به نام خداوند بخشنده‌ی مهربان

گزارش کار آزمایش دوم آز معماری کامپیوتر

ضرب‌کننده‌ی ممیز ثابت

سید پارسا نشایی - 98106134

محمدطه جهانی‌نژاد - 98101363

امیرحسین باقری - 98105621

نیم‌سال سوم ۱۳۹۹-۱۴۰۰

دانشگاه صنعتی شریف

دانشکده‌ی مهندسی کامپیوتر

فهرست مطالب

مقدمه

در این آزمایش، یک ضرب‌کننده‌ی بدون علامت دودویی ۴ بیتی (در طی ایمیل‌نگاری با استاد محترم، بیان شد که ضرب بدون علامت، کفایت می‌کند) طراحی شده است. در مدار طراحی شده در نهایت، دو عدد ورودی ۴ بیتی در مبنای ۲ (به نام‌های A و B که به ترتیب، مضروب و مضروب‌فیه هستند) به مدار داده شده و سپس سیگنال ورودی Start توسط کاربر، یک می‌شود؛ پس از این تغییر، ضرب‌کننده شروع به کار کرده و با الگوریتم shift and add (که از دروس مدارهای منطقی، ساختار و زبان کامپیوتر و نیز معماری کامپیوتر با آن آشنا هستیم)، حاصل‌ضرب دو عدد ورودی را در طی چند کلاک، محاسبه کرده و آن را روی خروجی ۸ بیتی مدار (C) قرار می‌دهد و سیگنال خروجی End را یک می‌کند که به معنای اعلان پایان عملیات مدار از سوی مدار است. پس از طراحی مدار، درستی کارکرد آن با ورودی‌های مختلف ارزیابی شده است؛ به گونه‌ای که انتظار می‌رود پس از اجرای مدار، به ازای هر دو عدد ۴ بیتی مبنای ۲ که به مدار داده شود، حاصل‌ضرب ۸ رقمی آن دو، در خروجی مدار، مشاهده شود. جزئیات پیاده‌سازی بخش‌های مختلف مدار، در ادامه آمده است.

در دستور کار نوشته شده است: «هنگام طراحی مدار به کمک شبیه‌ساز سعی کنید از تراشه‌های TTL موجود در کتابخانه شبیه‌ساز و آزمایشگاه استفاده کنید. بدین ترتیب هنگام پیاده‌سازی عملی نیازی به تغییر مدار برای استفاده از تراشه‌های موجود نیست» که در طی ایمیل‌نگاری با استاد محترم، مقرر شد این قسمت در ترم کنونی نادیده گرفته شود.

طراحی مدار، شبیه‌سازی و بررسی درستی عملکرد آن، همان‌گونه که خواسته شده بود، در نرم‌افزار Proteus انجام شده است. نسخه‌ای از این نرم‌افزار که از آن استفاده شده، **نسخه‌ی 8.12** است.

توجه مهم: برای محاسبه‌ی حاصل‌ضرب در مدار، با توجه به روش در پیش گرفته شده، حتما باید اگر Start در ابتدا یک بوده است، **ابتدا صفر شود**، ورودی‌ها تنظیم شده و سپس Start یک شود تا حاصل پس از چند کلاک در خروجی نمایش داده شود.

طراحی و بلوک دیاگرام‌های طرح اولیه

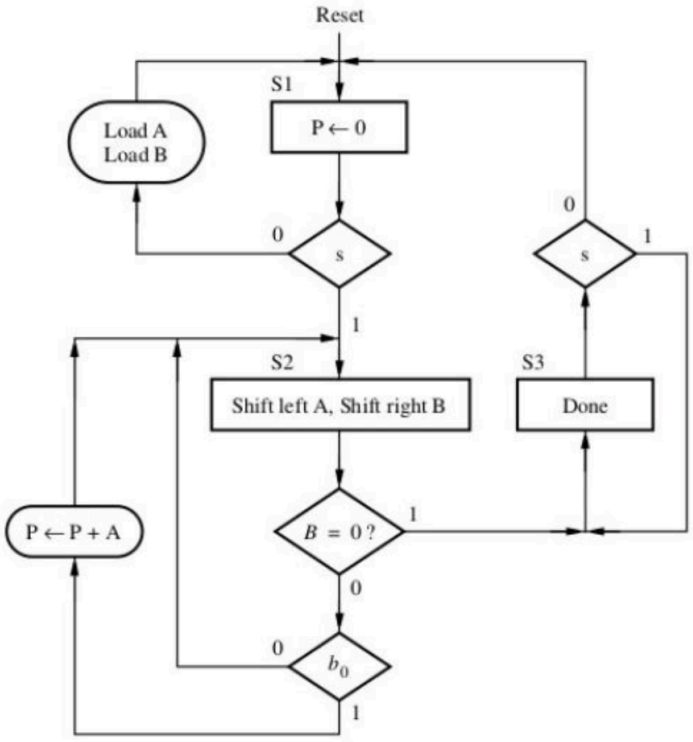
بلوک‌دیاگرام‌های این بخش، در صورت امکان، بر اساس سایر بلوک‌های موجود، ترسیم شده‌اند و در غیر این صورت، اگر از گیت‌های پایه تشکیل می‌شدند، بر اساس آن‌ها ترسیم شده‌اند.
به علت روش انجام ضرب در این مدار که در دستور کار نیز مشخص شده است، مدار به صورت سنکرون – یعنی با سیگنال پالس ساعت – کار خواهد کرد و در نتیجه از یک Clock Generator استفاده شده است. جهت ذخیره اعداد ورودی داده شده در طی چند پالس ساعت، نیاز به شیفت‌رجیستر داریم. برای ذخیره‌ی مضروب‌فیه، از یک شیفت رجیستر ۴ بیتی و برای ذخیره‌ی مضروب و حاصل‌ضرب از یک شیفت رجیستر ۸ بیتی استفاده شده است. دلیل این انتخاب، آن است که عملیات ضرب حداکثر ۴ کلاک طول می‌کشد. در طول این ۴ کلاک، هر بار مضروب به چپ و مضروب‌فیه به راست شیفت داده می‌شود؛ بنابراین برای جلوگیری از سرریز شدن بیت‌های معنادار مضروب، باید از یک شیفت رجیستر ۸ بیتی استفاده شود و این در حالی است که ۴ بیت فضا برای ذخیره سازی مضروب‌فیه کفایت می‌کند. هم‌چنین، چون حاصل ضرب دو عدد ۴ بیتی، حداکثر ۸ بیتی است، برای ذخیره سازی آن ۸ بیت در نظر گرفته می‌شود.

طراحی الگوریتم Shift and Add

روش shift and add عملا همان روش ضربی است که در دبستان آموخته‌ایم. برای ضرب دو عدد ۴ بیتی در یک‌دیگر به کمک این روش، حداکثر به ۴ مرحله نیاز است؛ در هر مرحله، مضروب را یک بیت به چپ shift داده تا ارزش آن دو برابر شود (همانند کاری که روی کاغذ در ضرب اعداد ده‌دهی انجام شده و در هر مرحله، ارزش مضروب ده برابر می‌شود). سپس، کل مضروب، در یک بیت متناظر از مضروب‌فیه «ضرب» می‌شود. در سیستم دودویی، چون هر بیت برابر ۰ یا ۱ است، حاصل‌ضرب هر بار به ترتیب یا ۰ و یا خود بیت متناظر از مضروب خواهد بود؛ بنابراین، معادل عمل ضرب در این سیستم، عمل AND بیتی است.

