طراحی ضرب‌کننده‌های ممیز- شناور با قابلیت کار در مدهای عادی و تحمل‌پذیر اِشکال با استفاده از کاهش دقت محاسبات

مریم مهاجر و مجتبی ولی‌نتاج

چکیده: عملیات ضرب یکی از مهم‌ترین محاسبات مورد استفاده در انواع پردازش‌های سیگنال خصوصاً صوت و تصویر محسوب می‌شود. با این حال، ضرب‌کننده‌ها به عنوان مدارهای دیجیتالی به خاطر وجود عوامل محیطی گوناگون مانند انواع نویزها مستعد تولید خروجی‌های نادرست هستند. در این مقاله، روشی جدید برای طراحی ضرب‌کننده ممیز- شناور ۳۲بیتی ارائه می‌شود که می‌تواند با توجه به شرایط محیطی که در آن استفاده می‌شود، در دو مد کاری عادی یا تحمل‌پذیر اشکال عمل کند. در مد تحمل‌پذیر اشکال، با کاهش دقت محاسبات و قبول مقدار ناچیزی خطای محاسباتی در خروجی، بخشی از مدار اولیه آزاد شده و برای فراهم‌کردن محاسبات افزونه به منظور تشخیص یا تصحیح خطاهای ناشی از اشکال‌ها استفاده می‌شود. بدین روش، دو معماری ضرب‌کننده با قابلیت تشخیص یا تصحیح خطا پیشنهاد می‌شوند که در مد کاری تحمل‌پذیر اشکال، دارای قابلیت اطمینان مناسبی در برابر انواع اشکال‌های دائمی و گذرا هستند. نتایج پیاده‌سازی نشان می‌دهد که در مد تحمل‌پذیر اشکال به جای ۲۳ بیت مانتیس اولیه، حفظ ۱۳ بیت برای دست‌یافتن به ضرب‌کننده با قابلیت تشخیص خطا و حفظ ۱۱ بیت برای دست‌یافتن به ضرب‌کننده با قابلیت تصحیح خطا، با سربار مساحت و توان قابل قبول که از ۱۲٪ تا ۲۶٪ خواهد بود و همچنین حفظ دقت مورد نیاز برای اکثر کاربردها، مناسب است.

کلیدواژه: دقت کاهش‌یافته، تحمل‌پذیری اشکال، تشخیص خطا، تصحیح خطا، ضرب‌کننده ممیز- شناور.

۱- مقدمه

الگوریتم‌های پردازش سیگنال، کاربردهای چندرسانه‌ای و بسیاری از سیستم‌های نهفته به انجام محاسبات ریاضی خصوصاً عملیات ضرب وابسته هستند [۱]. در این راستا انواع پردازش‌های سیگنال، ویدئو، صوت و تصویر به محاسبات حاوی ضرب ممیز- شناور وابسته‌اند. همچنین برنامه‌های کاربردی علمی که نیازمند شبیه‌سازی انواع پدیده‌های فیزیکی هستند از محاسبات ممیز- شناور بهره می‌برند. علاوه بر این، کاهش اندازه ترانزیستورها منجر به بروز مشکلات مرتبط با قابلیت اطمینان[1] مدارهای دیجیتال شده است [۲]. با توجه به اهمیت این نوع از محاسبات، بروز خطا حین انجام کار این نوع از مدارها می‌تواند نتیجه عملیات را بسیار تغییر داده و مشکل‌ساز گردد. بنابراین علاوه بر توان مصرفی کم، قابلیت اطمینان نیز از اهداف مهم در طراحی سیستم‌های محاسباتی امروزی و در نتیجه واحدهای عملیاتی ممیز- شناور حاوی ضرب‌کننده‌ها محسوب می‌گردد.

بسیاری از برنامه‌های کاربردی ممیز- شناور مانند کاربردهای چندرسانه‌ای به دقت کاملی که مطابق استانداردهای IEEE تعریف شده است نیازی ندارند. این برنامه‌ها اگرچه تحمل‌پذیر اشکال[2] نیستند اما می‌توانند محاسبات تقریبی (یا نادقیق)[3] را تحمل کنند و در نتیجه یک مقدار تخمینی برای خروجی آنها کافی است [۳] و [۴]. بنابراین می‌توان این نوع عملیات ممیز- شناور را با دقت کاهش‌یافته[4] اجرا نمود تا با کوچک‌ترشدن مدارهای لازم مساحت مورد نیاز و توان مصرفی کاهش یابد. اما هدف از این مقاله، استفاده‌ای متفاوت از این ویژگی و در واقع بهره‌برداری از واحدهای آزادشده به همراه واحدهای جدید لازم، برای انجام محاسبات افزونه به منظور دست‌یابی به قابلیت اطمینان به صورت تشخیص یا تصحیح خطا است. معماری ضرب‌کننده پیشنهادی در این مقاله مبتنی بر عملیات ممیز- شناور ۳۲بیتی مطابق با قالب نمایش دقت ساده در استاندارد IEEE-۷۵۴ [۵] است. ویژگی بعدی معماری پیشنهادی قابلیت عمل در دو مد کاری عادی و تحمل‌پذیر اشکال است تا با توجه به شرایط کاری محیطی که مدار ضرب‌کننده در آن قرار دارد، قابلیت انتخاب نحوه عمل وجود داشته باشد.

با توجه به این که در طرح‌های پیشنهادی، مد تحمل‌پذیر اشکال بر اساس محاسبات با دقت کاهش‌یافته قابل استفاده است برخی خطاهای محاسباتی در خروجی تولید می‌شود که با توجه به نتایج ارائه‌شده در [۶] در مورد دقت لازم و محدوده خطاهای محاسباتی برای کاربردهای گوناگون، قابل قبول خواهند بود. بنابراین در این مقاله با پذیرفتن درصدی ناچیز از خطای محاسباتی در خروجی به خاطر محاسبات تقریبی و تغییر ساختار ضرب‌کننده اصلی درونی، دو معماری ضرب‌کننده ممیز- شناور یکی با ویژگی تشخیص خطا و دیگری با ویژگی تصحیح خطا برای مقابله با انواع اشکال‌های دائمی و گذرا پیشنهاد می‌شود که با توجه به محدودبودن خطای محاسباتی، خروجی آنها برای بسیاری از برنامه‌های کاربردی چندرسانه‌ای رضایت‌بخش است، چون بسیاری از اعوجاج‌ها در خروجی به دلیل محدودیت حواس بینایی و شنوایی انسان به طور طبیعی پوشش داده می‌شوند.

بخش‌های بعدی مقاله به صورت زیر سازماندهی شده‌اند: در بخش دوم تحقیقات پیشین و مرتبط و در بخش سوم یک پیش‌زمینه از استاندارد IEEE برای اعداد ممیز- شناور و ضرب‌کننده ممیز- شناور با دقت ساده ارائه می‌شود. در ادامه در بخش چهارم معماری‌های پیشنهادی برای ضرب‌کننده ممیز- شناور در دو مد کاری عادی و تحمل‌پذیر اشکال، یکی با قابلیت تشخیص خطا و دیگری با قابلیت تصحیح خطا توضیح داده می‌شوند. سپس در بخش پنجم، طرح‌های پیشنهادی از نظر تأخیر، توان

این مقاله در تاریخ ۱۲ مهر ماه ۱۳۹۶ دریافت و در تاریخ ۱۶ اردیبهشت ماه ۱۳۹۷ بازنگری شد.

مریم مهاجر، کارشناس ارشد معماری سیستم‌های کامپیوتری، دانشگاه صنعتی نوشیروانی بابل، بابل، (email: maryam.mohajer@stu.nit.ac.ir).

مجتبی ولی‌نتاج، استادیار دانشکده مهندسی برق و کامپیوتر، دانشگاه صنعتی نوشیروانی بابل، بابل، (email: m.valinataj@nit.ac.ir).

1. Reliability
2. Fault-Tolerant
3. Approximate (or Inexact) Computations
4. Reduced Precision

مصرفی و مساحت مورد ارزیابی قرار گرفته و در پایان در بخش ششم، نتیجه‌گیری نهایی ارائه می‌گردد

۲- تحقیقات مرتبط

با توجه به پیچیدگی واحدهای محاسباتی ممیز- شناور از جمله ضرب‌کننده‌های ممیز- شناور، تحمل‌پذیر کردن آنها در برابر اشکال‌ها و خرابی‌های[1] احتمالی یکی از مسایل مهمی است که باید توجه دقیقی به آن نمود. چندین روش پایه برای بهبود قابلیت اطمینان وجود دارد اما این روش‌ها سربار توان و مساحت زیادی را تحمیل می‌کنند. روش‌های سنتی تشخیص یا تصحیح خطا مانند دونسخه‌سازی و مقایسه[2]، TMR[3] و افزونگی زمانی (با اجرای مجدد) که در پردازنده‌های ممیز- شناور استفاده شده‌اند، اغلب دارای سربار مساحت و توان زیادی هستند.

تاکنون واحدهای محاسباتی ممیز- ثابت متنوعی به صورت تحمل‌پذیر اشکال طراحی شده‌اند مانند [۷] تا [۱۱] اما به طراحی واحدهای محاسباتی ممیز- شناور توجه کمتری شده است. در [۱۲] یک واحد محاسباتی ممیز- شناور تشخیص‌دهنده خطا که از کدهای مانده کم‌هزینه استفاده می‌کند پیشنهاد شده است. اما با توجه به ماهیت کدهای مانده، طرح پیشنهادی تنها برای تشخیص خطا می‌تواند به کار رود و کمکی به تصحیح خطا نمی‌کند. در [۱۳] یک روش فقط برای حفاظت مدار کنترلی واحد ممیز- شناور با بررسی بخش نمای عدد ممیز- شناور ارائه شده است. این روش با توجه به عدم بررسی خطا در مانتیس[4]، تعداد خطاهای کمتری را نسبت به [۱۲] تشخیص می‌دهد و علاوه بر این، فقط برای تشخیص خطاهای گذرا در مدار کنترلی واحد ممیز- شناور مناسب است.

