

صلى الله عليه وسلم



دانشکده فنی و مهندسی
گروه برق - الکترونیک

پایان نامه جهت اخذ درجه کارشناسی ارشد رشته برق
گرایش الکترونیک

عنوان پایان نامه

طراحی و پیاده سازی یک ALU دیجیتال ۳۲ بیتی
با سرعت $1GHZ$ در پروسه $0.18\mu m CMOS$

استاد راهنما :

دکتر محمد مهدی کارخانه چی

نگارش :

شهاب تیموری

مهر ۱۳۹۲



دانشگاه رازی

دانشکده فنی مهندسی

گروه الکترونیک

پایان نامه جهت اخذ درجه کارشناسی ارشد رشته برق گرایش الکترونیک

نام دانشجو:

شهاب تیموری

تحت عنوان :

طراحی و پیاده سازی یک ALU دیجیتال ۳۲ بیتی

با سرعت 1GHZ در پروسه $0.18\mu m$ CMOS

در تاریخ ۲۱ مهر ۱۳۹۲ توسط هیأت داوران زیر بررسی و با درجه به تصویب نهایی رسید.

۱- استاد راهنما آقای دکتر محمد مهدی کارخانه چی با مرتبه ی علمی استادیار امضاء

۱- استاد داور داخل گروه آقای دکتر محسن حیاتی با مرتبه ی علمی دانشیار امضاء

۲- استاد داور داخل گروه آقای دکتر آرش احمدی با مرتبه ی علمی استادیار امضاء

کلیه حقوق مادی مترتب بر نتایج مطالعات ، ابتکارات و
نوآوری های ناشی از تحقیق موضوع این پایان نامه
متعلق به دانشگاه رازی است.

تقدیم به:

روح پاک خواهرم

که عالمانه به من آموخت تا چگونه در عرصه زندگی، ایستادگی را تجربه نمایم.

چکیده

با پیشرفت سریع تکنولوژی و تمرکز بر کارهای پیچیده علمی، که نیاز به شبیه سازیهای سنگین و زمان گیر دارد اهمیت پردازنده های سریع در سیستم های کامپیوتری بیش از پیش نمایان می شود. در این میان ضرب کننده های دیجیتال و واحد محاسبات منطقی که به عنوان بخشی از میکروپروسورها و DSP ها که دارای بحرانی ترین تاخیر است اهمیت ویژه ای دارد چون کارایی کل چیپ مستقیماً با کارایی ضرب کننده داخلی در ارتباط است. یکی از روشهای افزایش سرعت و کارایی در پردازنده ها استفاده از تکنیک خط لوله^۲ است که در سالهای اخیر در بسیاری از کاربردها اعمال می شود. عناصر خط لوله اغلب به صورت موازی و یا در مد زمان قطعه قطعه اجرا می شوند که در آن صورت برای ذخیره سازی بین عناصر به المان های حافظه دار نیاز می باشد. هدف ما طراحی و پیاده سازی یک ضرب کننده ۳۲ بیتی با سرعت ۱ گیگا هرتز در پروسه $0.18\mu m$ CMOS با حداقل تعداد سیکل کلاک ممکن بود، تا علاوه بر تاخیر کلی کم، سخت افزار و توان مصرفی کمی داشته باشد.

برای این منظور سعی شده با طراحی مدارات جدید در بعضی از قسمت ها، توازن در تاخیر طبقات برقرار شود تا ماکزیمم بهره برداری از تکنیک Pipeline صورت گیرد. بطور مثال مدار Booth Encoder/Decoder جدید طوری طراحی شده که حاصلضرب های جزئی^۳ در یک سیکل و با تاخیر کوچک ($380 ps$) تولید شوند. از طرفی با استفاده از یک تکنیک بهینه یافته ی ریاضی و مداری، ردیف اضافی ساختار Modified Booth بدون استفاده از یک سیکل اضافی و همزمان با تولید Partial Product ها در سیکل اول، حذف می شود حذف این ردیف علاوه بر کاهش توان مصرفی، سخت افزار و تاخیر کلی، به منظم تر شدن کل ضرب کننده می انجامد. از آنجاییکه روش های قبلی برای حذف این بیت، تاخیر بسیار زیادتری داشتند عملاً حذف بیت اضافی امکان پذیر نبود. ولی در این ضرب کننده مدارات سیکل اول طوری بهینه شده اند که ۱۶ ردیف پس از تاخیری حدود $380 ps$ تولید می شوند و به دلیل منظم بودن ساختار ایجاد شده برای جمع عمودی حاصلضرب های جزئی فقط کمپرسور ۴ به ۲ کفایت تا پس از ۴ سیکل ورودی های جمع کننده ی نهایی تولید شوند. همچنین جمع کننده ی نهایی جدید طوری طراحی شده که برای سرعت های مختلف در ساختارهای Pipeline شده قابل شکسته شدن بوده و علاوه بر سخت افزار کوچک، تاخیر کمی دارد. بطوریکه در ۳ سیکل جمع ۵۱ بیت در این ضرب کننده تحقق می یابد. مدار CLA سریع و همچنین ارائه ی یک مدار تعیین کننده ی رقم نقلی هر بلوک (BCG) از اصلی ترین عللی هستند که سرعت این جمع کننده ی کوچک را قابل مقایسه با جمع کننده های پیچیده و حجیم کرده است در نهایت این ضرب کننده می تواند ۱ میلیارد نمونه ی ۳۲ بیتی را در هر ثانیه از ورودی ها گرفته و پس از ۸ سیکل نتیجه ی ضرب علامت دارشان را تولید کند. نتایج شبیه سازی نشان می دهد باروش Pipeline سرعت ضرب کننده به بیش از ۶ برابر حالت عادی میرسد، درحالیکه مساحت سخت افزار ($1.5mm \times 1.87mm$) تقریباً ۲/۲ برابر و توان مصرفی آن ($543 mw$) می شود.