در این الگوریتم، در مرحله i ام از محاسبات، باید مضروب در بیت i ام از مضروب فیه AND شود. برای اینکه اندکی پیچیدگی مدار کاهش یابد، می‌توان به جای پیدا کردن بیت i ام، در هر مرحله مضروب فیه را به راست شیفت داد تا بیت مدنظر در جایگاه کم‌ارزش‌ترین بیت عدد حاصل قرار گیرد و سپس کافیست که در انتهای هر مرحله، مضروب در بیت اول مضروب‌فیه AND شود. در نهایت، تمامی اعداد محاسبه شده باید با یک‌دیگر جمع شوند تا حاصل ضرب تولید شود؛ برای انجام این کار، در هر مرحله، حاصل ضربی که تاکنون به دست آورده شده، با عدد محاسبه شده در این مرحله جمع شده و در رجیستر حاوی حاصل‌ضرب، قرار داده می‌شود.

برای مشاهده‌ی جزئیات بیش‌تر الگوریتم مرسوم به کار گرفته شده، ASM Chart موجود در شکل (۱) راه‌گشاست (P رجیستری است که در آن حاصل عملیات قرار داده می‌شود).



شکل ۱ – نمودار ASM الگوریتم مرسوم Shift and Add (لینک)

طراحی مسیر داده و واحد کنترل

برای طراحی مدار متناظر با ASM Chart فوق، بر اساس مطالبی که از دروس مدارهای منطقی و طراحی سیستم‌های دیجیتال فرا گرفته‌ایم، مدار در دو بخش مسیر داده (Data Path) و واحد کنترل (Control Unit) طراحی شده است.

موارد مهم برای تعبیه در بخش مسیر داده، عبارت هستند از:

- یک شیفت رجیستر با قابلیت بارگذاری موازی برای ذخیره مضروب
- یک شیفت رجیستر با قابلیت بارگذاری موازی برای ذخیره مضروب‌فیه
- یک رجیستر با قابلیت بارگذاری موازی برای ذخیره حاصل‌ضرب
- یک جمع‌کننده
- یک ماژول AND کننده‌ی بیتی (که معادل با واحد ضرب‌کننده در سیستم دهدهی دبستانی است؛ پیش‌تر، چرایی AND کردن به جای ضرب کردن، توضیح داده شده است)

در این الگوریتم و بر اساس شکل (۱)، چون مضروب همان A و مضروب‌فیه همان B است، شیفت‌رجیستر استفاده شده برای ذخیره‌سازی مضروب، باید قابلیت شیفت به چپ و شیفت رجیستر استفاده شده برای ذخیره‌سازی مضروب‌فیه، باید قابلیت شیفت به راست را داشته باشد. جمع‌کننده‌ی مورد استفاده، باید ۸ بیتی باشد (تا بتواند ورودی‌های حداکثر ۸ بیتی که یکی از این ورودی‌ها از خروجی شیفت‌رجیستر حاصل‌ضرب که ۸ بیتی است می‌آید را جمع کرده و خروجی ۸ بیتی تولید کند که بتواند آن را در شیفت‌رجیستر حاصل‌ضرب که ۸ بیتی است، بارگذاری موازی کند) که با استفاده از دیگر ماژول‌های آماده، ساخته شده و جلوتر توضیح داده شده است. تراشه‌های استفاده شده برای این شیفت‌رجیسترها و نیز طرح کلی مدار، در بخش‌های بعدی آمده است.

ماژول AND کننده‌ی بیتی، طبق توضیحات داده شده، باید ۸ بیت ورودی (مضروب) دریافت کند و آن‌را با یک بیت ورودی دیگر (کم‌ارزش‌ترین بیت مضروب‌فیه) AND کند و در خروجی ۸ بیت حاصل را نشان دهد. ساخت

این ماژول با ۸ گیت AND ممکن است که هر یک از این گیت‌ها، کم‌ارزش‌ترین بیت مضروب‌فیه را با یکی از بیت‌های ورودی AND کرده و حاصل را در بیت متناظر از خروجی نشان می‌دهد.

واحد کنترل

بخش Control Unit اصولا باید وضعیت (State) کنونی را ذخیره کند و در هر مرحله با توجه به سیگنال‌های ورودی و وضعیت کنونی، سیگنال‌هایی را برای کنترل مسیر داده در خروجی تولید کند. با توجه به ASM داده شده و تعداد State box ها، در کل سه حالت مختلف داریم:

- **حالت اولیه** که ورودی‌ها در آن بارگذاری می‌شوند و ورودی Start هنوز فعال نشده است (با فعال شدن Start، مدار به حالت محاسبه می‌رود) و متناظر با جعبه حالت P <- 0 است
- **حالت محاسبه** که در آن، با توجه به ورودی‌های داده شده، محاسبات انجام می‌شوند، شامل یک حلقه به قبل از State box متناظر با خود (Shift left A, Shift right B) است و پس از پایان محاسبات، مدار به حالت نهایی می‌رود
- **حالت نهایی** که در آن، پاسخ آماده شده، سیگنال Finish فعال می‌شود و مادامی که سیگنال Start فعال باشد، خروجی‌ها بدون تغییر می‌مانند (و در صورتی که صفر شود تا بخواهد بعدا مجددا یک شود و نشانگر شروع محاسبه‌ی بعدی باشد، مدار به حالت اولیه برمی‌گردد) و متناظر با جعبه حالت Done است

بنا بر توضیحات فوق، به دو سیگنال Finish (وقتی یک می‌شود که پایان عملیات مشخص شود) و Shift (وقتی یک می‌شود که این معنا را منتقل کند که باید shift انجام شود) نیاز است که این دو سیگنال، به ترتیب با نام‌های F و S در خروجی ماژول واحد کنترل آورده شده‌اند. هم‌چنین، چون ۳ حالت مختلف وجود دارد، می‌توان از

دو D-flip flop (که توانایی نمایش اعداد صفر تا $2^2 - 1 = 3$ را دارند)، برای ذخیره‌سازی حالت کنونی استفاده کرد.

توضیحات مربوط به پیاده سازی دقیق ماژول‌ها در ادامه آمده است.

طراحی ماژول واحد کنترل

مطابق آن‌چه از مدار منطقی می‌دانیم، می‌توان به حالت‌ها، کد تخصیص داد. حالت اولیه با کد صفر (۰۰ باینری)، حالت محاسبه با کد یک (۰۱ باینری) و حالت نهایی با کد دو (۱۰ باینری) در نظر گرفته می‌شود. این کدهای خروجی Q_1Q_0 در نظر گرفته می‌شوند (و با دو عدد D-FlipFlop نگه‌داری می‌شوند)، یعنی Q_1Q_0 یا برابر ۰۰ است، یا برابر ۰۱ است و یا برابر ۱۰ است.

- در حالت اولیه، نیاز به شیفت نیست و پایان عملیات نیز صورت نگرفته است، پس F = 0 و S = 0.
- در حالت محاسبه، نیاز به شیفت وجود دارد، ولی پایان عملیات صورت نگرفته است، پس F = 0 و S = 1.
- در حالت نهایی، نیاز به شیفت نیست، ولی پایان عملیات نیز صورت گرفته است، پس F = 1 و S = 0.

در نتیجه،

- در حالت Q_1Q_0 برابر 10 (و 11 که don't care است و در نتیجه خودمان آن را در نظر می‌گیریم تا بتوان مدار را ساده‌تر کرد)، F برابر یک است، پس می‌توان در نظر گرفت $F = Q_1Q_0'$.
- در حالت Q_1Q_0 برابر 01، S برابر یک است، پس می‌توان در نظر گرفت $S = Q_1'Q_0$.