در زمینه محاسبات تقریبی در ضرب‌کننده‌های ممیز- شناور که کاهش دقت محاسبات در آنها با حذف بیت‌های کم‌ارزش‌تر مانتیس انجام می‌شود، کارهای متنوعی انجام شده است. اما بیشتر روش‌های قبلی مبتنی بر محاسبات تقریبی مانند [۱۴] تا [۱۹] با هدف کاهش توان مصرفی بخش اصلی ضرب‌کننده که ضرب مانتیس‌ها را انجام می‌دهد، ابداع شده‌اند؛ بدین دلیل که این بخش، بیشترین توان مصرفی (حدود ۸۰٪) را به خود اختصاص داده است [۲۰]. بعضی از روش‌های مرتبط با محاسبات تقریبی نیز بر روی کاهش مساحت مصرفی و افزایش سرعت تمرکز داشته‌اند مانند [۲۱] و [۲۲]. همچنین روش‌های دیگری مبتنی بر محاسبات تقریبی وجود دارند که به صراحت برای کاربردهای مقاوم در برابر خطا ابداع شده‌اند مانند [۲۳] و [۲۴]. اما بایستی توجه داشت که خود این روش‌ها از نوع تحمل‌پذیر خطا یا اشکال نیستند، بلکه روش‌های ابداع‌شده برای کاربردهایی مناسب هستند که مقداری خطای محاسباتی ناشی از محاسبات تقریبی در خروجی آنها قابل قبول است.

نتایج ارائه‌شده در [۶] نشان می‌دهد که همه برنامه‌های کاربردی برای رسیدن به نتایج معتبر، به دقت اولیه فراهم‌شده توسط سخت‌افزار ممیز- شناور نیاز ندارند و می‌توانند تعداد بیت‌های کمتری از مانتیس و نما را در سخت‌افزار مربوطه به کار گیرند و در نتیجه، مساحت و توان مصرفی می‌تواند کاهش یابد. طبق نتایج ارائه‌شده در [۶]، با کاهش عرض بیت مانتیس از ۲۳ بیت به ۱۱ بیت در نمایش ممیز- شناور ۳۲بیتی، هیچ یک از برنامه‌های ممیز- شناور تحت آزمون کاهش قابل توجهی در دقت را نشان ندادند و حتی خروجی برخی از برنامه‌ها با کمتر از ۵ بیت مانتیس نیز قابل قبول بوده است. در نتیجه، بسیاری از برنامه‌ها که با حواس انسان سروکار دارند مانند برنامه‌های مرتبط با پردازش تصویر، صوت و ویدئو، با کاهش دقت محاسبات هنوز می‌توانند خروجی‌های مناسبی تولید نمایند.

در [۲۵] یک روش تشخیص خطا برای جمع‌کننده ممیز- شناور ارائه شده که از یک واحد جمع وارسی‌کننده[5] با دقت کاهش‌یافته برای تعیین این که آیا نتیجه در محدوده خطای قابل قبول درست است یا خیر استفاده می‌کند. در این روش با کاهش تعدادی از بیت‌های مانتیس، مساحت و توان مصرفی کاهش می‌یابد. محاسبه کامل جمع به موازات محاسبه افزونه که در واحد جمع وارسی‌کننده انجام می‌شود و در آن فقط بیت‌های پرارزش عملوندها دریافت می‌شود، انجام می‌گردد. در مرحله آخر، نتایج این دو واحد با هم مقایسه شده و وقوع خطا تشخیص داده می‌شود. در [۲۶] تکنیک وارسی مبتنی بر کاهش دقت (RPC)[6] مشابه [۲۵] به ضرب ممیز- شناور اعمال شده تا خطاها را تشخیص دهد. با توجه به این طرح، تشخیص خطا در ضرب‌کننده ممیز- شناور با هزینه معقول امکان‌پذیر است در حالی که امکان تصحیح خطا وجود ندارد.

در [۲۷] دو طرح ساده برای تشخیص و تصحیح خطا در ضرب‌کننده ممیز- شناور ارائه شده که با توجه به کوچک‌تر شدن مدار به خاطر استفاده از محاسبات تقریبی، افزونگی‌های لازم به نحوی اعمال شده‌اند که مساحت یا توان مصرفی مدار جدید در همان حدود مدار اولیه باقی بماند. طرح‌های پیشنهادی در [۲۷] همانند روش‌های قبلی تنها در یک مد کاری قابل استفاده هستند.

با بررسی کارهای گذشته می‌توان دریافت که طرحی مشابه طرح‌های پیشنهادی در این مقاله که قابلیت عمل در دو مد کاری عادی و تحمل‌پذیر اشکال را داشته باشد وجود ندارد. علاوه بر این، غیر از یکی از طرح‌های ارائه‌شده در [۲۷] طرحی مخصوص ضرب‌کننده ممیز- شناور با قابلیت تصحیح خطا در کارهای پیشین وجود ندارد.

۳- مبانی

۳-۱ استاندارد IEEE برای اعداد ممیز- شناور

معمول‌ترین نمایش‌هایی که برای اعداد حقیقی استفاده می‌شوند، روش‌های نمایش ممیز- ثابت و ممیز- شناور هستند [۲۸]. نمایش ممیز- شناور محدوده پویای وسیع‌تر و دقت بالاتری را در مقایسه با نمایش ممیز- ثابت، فراهم اما مساحت و توان مصرفی بیشتری را نیز تحمیل می‌کند. نمایش معمول اعداد ممیز- شناور مطابق استاندارد IEEE -۷۵۴ [۵] در دو قالب نمایش دقت ساده[7] و دقت مضاعف[8] انجام می‌شود. قالب قالب نمایش دقت ساده ۳۲بیتی است که شامل ۱ بیت علامت، ۸ بیت نما[9] و ۲۳ بیت مانتیس است. در این مقاله، تمرکز روی قالب نمایش دقت دقت ساده است. شکل ۱ نمایش دودویی قالب دقت ساده و (۱) نحوه محاسبه مقدار عدد را نشان می‌دهد

$$Number = (-1)^{Sign} \times 1.Mantissa \times 2^{Exponent-127} \qquad (1)$$

1. Failure
2. Duplication and Comparison
3. Triple Modular Redundancy
4. Mantissa
5. Checker Adder
6. Reduced Precision Checking
7. Single Precision
8. Double Precision
9. Exponent

Sign (1-bit)	Biased Exponent (8-bit)	Mantissa (23 bit)

شکل ۱: نمایش دودویی قالب دقت ساده.

جدول ۱: مقادیر ممیز- شناور به ازای مقادیر مختلف نما و مانتیس.

عدد صفر	$E=۰\ ,\ M=۰$
اعداد ناهنجار	$E=۰\ ,\ M\neq ۰$
اعداد هنجار ممیز- شناور	$۰<E<۲۵۵$
بی‌نهایت	$E=۲۵۵\ ,\ M=۰$
اعداد غیر ممیز- شناور	$E=۲۵۵\ ,\ M\neq ۰$

طبق استاندارد IEEE -۷۵۴ بخش مانتیس یک عدد ممیز- شناور همیشه کوچک‌تر از یک است اما همواره یک بیت ضمنی '۱' در سمت چپ نقطه اعشار در نظر گرفته می‌شود و همراه با مانتیس، significand ۲۴بیتی را تشکیل می‌دهد. بنابراین significand عددی است که در محدوده $[۱,۲)$ قرار دارد و به عدد ممیز- شناور حاوی چنین significandای عدد هنجار می‌گویند. بیت علامت، علامت significand را نشان می‌دهد در حالی که علامت نما به طور ضمنی درون آن قرار دارد. در واقع، نما به شکل افزون بر ۱۲۷ نمایش داده شده که با توجه به غیر منفی بودن نما هنگام ذخیره‌شدن (از ۰ تا ۲۵۵) با کم‌کردن عدد ۱۲۷ از آن نمای واقعی منفی یا مثبت می‌تواند ساخته شود. مقادیر ممیز- شناور به ازای مقادیر مختلف نما و مانتیس در جدول ۱ نشان داده شده است.

۳-۲ ضرب‌کننده ممیز- شناور با دقت ساده

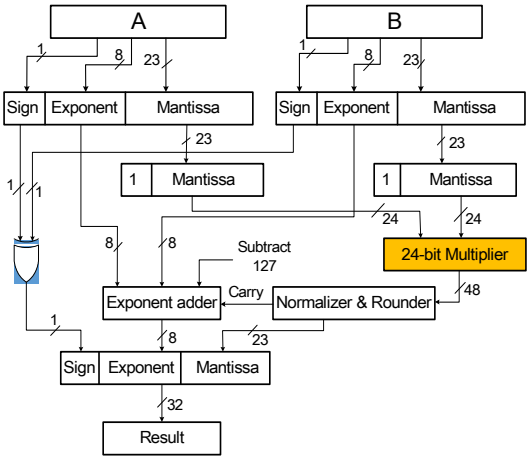
ضرب‌کننده ممیز- شناور ۳۲بیتی عمل ضرب دو عملوند ورودی با دقت ساده را انجام داده و حاصل‌ضرب را نیز در دقت ساده ۳۲بیتی تولید می‌کند. الگوریتم ضرب دو عدد ممیز- شناور طبق ۶ مرحله انجام می‌شود:

مرحله ۱: ضرب مانتیس‌ها به همراه بیت ضمنی (ضرب‌کننده ۲۴بیتی)
مرحله ۲: هنجارسازی و گردکردن مانتیس نتیجه
مرحله ۳: جمع نماها
مرحله ۴: محاسبه علامت نتیجه
مرحله ۵: بررسی وقوع سرریز (یا فروریز)
مرحله ۶: ساختن نتیجه استاندارد

شکل ۲ طرح کلی ضرب‌کننده ممیز- شناور ۳۲بیتی را نشان می‌دهد. با توجه به این که در این طرح، بلوک ضرب‌کننده اصلی ۲۴بیتی است، خروجی این ضرب‌کننده عددی ۴۸بیتی است اما طبق استاندارد برای قالب با دقت ساده، پس از جداسازی بیت ضمنی تنها به ۲۳ بیت از آن برای مانتیس نتیجه نیاز است. علاوه بر این، خروجی ۴۸بیتی ضرب‌کننده از یک هنجارساز و گردکننده عبور می‌کند تا در صورت نیاز باعث تغییر مقدار نمای نهایی شود.