1 - Critical path delay

2 - Pipeline

3 - Partial Product

فهرست مطالب

عنوان	صفحه
فصل اول: اهداف و الگوریتم های مختلف ضرب کننده	
۱-۱- مقدمه.....	۲
۲-۱- اهداف.....	۴
۳-۱- سازماندهی.....	۵
۴-۱- ساختار ضرب کننده ی دیجیتال.....	۶
۵-۱- تولید حاصلضرب های جزئی.....	۷
۶-۱- کاهش حاصلضرب های جزئی.....	۸
۱-۶-۱- کاهش به روش آرایه ای.....	۹
۱-۶-۱- کاهش به روش درخت والاس.....	۱۰
۲-۶-۱- کاهش به روش <i>Dadda</i>	۱۱
۳-۶-۱- کاهش حاصلضرب های جزئی به روش <i>Booth Recoding</i>	۱۲
۱-۳-۶-۱- ضرب در میناهای مختلف.....	۱۳
۱-۳-۶-۱- تکنیک <i>Booth Recoding</i>	۱۳
۷-۱- ضرب کننده ی <i>Baugh-Wooley</i>	۱۶
۸-۱- کمپرسور.....	۱۹
۱-۸-۱- کمپرسور ۴ به ۲.....	۱۹
۲-۸-۱- کمپرسور ۵ به ۲.....	۲۱
۹-۱- ضرب کننده هیتاچی.....	۲۲
۱۰-۱- ضرب کننده ی <i>Inoue</i>	۲۳
۱۱-۱- کاهش توان مصرفی ضرب کننده.....	۲۳
۱۲-۱- بهینه سازی ضرب کننده در سطح گیت های منطقی.....	۲۵
۱۳-۱- مقایسه ی ضرب کننده ها.....	۲۵
فصل دوم: انواع جمع کننده	
۱-۲- تمام جمع کننده.....	۲۸
۱-۲- جمع کننده ی نقلی موج دار.....	۳۱
۲-۲- جمع کننده با بیت نقلی جهشی.....	۳۲
۳-۲- جمع کننده با بلوک های متغیر.....	۳۳
۴-۲- جمع کننده ی پیش بینی کننده نقلی.....	۳۵
۶-۲- جمع کننده ی انتخابگر نقلی.....	۴۱
فصل سوم: ساختار <i>ALU</i> طراحی شده	
۱-۳- ساختار کلی.....	۴۳
۱-۳- الگوریتم <i>Booth</i> اصلاح شده.....	۴۶
۲-۳- مدار ارائه شده برای الگوریتم <i>Booth</i>	۵۱

۵۸	۳-۳- روش های ارائه شده برای حذف بیت اضافی
۶۴	۳-۴- کاهش حاصلضرب های جزئی با کمپرسور ۴ به ۲
۶۷	۳-۶- جمع کننده نهایی
۶۹	۳-۶-۱- جمع کننده ی پیشبینی کننده ی نقلی ۸-بیتی
۷۱	۳-۶-۱- مدار <i>First Zero Finder (FZF)</i> یا جمع کننده با یک
۷۴	۳-۶-۲- مدار تولیدکننده ی بیت نقلی بلوک ها
۷۴	۳-۶-۳- مدار انتخاب کننده ی نهایی
۷۷	۳-۷- المان حافظه

فصل چهارم: واحد محاسبات *ALU*

۷۹	۴-۱- عملکرد <i>ALU</i>
۸۲	۴-۱- رمزگشا <i>Decoder</i>
۸۲	۴-۲- واحد منطقی <i>Logical</i>
۸۴	۴-۳- واحد عملیات <i>Bit-shifting</i>
۸۴	۴-۴- طراحی بلوک <i>Adder</i>
۸۴	۴-۶- طراحی بلوک <i>Subtractor</i>