حال که خروجی‌های مدار FSM واحد کنترل بر اساس حالت کنونی آن به دست آمدند، باید تعیین شود که حالت بعدی چگونه از روی حالت قبلی و نیز ورودی‌های مدار به دست می‌آید. سه سیگنال به عنوان ورودی‌های واحد کنترل، در نظر گرفته شده‌اند:

- سیگنال پالس ساعت (کلاک)؛ چون این مدار فلیپ فلاپ دارد، به پالس ساعت نیاز است. طبیعتا این سیگنال در معادلات تعیین حالت بعدی، اثری ندارد.
- سیگنال Start؛ این سیگنال، در اصل، از ورودی کاربر گرفته می‌شود و برای «خروج از حالت اولیه و وارد شدن به حالت محاسبه» و هم‌چنین «خروج از حالت نهایی و ورود به حالت اولیه» به آن نیاز است.
- سیگنال مشخص کننده‌ی b = 0؛ در پایان عملیات ضرب، ورودی b یا همان مضروب‌فیه برابر صفر می‌شود و در ورودی‌های واحد کنترل، سیگنالی که مشخص کند آیا مضروب فیه برابر صفر شده یا خیر، مورد نیاز است. طبیعتا، مقدار OR بیت به بیت ورودی b مشخص می‌کند که آیا مقدار داخل این رجیستر صفر شده است یا خیر (زیرا اگر حتی یک بیت b ناصفر بوده و در نتیجه b خود ناصفر باشد، OR همه‌ی بیت‌های b برابر یک خواهد بود و پس می‌توان گفت اگر OR همه‌ی بیت‌های b (Reduction OR عدد b) برابر یک باشد، b ناصفر و اگر OR همه‌ی بیت‌های b برابر صفر باشد، b صفر است). این ورودی، Or_b نام‌گذاری شده است (و وقتی یک است که b ناصفر است).

در نتیجه‌ی بندهای فوق، ورودی‌های تعیین کننده‌ی حالت بعد، Start و Or_b هستند و:

- اگر حالت کنونی ۰۰ (اولیه) باشد،
  - اگر ورودی Start صفر باشد، مستقل از Or_b، مدار در حالت اولیه می‌ماند و در نتیجه حالت بعدی ۰۰ است.
  - اگر ورودی Start یک باشد، مستقل از Or_b، مدار محاسبه را شروع کرده (به حالت محاسبه می‌رود) و در نتیجه حالت بعدی ۰۱ است.
- اگر حالت کنونی ۰۱ (محاسبه) باشد،
  - اگر ورودی Or_b صفر باشد، مستقل از Start، به این معناست که b صفر شده، محاسبه با صفر شدن مضروب‌فیه (b) به پایان رسیده و لازم است به حالت نهایی رفته و در نتیجه حالت بعدی ۱۰ است.

- اگر ورودی Or_b صفر باشد، مستقل از Start، به این معناست که b (هنوز) ناصفر است، محاسبه هنوز با صفر شدن مضروب‌فیه (b) به پایان نرسیده و لازم است دوباره دور بعد محاسبه اجرا شده و در اصل مدار باز هم به حالت محاسبه برود و در نتیجه حالت بعدی ۰۱ است.
- اگر حالت کنونی ۱۰ (نهایی) باشد،
- اگر ورودی Start صفر باشد، مستقل از Or_b، به این معناست که پس از اتمام محاسبه، Start که در ابتدا یک شده بود، صفر شده و در نتیجه مدار باید منتظر بماند تا دوباره Start پس از بازتنظیم ورودی‌های جدید، یک شود و بتواند محاسبه را آغاز کند و این انتظار، لازم است در حالت اولیه رخ دهد، در نتیجه مدار لازم است به حالت اولیه برود و در نتیجه حالت بعدی ۰۰ است.
- اگر ورودی Start صفر باشد، مستقل از Or_b، به این معناست که پس از اتمام محاسبه، Start که در ابتدا یک شده بود، هنوز صفر نشده و در نتیجه مدار باید منتظر بماند تا Start صفر شود که به حالت اولیه برود و منتظر یک شدن مجدد آن بماند (بر اساس طراحی انجام شده و نیز ASM Chart موجود در شکل (۱)، این استدلال آورده شده است)، پس مدار لازم است در حالت نهایی (Done) بماند و در نتیجه حالت بعدی ۱۰ است.

بر اساس موارد فوق، جدول تغییرات حالات مدار بر اساس ورودی‌ها به صورت مشخص شده در جدول (۱) است.

Or_b	$Start$	Q_1Q_0	$Q_1^+Q_0^+$
X	0	00	00
X	1	00	01
1	X	01	01
0	X	01	10
X	1	10	10

Or_b	$Start$	Q_1Q_0	$Q_1^+Q_0^+$
X	0	10	00

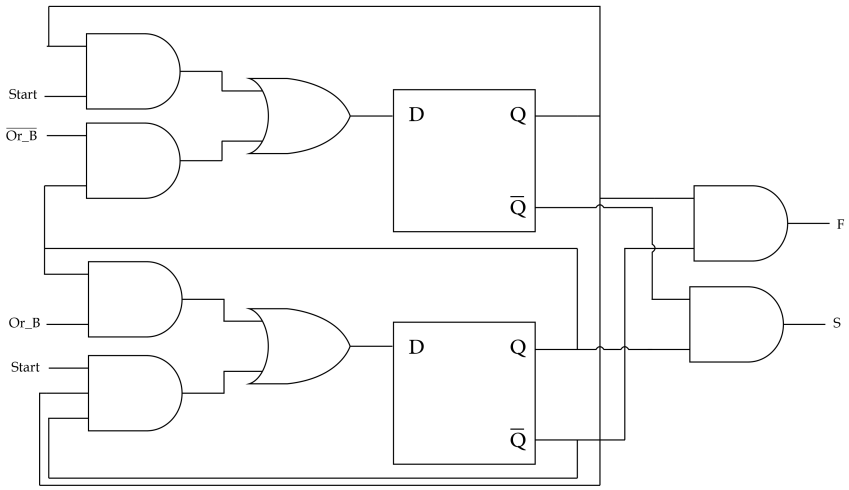
جدول ۱ – تغییرات حالات مدار واحد کنترل بر اساس ورودی‌های آن

(علامت + در بالای هر حرف به معنای «حالت بعدی» بوده و X نیز به معنای don't care است)

بر اساس جدول فوق (حالت‌های بررسی نشده، don't care اند):

- تنها وقتی قرار است Q_1 حالت بعدی، یک شود که $Start\,.\,Q_1 + (Or\_b)'\,.\,(Q_0)'$ یک باشد (و در نتیجه، $Q_1^+ = Start\,.\,Q_1 + (Or\_b)'\,.\,(Q_0)'$)، یعنی وقتی:
  - Start یک باشد و Q_1 از حالت قبلی، یک باشد (یعنی $Start\,.\,Q_1$ و معادل سطر دوم از پایین جدول)
  - یا Or_b صفر باشد (نقیض یک باشد) و Q_0 از حالت قبلی، یک باشد (یعنی $(Or\_b)'\,.\,Q_0$ و معادل سطر سوم از پایین جدول)
- تنها وقتی قرار است Q_0 حالت بعدی، یک شود که $Start\,.\,(Q_0)'\,.\,(Q_1)' + Or\_b\,.\,Q_0$ یک باشد (و در نتیجه، $Q_0^+ = Start\,.\,(Q_0)'\,.\,(Q_1)' + Or\_b\,.\,Q_0$)، یعنی وقتی:
  - Or_b یک باشد و Q_0 از حالت قبلی، یک باشد (یعنی $Or\_b\,.\,Q_0$ و معادل سطر چهارم از پایین جدول)
  - یا Start یک باشد، Q_0 از حالت قبلی، صفر باشد (یعنی نقیض یک باشد) و Q_1 از حالت قبلی، صفر باشد (یعنی نقیض یک باشد)، یعنی $Start\,.\,(Q_0)'\,.\,(Q_1)'$ (و معادل سطر پنجم از پایین جدول)

بنا به توضیحات داده شده و قرار دادن گیت‌های منطقی برای تولید عبارات فوق (به گونه‌ای که Q_0^+ و Q_1^+ در ورودی‌های D ی فلیپ‌فلاپ‌ها قرار داشته باشند)، بلوک‌دیاگرام واحد کنترل (Control Unit)، به صورت مشخص شده در شکل (۲) خواهد بود.