۴- طرح‌های پیشنهادی

با توجه به این که در بسیاری از کاربردها قابلیت تحمل محاسبات تقریبی وجود دارد، در این مقاله از روش کاهش دقت محاسبات از طریق حذف بیت‌های کم‌ارزش‌تر استفاده می‌شود تا با توجه به نیاز به سخت‌افزاری کمتر بتوان یک سری افزونگی را برای دست‌یابی به قابلیت اطمینان از نوع تشخیص یا تصحیح خطا در مد کاری تحمل‌پذیر اشکال اعمال نمود. در واقع ذکر این نکته ضروری است که گرچه بسیاری از کاربردها می‌توانند محاسبات تقریبی را تحمل کنند، اما انحراف در خروجی



شکل ۲: ضرب‌کننده ممیز- شناور ۳۲بیتی پایه.

و نتایج کاربردهای گوناگون به دلیل رخداد خطاهای تصادفی هنگام انجام محاسبات، اکثراً غیر قابل قبول هستند. در این مقاله منظور از انحراف در خروجی، خطاهای تولیدشده در خروجی است که به علت رخداد اشکال‌های گذرا یا دائمی درون مدار اصلی، ایجاد می‌شوند. همچنین مدل اشکال فرض‌شده مبتنی بر تغییرات رخداده در سطح منطقی و صفر یا یک‌شدن سیگنال‌ها است.

ضرب‌کننده ۲۴بیتی نشان داده شده در شکل ۲ که برای ضرب مانتیس‌ها به کار می‌رود، بلوک اصلی ضرب‌کننده ممیز- شناور ۳۲بیتی است. این ضرب‌کننده حدود ۸۰٪ مساحت و توان مصرفی کل ضرب‌کننده ممیز- شناور را به خود اختصاص داده است. بنابراین احتمال رخداد خطا در آن بسیار بیشتر از سایر بلوک‌های درونی است. به همین دلیل، قابلیت تشخیص یا تصحیح خطا را اساساً روی این بلوک اعمال می‌نماییم.

۴-۱ معماری ضرب‌کننده ممیز- شناور با قابلیت تشخیص خطا

ضرب‌کننده مانتیس استفاده‌شده در معماری پیشنهادی، یک ضرب‌کننده با ساختار ماجولار است. این نوع ضرب‌کننده بر اساس جمع حاصل‌ضرب‌های پاره‌ای با ساختارهای تکرارشونده عمل می‌کند. بنابراین اگر ضرب‌کننده مورد نظر $۲n\times ۲n$ بیتی باشد (یعنی عملوندهای آن $۲n$ بیتی باشند) آن گاه می‌توانیم هر عملوند ورودی را به دو نیمه n بیتی مرتبه بالا و مرتبه پایین تقسیم نموده و ضرب‌کننده را بر اساس ضرب $n\times n$ بیتی طراحی نماییم [۲۸]. برای مثال اگر عملوند $۲n$ بیتی A را به صورت $A=A_H.A_L$ نشان دهیم، بدین معنا است که دو بخش مرتبه بالا و مرتبه پایین A_H و A_L، n بیتی هستند. بدین ترتیب می‌توان ضرب‌کننده $۲n$ بیتی را با استفاده از ماجول‌های کوچک‌تر ضرب‌کننده n بیتی طراحی نمود. به عنوان مثال، روند انجام عملیات ضرب میان دو عملوند $۲n$ بیتی A و B که به دو نیمه بالا و پایین تقسیم شده‌اند در شکل ۳ نشان داده شده است. تقسیم‌کردن هر عملوند به دو نیمه، چهار حاصل‌ضرب پاره‌ای را تولید می‌کند. در این شکل، ضرایب ۲^{n} و $۲^{۲n}$ برای هم‌ترازکردن وزن حاصل‌ضرب‌های پاره‌ای استفاده شده‌اند. شکل ۴ نیز به نحوی تفاوت وزن حاصل‌ضرب‌های پاره‌ای و نیاز به هم‌ترازی آنها را قبل از جمع‌نمودن آنها با هم نشان می‌دهد. برای افزایش سرعت جمع و انجام آن به طور موازی، هرگاه تعداد عملوندهای جمع‌شونده بیش از دو باشد، استفاده از جمع‌کننده‌ای با نام CSA[1] سودمند خواهد بود. شکل ۵

1. Carry Save Adder

$$P = A \times B = A_H.A_L \times B_H.B_L$$
$$= A_L \times B_L + A_H \times B_L \times ۲^n$$
$$+ A_L \times B_H \times ۲^n + A_H \times B_H \times ۲^{۲n}$$

شکل ۳: ضرب دو عملوند A و B با تجزیه به ضرب‌های کوچک‌تر.

$$\begin{array}{rl} & A_H.A_L \\ \times & B_H.B_L \\ \hline & A_L \times B_L \\ + & A_H \times B_L \\ + & A_L \times B_H \\ + & A_H \times B_H \\ \hline & \text{Product} \end{array}$$

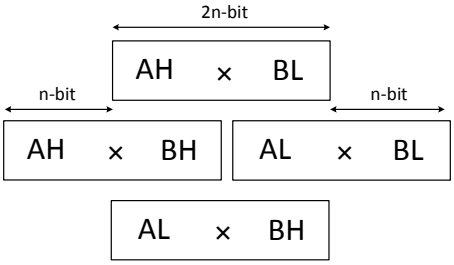
شکل ۴: ضرب $A \times B$ و تولید حاصل‌ضرب‌های پاره‌ای با وزن‌های متفاوت.

سازماندهی مناسب چهار حاصل‌ضرب پاره‌ای تولیدشده را برای کاهش تعداد سطوح در استفاده از CSA نشان می‌دهد. با این ساختار عمل ضرب در دو مرحله انجام خواهد شد. در مرحله اول، چهار ضرب‌کننده کوچک‌تر n بیتی حاصل‌ضرب‌های پاره‌ای را تولید می‌کنند. سپس در مرحله دوم حاصل‌ضرب‌های پاره‌ای با استفاده از چینش نشان داده شده در شکل ۵ با هم جمع شده و نتیجه نهایی $۴n$ بیتی تولید می‌شود.

اعمال معماری ضرب‌کننده ماجولار توضیح داده شده به بخش اصلی ضرب‌کننده ممیز- شناور با دقت ساده که در شکل ۲ با رنگ متفاوت نشان داده شده است امکان‌پذیر است. بنابراین با انجام این کار امکان استفاده از ضرب‌کننده‌های کوچک‌تر درونی برای انجام محاسبات افزونه برای دسترسی به قابلیت اطمینان در مد تحمل‌پذیر اشکال فراهم خواهد شد. بنابراین ضرب‌کننده ۲۴بیتی در شکل ۲ که برای ضرب دو significand عملوندهای ورودی به کار می‌رود، می‌تواند با استفاده از چهار ضرب‌کننده ۱۲بیتی مشابه شکل ۵ پیاده‌سازی شود.

ویژگی ضرب‌کننده ممیز- شناور پیشنهادی در این بخش این است که می‌تواند در مد کاری عادی (با دقت کامل شامل یک ضرب‌کننده ۲۴بیتی) یا مد کاری تحمل‌پذیر اشکال با قابلیت تشخیص خطا (با دقت کاهش‌یافته شامل دو ضرب‌کننده کوچک‌تر m بیتی) کار کند. با توجه به وجود تعداد کافی ضرب‌کننده ۱۲بیتی، حداقل مقدار m را برابر با ۱۲ در نظر می‌گیریم که به معنای مانتیس ۱۱بیتی و حذف ۱۲ بیت کم‌ارزش‌تر مانتیس عملوندهای ورودی است. در مد تحمل‌پذیر اشکال با توجه به این نوع حذف، برخی از ضرب‌کننده‌های ۱۲بیتی آزاد می‌شوند که می‌توانند برای انجام محاسبات افزونه با هدف تشخیص (یا تصحیح خطا) به کار روند. بنابراین برای دست‌یابی به قابلیت تشخیص خطا در ضرب‌کننده ممیز- شناور، به جای یک ضرب‌کننده ۲۴بیتی از دو ضرب‌کننده با دقت کاهش‌یافته m بیتی ($m = ۱۲ + t$ و $t \le ۶$ که دلیل آن در ادامه توضیح داده خواهد شد) طبق مفهوم روش دونسخه‌سازی و مقایسه استفاده می‌نماییم. بنابراین هر مانتیس ۲۳بیتی از عملوندهای ورودی به یک مانتیس k بیتی جدید ($k = ۱۱ + t$) تبدیل می‌شود (یعنی حذف $۲۳ - k$ بیت کم‌ارزش‌تر مانتیس) و این مانتیس به همراه یک بیت ضمنی، یک significand m بیتی ($m = k + ۱$) را تشکیل می‌دهد. در معماری با قابلیت تشخیص خطا به جای یک ضرب‌کننده ۲۴بیتی با خروجی ۴۸بیتی، دو ضرب‌کننده با دقت کاهش‌یافته m بیتی استفاده می‌شود تا دو خروجی $۲m$ بیتی تولیدشده بعد از عبور از واحد هنجارساز و گردکننده توسط مقایسه‌کننده با هم مقایسه شوند تا خطای احتمالی تشخیص داده شود.

