به نام خدا

آزمایش شماره ۲

آز معماری – دکتر سربازی آزاد

دانشکده مهندسی کامپیوتر

دانشگاه صنعتی شریف

نیم‌سال اول ۰۰-۰۱

گروه:

امیرحسین هادیان - ۹۷۱۰۲۶۰۹

محمدرضا مفیضی - ۹۸۱۰۶۰۵۹

علی حاتمی تاجیک - ۹۸۱۰۱۳۸۵

۱ هدف

در این آزمایش قصد طراحی و پیاده‌سازی یک ضرب‌کننده چهاربیتی ممیز ثابت را داریم که دو عدد چهاربیتی (دو بیت قبل از ممیز و دو بیت بعد از ممیز) و یک سیگنال شروع را در ورودی دارد و پس از اینکه سیگنال شروع ارسال شد ضرب کننده شروع به کار می‌کند و حداکثر پس از شش پالس ساعت جواب را در خروجی باز می‌گرداند.

۲ طراحی

می‌دانیم که قصد تولید یک مدار ضرب کننده با الگوریتم Shift and Add داریم. ابتدا نیازهای مدار را برسی می‌کنیم. دو عدد ورودی چهار بیتی خواهیم‌داشت و یک خروجی هشت بیتی(چهار بیت قبل ممیز و چهاربیت بعد از ممیز)، اما باید توجه داشته باشیم که برای عملیات شیفت به چپ که روی عامل اول ضرب انجام می‌شود نیاز به ۸ بیت فضا خواهیم داشت. همینطور رجیسترها نیاز به لود موازی دارند. این رجیسترها دارای قابلیت بارگذاری موازی و شیفت هستند (البته طراحی مدار بدون قابلیت شیفت هم ممکن بود. تنها کاری که نیاز بود انجام بشود این بود که سیگنال‌هایی خروجی ثبات را با یکی اختلاف، حالا کمتر یا بیشتر بستگی به نوع شیفت دارد، به ورودی لود رجیستر می‌دهیم و آن سیگنالی که شیفت را کنترل می‌کرد به جای اینکه به خود ثبات بدهیم آنرا به مالتیپلکسری وصل می‌کنیم که سیگنال‌های لود معمولی و لود شیفت را کنترل می‌کند. پس در پیاده سازی تاثیری آنچنانی نخواهد داشت اما برای جلوگیری از شلوغی شکل مدار سنتز شده نهایی از شیفت‌رجیسترهای دوطرفه استفاده شده است).

همینطور درباره ضرب اعداد با این الگوریتم می‌دانیم که روی اعداد علامت دار کاربردی ندارد پس اعداد ورودی ما هر دو بدون علامت هستند.

یک سیگنال ورودی برای اعلام شروع ضرب به مدار نیاز داریم. همینطور دو سری سیگنال ۴ بیتی نیز برای ورودی گرفتن اعداد نیاز دارد. یک سیگنال نیز برای اعلام اتمام عملیات نیاز است. فرض شده است که سیگنال شروع تنها یک لحظه یک می‌شود و پس از آن تا دریافت جواب نهایی صفر خواهد بود و تا رسیدن مدار به حالت اولیه سیگنال S برابر با ۱ باقی نخواهد ماند و نیازی به بلاک فاینال نخواهیم داشت (این فرض تاثیری چنان بر روندکار نخواهد داشت چون تنها یک بلاک اضافه می‌شود و یک سیگنال ریست که آوردن آنها در مدار نهایی کار ساده‌ای خواهد بود. البته این کار به هر نوع جواب نهایی را به ما خواهد داد چون بالاخره دست از روی پوش‌باتن برداشته خواهد شد و اعداد ورودی در این میان ثابت می‌مانند اما ممکن است در این حین چندین بار عملیات ضرب تا انتها انجام شود و دوباره از سر گرفته شود).