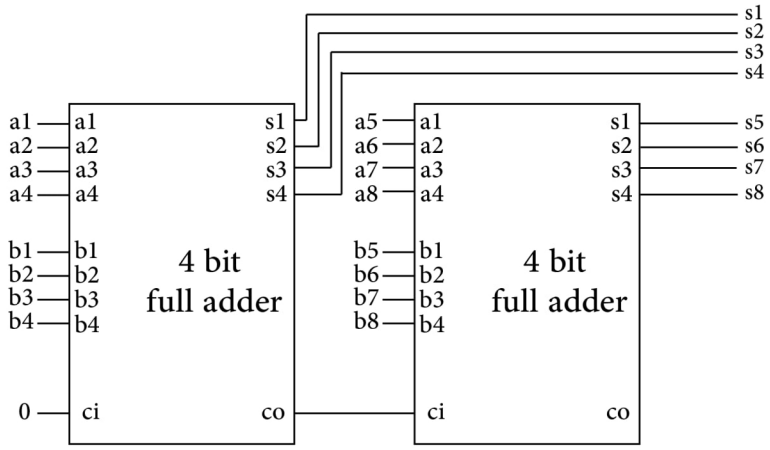


شکل ۲ – بلوک‌دیاگرام واحد کنترل

فلیپ‌فلاپ بالا مربوط به Q_1 و فلیپ‌فلاپ پایین مربوط به Q_0 است

طراحی ماژول جمع‌کننده‌ی مسیر داده

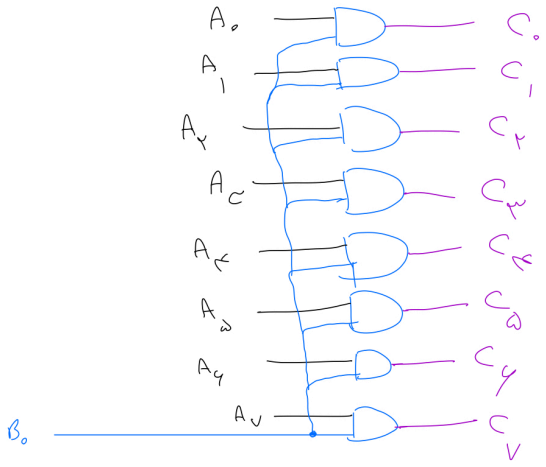
با اتصال آبشاری (Cascade) دو عدد جمع‌کننده‌ی چهار بیتی (مشابه آن‌چه در دروس مدارهای منطقی، ساختار و زبان کامپیوتر و نیز معماری کامپیوتر خوانده‌ایم)، یک جمع‌کننده‌ی هشت بیتی ساخته می‌شود. نحوه اتصال به این شکل است که خروجی CO (بیت نقلی تولید شده)ی ماژول اول، به ورودی CI (بیت نقلی مصرفی / ورودی) ماژول دوم، متصل می‌شود. به این صورت، ۴ بیت کم‌ارزش‌تر در ماژول اول محاسبه شده و ۴ بیت پرارزش‌تر در ماژول دوم محاسبه می‌شوند. بلوک‌دیاگرام این ماژول، در شکل (۳) آمده است.



شکل ۳ – بلوک‌دیاگرام ماژول جمع‌کننده‌ی مسیر داده

طراحی ماژول And بیتی (Bitwise AND) مسیر داده

این زیرمدار، تنها حاوی ۸ واحد AND است؛ این ماژول، کم‌ارزش‌ترین بیت ورودی B (مضروب‌فیه) را گرفته، تک‌تک بیت‌های ورودی A (مضروب) را در آن AND کرده و حاصل را به صورت ۸ بیت خروجی می‌دهد. بلوک‌دیاگرام این ماژول، در شکل (۴) آمده است.



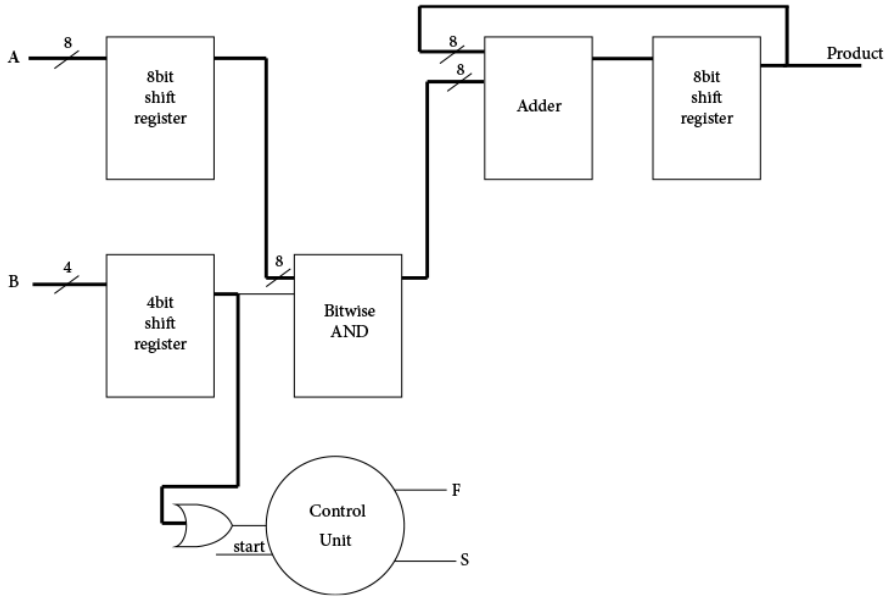
شکل ۴ – بلوک دیاگرام ماژول AND کننده‌ی بیتی مسیر داده

حال کافی است طبق توضیحات فوق، مسیر داده (Data Path) و واحد کنترلی (Control Unit) شکل داده شده و کنار هم قرار داده شوند (که شکل نهایی مدار در بخش‌های بعدی آمده است). همانطور که بالاتر عنوان شد، مسیر داده از سه شیفت‌رجیستر و دو زیر مدار که جزئیات تشکیل هر یک در فوق نوشته شده، تشکیل شده است. افزون بر موارد فوق، کافی است سیگنال‌های کنترلی واحد کنترل، به ورودی شیفت‌رجیسترها متصل شوند. زمانی که مدار در حالت اولیه باشد، اتفاقات زیر باید رخ دهند:

- خروجی مدار ریست شود
- ورودی‌ها داخل رجیسترهای مخصوص بارگذاری موازی شوند

در حالت محاسبه، باید خروجی با استفاده از خروجی بخش جمع‌کننده، بارگذاری موازی شود و هم‌چنین ورودی‌ها یک واحد شیفت پیدا کنند.

جزئیات بیش‌تر از نحوه‌ی اتصال واحد کنترل و مسیر داده به هم، در بخش‌های بعدی آمده است. بلوک‌دیاگرام کلی مسیر داده و نیز اتصال مسیر داده و واحد کنترل، در شکل (۵) آمده است.



شکل ۵ – بلوک‌دیاگرام کلی مسیر داده و نیز اتصال مسیر داده و واحد کنترل

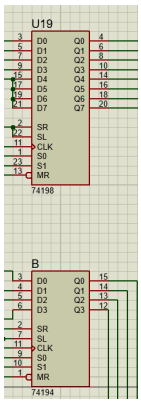
شرح انجام آزمایش و توضیح پیاده‌سازی

طراحی ماژول‌ها به کمک ساختن Sub Module در Proteus انجام گرفته است. برای تولید مقدار منطقی صفر، از Ground و برای تولید مقدار منطقی یک، از Power استفاده شده است.

