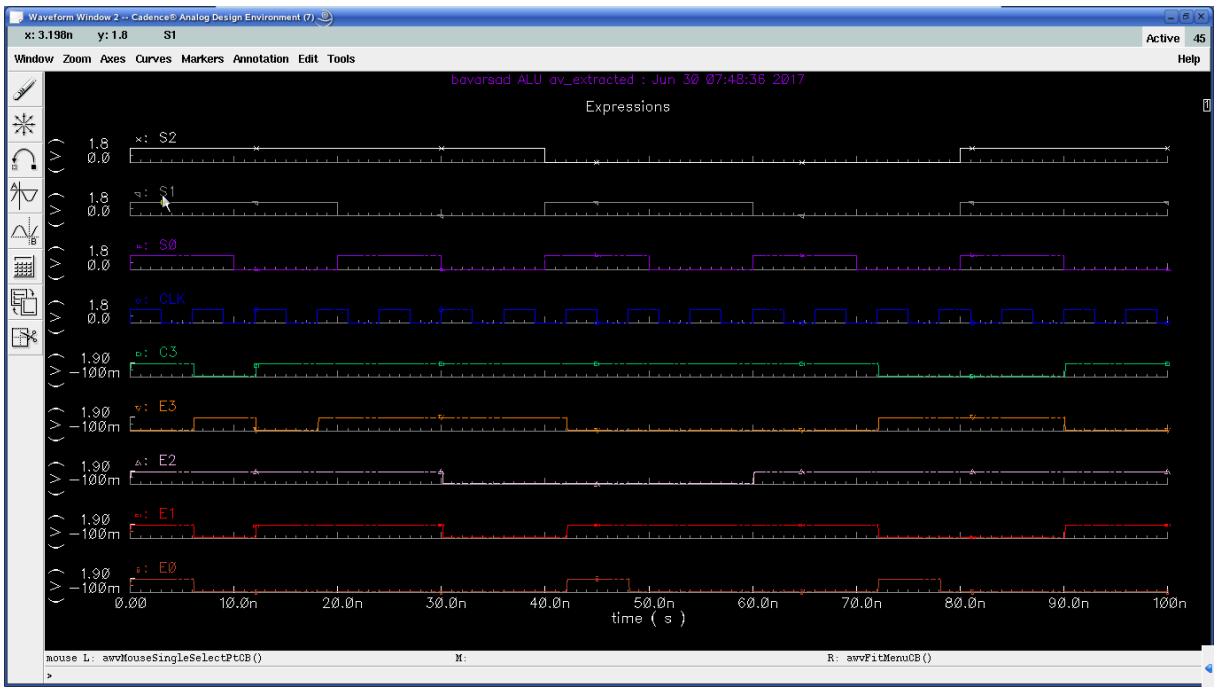
Post Layout

پس از انجام آنالیز LVS و اطمینان از match بودن layout رسم شده با شماتیک، می توانیم عمل extraction به همراه المان های پارازیتی و به دنبال آن post layout simulation را انجام دهیم. از منوی Assura گزینه‌ی RunRCX را انتخاب می‌کنیم و در پنجره‌ی باز شده گزینه‌ی Output را روی Extracted View قرار می‌دهیم. مقابل گزینه View نام دلخواهی را وارد می‌کنیم. سپس در بخش بالایی پنجره، تب Extraction را انتخاب می‌کنیم وگزینه‌ی Extraction Mode را روی RC ست می‌کنیم. به این ترتیب عمل extraction مقاومتی-خازنی انجام می گیرد. در بخش Ref Node گره GND را به عنوان گره مرجع وارد می‌کنیم و در نهایت روی OK کلیک می‌کنیم. پس از انجام عمل Extraction یک View با همان نامی که انتخاب کرده بودیم در آن سلول ایجاد می‌شود. نتیجه در شکل زیر آمده است.