فصل پنجم: نتیجه گیری و نتایج شبیه سازی شده

۸۷	۶-۱ نتایج شبیه سازی شده
۹۳	منابع

فهرست اشکال

صفحه	عنوان
۶	شکل ۱-۱- ساختار کلی ضرب کننده ی دیجیتال (۶- بیتی).....
۷	شکل ۱-۱- تولید حاصلضرب های جزئی در یک ضرب کننده ی ۱۶بیتی.....
۸	شکل ۲-۱- کاهش حاصلضرب های جزئی با روش <i>Carry Save Addition</i>
۹	شکل ۳-۱- کاهش حاصلضرب های جزئی به روش <i>Array style</i>
۱۰	شکل ۴-۱- کاهش حاصلضرب های جزئی یک ضرب کننده ی ۸×۸ بیتی به روش <i>Wallace Tree</i>
۱۱	شکل ۶-۱- کاهش حاصلضرب های جزئی یک ضرب کننده ی ۸×۸ بیتی به روش <i>Dadda</i>
۱۶	شکل ۷-۱- ضرب بدون علامت (۵×۵ بیتی).....
۱۶	شکل ۸-۱- ضرب مکمل-۲ (۵×۵ بیتی).....
۱۷	شکل ۹-۱- ضرب مکمل-۲ به روش <i>Baugh-wooley</i> (۵×۵ بیتی).....
۱۸	شکل ۱۰-۱- ضرب مکمل-۲ به روش <i>Baugh-wooley</i> اصلاح شده (۵×۵ بیتی).....
۱۹	شکل ۱۱-۱- کمپرسور ۴ به ۲ (بلوکی).....
۲۰	شکل ۱۱-۱- کمپرسور ۴ به ۲ (با گیت های منطقی).....
۲۱	شکل ۱۲-۱- کمپرسور ۵ به ۲ (بلوکی).....
۲۱	شکل ۱۳-۱- نحوی اتصال دو کمپرسور ۵ به ۲ مجاور.....
۲۲	شکل ۱۴-۱- ساختار کلی ضرب کننده ی هیتاچی.....
۲۹	شکل ۱-۲- مدار تمام جمع کننده ی <i>Static CMOS</i>
۲۹	شکل ۱-۲- پیاده سازی منطقی تمام جمع کننده.....
۳۰	شکل ۲-۲- : پیاده سازی تمام جمع کننده در منطق سریع <i>DPL</i>
۳۱	شکل ۳-۲- جمع کننده ی نقلی موج دار با استفاده از مالتی پلکسر.....
۳۲	شکل ۴-۲- جمع کننده ی نقلی جهشی.....
۳۳	شکل ۶-۲- جمع کننده ی ۳۱-بیتی با بلوک های متغییر.....
۳۵	شکل ۷-۲- جمع کننده ی ۱۶-بیتی <i>CLA</i> با بلوک های ۳-بیتی.....
۳۷	شکل ۸-۲- جمع کننده ی ۳۲-بیتی <i>Super-Group CLA</i>
۳۹	شکل ۹-۲- جمع کننده ی <i>MCC</i>
۴۰	شکل ۱۰-۲- جمع کننده ی ۶۳-بیتی <i>CLA</i> با استفاده از منطق دومینو <i>CMOS</i>
۴۰	شکل ۱۱-۲- مسیر بحرانی در جمع کننده ی ۶۳-بیتی <i>CLA</i> موتورلا.....
۴۱	شکل ۱۲-۲- یک <i>CSA</i> ۱۶-بیتی با ۴ بلوک ۳-بیتی.....
۴۵	شکل ۱-۳- ساختار کلی ضرب کننده ی ۳۲ بیتی طراحی شده.....
۴۶	شکل ۲-۳- تبدیل موازی مبنای ۲.....
۴۷	شکل ۳-۳- مدار <i>Booth Encoder</i> و <i>Booth Decoder</i>
۴۸	شکل ۴-۳- مدار <i>Booth Encoder</i> و <i>Booth Decoder</i>
۴۸	شکل ۵-۳- مدار <i>Booth Encoder</i> و <i>Booth Decoder</i>
۵۰	شکل ۶-۳- مدار <i>Booth Encoder</i> و <i>Booth Decoder</i> چهار سیگناله.....

- شکل ۳-۷- مدار *Booth Encoder* و *Booth Decoder* چهار سیگنال با تاخیر سه گیت ۵۱
- شکل ۳-۸- مدار *Booth Encoder* و *Booth Decoder* ۵۲
- شکل ۳-۹- نمودار نقطه ای حاصل ضرب های جزئی ضرب کننده *Booth* ۱۶ بیتی ۵۴
- شکل ۳-۱۱- نمودار نقطه ای حاصل ضرب های جزئی ضرب کننده *Booth* ۸ بیتی مکمل ۲ ۵۵
- شکل ۳-۱۲- نمای کلی حذف ردیف اضافه ۵۷
- شکل ۳-۱۳- ساختار ارائه شده برای گرفتن مکمل ۲ ۵۷
- شکل ۳-۱۴- ساختار ارائه شده اول برای حذف ردیف اضافی الگوریتم *Booth* ۵۹
- شکل ۳-۱۵- تحقق عمل مکمل ۲ برای پنج بیت با استفاده از مدار *Add-one* و مولتی پلکسر ۵۹
- شکل ۳-۱۶- ساختار ارائه شده دوم برای حذف ردیف اضافی الگوریتم *Booth* ۶۱
- شکل ۳-۱۷- تبدیل سیگنال *E* به سیگنال *S* ۶۱
- شکل ۳-۱۸- تولید سیگنال های *S* با تاخیر معادل تاخیر سه گیت ۶۲
- شکل ۳-۱۹- مدار *Booth Encoder* و *Booth Decoder* ارائه شده دوم ۶۳
- شکل ۳-۲۰- کمپرسور و شمارنده ۶۵
- شکل ۳-۲۱- نمای کلی کاهش حاصل ضرب های جزئی با استفاده از کمپرسور ۴ به ۲ ۶۶
- شکل ۳-۲۲- نمای کلی کمپرسور ۴ به بهبود داده شده ۲ با تاخیر سه تا گیت ۶۶
- شکل ۳-۲۳- نمای کلی کمپرسور ۴ در سطح ترانزیستور ۶۷
- شکل ۳-۲۴- نمای کلی جمع کننده ی نهایی که ۵۱ بیت را در سه سیکل جمع می کند ۶۸
- شکل ۳-۲۵- مدار جمع کننده *CLA* ارائه شده ۷۰
- شکل ۳-۲۶- مدار جمع کننده *CSA* متعارف ۷۲
- شکل ۳-۲۷- مثالی برای منطق "*First Zero One*" ۷۲
- شکل ۳-۲۸- مدار *FZF* ۸-بیتی استفاده شده ۷۳
- شکل ۳-۲۹- مدار تولید کننده بیت نقلی بلوک ها در سطح گیت منطقی ۷۴
- شکل ۳-۳۰- مدار انتخاب کننده ی نهایی ۷۵
- شکل ۳-۳۱- ساختار کامل *Adder* ۸-بیتی ۷۶
- شکل ۳-۳۲- مدار فلیپ فلاپ استفاده شده ۷۷
- شکل ۴-۱- ساختار کلی یک *ALU* ۷۹
- شکل ۴-۲- یک نمونه *ALU* طراحی شده با مالتی پلکسر کنترل شده توسط سیگنال کنترلی در خروجی ۸۰
- شکل ۴-۳- اعمال کردن خروجی های دیکدر به بلوک *ALU* ۸۱
- شکل ۴-۴- دیکدر ۳ به ۸ ۸۲
- شکل ۴-۵- ساختار بلوک ها در سطح گیت های منطقی ۸۳
- شکل ۴-۶- ساختار بلوک ها در سطح ترانزیستوری ۸۳