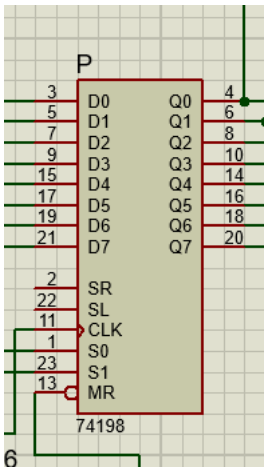
مسیر داده

شیفت‌رجیسترها

پیش‌تر گفته شد که شیفت‌رجیستر استفاده شده برای ذخیره‌سازی مضروب، باید قابلیت شیفت به چپ و شیفت رجیستر استفاده شده برای ذخیره سازی مضروب‌فیه، باید قابلیت شیفت به راست را داشته باشد. ماژول‌های استفاده شده برای این دو، ماژول‌های شماره ۷۴۱۹۸ و ۷۴۱۹۴ هستند که به ترتیب ۸ بیتی و ۴ بیتی بوده و قابلیت شیفت دوطرفه و بارگذاری موازی را نیز دارند. برای ذخیره‌سازی حاصل‌ضرب هم از همان ماژول ۷۴۱۹۸ استفاده شده است. این ماژول‌ها در شکل‌های (۶) و (۷) قابل مشاهده‌اند. در حالت محاسبه، باید خروجی با استفاده از خروجی بخش جمع‌کننده بارگذاری موازی شود و همچنین ورودی‌ها یک واحد شیفت پیدا کنند. با توجه به داکیومنت ماژول ۷۴۱۹۸ و ۷۴۱۹۴ (که البته مشابهت‌های زیادی نیز دارند)، ورودی‌های S0، SL، SR و S1 شیفت‌رجیسترها، با استفاده از دو سیگنال S و F تولید شده است. مستند (document) این رجیستر، از این لینک، مطالعه و به کار گرفته شده است.



شکل ۶ – ماژول‌های ذخیره‌سازی
مضروب و مضروب‌فیه

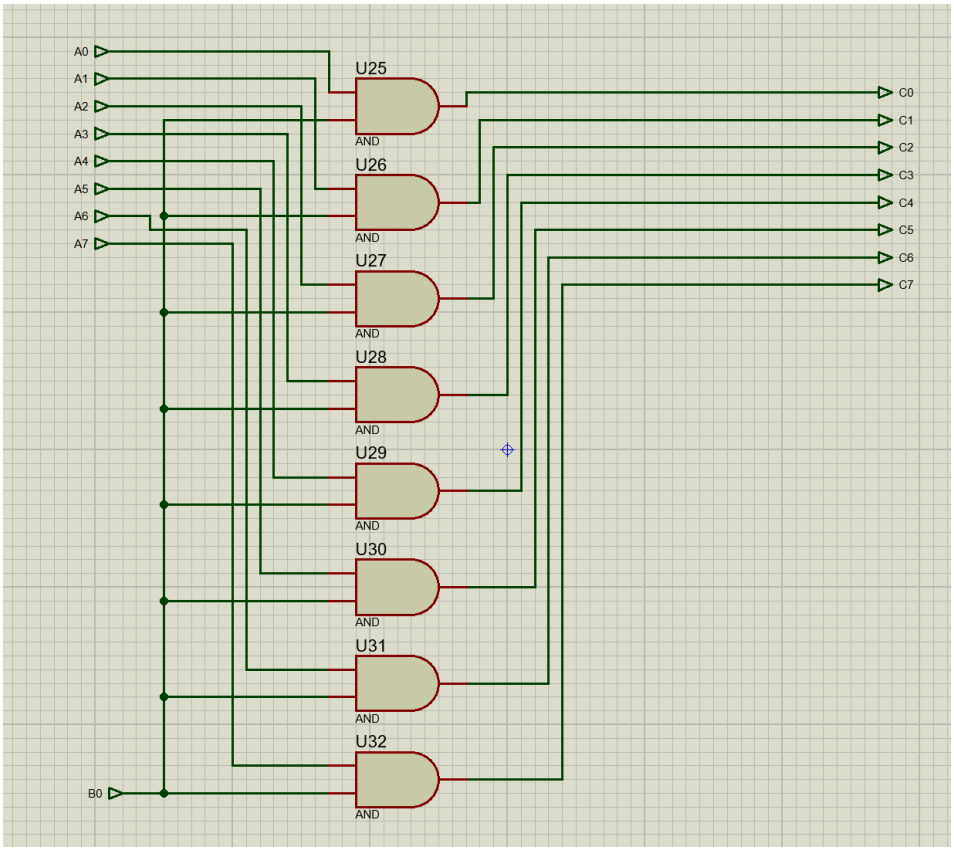


شکل ۷ – ماژول ذخیره‌سازی
حاصل‌ضرب

در تمام ماژول‌ها، ورودی‌ها و خروجی‌ها توسط Terminals در Proteus و مشابه آن‌چه در آزمایش قبلی داشتیم، قرار داده شده و سپس از بیرون ماژول، مشابه آن‌چه در جلسه‌ی نخست کلاس این ترم دیدیم، map شده‌اند.

ماژول AND کننده‌ی بیتی

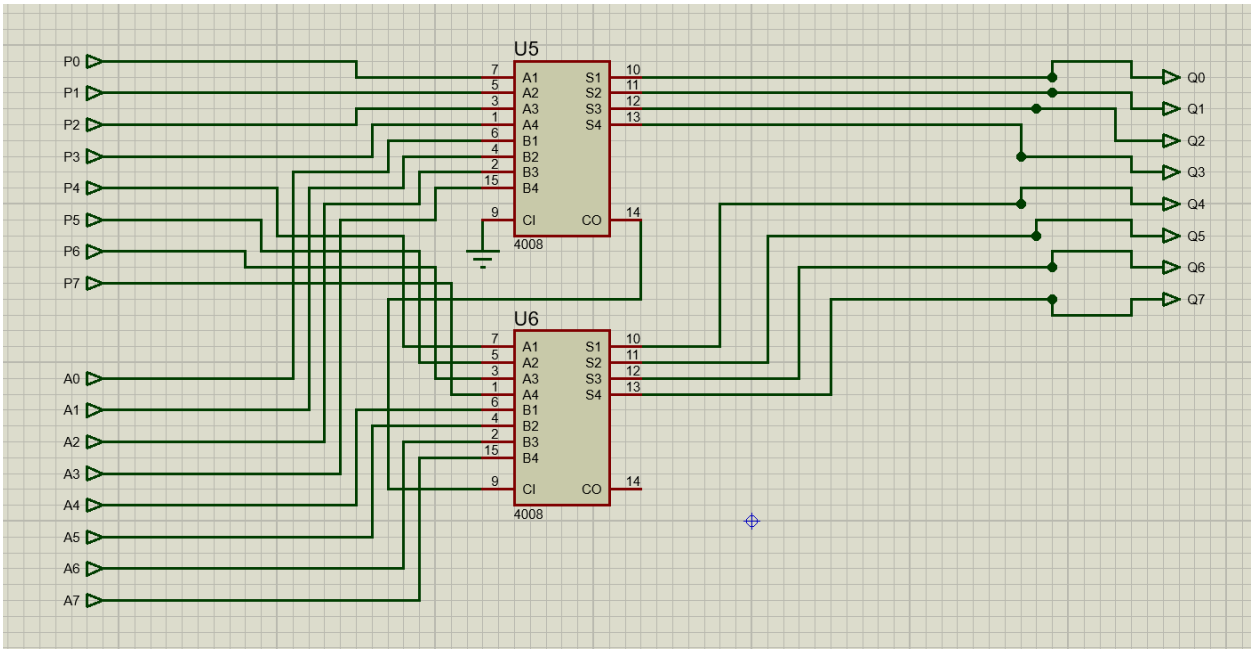
برای طراحی این ماژول، از ۸ عدد گیت AND در Proteus و اتصالات آن‌ها، مطابق توضیحاتی که پیش‌تر داده شده، استفاده شده است. نمایی از این ماژول طراحی شده، در شکل (۸) موجود است.