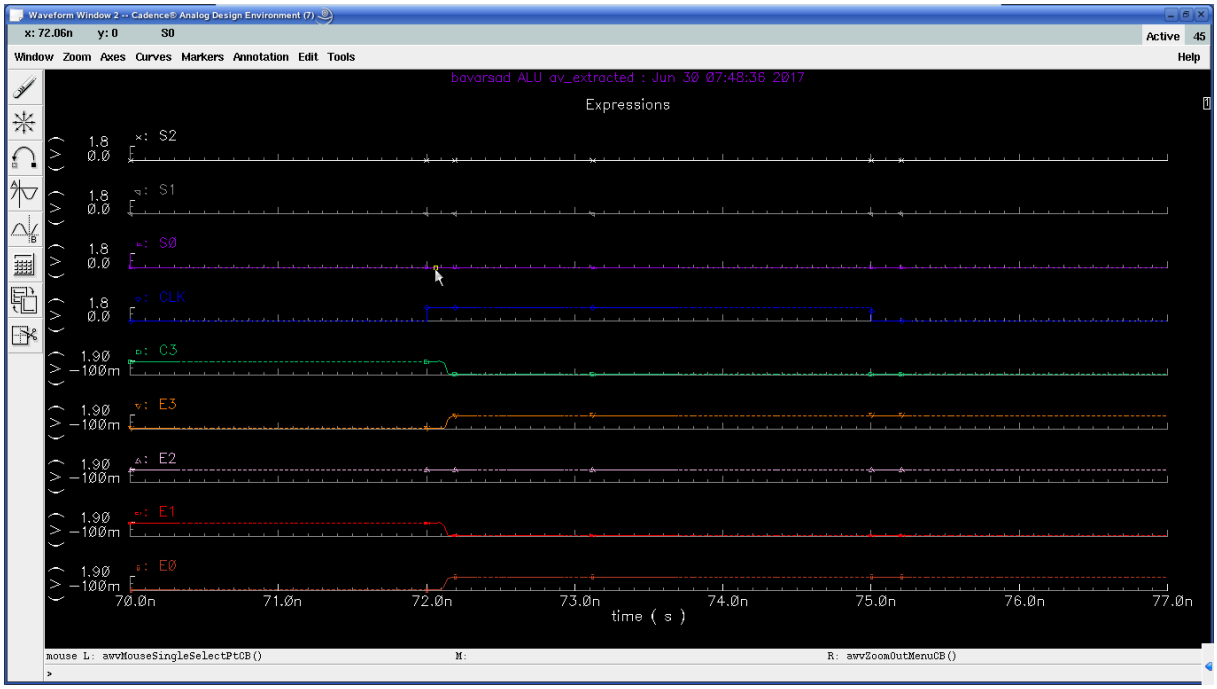


شکل (۲۸) Post Layout

سپس PostLayout شبیه سازی شد و مشاهده شد که زمان صعود و نزول سیگنال‌ها کمی افزایش یافته است. نتیجه در شکل زیر آورده شده است.



شکل (۲۹) شبیه سازی Post Layout

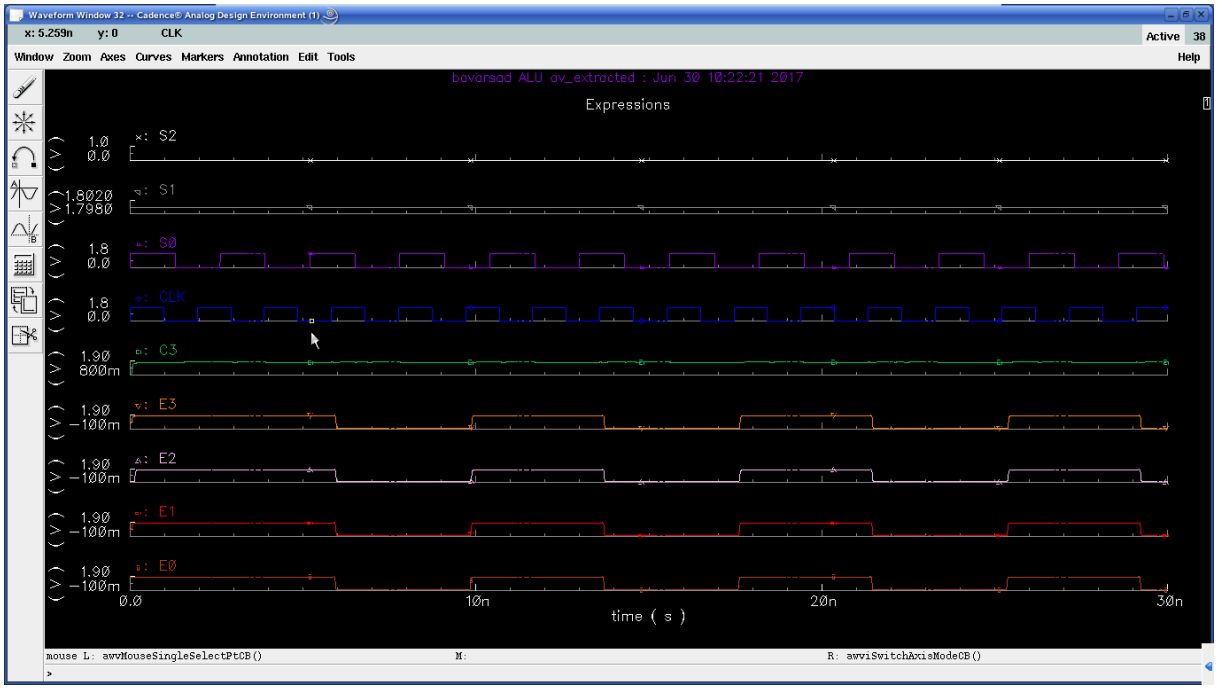


شکل (۳۰) نتایج شبیه سازی شده‌ی Post Layout در حالت بزرگ نمایی شده برای مشاهده‌ی بهتر زمان صعود و نزول