- شکل ۴-۷- شیفیت چرخشی به چپ بر روی A و شیفیت چرخشی به راست بر روی B ۸۴
- شکل ۴-۸- ساختار بلوک تفریق کننده ۸۵
- شکل ۵-۱- خروجی دیکدرها ۸۹
- شکل ۵-۲- نتایج خروجی بلوک جمع کننده و تفریق کننده ۳۲ - بیتی ۸۹
- شکل ۵-۳- نتایج شبیه سازی شده بلوک مربوط به الگوریتم Booth ۹۰
- شکل ۵-۴- شبیه سازی کمپرسور ۴ به ۲ ۹۰
- شکل ۵-۵- خروجی هر یک از طبقات ضرب کننده ۹۱
- شکل ۵-۶- خروجی طبقه والاس تری ۹۱

فهرست جداول

عنوان	صفحه
جدول ۱-۱- جدول الگوریتم <i>Booth Recording</i> در مبنای ۴.....	۱۵
جدول ۲-۱- توضیح جدول الگوریتم <i>Booth Recording</i> در مبنای ۴.....	۱۵
جدول ۳-۱- مقایسه ی کیفی ساختارهای مختلف ضرب کننده.....	۲۶
جدول ۱-۲- جدول درستی یک <i>Full Adder</i>	۲۸
جدول ۱-۳- جدول درستی الگوریتم <i>Booth</i> با اینکدر سه حالت.....	۴۷
جدول ۱-۳- جدول درستی الگوریتم <i>Booth</i> با اینکدر چهار سیگناله.....	۴۹
جدول ۲-۳- جدول درستی الگوریتم <i>Booth</i> با اینکدر چهار سیگناله.....	۵۰
جدول ۳-۳- جدول درستی الگوریتم <i>Booth</i> با اینکدر سه حالت.....	۵۳
جدول ۱-۴- اعمال منطقی و حسابی انجام شده توسط <i>ALU</i> به همراه کدهای کنترلی.....	۸۰
جدول ۱-۵- مقایسه تاخیر روش های حذف بیت اضافی در الگوریتم <i>Booth</i> برای ضرب کننده های مختلف.....	۸۷
جدول ۲-۵- مقایسه کمپرسور ۴ به ۲ با <i>counter</i> های دیگر.....	۸۷
جدول ۳-۵- مقایسه تاخیر هر یک از بلوک های جدول ۲-۵ در کاهش ۱۶ ردیف به ۲ ردیف.....	۸۸
جدول ۴-۵- مقایسه جمع کننده طراحی شده با جمع کننده های سریع قبلی.....	۸۸
جدول ۵-۵- مقایسه ضرب کننده های سریع با ضرب کننده ارائه شده.....	۹۲

فصل اول

اهداف و الگوریتم های مختلف ضرب کننده ها

۱-۱- مقدمه

ریزپردازنده^۱، واحد پردازش مرکزی در هر کامپیوتر است، که پردازش های مختلف در آن انجام می گیرد. با استفاده از تکنولوژی ساخت قطعات نیمه هادی می توان این واحد پردازنده (CPU) را کوچک نموده و در یک بسته بندی یا تراشه^۲ قرار داد، که به آن ریزپردازنده یا میکروپروسور گویند. پس در حالت کلی کلمه CPU یک مفهوم کلی تری از کلمه ریزپردازنده است. با پیشرفت تکنولوژی، هم اکنون تمام CPUها بصورت ریزپردازنده ساخته می شود، در صورتی که در کامپیوترهای نسل های پایین تر به دلیل پایین بودن تکنولوژی، امکان ساخت CPU بصورت متمرکز و در داخل یک بسته بندی (تراشه) وجود نداشت. یک ریزپردازنده مجموعه هایی از دستورات به زبان ماشین را اجرا می کند. براساس این دستورات، ریزپردازنده سه نوع عملیات اصلی را انجام میدهد:

ریزپردازنده با استفاده از واحد (Arithmetic Logic Unit) ALU قادر به انجام عملیات محاسباتی نظیر جمع، تفریق، ضرب و ... ، عملیات منطقی نظیر AND، NAND و... می تواند داده ها را از یک حافظه به حافظه دیگر منتقل کند. قادر به تصمیم گیری و جهش به مجموعه جدیدی از دستورات بر اساس این تصمیم گیری است. از معیارهای طبقه بندی میکروپروسور، سرعت پردازش اطلاعات در آنهاست، که با واحد مگا هرتز بیان می شود. هر مگا هرتز، معادل انجام یک میلیون دستور در ثانیه می باشد. هرچقدر سرعت ساعت یک ریزپردازنده بیشتر باشد، آن ریزپردازنده سریع تر عمل می کند. CPU به طور کلی از چند قسمت تشکیل شده است. مانند واحد محاسبه و منطق، واحد کنترل، واحد حافظه ثبات و ...