پس تا اینجا درباره مدار می‌دانیم:

- دو ورودی ممیزثابت چهاربیتی داریم (IN1, IN2)
- یک خروجی هشت بیتی ممیز ثابت داریم (C)
- یک سیگنال شروع برای عملیات داریم (S)
- یک سیگنال نمایش پایان عملیات داریم (F)

۱.۲ طراحی ASM

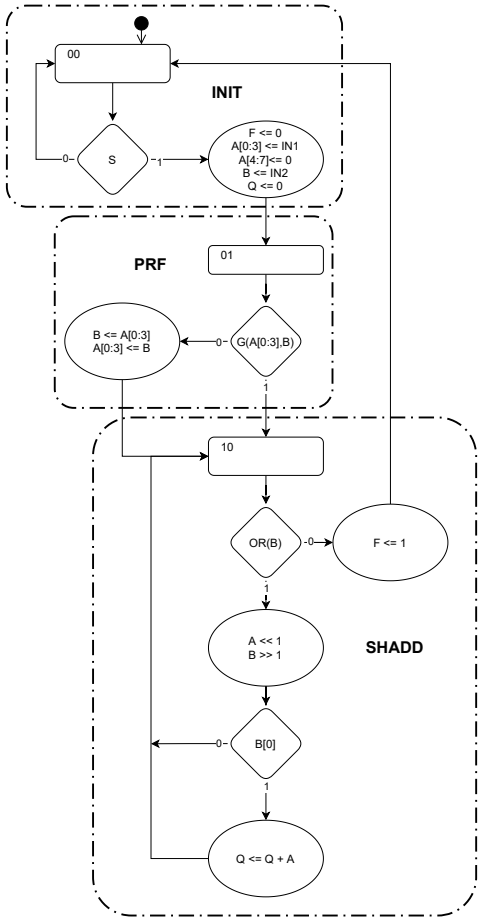
در ابتدای کار حالت‌های شکل ۱ در نظر گرفته شده بود. حالت‌های مختلف آن به شرح زیر است:

INIT زمانی که سیگنال شروع یک بشود، ابتدا در همان بلاک مقادیر اولیه ورودی‌ها در رجیسترها لود می‌شود. ۴ بیت مربوط به عامل دوم کامل به درون رجیستر B لود خواهد شد اما برای لود کردن A نیاز داریم که تعداد بیت آنرا اکستند کنیم چون مقدار این ثبات باید تا ۳۲بیت قابل شیفت باشد و اطلاعات آن از بین نرود بنابراین در هنگام لود کردن ورودی اول ۱۶بیت ورودی را در سمت کم‌ارزش بارگذاری کرده و ۱۶بیت پرارزش را صفر میگذاریم. همینطور سیگنال اتمام کار را نیز صفر می‌کنیم تا در پایان کار دوباره یک بشود. رجیستر C نیز باید مقدار صفر را بگیرد تا هنگام جمع کردن آن با عامل اول مقدار غیر منتظره ای درون آن نباشد چون از این لحظه به بعد چیزی که تنها چیزی که در این ثبات باگذاری خواهد شد مقدار این ثبات به علاوه مقدار عامل اول شیفت‌داده‌شده است.

PRF سپس که ورودی‌ها را دریافت کردیم، برای افزایش کارایی مدار و حداقل زمان لازم برای انجام عملیات، در یک کلاک چک می‌کنیم که حتما ورودی کوچکتر در عامل دوم (B) باشد.

SHADD این بلاک وظیفه اصلی عملیات ضرب را بر عهده دارد. در هر بلاک چند اتفاق به صورت همزمان می‌افتد:

- اگر عملیات به پایان رسیده باشد مقدار سیگنال پایان را یک کرده و به بلاک ابتدایی می‌رود.
- اگر عملیات به اتمام نرسیده باشد، عامل اول یکی به چپ و عامل دوم یکی به راست شیفت می‌خورد.



شکل ۱: نسخه ابتدایی طراحی شده

- اگر عملیات به اتمام نرسیده باشد و کم‌ارزشترین بیت عامل دوم نیز یک باشد عدد فعلی پاسخ با عدد فعلی عامل اول جمع می‌شود.