ساختار کلی ضرب‌کننده ممیز- شناور پیشنهادی در دو مد کاری عادی

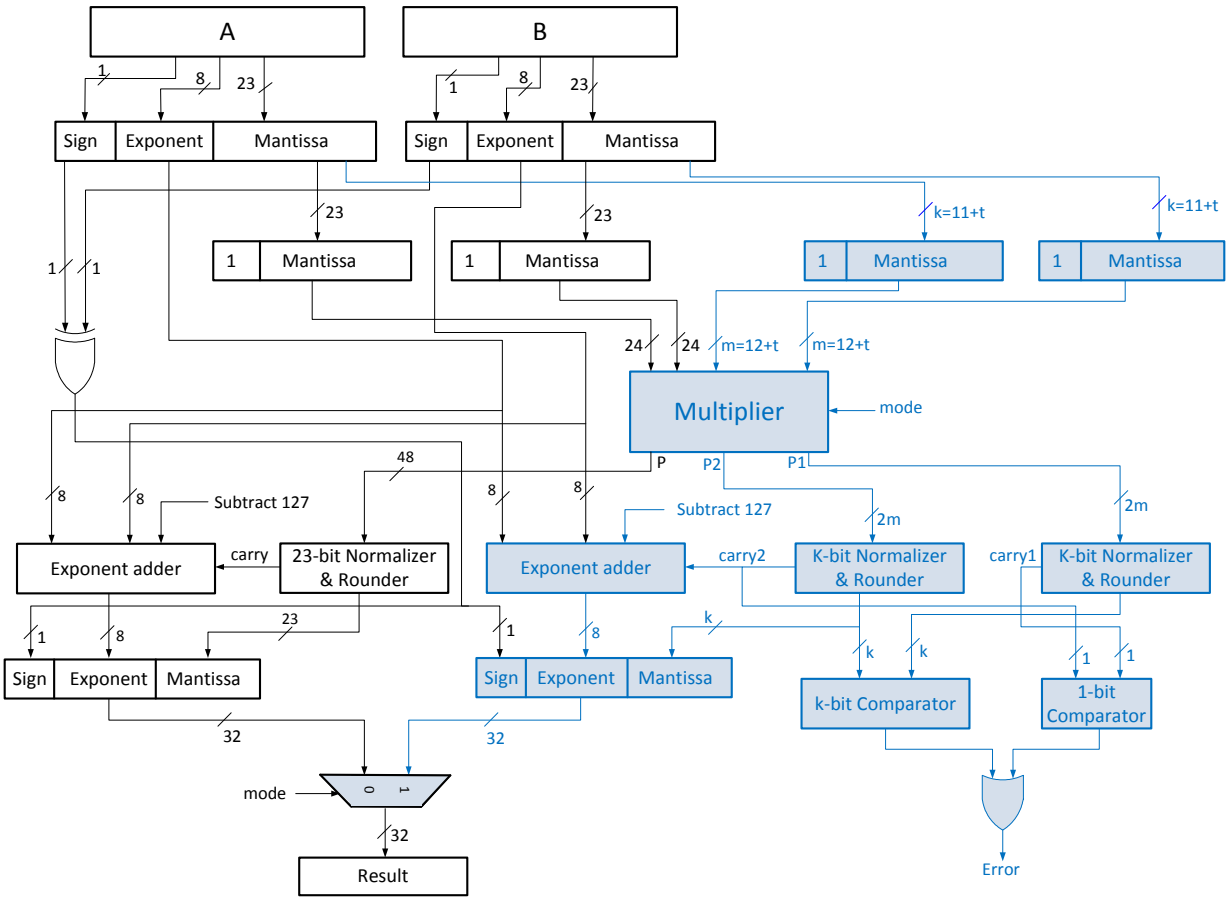


شکل ۵: سازماندهی مناسب حاصل‌ضرب‌های پاره‌ای برای انجام ضرب $۲n$ بیتی $A \times B$.

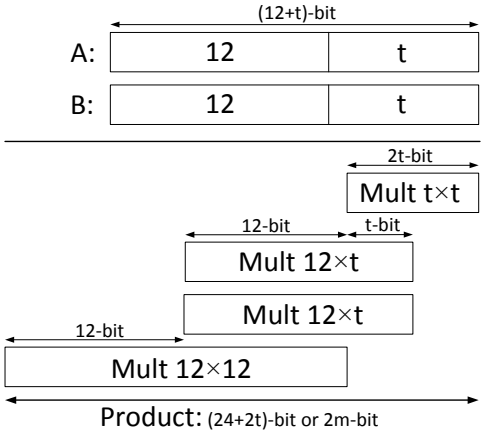
و تحمل‌پذیر اشکال با قابلیت تشخیص خطا در شکل ۶ نشان داده شده است. در این شکل، بخش‌های رنگی نشان‌دهنده بلوک‌های تغییریافته ساختار اولیه ضرب‌کننده ممیز- شناور یا بلوک‌های جدیدی هستند که به ساختار اولیه اضافه شده‌اند. با توجه به شکل ۶، دو حاصل‌ضرب $۲m$ بیتی تولیدی از بلوک‌های هنجارساز و گردکننده عبور کرده و حاصل آنها به k بیت گرد می‌شود. اگر خروجی‌های متناظر با دو ضرب‌کننده m بیتی با هم برابر نباشند یا رقم‌های نقلی تولیدشده توسط هنجارسازها با هم برابر نباشند، سیگنال خطا برابر یک شده و بروز خطا اعلام می‌شود. در طرح پیشنهادی شکل ۶ فرض بر این است که اگر دو عملوند ورودی یک مقایسه‌کننده با هم برابر نباشند، خروجی آن برابر یک می‌شود. ضمناً در این شکل، سیگنال ورودی mode با مقادیر صفر یا یک به ترتیب تعیین‌کننده مدهای کاری عادی یا تحمل‌پذیر اشکال است.

پیاده‌سازی هر کدام از دو نسخه ضرب‌کننده مورد نیاز با دقت کاهش‌یافته m بیتی ($m = ۱۲ + t$) طبق شکل ۷ انجام می‌شود. با توجه به حاصل‌ضرب‌های پاره‌ای تولیدشده در این ضرب‌کننده، هر ضرب‌کننده m بیتی به یک ضرب‌کننده ۱۲بیتی (۱۲×۱۲)، دو ضرب‌کننده $۱۲ \times t$ و یک ضرب‌کننده t بیتی ($t \times t$) احتیاج دارد. از طرف دیگر، همان طور که قبلاً بیان شد، ضرب‌کننده ۲۴بیتی اصلی را می‌توان با توجه به شکل ۵ با استفاده از چهار ضرب‌کننده ۱۲بیتی ساخت. علاوه بر این، دو ضرب‌کننده $۱۲ \times t$ را تا زمانی که t بزرگ‌تر از ۶ نباشد می‌توان به طور کامل با یک ضرب‌کننده ۱۲بیتی پیاده‌سازی کرد زیرا هر ضرب‌کننده ۱۲بیتی خود می‌تواند از چهار ضرب‌کننده ۶بیتی تشکیل شود (هر ضرب‌کننده $۱۲ \times t$ از دو ضرب‌کننده $۶ \times t$ تشکیل می‌شود). در نتیجه، مقدار t در رابطه $m = ۱۲ + t$ حداکثر برابر ۶ انتخاب می‌شود ($t \le ۶$). مسلماً برای tهای بزرگ‌تر امکان پیاده‌سازی وجود دارد اما به علت عدم وجود مدار اولیه برای انجام کامل آن، سربار زیادی تحمیل خواهد شد.

در ادامه، معماری پیشنهادی برای تجزیه ضرب‌کننده اصلی ۲۴بیتی به ضرب‌کننده‌های کوچک‌تر برای استفاده در دو مد کاری شرح داده می‌شود. ساختار ضرب‌کننده اصلی ۲۴بیتی برای استفاده در مدهای کاری عادی و تحمل‌پذیر اشکال با قابلیت تشخیص خطا به صورت شکل ۸ پیشنهاد می‌شود. در این شکل، نسخه اول ضرب‌کننده با دقت کاهش‌یافته m بیتی به کمک دو ضرب‌کننده ۱۲بیتی بالایی و سمت راست و نسخه دوم ضرب‌کننده با دقت کاهش‌یافته m بیتی به کمک دو ضرب‌کننده ۱۲بیتی پایینی و سمت چپ پیاده‌سازی شده است. با توجه به این که ضرب‌کننده تغییریافته بایستی قابلیت اجرا در هر کدام از مدهای کاری را داشته باشد، یک سری مالتی‌پلکسر برای انتخاب ورودی‌های متفاوت، درون ضرب‌کننده قرار داده شده‌اند. در هر نسخه ضرب‌کننده m بیتی یک ضرب‌کننده ۱۲بیتی (در نسخه اول ماجول $L \times L$ و در نسخه دوم ماجول $H \times H$) بدون تغییر در ساختار درونی برای پیاده‌سازی ضرب‌کننده ۱۲×۱۲ شکل ۷ استفاده می‌شود. اما در هر دو نسخه، ضرب‌کننده



شکل ۶: ساختار ضرب‌کننده پیشنهادی ممیز- شناور در دو مد کاری عادی و تحمل‌پذیر اشکال با قابلیت تشخیص خطا.