ثباتها، حافظه های ناپایدار برای ذخیره سریع و موقتی داده هایی هستند که باید پردازش شوند و در CPU قرار گرفته اند. این نوع حافظه ها می توانند داده ها و دستورات عملی های در حال پردازش را به سرعت دریافت، ذخیره و منتقل کنند. انباره^۳، ثباتی است که نتایج عملیات ریاضی و منطقی انجام شده به وسیله واحد ALU را در خود نگه می دارد. وظیفه ی واحد کنترل^۴، کنترل دریافت داده ها از واحد ورودی، کنترل عملیات داخلی CPU و کنترل ارسال اطلاعات به واحد خروجی می باشد. این واحد مشابه یک سیستم عصبی برای کنترل سایر بخشهای کامپیوتر عمل می کند.

1 - CPU

2 - IC

3 - Accumulator

4 - CU- Control Unit

در علم کامپیوتر، واحد حساب و منطق (به اختصار *ALU*)، یک مدار دیجیتالی است که عملیات محاسباتی و منطقی را انجام می دهد. *ALU* بخش بنیادی واحد پردازش مرکزی (*CPU*) یا هر ریز پردازنده ی دیگری است. پردازنده های مرکزی یا پردازنده های گرافیکی مدرن، شامل *ALU* های بسیار قدرتمند و پیچیده هستند و ممکن است بیش از یک *ALU* داشته باشند. اکثر عملیات پردازنده به وسیله یک یا چند *ALU* انجام می شود. *ALU* دیتا را از رجیسترهای ورودی بارگذاری می کند، سپس یک واحد کنترل (*Control unit*) به *ALU* می گوید که چه عملیاتی را بر آن اطلاعات انجام دهد. و در آخر *ALU* نتایج را بر روی یک رجیستر خروجی ذخیره می کند.

اکثر *ALU* ها می توانند این عملیات را انجام دهند:

- عملیات محاسباتی صحیح (جمع، تفریق و بعضی ضرب و ...)
- عملیات بیتی منطقی (*NAND*، *NOR*، *XOR*، *XNOR* و ...)
- عملیات انتقال بیتی^۵ انتقال یا چرخش یک کلمه با تعداد مشخصی بیت به راست یا چپ/بدون گسترش علامت.

۱-۲- اهداف :

با پیشرفت سریع تکنولوژی و تمرکز بر کارهای پیچیده علمی، که نیاز به شبیه سازی های سنگین و زمان گیر دارد اهمیت پردازنده های سریع در سیستم های کامپیوتری بیش از پیش نمایان می شود. در این میان ضرب کننده های دیجیتال و واحد محاسبات منطقی که به عنوان بخشی از میکروپروسور ها و *DSP* ها که دارای بحرانی ترین تاخیر (*Critical path delay*) است اهمیت ویژه ای دارد چون کارایی کل چیپ مستقیماً با کارایی ضرب کننده داخلی در ارتباط است. یکی از روش های افزایش سرعت و کارایی در پردازنده ها استفاده از تکنیک *Pipeline* (خط لوله) است که در سالهای اخیر در بسیاری از کاربردها اعمال می شود. عناصر خط لوله اغلب به صورت موازی و یا درمد زمان قطعه قطعه اجرا می شوند که در آن صورت برای ذخیره سازی بین عناصر به المان های حافظه دار نیاز می باشد. هدف ما طراحی و پیاده سازی یک ضرب کننده ی ۳۲ بیتی با سرعت ۱ گیگا هرتز در پروسه ی *0.18μm CMOS* با حداقل تعداد سیکل کلاک ممکن بود، تا علاوه بر تاخیر کلی کم، سخت افزار و توان مصرفی کمی داشته باشد.

همچنین جمع کننده ی نهایی جدید طوری طراحی شده که برای سرعت های مختلف در ساختارهای *Pipeline* قابل شکسته شدن بوده و علاوه بر سخت افزار کوچک، تاخیر کمی دارد. بطوریکه در ۴ سیکل جمع ۶۴ بیت در این ضرب کننده تحقق می یابد. مدار *CLA* سریع و همچنین ارائه ی یک مدار تعیین کننده ی رقم نقلی هر بلوک (*BCG*) از اصلی ترین عللی هستند که سرعت این جمع کننده ی کوچک را قابل

مقایسه با جمع کننده های پیچیده و حجیم کرده است در نهایت این ضرب کننده می تواند ۱ میلیارد نمونه ی ۳۲ بیتی رادره ثانیه از ورودی ها گرفته و پس از ۸ سیکل نتیجه ی ضرب علامت دارشان را تولید کند. نتایج شبیه سازی نشان می دهد با روش Pipeline سرعت ضرب کننده به بیش از ۶ برابر حالت عادی می رسد، در حالیکه مساحت سخت افزار ($1.5mm \times 1.87mm$) تقریباً ۲/۲ برابر و توان مصرفی آن ($543 m$) می شود.