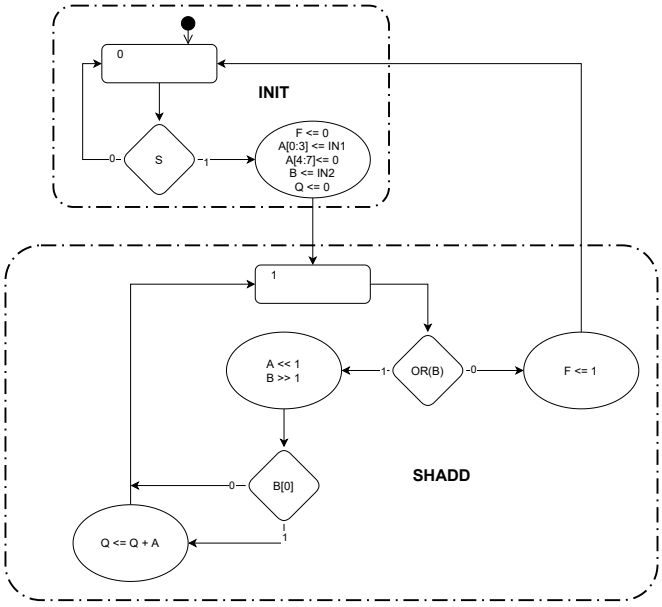
لازم به ذکر است که انجام عمل شیفت و جمع‌کردن در یک بلاک به صورت همزمان صورت می‌گیرد و این کار به خاطر تاخیر جزئی رجیسترها مشکلی برای ما بوجود نخواهد آورد. مثل این می‌ماند که اگر شرایط برقراربود ابتدا عمل جمع صورت بگیرد و سپس شیفت‌ها به میزان لازم انجام بشود.

پایان یافتن عملیات هنگامی رخ می‌دهد که تمام بیت‌های عامل دوم صرف شده باشند (صفر شدن بیت‌های عامل اول غیر ممکن است مگر اینکه هر دو آنها از ابتدا صفر بوده باشند که باز تغییری در شرط ایجاد نخواهد کرد). همینطور طبق الگوریتم تنها زمانی باید عمل جمع انجام بشود که کم‌ارزشترین بیت عامل دوم، یک باشد.

اما در زمان سنتز اولیه این چارت به مشکلی برخوردیم و آن نبود تراشه‌های مناسب مالتیپلکسر TTL بود که زمانی که چند مورد ورودی برای یک شیفت رجیستر داشتیم باید از آنها استفاده می‌کردیم. به همین دلیل بلوک پرفورمنس را از چارت خارج کردیم تا چارت نهایی به شکل ۲ دربیاید (توضیحات همان است فقط بلوک کارایی حذف شده است و استیت‌ها به دو کاهش یافته است).

۲.۲ سنتز و ساخت مدار

در این مرحله به سنتز و ساخت مدار می‌پردازیم. در ابتدای کار مدار اصلی را طبق چارت طراحی شده پیاده‌سازی می‌کنیم. مدار از دو بخش اصلی Data Path (DP) و Control Unit (CU) تشکیل شده است که خطوط اصلی مدار در قسمت DP تشکیل شده و اجزای نگه‌دارنده



شکل ۲: نسخه اصلاح شده

داده و انتقال دهنده آنها در این بخش وجود دارد. خطوط کنترل این خطوط و کنترل وضعیت مدار در بخش CU وجود دارد. سیگنال‌های کنترلی از بخش کنترل به بخش مسیرداده می‌روند.

در ادامه ابتدا تراشه‌هایی که مورد استفاده قرار گرفته است بررسی می‌شوند و سپس بخش‌های مختلف مدار بررسی می‌شوند.