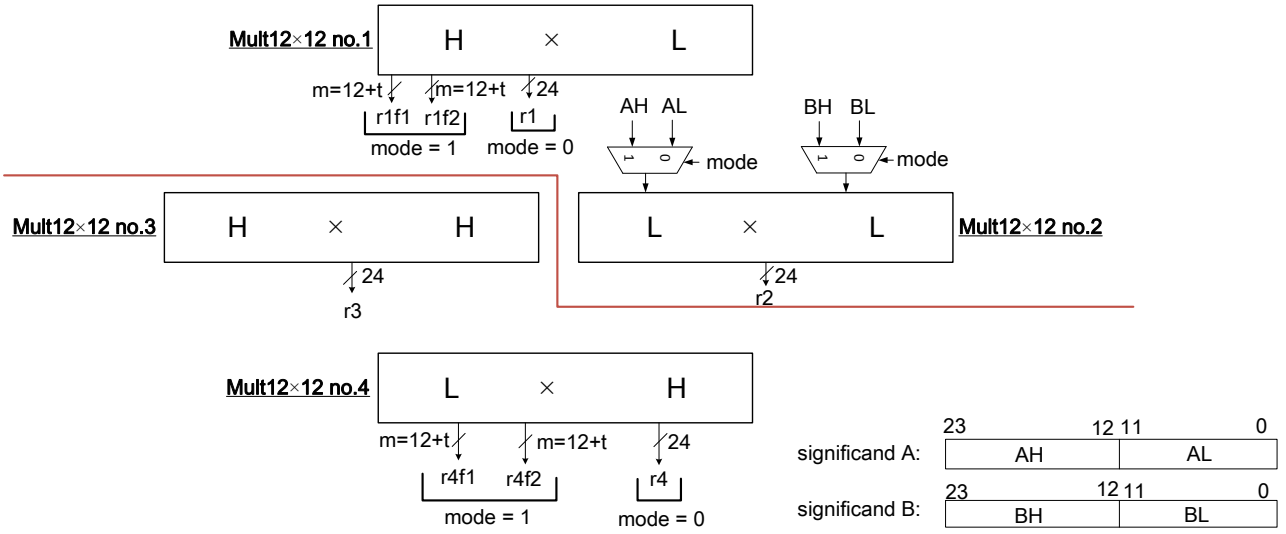
شکل ۷: نحوه انجام ضرب m یا $(12+t)$ بیتی.

۱۲بیتی باقیمانده (ماجول‌های $H\times L$ و $L\times H$) باید مطابق شکل ۹ برای پیاده‌سازی دو ضرب‌کننده $12\times t$ درونی طراحی شود. البته شکل ۹ جزئیات معماری درونی ضرب‌کننده ۱۲بیتی $H\times L$ را نشان می‌دهد که قابل تعمیم به ضرب‌کننده ۱۲بیتی $L\times H$ است. بدین ترتیب تنها ضرب‌کننده tبیتی $(t\times t)$ مورد نیاز درون هر ضرب‌کننده mبیتی (کوچک‌ترین حاصل‌ضرب پاره‌ای در شکل ۷) بایستی به طور جداگانه طراحی شده و به مدار اضافه گردد.

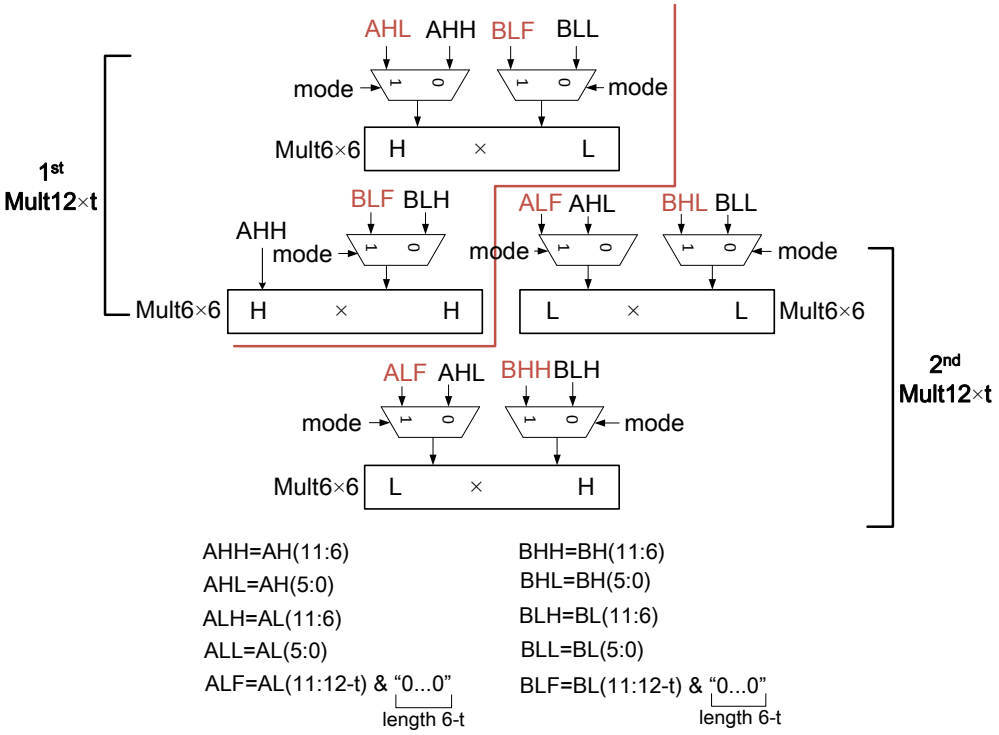
با توجه به شکل ۸، ضرب‌کننده ۱۲بیتی شماره ۲ (ماجول $L\times L$)، در مد تحمل‌پذیر اشکال حاصل‌ضرب ۲۴بیتی $A_H\times B_H$ را برای نسخه اول ضرب‌کننده mبیتی تولید می‌کند (r_2)، در حالی که در نسخه دوم، ضرب‌کننده ۱۲بیتی شماره ۳ (ماجول $H\times H$) همین عملیات ضرب را انجام می‌دهد که با توجه به نیاز به اجرای این عملیات در هر دو مد کاری، نیازی به استفاده از مالتی‌پلکسر در ورودی‌های آن نیست. با توجه به قابلیت پیاده‌سازی ضرب‌کننده $12\times t$ با استفاده از دو ضرب‌کننده ۶بیتی، پیاده‌سازی دو ضرب‌کننده $12\times t$ مورد نیاز در هر نسخه ($A_H\times B_{LF}$ و $B_H\times A_{LF}$)، به چهار ضرب‌کننده ۶بیتی نیاز دارد. از طرف دیگر، هر ضرب‌کننده ۱۲بیتی خود می‌تواند از چهار ضرب‌کننده ۶بیتی تشکیل شود. بنابراین ضرب‌کننده ۱۲بیتی شماره ۱ (ماجول $H\times L$) مطابق شکل ۹ به چهار ضرب‌کننده ۶بیتی تجزیه می‌شود تا برای تولید دو حاصل‌ضرب $12+t$بیتی (خروجی‌های ضرب‌کننده‌های $12\times t$) استفاده شود. همین تغییر در معماری برای ضرب‌کننده ۱۲بیتی شماره ۴ (ماجول $L\times H$) نیز انجام می‌گردد و هر دو نسخه ضرب‌کننده mبیتی ضرب‌های درونی خود را به طور موازی انجام می‌دهند. در نتیجه، همه حاصل‌ضرب‌های پاره‌ای مورد نیاز طبق شکل ۷ برای هر دو نسخه فراهم می‌گردند که پس از آن با استفاده از جمع‌کننده از نوع CSA با هم جمع شده و برای هر نسخه یک حاصل $2m$بیتی تولید می‌شود. ادامه کار مطابق شکل ۶ انجام خواهد شد.

۴-۲ معماری ضرب‌کننده ممیز- شناور با قابلیت تصحیح خطا

طرح کلی پیشنهادی برای ضرب‌کننده ممیز- شناور در دو مد کاری عادی و تحمل‌پذیر اشکال با قابلیت تصحیح خطا در شکل ۱۰ آمده است. در این شکل، بخش‌های رنگی نشان‌دهنده بلوک‌های تغییریافته ساختار اولیه ضرب‌کننده ممیز- شناور یا بلوک‌های جدیدی هستند که به ساختار اولیه اضافه شده‌اند. در این طرح، برای دست‌یافتن به قابلیت تصحیح خطا در مد تحمل‌پذیر اشکال، ضرب‌کننده ۲۴بیتی اولیه که خروجی ۴۸بیتی P را تولید می‌کند به گونه‌ای تنظیم می‌شود که سه عملیات ضرب با



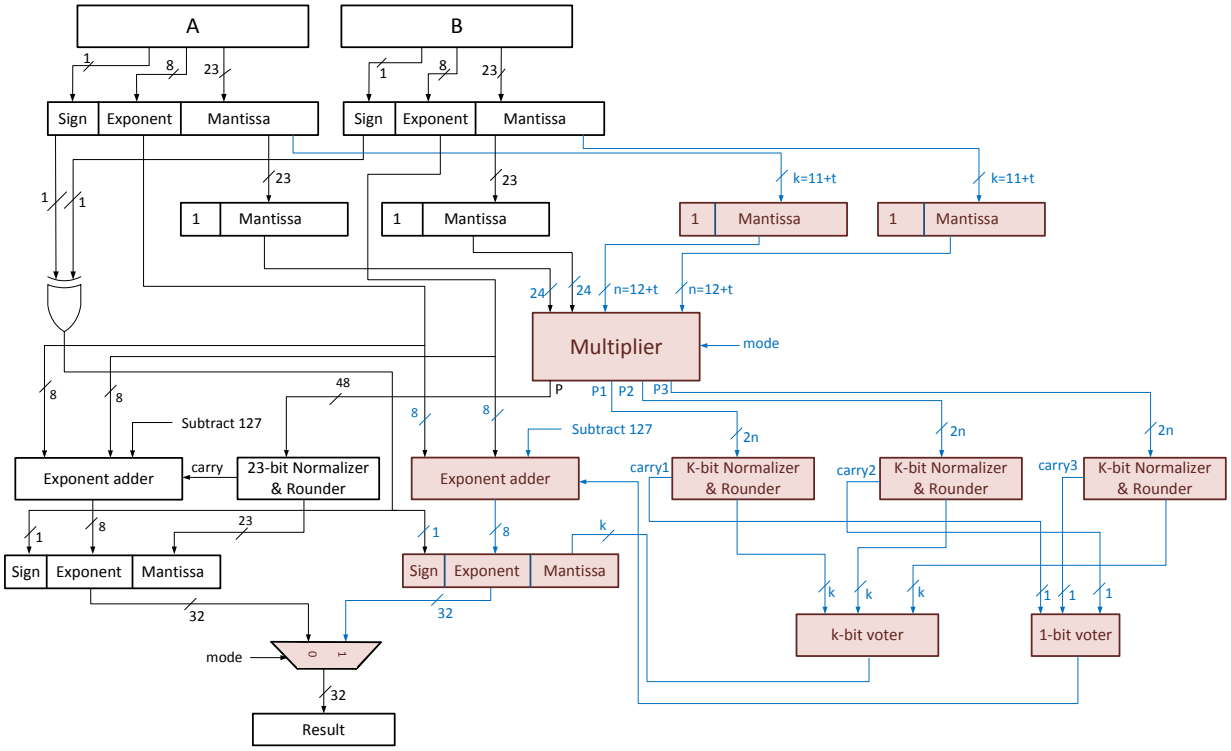
شکل ۸: ساختار ضرب‌کننده اصلی ۲۴بیتی پیشنهادی برای استفاده در مدهای کاری عادی و تحمل‌پذیر اشکال با قابلیت تشخیص خطا.