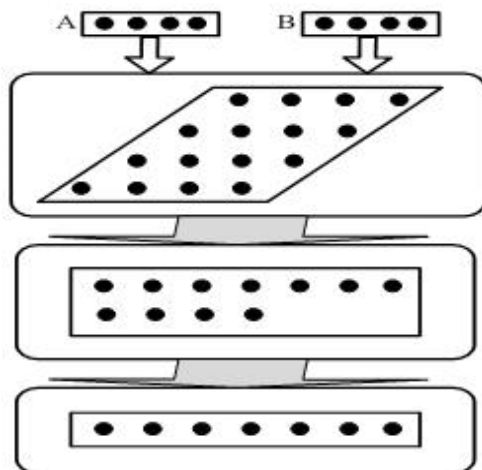
۱-۳- سازماندهی :

در بخش های این پایان نامه که شامل شش فصل می باشد قسمت های مختلف پایان نامه مورد بررسی و تجزیه و تحلیل قرار می گیرد. فصل ۱ مروری بر مفاهیم و اجزاء یک ضرب کننده ی دیجیتال و ساختارهای متداول دارد. فصل ۲ شامل بررسی ساختارهای جمع کننده های دیجیتال و ویژگیهای هر یک است. فصل ۳ به توضیح ساختار ضرب کننده ی ارائه شده و طراحی مدارات قسمت های مختلف اختصاص یافته است. در فصل ۴ ساختار ALU مورد بررسی قرار می گیرد. و در فصل ۵ نتایج شبیه سازی آورده شده به همراه جمع بندی نتایج کلی پایان نامه و مقایسه با کار های انجام شده در زمینه مشابه، نیز در پایان فصل گنجانده شده است.

در این بخش توصیف مختصری از ضرب کننده های دیجیتالی شامل ساختار و اجزای وابسته آنها نمایش خواهیم داد. همچنین بعضی تکنیک ها که برای افزایش سرعت ضرب کننده ها ارائه شده اند، مورد بحث قرار خواهد گرفت. سپس مروری بر توان مصرفی در مدار های CMOS به همراه تکنیک های اولیه برای کاهش توان مصرفی خواهیم داشت.

۱-۴- ساختار ضرب کننده ی دیجیتال:

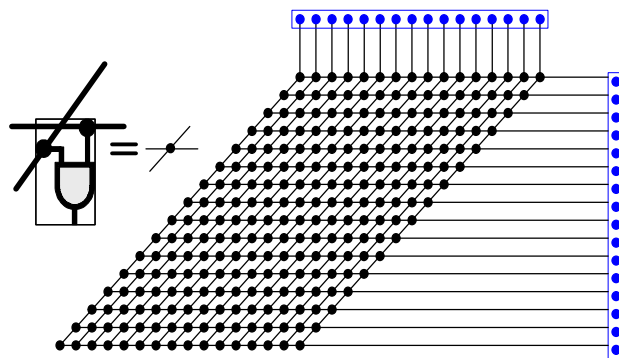
ضرب دیجیتالی شیفت و جمع یافتن یک سری از بیت ها است که در آن دو عدد، ضرب کننده و ضرب شونده، برای تولید خروجی ترکیب می شوند، با فرض اینکه نمایش بیتی ضرب شونده بصورت: $X = X_{n-1} \dots X_1 X_0$ و ضرب کننده بصورت $Y = Y_{n-1} \dots Y_1 Y_0$ باشد. برای شکل گیری حاصلضرب، حداکثر N ردیف از ضرب شونده بصورت شیفت یافته (برای ضرب بدون علامت) جمع می شوند. کل فرآیند شامل ۳ مرحله می باشد، تولید حاصلضرب جزئی (Partial Product)، کاهش حاصلضرب جزئی و جمع نهایی. ساختار کلی ضرب دیجیتال در شکل (۱-۱) نشان داده شده است.



شکل ۱-۱: ساختار کلی ضرب کننده ی دیجیتال (۳-بیتی)

۱-۵- تولید حاصلضرب های جزئی (*Partial Product*):