۱.۲.۲ تراشه‌هایی که مورد استفاده قرار گرفته است

74283

74194

74198

7474

2732

۲.۲.۲ طراحی مسیرداده

در این بخش باید داده‌ها و روابط بین آنها را که در دسیژن‌باکس‌ها آمده است را بیاوریم. با توجه به شکل ۲ این بخش را سنتز می‌کنیم:

A یک شیفت رجیستر ۸ بیتی که چهاربیت داده ورودی آن به عدد اول و چهاربیت پرارزشتر آن به عدد ثابت صفر (زمین) متصل است.

B یک شیفت رجیستر ۴ بیتی که داده ورودی آن به ورودی دوم متصل است. از خروجی آن برای تولید دو سیگنال کنترلی استفاده می‌کنیم.

C یک شیفت رجیستر ۸ بیتی که جواب نهایی را در خود نگه‌میدارد و ورودی آن هم خروجی جمع کننده هشت بیتی است.

- یک جمع کننده که همواره مقدار درون شیفت رجیستر A و رجیستر C را در خود نگه میدارد.

F یک فلیپ‌فلاپ نوع دی که سیگنال اتمام را به خروجی میرساند و آنرا نگه میدارد.

همچنین به سیگنال‌های وضعیت زیر نیز نیاز خواهیم داشت:

B0 این سیگنال بیت اول شیفت رجیستر دوم است.

ORB این سیگنال «یا» منطقی چهار بیت شیفت رجیستر دوم است.

S این سیگنال شروع کار است.

۳.۲.۲ طراحی واحد کنترل

در این بخش باید سیگنال‌های کنترلی را برای مسیر داده تولید کنیم. همچنین از یک فلیپفلاپ نوع دی برای نگه‌داری وضعیت فعلی مدار استفاده می‌کنیم. هر شیفت رجیستر (تراشه‌های ۷۴۱۷۹۴/۷۴۱۹۸) نیاز به دو سیگنال انتخاب (تغییر بین کارایی‌ها طبق بخش ۱.۲.۲) و یک سیگنال ریست دارند. یک سیگنال برای ورودی و یک سیگنال برای ریست فلیپ‌فلاپ F و همین‌ها را برای فلیپ‌فلاپی که حالت را نگه می‌دارد نیز نیاز خواهیم داشت.

از آنجایی که شیفت و لود رجیسترهای A,B با یکدیگر انجام می‌شود از یک سیگنال تولید کرده و از آن برای هردو (با تغییرات جزئی در نحوه فرستادن آنها) استفاده می‌کنیم (Q نمایش دهنده خروجی فلیپ‌فلاپ استیت‌های ماست).

loadAB $= \overline{Q} \times S$
shiftAB $= Q \times OR(B)$
clearAB = 1 (Never will be cleared)
clearC $= \overline{Q} \times S =$ loadAB
loadC $= Q \times OR(B) \times B[0]$
shiftC $= 0$ (Never will be shifted)

با استفاده از این روابط سیگنال‌های مربوط به رجیسترها را به دست می‌آوریم:

$A_{S0} = loadAB + shiftAB, A_{S1} = loadAB$
$B_{S0} = loadAB = A_{S1}, B_{S1} = loadAB + shiftAB = A_{S0}$
$C_{S0} = loadC, C_{S1} = loadC$
$C_{clear} = \overline{clearC}$

با فلیپ‌فلاپ F به اینصورت رفتار می‌کنیم که ورودی اش را به خروجی اش می‌بریم تا همیشه حالت خود را حفظ کند تا اینکه سیگنال ست و ریست به آن برسد. سیگنال ست و ریست آن به صورت زیر خواهد بود:

$F_{reset} = \overline{\overline{loadAB}} = C_{clear}$
$F_{set} = \overline{Q \times \overline{OR(B)}} = \overline{Q} + OR(B)$

با یک نگاه به جدول حالت می‌توانیم بگوییم که $Q_D = (Q + S)(F_{set})$ (دقت کنید که سیگنال ست در زمانی که میخواهیم ست رخ بدهد صفر میشود نه یک).