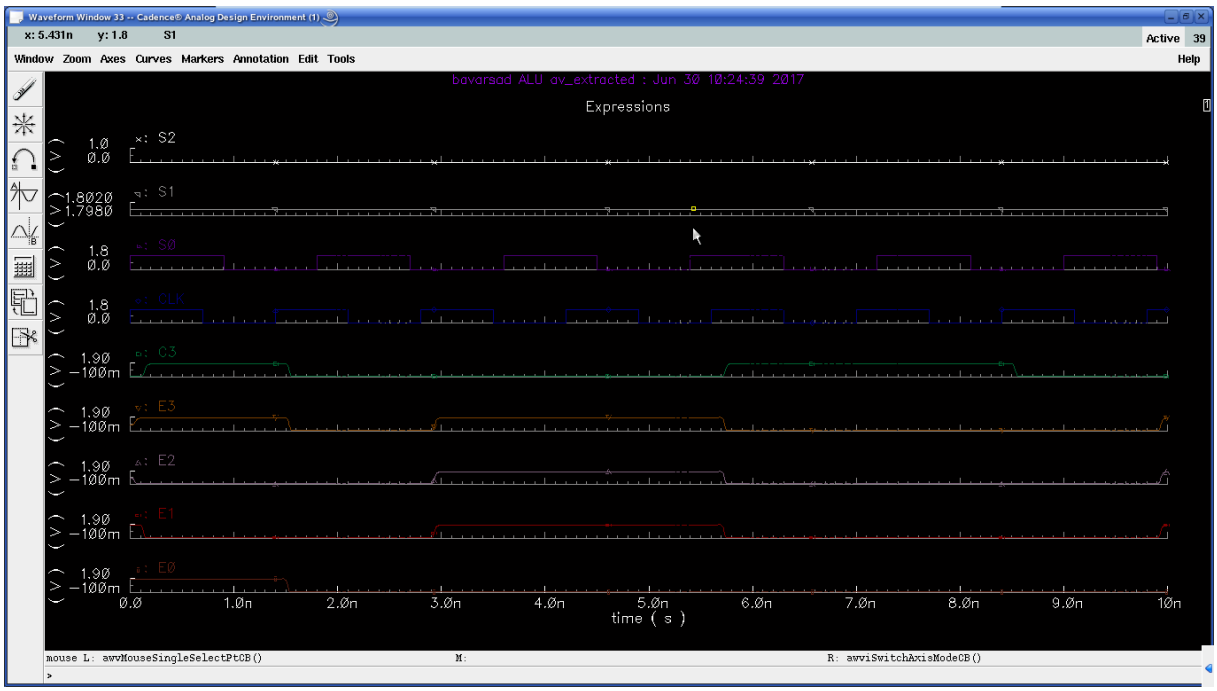
حداکثر فرکانس

بیشترین تاخیر مدار در حالتی است که دستورات جمع و تفریق را انجام می‌دهد. بنابر این ما برای به دست آوردن حداکثر فرکانس کاری، مدار را در بدترین حالت امتحان کردیم.برای اطمینان مدار را در شرایط بهتری نیز آزمایش کردیم که در حالت دوم فرکانس بالاتری به دست آمد. در هر حالت دو عدد ثابت به A و B اختصاص داده شد و فرکانس CLK تغییر داده شد همچنین به صورت متوالی کد دستور را بین دستورات A+B و A-B تغییر دادیم. مشاهده شد که هنگامی که دوره تناوب CLK از حدود ۱/۹۴ns بیشتر می‌شود مدار به درستی کار نمی‌کند. با این حساب حداکثر فرکانس کاری مدار حدود ۵۱۵MHz به دست می‌آید.

حالت	ورودی‌ها			مقادیر مورد انتظار		دوره تناوب و فرکانس	
	A	B	CI	A+B	A-B	حداقل دوره تناوب	حداکثر فرکانس
۱	1111	1111	1	11111	10000	۱/۹۴ ns	۵۱۵MHz
۲	0111	0111	0	01110	10000	۱/۴ ns	۷۲۰ MHz



شکل (۳۱) شبیه سازی مدار در فرکانس ۵۱۵ MHz (حالت اول)



شکل (۳۲) شبیه سازی مدار در فرکانس ۷۲۰ MHz (حالت دوم)