شکل ۸ – ماژول AND کننده‌ی بیتی

ماژول جمع‌کننده‌ی ۸ بیتی

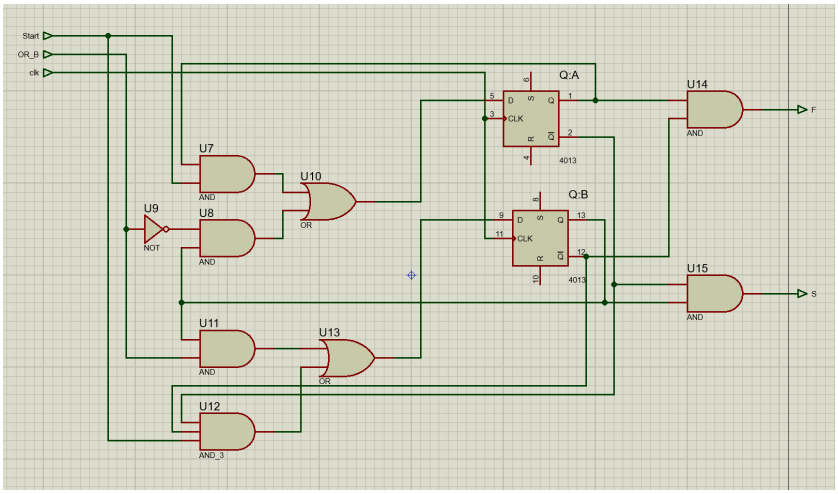
برای طراحی این ماژول، از دو عدد تمام‌جمع‌کننده‌ی ۴ بیتی (شماره تراشه ۴۰۰۸) در Proteus و اتصال آبشاری آن‌ها (اتصال خروجی CO مربوط به ماژول جمع‌کننده‌ی مربوط به ۴ بیت کم‌ارزش، به ورودی CI مربوط به ماژول جمع‌کننده‌ی مربوط به ۴ بیت پرارزش)، مطابق توضیحاتی که پیش‌تر داده شده، استفاده شده است. نمایی از این ماژول طراحی شده، در شکل (۹) موجود است.



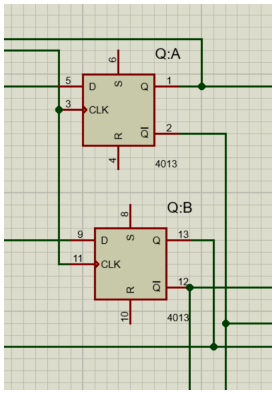
شکل ۹ – ماژول جمع‌کننده‌ی ۸ بیتی

واحد کنترل

برای طراحی Control Unit، از دو دی‌فلیپ‌فلاپ و تعدادی گیت پایه، مطابق بلوک‌دیاگرام ارائه‌شده در قبل، استفاده شده است. تصویری از طراحی واحد کنترل در Proteus، در شکل‌های (۱۰) و (۱۱) آمده است.



شکل ۱۰ – طراحی واحد کنترل



شکل ۱۱ – فلیپ‌فلاپ‌های واحد کنترل

مدار اصلی

برای طراحی مدار اصلی، Control Unit و واحدهای مسیر داده کنار هم قرار گرفته و اتصالات‌شان برقرار شده است. ورودی A در چهار بیت A0 تا A3 (توسط Logic Toggle ها) به مدار داده شده که A0 بیت کم‌ارزش و A3 بیت پرارزش آن است. ورودی B در چهار بیت B0 تا B3 (توسط Logic Toggle ها) به مدار داده شده که B0 بیت کم‌ارزش و B3 بیت پرارزش آن است. ورودی Start نیز همان ورودی‌ای است که کاربر برای اعلان شروع به کار به مدار، آن را یک می‌کند (توسط Logic Toggle). خروجی‌های مدار، خروجی ۸ بیتی C (که در هشت بیت C0 تا C7 – (توسط Logic Probe (Big ها) – از مدار خارج شده که C0 بیت کم‌ارزش و C7 بیت پرارزش آن است) و نیز سیگنال خروجی End (که توسط Logic Probe (Big ها قرار داده شده و وقتی توسط مدار یک می‌شود که عملیات ضرب به پایان رسیده باشد) هستند.

در طراحی این بخش، ورودی‌های مدار به ورودی parallel load (D های) رجیسترها (دو رجیستر A و B که شامل عملوندهای ضرب بوده و در بالا سمت چپ مدار هستند، و هم‌چنین رجیستر سمت راست که شامل مقدار حاصل‌ضرب است) متصل شده (تا بتوان بارگذاری موازی را انجام داد) و سیگنال‌های کنترلی ورودی رجیسترها نیز طبق مستندات datasheet شیفت‌رجیسترها تعیین شده است (و نیز، چگونگی تعیین مقدار ورودی پایه‌های کنترلی رجیسترها، در جدول (۲) آورده شده است). چهار بیت ورودی A به چهار بیت اول ورودی رجیستر ۸ بیتی

متصل شده‌اند و به باقی بیت‌های پرارزش‌تر این رجیستر، صفر متصل شده است، زیرا صفر در سمت چپ عدد، ارزشی ندارد.

MODE SELECT TABLE

INPUTS				RESPONSE
$\overline{MR}$	CP	S_0*	S_1*	
L	X	X	X	Asynchronous Reset; Outputs = LOW
H	↑	H	H	Parallel Load; $P_n \rightarrow Q_n$
H	↑	L	H	Shift Right; $D_{SR} \rightarrow Q_0$, $Q_0 \rightarrow Q_1$, etc.
H	↑	H	L	Shift Left; $D_{SL} \rightarrow Q_7$, $Q_7 \rightarrow Q_6$, etc.
H	X	L	L	Hold

*Select inputs should be changed only while CP is HIGH
H = HIGH Voltage Level
L = LOW Voltage Level
X = Immaterial

جدول ۲ – کارکرد مقادیر مختلف ورودی پایه‌های کنترلی رجیستر ۷۴۱۹۸ (و مشابها ۷۴۱۹۴، با این تفاوت که به جای ۸ بیت از ۴ بیت ذخیره‌سازی بهره می‌برد)

ورودی‌های SR و SL هر دو رجیستر عملوندهای ضرب، صفر تعیین شده‌اند تا با شیفت دادن، مقدار صفر به صورت سریال وارد شود (که مطلوب بوده و با الگوریتم هم‌خوانی دارد). ورودی CLK رجیسترها نیز به تولیدکننده‌ی پالس ساعت متصل شده است. ورودی MR (Master Reset) این دو رجیستر نیز به Power متصل شده‌اند تا به دلیل Active Low بودن آن‌ها، ریست انجام نپذیرد (و در صورت نیاز به صفر کردن، با parallel load و یا شیفت‌های متوالی که پس از مدتی B را صفر می‌کنند، این کار صورت گیرد).

با توجه به آن‌که می‌دانیم کار A شیفت به چپ و کار B شیفت به راست است، با در نظر گرفتن خروجی‌های واحد کنترل، داریم (حالت‌های don’t care نوشته نشده‌اند):

- اگر F=0 و S=0 آن‌گاه در حالت اولیه هستیم و باید Parallel Load انجام دهیم، پس طبق جدول (۱)، باید S0=S1=1 باشد.

- اگر F=1 و S=0 آن‌گاه در حالت پایانی هستیم و باید کاری انجام ندهیم (در اصل Hold انجام دهیم و مقادیر رجیسترها را تا اطلاع ثانوی، نگه‌داری کنیم)، پس طبق جدول (۱)، باید S0=S1=0 باشد.
- اگر F=0 و S=1 آن‌گاه در حالت محاسبه هستیم، به گونه‌ای که باید شیفت انجام دهیم، پس طبق جدول (۱)، باید برای رجیستر A (بالایی) که قصد داریم محتوای آن را به چپ شیفت دهیم، S0=1, S1=0 بوده و برای رجیستر B (پایینی) که قصد داریم محتوای آن را به راست شیفت دهیم، S0=0, S1=1 باشد.