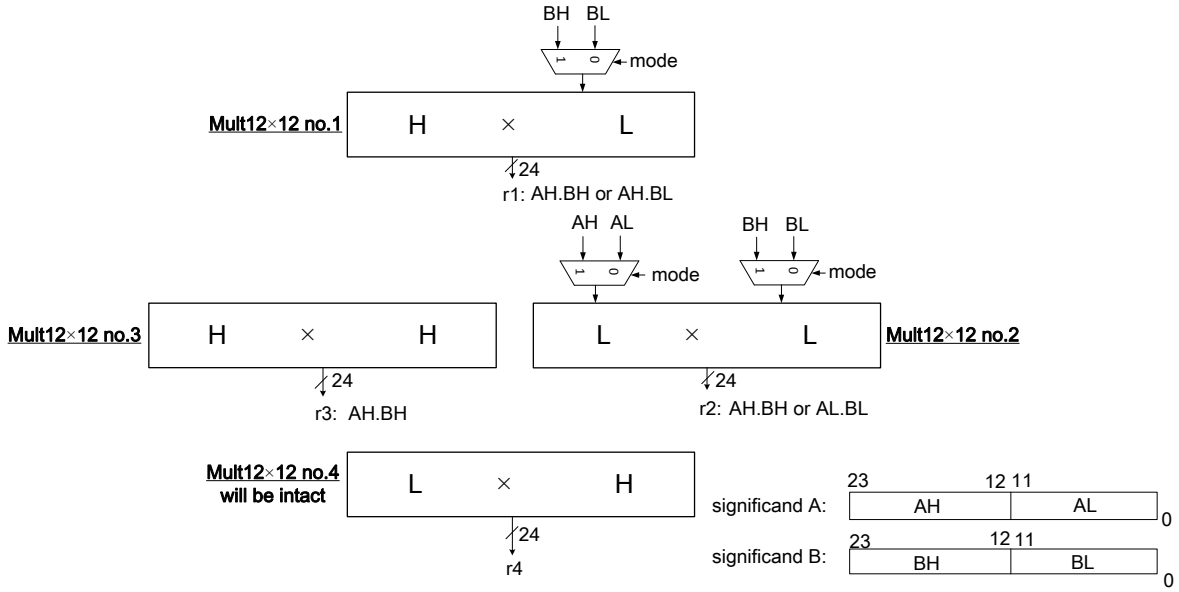
شکل ۹: جزئیات معماری درونی ضرب‌کننده ۱۲بیتی ماجول $H \times L$ در شکل ۸.

دقت کاهش‌یافته nبیتی $(n=12+t$ و $t \le 6)$ را انجام دهد تا سه خروجی $2n$بیتی $P1$، $P2$ و $P3$ تولید شود. این سه خروجی بعد از عبور از بلوک‌های هنجارساز و گردکننده توسط رأی‌دهنده با هم مقایسه شده و خطای احتمالی تصحیح یا پوشانده می‌شود. در شکل ۱۰ برای رقم‌های نقلی تولیدشده توسط بلوک‌های هنجارساز و گردکننده نیز رأی‌دهنده یک‌بیتی گذاشته شده تا به طور جداگانه تصحیح خطا در رقم نقلی را انجام دهد. در ضرب‌کننده جدید، هر مانتیس ۲۳بیتی به یک مانتیس kبیتی جدید $(k=11+t)$ تبدیل می‌شود که به همراه یک بیت ضمنی، یک significand nبیتی $(n=k+1)$ را تشکیل می‌دهد. همان طور که در شکل ۱۰ مشاهده می‌شود، بلوک هنجارساز و گردکننده سه‌تایی شده است تا قابلیت تصحیح خطا در کل طرح افزایش یابد. همچنین همانند شکل ۶، بلوک سازنده نتیجه استاندارد، نتیجه خارج‌شده از رأی‌دهنده kبیتی را با گذاشتن تعداد مورد نیاز صفر در سمت راست آن و ترکیب با توان و علامت عدد نهایی به قالب نمایش دقت ساده ۳۲بیتی تبدیل می‌کند. نحوه عملکرد ساختار شکل ۱۰ در مد عادی مشابه ساختار شکل ۶ است.

اگرچه در مد تحمل‌پذیر اشکال از ضرب‌کننده ۲۴بیتی اولیه برای تولید سه حاصل‌ضرب با طول مساوی استفاده می‌شود، اما با توجه به اندازه ضرب‌کننده‌های لازم، ممکن است به ضرب‌کننده‌های اضافه کوچک‌تری نیز احتیاج باشد. بنابراین اگر در ضرب‌کننده با دقت کاهش‌یافته nبیتی $(n=12+t)$ t برابر صفر باشد، تنظیم و استفاده از ضرب‌کننده‌های ۱۲بیتی درون ضرب‌کننده ۲۴بیتی به راحتی و مطابق شکل ۱۱ انجام خواهد شد. با توجه به نیاز به سه ضرب‌کننده ۱۲بیتی، یکی از ضرب‌کننده‌های درونی (شماره ۴ در شکل ۱۱) بلااستفاده خواهد ماند. واضح است که برای nهای بزرگ‌تر از ۱۲ $(t>0)$، ساختاری مشابه شکل ۹ اما برای سه ضرب‌کننده nبیتی مورد نیاز است که با توجه به نیاز به ضرب‌کننده‌های اضافه جدید، سربار کلی افزایش واضحی خواهد داشت.



شکل ۱۰: ساختار ضرب‌کننده پیشنهادی ممیز-شناور در دو مد کاری عادی و تحمل‌پذیر اشکال با قابلیت تصحیح خطا.



شکل ۱۱: ساختار ضرب‌کننده اصلی ۲۴بیتی پیشنهادی برای استفاده در مدهای کاری عادی و تحمل‌پذیر اشکال با قابلیت تصحیح خطا برای significand ۱۲بیتی.

۵- نتایج پیاده‌سازی و ارزیابی

طرح‌های پیشنهادی با تمام افزونگی‌های اعمال‌شده به همراه طرح پایه با استفاده از زبان توصیف سخت‌افزار VHDL پیاده‌سازی و شبیه‌سازی شده و سپس سنتز آنها توسط ابزار synopsys انجام شده است تا مساحت و توان مورد نیاز آنها به دست آید. همچنین مساحت و توان مصرفی طرح‌های پیشنهادی به ازای عرض بیت‌های مختلف مانتیس (k) اندازه‌گیری شده است. نتایج پیاده‌سازی با استفاده از کتابخانه استاندارد CMOS ۶۵ nm LPLVT STMicroelectronics (با ولتاژ تغذیه ۱٫۲ V و دمای ۲۵°C) در جدول‌های ۲ و ۳ به ترتیب برای ساختارهای با قابلیت تشخیص خطا و تصحیح خطا آورده شده است. در این دو جدول، تأخیر، مساحت، توان و سربارهای مربوط به آنها در ضرب‌کننده‌های پیشنهادی به ازای عرض بیت‌های مختلف مانتیس، در مقایسه با ضرب‌کننده پایه نشان داده شده است. با توجه به این جدول‌ها می‌توان نتیجه گرفت که انتخاب عرض مانتیس برابر با ۱۱ برای هر دو طرح پیشنهادی، سربار ناچیزی را خصوصاً برای تأخیر و مساحت مورد نیاز تحمیل می‌کند در حالی که مطابق [۶] برای این اندازه مانتیس بسیاری از برنامه‌های کاربردی دارای دقت کافی در خروجی خود خواهند بود. همچنین می‌توان برای ساختار با قابلیت تشخیص خطا عرض مانتیس برابر با ۱۲ یا ۱۳ را نیز مورد استفاده قرار داد زیرا به سربارهای متعارفی احتیاج دارند.

با توجه به نتایج ارائه‌شده در این مقاله و [۶] می‌توان دریافت که برنامه‌هایی که به نحوی با پردازش صوت، تصویر یا ویدئو در ارتباط هستند، در صورت استفاده از مانتیس حداقل ۱۱بیتی در ضرب‌کننده

جدول ۲: نتایج پیاده‌سازی طرح ضرب‌کننده ممیز- شناور پیشنهادی حاوی دو مد عادی و تحمل‌پذیر اشکال با قابلیت تشخیص خطا در مقایسه با طرح پایه.

اندازه مانتیس بر حسب بیت در مد تحمل‌پذیر اشکال	تأخیر (نانوثانیه)	سربار تأخیر	مساحت (میکرومتر مربع)	سربار مساحت	توان (میلی‌وات)	سربار توان	احتمال تشخیص خطا
$k=۲۳$ (طرح پایه)	۸٫۷۰	–	۱۱۲۹۰٫۲	–	۵٫۶۲	–	۰
$k=۱۱$	۹٫۰۴	۳٫۹٪	۱۲۵۲۴٫۷	۱۰٫۹٪	۶٫۴۹	۱۵٫۵٪	۸۱٫۵٪
$k=۱۲$	۸٫۸۷	۲٫۰٪	۱۳۶۴۹٫۵	۲۰٫۹٪	۶٫۵۹	۱۷٫۳٪	۸۳٫۱٪
$k=۱۳$	۸٫۸۷	۲٫۰٪	۱۴۰۸۲٫۶	۲۴٫۷٪	۷٫۰۹	۲۶٫۲٪	۸۳٫۳٪
$k=۱۵$	۸٫۸۷	۲٫۰٪	۱۴۶۹۴٫۷	۳۰٫۲٪	۷٫۹۰	۴۰٫۶٪	۸۳٫۷٪

جدول ۳: نتایج پیاده‌سازی طرح ضرب‌کننده ممیز- شناور پیشنهادی حاوی دو مد عادی و تحمل‌پذیر اشکال با قابلیت تصحیح خطا در مقایسه با طرح پایه.