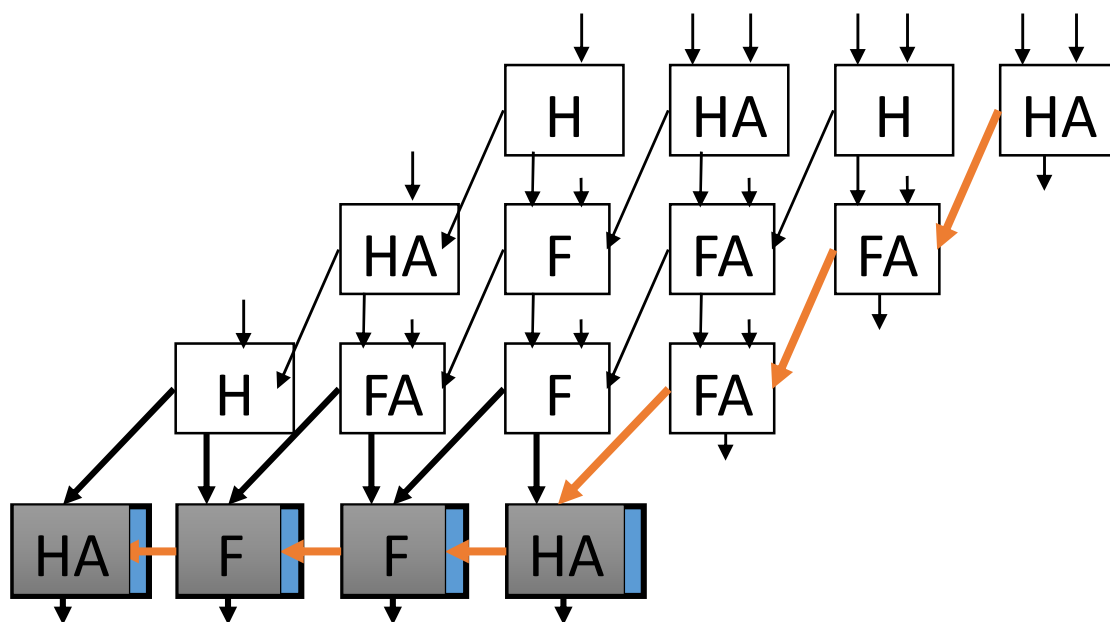
به عنوان مرحله اولیه در ضرب دیجیتال، نیاز به تولید N کپی شیفت یافته از ضرب شونده می باشد، که باید در مراحل بعدی جمع شوند. ارزش بیت ضرب کننده مشخص می کند که کپی شیفت یافته از ضرب شونده باید جمع شود یا نه. به این صورت که اگر بیت i -ام $(0 \leq i \ll n - 1)$ از ضرب کننده "1" باشد، کپی شیفت یافته از ضرب شونده جمع می شود و اگر "0" باشد کپی شیفت یافته از ضرب شونده جمع نمی شود. یک گیت *AND* می تواند این عملیات را بصورت $(X_i Y_j)$ *AND* $(0 \leq i \ll n - 1$ و $0 \leq j \ll n - 1)$ انجام دهد و نتیجه ی این عملیات حاصلضرب جزئی (*Partial Product*) نامیده می شود. شکل (۱-۲)، یک ساختار دوزنقه ای با نام (*PPA*) به نمایش می گذارد که در آن بیت های حاصلضرب جزئی در ستون هایی برای جمع شدن جهت تشکیل حاصلضرب نهایی مرتب شده اند. این فرآیند *Partial Product Generation* نامیده می شود. همان طور که اشاره شد تمام بیت ها به صورت موازی در *PPA* شکل گرفته اند، بنابراین تأخیر استاتیک تمام بیت ها برابر است. عرض *PPA* متناسب با اندازه ضرب شونده است، در حالی که ارتفاع آرایه متناسب با اندازه ضرب کننده می باشد. بعضی ستون های بیشتر نسبت به بقیه دارند و بنابراین عمل جمع بیشتری باید بصورت عمودی بیت های این ستون ها را به دو ردیف کاهش دهد. همانطور که در شکل زیر مشخص است این ستون ها در بیت های میانی قرار دارند.



شکل ۱-۲: تولید حاصلضرب های جزئی در یک ضرب کننده ۱۶ بیتی

۱-۶- کاهش حاصلضرب های جزئی:

پیاده سازی یک ضرب کننده ی دیجیتالی با قابلیت بالا، تا حدود زیادی بستگی به روش جمع کردن بیت حاصلضرب های جزئی آن ضرب کننده دارد. از آنجایی که تولید هر ردیف شیفت یافته از ضرب شونده تأخیری متناسب با عرض ضرب شونده خواهد داشت بلوک ضرب کننده نیاز به مقدار زیادی زمان برای انجام عملیات خواهد داشت، که اگر برای کاهش این ردیف ها از جمع کننده های معمولی استفاده شود، تاخیر زیادی ایجاد خواهد شد. به خصوص زمانی که تعداد بیت ضرب کننده زیاد باشد. به همین خاطر حاصلضرب های جزئی اغلب توسط یک تکنیک به نام *Carry Save Addition* کاهش داده می شوند (شکل ۱-۳).



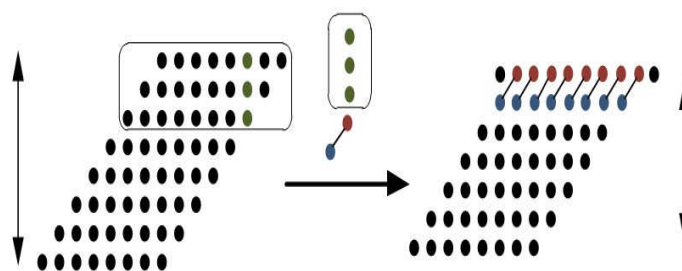
شکل ۱-۳: کاهش حاصلضرب های جزئی با روش *Carry Save Addition*

در این روش جمع شدن پیاپی در یک مرحله کلی اجازه می دهد تا جمع دو بیت از هر دو ردیف ضرب شونده های شیفت یافته، براحتی توسط تمام جمع کننده ی معمولی تحقق یابد. همان طور که در شکل ۱-۳

مشخص شده است با این روش در یک ضرب کننده N بیتی، تبدیل حاصلضرب های جزئی به نتیجه ی نهایی کل ضرب کننده تقریباً تاخیری معادل $2N$ برابر تاخیر یک تمام جمع کننده می باشد و با این حال اگر N عدد بزرگی باشد، تاخیر ضرب کننده نسبتاً خیلی زیاد می شود و برای سرعت های بالا (و بخصوص ضرب کننده های با بیت بالا) باید دنبال روش های دیگری باشیم. در ادامه به چند روش سریعتر و متداول برای کاهش حاصلضرب های جزئی اشاره می کنیم.