بر اساس توضیحات فوق، در خصوص رجیستر A کافیست $S_0 = F'$ باشد، زیرا هرجا F صفر است (موارد اول و سوم)، S_0 یک است و هرجا F یک است (مورد دوم)، S_0 صفر است. هم‌چنین، در خصوص رجیستر A کافیست $S_1 = F\ XNOR\ S$ باشد، زیرا هرجا F و S ناهم‌علامت بوده و در نتیجه XNOR شان صفر باشد (موارد دوم و سوم)، S_1 صفر است و هرجا F و S هم‌علامت بوده و در نتیجه XNOR شان یک باشد (مورد اول)، S_1 یک است. در خصوص رجیستر B، ورودی‌های S_0 و S_1 آن مشابه رجیستر A است، با این تفاوت که جابه‌جا قرار گرفته است؛ یعنی در هر حالتی که S_0 و S_1 یکسان باشد، برای B هم همان مقادیر S_0 و S_1 وجود دارد و در هر حالتی که S_0 و S_1 متفاوت باشد، مقدار S_0 رجیستر A برابر مقدار S_1 رجیستر B و مقدار S_0 رجیستر B برابر مقدار S_1 رجیستر A است؛ در نتیجه، کافی است که دو مدار (NOT و XNOR) به دست آمده در فوق برای A، عینا برای B هم استفاده شود، با این تفاوت که به صورت جابه‌جا متصل شود؛ یعنی خروجی گیت XNOR به جای S_1 به S_0 در B و خروجی گیت NOT به جای S_0 به S_1 در B متصل شود.

در رجیستر نگه‌دارنده‌ی حاصل‌ضرب، ورودی‌های SR و SL به جایی متصل نیستند، زیرا قصد انجام شیفت در آن را نداشته و تنها نیاز است که بتوان در این رجیستر، Parallel Load و Hold را انجام داد. ورودی CLK این رجیستر به ایجادکننده‌ی پالس کلاک متصل است.

تنها زمانی نیاز است در رجیستر حاصل‌ضرب مقداری load شود (به جز هنگام reset) که مدار در حال انجام محاسبه (عملیات شیفت دادن) بوده و لازم باشد حاصل در رجیستر ذخیره شود؛ در حالت‌های اولیه و نهایی، نیازی به Parallel Load نیست و باید Hold انجام شود. در حالت‌هایی که مدار در حال محاسبه است، S (از خروجی‌های واحد کنترل) یک است، پس کافیست S به ورودی‌های S_0 و S_1 این رجیستر متصل شود. در این صورت، اگر مدار در حال شیفت و محاسبه باشد، S یک است و در نتیجه ورودی‌های S_0 و S_1 این رجیستر، یک بوده و طبق جدول (۱)، Parallel Load از خروجی جمع‌کننده، انجام می‌شود؛ در غیر این صورت، S صفر است و در نتیجه ورودی‌های S_0 و S_1 این رجیستر، صفر بوده و طبق جدول (۱)، Hold انجام می‌شود (و خروجی رجیستر تغییری نمی‌کند و ثابت می‌ماند).

هنگامی مقدار رجیستر حاصل‌ضرب لازم است ریست (صفر) شود که هم F و هم S (از خروجی‌های واحد کنترل) صفر باشند (و به عبارت دیگر، مدار در حالت اولیه باشد) و در نتیجه، نقیض F و نقیض S هر دو یک باشند، یعنی وقتی ریست لازم است که $F'S' = 1$. چون ورودی MR (Master Reset) تراشه ۷۴۱۹۸، Active Low است، باید نقیض $F'S'$ – که بر اساس قانون دمورگان، برابر $F + S$ است – توسط یک گیت OR تشکیل داده شده و به ورودی MR رجیستر حاصل‌ضرب، داده شود.

به واحد BITWISE_AND (AND بیتی)، خروجی‌های رجیستر مضروب (که ورودی‌های آن از بیت‌های ورودی مدار A تامین می‌شد) و نیز کم‌ارزش‌ترین بیت خروجی رجیستر مضروب‌فیه (که ورودی‌های آن از بیت‌های ورودی مدار B تامین می‌شد) متصل شده است. خروجی این واحد، توسط جمع‌کننده‌ی ۸ بیتی (که با نام ADDER2 در مدار اصلی قرار داده شده)، با مقدار کنونی رجیستر حاصل‌ضرب تا کنون جمع زده شده و به عنوان مقدار بعدی رجیستر حاصل‌ضرب در ورودی آن قرار می‌گیرد تا با لبه‌ی مثبت کلاک، در آن رجیستر (و خروجی‌اش) بنشیند.

یک واحد Clock Generator توسط DCLOCK از بخش Generators نیز قرار داده شده (و به پیشنهاد استاد محترم در جلسه‌ی رفع اشکال مورخ چهارشنبه ۲۰ مرداد ۱۴۰۰، فرکانس آن، ۱۰۰۰ هرتز تنظیم شده) و به ورودی کلاک تمامی ماژول‌های نیازمند به پالس ساعت، متصل شده است.
افزون بر موارد فوق که مربوط به مسیر داده هستند، واحد کنترلی نیز با ورودی‌های زیر قرار داده شده است:

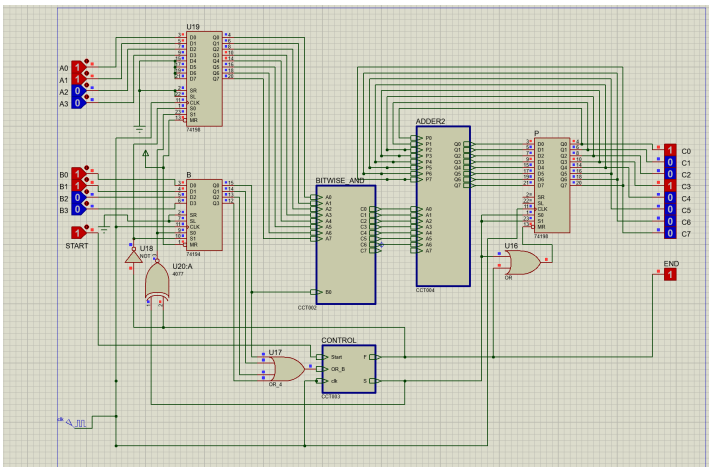
- ورودی Start که از ورودی Start مدار اصلی تامین می‌شود
- ورودی OR_B که از OR هر چهار بیت خروجی رجیستر مربوط به مضروب‌فیه (B) تامین می‌شود
- ورودی clk (پالس ساعت)

خروجی‌های واحد کنترلی، F و S هستند که پیش‌تر نیز توضیح داده شده‌اند. چون یک شدن خروجی F توسط واحد کنترل به معنای پایان محاسبات است، این خروجی به خروجی End مدار اصلی نیز متصل شده است.
نمایی از مدار اصلی به همراه توضیحات گفته شده، در شکل (۱۰) آمده است.