اندازه مانتیس بر حسب بیت در مد تحمل‌پذیر اشکال	تأخیر (نانوثانیه)	سربار تأخیر	مساحت (میکرومتر مربع)	سربار مساحت	توان (میلی‌وات)	سربار توان	احتمال تصحیح خطا
$k=۲۳$ (طرح پایه)	۸٫۷۰	–	۱۱۲۹۰٫۲	–	۵٫۶۲	–	۰
$k=۱۱$	۹٫۰۰	۳٫۴٪	۱۲۶۹۸٫۴	۱۲٫۵٪	۶٫۷۰	۱۹٫۲٪	۸۱٫۶٪
$k=۱۲$	۸٫۸۸	۲٫۱٪	۱۶۳۶۲٫۳	۴۵٫۰٪	۷٫۷۷	۳۸٫۳٪	۸۵٫۵٪
$k=۱۳$	۸٫۸۸	۲٫۱٪	۱۷۲۱۹٫۳	۵۲٫۵٪	۸٫۳۹	۴۹٫۳٪	۸۶٫۱٪
$k=۱۵$	۸٫۸۸	۲٫۱٪	۱۹۱۴۹٫۰	۶۹٫۶٪	۹٫۸۱	۷۴٫۶٪	۸۷٫۲٪

جدول ۴: ضرب‌کننده‌های ممیز- شناور پیشنهادی در مقایسه با روش‌های قبلی.

طرح به همراه اندازه مانتیس (دقت نتایج خروجی)	توانایی کار در دو مد عادی و تحمل‌پذیر اشکال	قابلیت تحمل‌پذیری اشکال	سربار مساحت	سربار توان	حداکثرخطای نسبی خروجی
DMR با مانتیس ۲۳بیتی	خیر	تشخیص	بیش از ٪۱۰۰	بیش از ٪۱۰۰	۰
TMR با مانتیس ۲۳بیتی	خیر	تصحیح	بیش از ٪۲۰۰	بیش از ٪۲۰۰	۰
طرح‌های ارائه‌شده در [۱۲] با مانتیس ۲۳بیتی	خیر	تشخیص	از ٪۳۰ تا ٪۵۰	نامعلوم	۰
طرح ارائه‌شده در [۲۶] با مانتیس ۷بیتی	خیر	تشخیص	٪۱۷٫۸	٪۲۸	۰٫۰۰۷۸۱
طرح ارائه‌شده در [۲۷] با مانتیس ۱۵بیتی	خیر	تصحیح	٪۲۸٫۹	٪۱۲٫۶	۰٫۰۰۰۰۳
معماری پیشنهادی با مانتیس ۱۱بیتی (شکل ۶)	بله	تشخیص	٪۱۰٫۹	٪۱۵٫۵	۰٫۰۰۰۴۹
معماری پیشنهادی با مانتیس ۱۱بیتی (شکل ۱۰)	بله	تصحیح	٪۱۲٫۵	٪۱۹٫۲	۰٫۰۰۰۴۹

ممیز- شناور از کیفیت مناسبی برخوردار خواهند بود. نمونه‌های مناسبی از این برنامه‌ها که در [۶] مورد ارزیابی قرار گرفته‌اند عبارتند از برنامه شناسایی سیگنال صحبت، برنامه طبقه‌بندی کننده اثر انگشت در سطح الگو و الگوریتم‌های دوبعدی DCT[1] و معکوس DCT که دقت خروجی همه آنها بیش از ٪۹۰ بوده است.

با توجه به جدول‌های ۲ و ۳، در طرح‌های پیشنهادی برای تمام عرض مانتیس‌های ذکرشده، احتمال تشخیص یا تصحیح خطا از ٪۸۱ تا ٪۸۷ تغییر می‌کند. در اینجا احتمال تشخیص یا تصحیح خطا در کل طرح با محاسبه نسبت مساحت بلوک‌های دارای ویژگی تشخیص یا تصحیح خطا به کل مساحت مصرفی معماری پیشنهادی به دست آمده است. در واقع با توجه به این که مشخص است در چه بلوک‌هایی تشخیص یا تصحیح خطا به طور کامل انجام می‌شود و مساحت این بلوک‌ها نیز در سنتز نهایی به دست آمده است، این محاسبات انجام شده و نیازی به تزریق اشکال[2] در معماری‌های پیشنهادی نیست. در طرح‌های پیشنهادی، مساحت کل ضرب‌کننده اصلی و اجزای آن به همراه بخش‌های تکرارشده دارای قابلیت اطمینان در برابر اشکال هستند.

با توجه به ماهیت معماری‌های پیشنهادی در این مقاله که قابلیت عمل در دو مد کاری عادی و تحمل‌پذیر اشکال (از نوع تشخیص خطا یا تصحیح خطا) را با توجه به شرایط محیط کاری دارا هستند، طرح‌های پیشنهادی از لحاظ ساختاری کاری جدید محسوب شده و بنابراین در پژوهش‌های پیشین روش مشابهی برای مقایسه کامل و دقیق وجود ندارد. با این حال در میان کارهای گذشته، می‌توان روش‌های ارائه‌شده برای تشخیص خطا در [۱۲] و [۲۶] و روش ارائه‌شده برای تصحیح خطا در [۲۷] را به طور محدود با معماری‌های پیشنهادی در این مقاله مقایسه نمود. به عنوان نمونه، معماری ارائه‌شده در [۲۶] که سنتز آن با ابزار synopsys انجام شده است، برای تشخیص خطای ضرب‌کننده ممیز- شناور با مانتیس برابر با ۷ بیت به ٪۱۷٫۸ سربار مساحت احتیاج دارد، در صورتی که معماری پیشنهادی در این مقاله با قابلیت تشخیص خطا با مانتیس برابر با ۱۱ بیت که بسیار بهتر از ۷ بیت است، تنها به ٪۱۰٫۹ سربار احتیاج دارد. علاوه بر این، معماری پیشنهادی دارای قابلیت تشخیص خطا با مانتیس برابر با ۱۱ بیت دارای ٪۱۵٫۵ سربار توان مصرفی است در حالی که روش ارائه‌شده در [۲۶] با مانتیسی کوچک‌تر و برابر با ۷ بیت، ٪۲۸ سربار توان را تحمیل می‌کند. نتایج این مقایسه به همراه نتایج مقایسه با [۱۲] و [۲۷] و دو روش پایه در جدول ۴ ارائه شده است. با توجه به این جدول، هر دو معماری پیشنهادی با قابلیت عمل در دو مد کاری عادی و تحمل‌پذیر اشکال (هم از نوع تشخیص خطا و هم از نوع تصحیح خطا) با مانتیس ۱۱بیتی، دارای وضعیت مناسب‌تری نسبت به روش‌های قبلی هستند. باید توجه داشت که روش پایه دونسخه‌سازی

1. Discrete Cosine Transform
2. Fault Injection

و مقایسه که نام دیگر آن DMR[1] است، برای تشخیص خطا به بیش از ٪۱۰۰ سربار مساحت و توان مصرفی احتیاج دارد که مناسب‌بودن معماری‌های پیشنهادی را نشان می‌دهد. همچنین معماری پیشنهادی دوم در این مقاله که در مد تحمل‌پذیر اشکال دارای قابلیت تصحیح خطا است و با توجه به اندازه مانتیس‌های مشخص‌شده در جدول ۳ سربارهایی را تحمیل می‌کند، در مقایسه با روش TMR که حداقل به ٪۲۰۰ سربار مساحت و توان مصرفی احتیاج دارد به سربار بسیار کمتری نیاز دارد. علاوه بر این معماری پیشنهادی نسبت به روش ارائه‌شده در [۲۷] که تنها در یک مد کاری عمل می‌کند و قادر به تولید نتایج با دقت کامل نیست، در وضعیت مناسبی قرار دارد.

از آنجایی که طرح‌های تحمل‌پذیر اشکال پیشنهادی برای واحدهای ممیز- شناور در این پژوهش بر اساس کاهش دقت خروجی محاسبات است بنابراین برخی خطاهای محاسباتی در خروجی تولید خواهد شد که در بسیاری از برنامه‌های کاربردی به ویژه برنامه‌هایی که با حواس انسان سروکار دارند، قابل قبول است. تخمین حداکثر خطای نسبی برای یک خروجی ممیز- شناور که در جدول ۴ نیز انجام شده ساده است و طبق رابطه زیر به دست می‌آید

$$Maximum\ Relative\ Error = ۲^{-l} \qquad (۲)$$

در (۲)، l عرض بیت مانتیس مورد استفاده در معماری‌های مختلف است. بنابراین در طرح‌های با مانتیس کوچک‌تر که سربارهای کمتری تحمیل می‌شود مقدار خطای نسبی بیشتر خواهد بود.