۱-۶-۱- کاهش به روش آرایه ای γ :

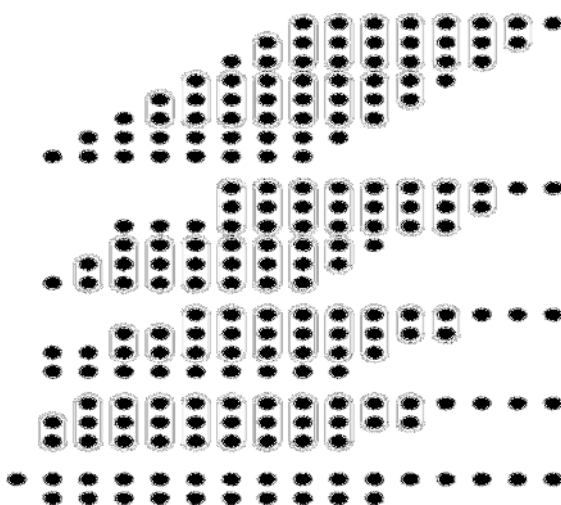
روش های مختلفی برای کاهش حاصلضرب های جزئی ذوزنقه ای شکل وجود دارد. اصلی ترین مزیتی که این روش ها نسبت به روش *Carry Save* دارند این است که حاصلضرب های جزئی با تاخیر کمی به دو ردیف نهایی تبدیل می شوند و چون بیت های این دو ردیف همزمان تولید می شوند، امکان استفاده از جمع کننده های سریعی مثل *Carry Select Adder (CSA)* یا *Carry Lookahead Adder (CLA)* به عنوان جمع کننده ی نهایی وجود دارد. در این بخش ساده ترین روش، به نام کاهش به روش آرایه ای، توصیف خواهد شد. در شکل (۴-۱) *PPA* ذوزنقه ای برای ضرب کننده ی 8×8 بیتی نشان داده شده است (هر نقطه نمایش دهنده یک بیت حاصلضرب است). سه ردیف اول توسط یک سری از تمام جمع کننده ها به ۲ ردیف تبدیل می شوند. بدین ترتیب دو ردیف تولید شده با پنج ردیف باقیمانده ترکیب شده و ۷ ردیف باقی می ماند. این روال ادامه می یابد تا حاصلضرب های جزئی در نهایت به ۲ ردیف کاهش یابند. این طراحی با این که ساده است، ولی سرعت کندی دارد و شاید تنها مزیتش وقتی است که قرار باشد تنها سیم های کوتاه برای اتصال عمودی-افقی و قطری سلول های تمام جمع کننده بکار برده شود [۱۸]. در ادامه روش های مکمل یافته ی این روش با سرعت بالاتر توضیح داده می شود.



شکل ۴-۱: کاهش حاصلضرب های جزئی به روش *Array style*

۱-۶-۲- کاهش به روش درخت والاس^۸:

در سال ۱۹۶۶، C.S Wallace به این نتیجه رسید که می توان ساختاری یافت تا عملیات جمع حاصلضرب های جزئی را به طور موازی انجام دهد. بنابراین تأخیر بطور چشم گیری کاهش می یابد. در این مقاله ی تاریخی Wallace، روشی متفاوت از جمع موازی بیت های حاصلضرب جزئی با استفاده از یک درخت جمع کننده ی ذخیره ی (Carry Save) معرفی کرد، که به درخت والاس (Wallace Tree) معروف شده است. در شکل (۱-۵) PPA دوزنقه ای برای ضرب کننده ی ۸×۸ بیتی نشان داده شده است. سه ردیف اول توسط یک سری از تمام جمع کننده ها به ۲ ردیف تبدیل می شوند. به همین ترتیب سه ردیف دوم نیز همزمان با سه ردیف اول به ۲ ردیف کاهش می یابند. بنابراین در یک ضرب کننده ی ۸×۸ بیتی حاصلضرب های جزئی از ۸ ردیف به ۶ ردیف و با تأخیر یک تمام جمع کننده کاهش می یابد. این روال ادامه می یابد. یعنی ۶ ردیف باقیمانده توسط تمام جمع کننده ها به ۴ ردیف و سپس ۳ و در نهایت به ۲ ردیف کاهش می یابند. این طراحی به دلیل موازی صورت گرفتن عمل جمع سرعت بالاتری نسبت به روش های ذکر شده دارد و تأخیر کاهش حاصلضرب های جزئی آن بصورت لگاریتمی با تعداد بیت ضرب کننده متناسب است. بنابراین برای ضرب کننده های با بیت بالا مناسب تر است.



شکل ۱-۵: کاهش حاصلضرب های جزئی یک ضرب کننده ی ۸×۸ بیتی به روش Wallace Tree

با این حال اصلی ترین عیب درخت Wallace چینش^۹ نامنظم آن نسبت به ساختار آرایه است. بی نظمی در چینش این ساختار اثر بار (Load) را هم زیاد می کند (به خاطر وجود سیم ها ی بیشتر). همچنین ذکر این نکته حائز اهمیت است که عرض (تعداد بیت) جمع کننده ی نهایی در درخت Wallace تقریباً $N - \log_2 N$ بوده، در حالیکه این عرض در ساختار (Carry Save) برابر با N می باشد [۲۳]. ولی با همه ی

8 - Wallace Tree

9 - Layout