تست و بررسی عملکرد

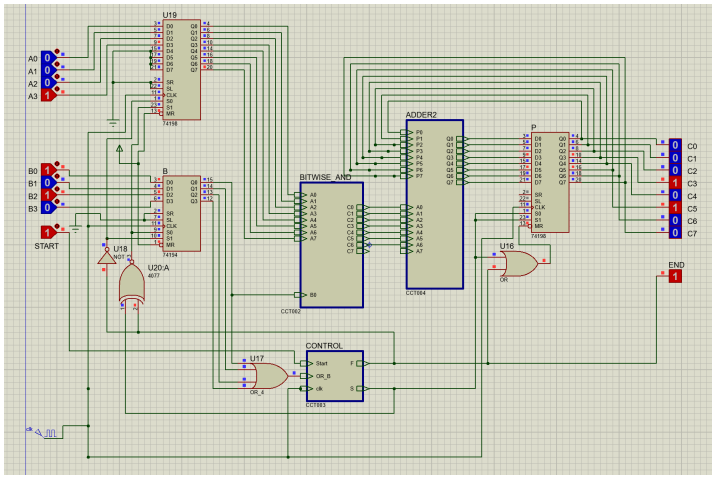
جهت بررسی عملکرد و صحت کارکرد مدار، به ازای چند ورودی نمونه، خروجی آن بررسی می‌شود. برای هر تست، ابتدا در حالتی که Start صفر است، مجموعه‌بیت‌های ورودی A و B تنظیم شده، سپس Start یک می‌شود و تا زمان یک شدن End (که چند کلاک به طول می‌انجامد)، Start را ۱ نگه می‌داریم و سپس خروجی را بررسی می‌کنیم؛ در نهایت، مجددا Start را صفر می‌کنیم تا برای تست بعدی آماده شویم. هرچه اندیس بزرگ‌تر باشد، یعنی آن بیت پرارزش‌تر است؛ مثلا C7 از C0 از لحاظ ارزش مکانی در عدد، پرارزش‌تر است. تست‌های انجام شده، به شرح زیر هستند:

- محاسبه‌ی $3 \times 3 = 9$ (یا همان $0011 \times 0011 = 00001001$ در مبنای ۲) که به درستی صورت گرفته و در شکل (۱۲) قابل رویت است.



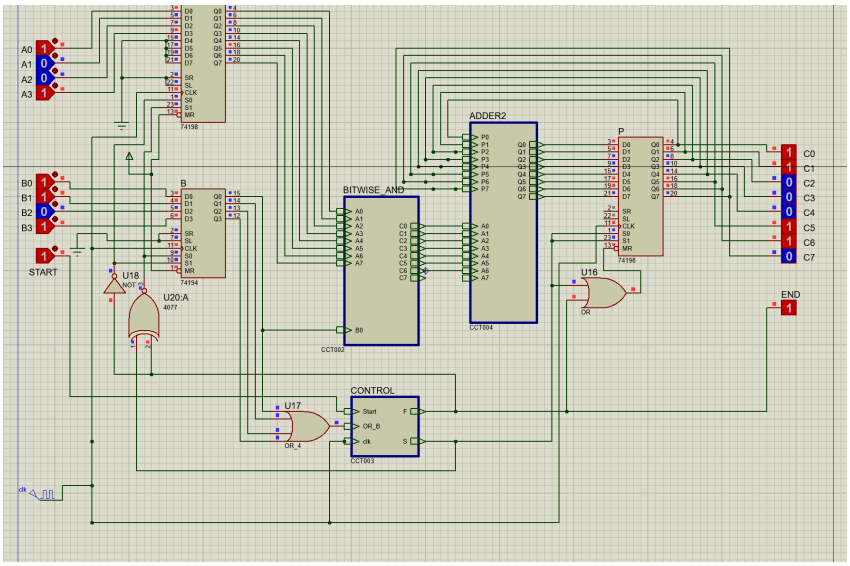
شکل ۱۲ – بررسی نمونه‌ی اول

- محاسبه‌ی $8 \times 5 = 40$ (یا همان $1000 \times 0101 = 00101000$ در مبنای ۲) که به درستی صورت گرفته و در شکل (۱۳) قابل رویت است.



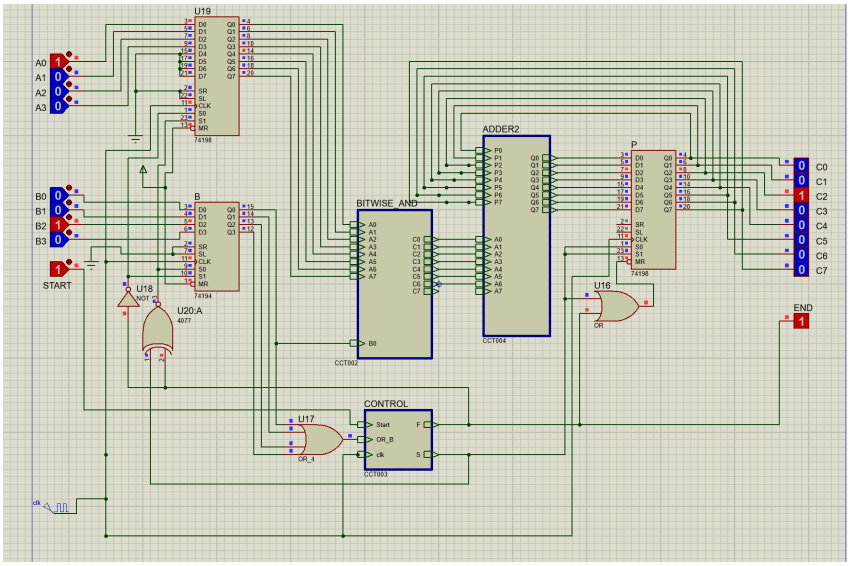
شکل ۱۳ – بررسی نمونه‌ی دوم

- محاسبه‌ی $9 \times 11 = 99$ (یا همان $1001 \times 1011 = 01100011$ در مبنای ۲) که به درستی صورت گرفته و در شکل (۱۴) قابل رویت است.



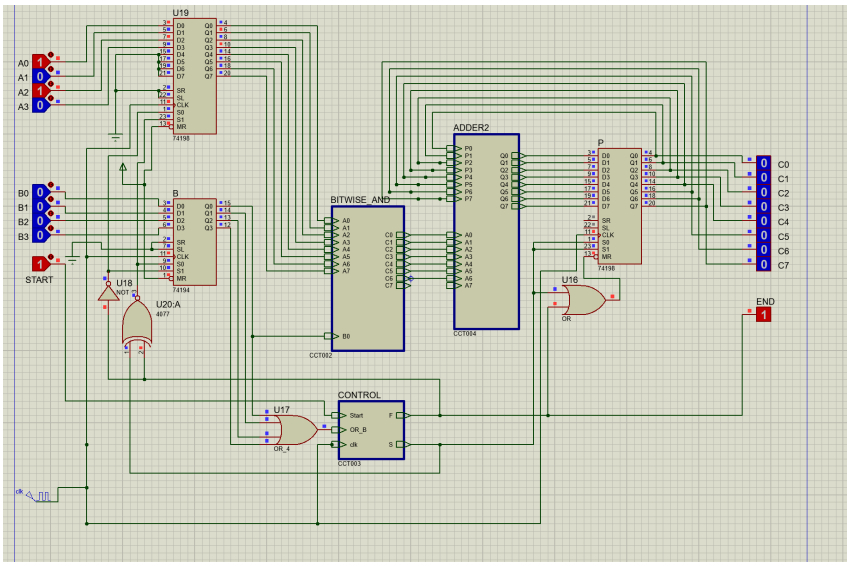
شکل ۱۴ – بررسی نمونه‌ی سوم

- محاسبه‌ی $1 \times 4 = 4$ (یا همان $0001 \times 0100 = 00000100$ در مبنای ۲) که یک حالت خاص بوده و به درستی صورت گرفته و در شکل (۱۵) قابل رویت است.



شکل ۱۵ – بررسی نمونه‌ی چهارم

- محاسبه‌ی $5 \times 0 = 0$ (یا همان $0000 \times 0101 = 00000000$ در مبنای ۲) که یک حالت خاص بوده و به درستی صورت گرفته و در شکل (۱۶) قابل رویت است.



شکل ۱۶ – بررسی نمونه‌ی پنجم

فایل‌های ضمیمه

فایل‌های پروتئوس در پوشه‌ی proteus-files ضمیمه شده‌اند.

مراجع

- مدارهای منطقی و سیستم‌های دیجیتال، دکتر اجلالی، انتشارات نصیر
- طراحی دیجیتال (مدارهای منطقی)، موریس مانو
- مستندات تراشه رجیستر ۷۴۱۹۸ در پروتئوس
- مستندات سایر تراشه‌ها – از جمله تراشه رجیستر ۷۴۱۹۴ – در پروتئوس