٦- نتیجه‌گیری

در این مقاله، دو معماری برای ضرب‌کننده ممیز- شناور با قابلیت کار در دو مد کاری عادی و تحمل‌پذیر اشکال پیشنهاد گردید. اگر محیطی که در آن از ضرب‌کننده استفاده می‌شود محیطی پرنویز باشد، استفاده از مد کاری تحمل‌پذیر اشکال بسیار مفید خواهد بود. اما خروجی عملیات با توجه به هزینه‌های مورد نیاز می‌تواند دارای یکی از قابلیت‌های تشخیص یا تصحیح خطا باشد. همچنین نشان داده شد که با کاهش دقت محاسبات در محدوده قابل قبول برای بسیاری از برنامه‌های کاربردی، می‌توان به قابلیت تشخیص یا تصحیح خطا دست یافت. معماری‌های پیشنهادی پس از حذف بیت‌های کم‌ارزش‌تر عملوندهای ورودی، با استفاده از تجزیه و طراحی ماجولار ضرب‌کننده ۲۴بیتی اصلی درون طرح ضرب‌کننده ممیز- شناور به منظور انجام محاسبات افزونه، به قابلیت‌های اشاره‌شده می‌رسند. نتایج پیاده‌سازی نشان می‌دهد که با سربارهای معقول می‌توان به ساختارهایی رسید که دارای درجه بالایی از قابلیت اطمینان در برابر خطاهای منفرد از نوع تشخیص یا تصحیح باشند.

مراجع

[1] C. H. Yu, K. Chung, D. Kim, and L. S. Kim, "An energy-efficient mobile vertex processor with multithread expanded VLIW architecture and vertex caches," *IEEE J. of Solid-State Circuits*, vol. 42, no. 10, pp. 2257-2269, Oct. 2007.

[2] B. Nicolescu, N. Lgnat, Y. Savaria, and G. Nicolescu, "Analysis of real-time systems sensitivity to transient faults using MicroC kernel," *IEEE Trans. Nuclear Science*, vol. 53, no. 4, pp. 1902-1909, Aug. 2006.

[3] A. B. Kahng and S. Kang, "Accuracy-configurable adder for approximate arithmetic designs," in *Proc. 49th Annual Design Automation Conf.*, *DAC'12,* pp. 820-825, San Francisco, CA, USA, 3-7 Jun. 2012.

[4] S. Venkataramani, S. T. Chakradhar, K. Roy, and A. Raghunathan, "Computing approximately, and efficiently," in *Proc. Design, Automation & Test in Europe Conf., DATE'15*, pp. 748-751, Grenoble, France, 9-13 Mar. 2015.

[5] IEEE Computer Society (2008), IEEE standard for floating-point arithmetic, IEEE Standard 754-2008.

[6] J. Ying, F. Tong, D. Nagle, and R. A. Rutenbar, "Reducing power by optimizing the necessary precision/range of floating-point arithmetic," *IEEE Trans. VLSI Systems*, vol. 8, no. 3, pp. 273-286, Jun. 2000.

[7] I. Z. Milovanovic, E. I. Milovanovic, M. K. Stojcev, and M. P. Bekakos, "Orthogonal fault-tolerant systolic arrays for matrix multiplication," *Microelectronics Reliability*, vol. 51, no. 3, pp. 711-725, Mar. 2011.

[8] M. Fazeli, A. Namazi, S. G. Miremadi, and A. Haghdoost, "Operand width aware hardware reuse: a low cost fault-tolerant approach to ALU design in embedded processors," *Microelectronics Reliability*, vol. 51, no. 12, pp. 2374-2387, Dec. 2011.

[9] P. Reviriego, S. Z. Can, C. Eryılmaz, J. A. Maestro, and O. Ergin, "Exploiting processor features to implement error detection in reduced precision matrix multiplications," *Microprocessors and Microsystems*, vol. 38, no. 6, pp. 581-584, Aug. 2014.

[10] A. Mukherjee and A. S. Dhar, "Real-time fault-tolerance with hot-standby topology for conditional sum adder," *Microelectronics Reliability*, vol. 55, no. 3-4, pp. 704-712, Feb./Mar. 2015.

[11] M. Valinataj, "Fault-tolerant carry look-ahead adder architectures robust to multiple simultaneous errors," *Microelectronics Reliability*, vol. 55, no. 12, pp. 2845-2857, Dec. 2015.

[12] D. Lipetz and E. Schwarz, "Self checking in current floating-point units," in *Proc. 20th IEEE Symp. Computer Arithmetic*, pp. 73-76, Tubingen, Germany, 25-27 Jul. 2011.

[13] M. Maniatakos, Y. Makris, P. Kudva, and B. Fleischer, "Exponent monitoring for low-cost concurrent error detection in FPU control logic," in *Proc. 29th IEEE VLSI Test Symp.*, pp. 235-240, Dana Point, CA, USA, 1-5 May 2011.

[14] A. Gupta, *et al.*, "Low power probabilistic floating point multiplier design," in *Proc. IEEE Computer Society Annual Symp. VLSI, ISVLSI'11*, pp. 182-187, Chennai, India, 4-6 Jul. 2011.

[15] H. Zhang, W. Zhang, and J. Lach, "A low-power accuracy-configurable floating point multiplier," in *Proc. 32nd IEEE Intl. Conf. Computer Design, ICCD'14*, pp. 48-54, Seoul, South Korea, 19-22 Oct. 2014.

[16] S. Hashemi, R. Iris Bahar, and S. Reda., "DRUM: a dynamic range unbiased multiplier for approximate applications," in *Proc. IEEE Int. Conf. on Comp. Aided Design, ICCAD'15*, pp. 418-425, Austin, TX, USA, 2-6 Nov. 2015.

[17] S. Narayanamoorthy, H. Asghari Moghaddam, Z. Liu, T. Park, and N. Sung Kim, "Energy-efficient approximate multiplication for digital signal processing and classification applications," *IEEE Trans. on VLSI*, vol. 23, no. 6, pp. 1180-1184, Jun. 2015.

[18] V. Camus, J. Schlachter, C. Enz, M. Gautschi, and F. K. Gurkaynak, "Approximate 32-bit floating-point unit design with 53% power-area product reduction," in *Proc. 42nd European Solid-State Circuits Conf.*, *ESSCIRC'16,* pp. 465-468, Lausanne, Switzerland, 12-15 Sept. 2016.

[19] M. Imani, D. Peroni, and T. Rosing, "CFPU: configurable floating point multiplier for energy-efficient computing," in *Proc. 54th ACM/EDAC/IEEE Design Automation Conf., DAC'17*, 6 pp., Austin, TX, USA, 18-22 Jun. 2017.

[20] P. Yin, C. Wang, W. Liu, and F. Lombardi, "Design and performance evaluation of approximate floating-point multipliers," in *Proc. IEEE Computer Society Annual Symp. VLSI, ISVLSI'16*, pp. 296-301, Pittsburgh, PA, USA, 11-13 Jul. 2016.

[21] A. Sunny, B. K. Mathew, and P. B. Dhanusha, "Area efficient high speed approximate multiplier with carry predictor," *Procedia Technology*, vol. 24, pp. 1170-1177, 2016.

[22] M. Fathi and H. Nikmehr, "Improving accuracy, area and speed of approximate floating point multiplication using carry prediction," *J. of Information Systems and Telecommunication*, vol. 5, no. 2, pp. 120-127, Apr./Jun. 2017.

[23] C. Liu, J. Han, and F. Lombardi, "A low-power, high-performance approximate multiplier with configurable partial error recovery," in *Proc. Design, Automation & Test in Europe Conf., DATE'14*, 4 pp. , Dresden, Germany, 24-28 Mar. 2014.

[24] P. Yin, C. Wang, W. Liu, E. E. Swartzlander Jr., and F. Lombardi, "Designs of approximate floating-point multipliers with variable accuracy for error-tolerant applications," *J. of Signal Processing Systems*, vol. 90, no. 4, pp. 641-654, Apr. 2018.

1. Double Modular Redundancy

[25] P. J. Eibl, A. D. Cook, and D. J. Sorin, "Reduced precision checking for a floating point adder," in *Proc. 24th IEEE Int. Symp. Defect and Fault Tolerance in VLSI Systems*, pp. 145-152, Chicago, IL, USA, 7-9 Oct. 2009.

[26] K. Seetharam, L. C. T. Keh, R. Nathan, and D. J. Sorin, "Applying reduced precision arithmetic to detect errors in floating point multiplication," in *Proc. IEEE Pacific Rim Intl. Symp. Dependable Computing, PRDC'13*, pp. 232-235, Vancouver, BC, Canada, 2-4 Dec. 2013.

[27] M. Mohajer and M. Valinataj, "A novel reduced-precision fault-tolerant floating-point multiplier," *Int. J. of Modern Education and Computer Science*, vol. 9, no. 6, pp. 17-24, Jun. 2017.

[28] B. Parhami, *Computer Arithmetic, Algorithms and Hardware Designs*, 2nd Ed., Oxford University Press, 2009.

مریم مهاجر تحصیلات خود را در مقاطع کارشناسی و کارشناسی ارشد مهندسی کامپیوتر به‌ترتیب در سال‌های ۱۳۹۲ و ۱۳۹۵ در دانشگاه صنعتی نوشیروانی بابل به پایان رسانده است. زمینه‌های تحقیقاتی مورد علاقه ایشان عبارتند از حساب کامپیوتری، محاسبات قابل اطمینان و سیستم‌های تحمل‌پذیر اشکال.

مجتبی ولی‌نتاج تحصیلات خود را در مقاطع کارشناسی، کارشناسی ارشد و دکترای مهندسی کامپیوتر به ترتیب در سال‌های ۱۳۷۹، ۱۳۸۱ و ۱۳۸۹ هر سه در دانشگاه تهران به پایان رسانده است. نام‌برده در سال‌های ۱۳۹۰، ۱۳۹۱ و ۱۳۹۶ پروژه‌های مشترکی را با گروه‌های تحقیقاتی در دانشگاه تورکو و دانشگاه صنعتی تامپره در فنلاند به انجام رسانده است. ایشان از سال ۱۳۸۹ تاکنون به‌عنوان عضو هیأت علمی در دانشکده مهندسی برق و کامپیوتر دانشگاه صنعتی نوشیروانی بابل مشغول به فعالیت است. زمینه‌های پژوهشی مورد علاقه ایشان شامل موضوعاتی مانند سیستم‌های قابل اطمینان، حساب کامپیوتری، شبکه بر تراشه، منطق برگشت‌پذیر و سیستم‌های چندپردازنده‌ای است.