بهره‌برداری از کاهش دقت محاسبات در طراحی ضرب‌کننده ممیز-شناور با قابلیت کار در مدهای عادی و تحمل‌پذیر اشکال

مریم مهاجر[1]، مجتبی ولی‌نتاج[2]

[1]کارشناسی ارشد کامپیوتر، دانشکده مهندسی برق و کامپیوتر، دانشگاه صنعتی نوشیروانی بابل، maryam.mohajer@stu.nit.ac.ir

[2]استادیار دانشکده مهندسی برق و کامپیوتر، دانشگاه صنعتی نوشیروانی بابل، m.valinataj@nit.ac.ir

چکیده

ضرب‌کننده یکی از مهمترین واحدهای عملیاتی مورد استفاده در انواع پردازش سیگنال، تصویر، صوت و سیستم‌های چندرسانه‌ای است. از طرفی، مدارهای دیجیتالی مستعد تولید خروجی‌های نادرست به علت عوامل محیطی گوناگون هستند. در این مقاله، یک معماری جدید برای ضرب‌کننده ممیز-شناور ارائه می‌شود که می‌تواند در دو مد کاری عادی و تحمل‌پذیر اشکال با قابلیت تشخیص یا تصحیح خطا عمل کند. در مد تحمل‌پذیری اشکال، با کاهش دقت محاسبات، بیت‌های کم‌ارزش‌تر مقادیر ممیز-شناور حذف می‌گردند. در نتیجه، بخشی از مدار اولیه آزاد شده و برای فراهم‌کردن محاسبات افزونه به منظور تشخیص یا تصحیح خطا استفاده می‌شود. معماری پیشنهادی با کاهش دقت محاسبات، در مد کاری تحمل‌پذیر اشکال قابلیت اطمینان مناسبی در برابر انواع اشکال‌های دائمی و گذرا دارد. نتایج پیاده‌سازی نشان می‌دهد که حفظ ۱۳ بیت مانتیس در مد تحمل‌پذیری اشکال، برای دست‌یافتن به ضرب‌کننده با قابلیت تشخیص خطا و حفظ ۱۱ بیت مانتیس در این مد، برای دست‌یافتن به ضرب‌کننده با قابلیت تصحیح خطا، با سربار مساحت و توان قابل قبول، مناسب است.

واژه‌های کلیدی

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

۱- مقدمه

کاربردهای چندرسانه‌ای وابستگی زیادی به محاسبات ریاضی دارند [۱،۲]. این کاربردها که شامل انواع پردازش‌های سیگنال، ویدئو، تصویر و صوت هستند به محاسبات جمع و ضرب ممیز-شناور نیز وابسته‌اند. همچنین، برنامه‌های کاربردی علمی که نیازمند شبیه‌سازی انواع پدیده‌های فیزیکی هستند، از محاسبات ممیز-شناور بهره می‌برند. با توجه به اهمیت این نوع از محاسبات، بروز خطا در نتیجه عملیات می‌تواند مشکل‌ساز باشد. علاوه بر این، کاهش اندازه ترانزیستورها منجر به بروز مشکلات مرتبط با قابلیت اطمینان مدارهای دیجیتالی شده است [۳،۴]. قابلیت اطمینان و توان مصرفی کم دو هدف اصلی در طراحی سیستم‌های محاسباتی امروزی و در نتیجه واحدهای عملیاتی ممیز-شناور مانند ضرب‌کننده‌ها است

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 نیستند، اما می‌توانند محاسبات نادقیق را تحمل کنند و در نتیجه یک مقدار تخمینی برای خروجی آن‌ها کافی است. بنابراین، می‌توان این نوع عملیات ممیز-شناور را با دقت کاهش یافته[1] اجرا نمود. هدف از این مقاله، استفاده از این ویژگی و بهره‌برداری از واحدهای آزاد شده و واحدهای افزونه جدید برای انجام محاسبات افزونه به منظور دستیابی به قابلیت اطمینان به صورت تشخیص یا تصحیح خطا است. معماری ضرب‌کننده پیشنهادی مبتنی بر عملیات ممیز-شناور ۳۲ بیتی مطابق با فرمت نمایش دقت ساده در استاندارد IEEE-754 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 تحمل‌پذیری اشکال بر اساس محاسبات با دقت کاهش‌یافته است، برخی خطاهای محاسباتی در خروجی تولید می‌شود. بنابراین، با پذیرفتن درصدی ناچیز از خطای محاسباتی در خروجی، یک ضرب‌کننده ممیز-شناور قابل اطمینان در برابر

[1] Reduced Precision

انواع اشکال‌های دائمی و گذرا به‌دست می‌آید که این امر در بسیاری از برنامه‌های کاربردی چندرسانه‌ای رضایت‌بخش است، چون برخی از انحرافات در خروجی، به دلیل محدودیت حواس انسان به طور طبیعی پوشش داده می‌شوند.

۲- کارهای مرتبط

روش‌های سنتی تشخیص (و تصحیح) خطا/ اشکال مانند duplication، TMR[۲] و افزونگی زمانی [۵] که در پردازنده‌های ممیز-شناور استفاده شده‌اند، اغلب دارای سربار مساحت و توان زیادی هستند. در [۶] یک واحد محاسباتی ممیز-شناور تشخیص‌دهنده خطا که از کدهای مانده استفاده می‌کند، پیشنهاد شده است. در [۷] یک روش فقط برای حفاظت مدار کنترلی واحد ممیز-شناور با بررسی بخش نمای عدد ممیز-شناور ارائه شده است. این روش برای تشخیص خطاهای گذرا در مدار کنترلی یک واحد ممیز-شناور مناسب است. روش‌های قبلی مبتنی بر کاهش دقت محاسبات، با حذف بیت‌های کم‌ارزش‌تر، بیشتر روی کاهش توان مصرفی تمرکز داشته‌اند. نتایج ارائه شده در [۸] نشان می‌دهد که همه برنامه‌های کاربردی به دقت اولیه فراهم شده توسط سخت‌افزار ممیز-شناور نیاز ندارند و می‌توانند تعداد بیت‌های کمتری از مانتیس و نما را در سخت‌افزار مربوطه به‌کار گیرند. درنتیجه، توان مصرفی می‌تواند کاهش یابد. طبق نتایج ارائه شده در [۸]، با کاهش عرض بیت مانتیس از ۲۳ بیت به ۱۱ بیت، هیچ یک از برنامه‌های ممیز-شناور کاهش قابل‌توجهی در دقت نداشتند و حتی برخی از برنامه‌ها با کمتر از ۵ بیت مانتیس نیز قابل قبول بودند.

در [۹] یک روش تشخیص خطا برای جمع‌کننده ممیز-شناور ارائه شده است که از یک جمع وارسی‌کننده[۳] 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 یابد. محاسبه کامل به موازات محاسبه افزونه که در آن فقط بیت‌های پرارزش عملوندها دریافت می‌شود، انجام می‌گردد. در مرحله آخر مداری نتایج این دو واحد را با هم مقایسه کرده و وقوع خطا را تشخیص می‌دهد. در [۱۰] تکنیک بررسی مبتنی بر کاهش دقت (RPC)[۴] به ضرب ممیز-شناور اعمال شده تا خطاها را تشخیص دهد. با توجه به این طرح، تشخیص خطاها در ضرب ممیز-شناور با هزینه معقول امکان‌پذیر است.

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 دارند، وجود ندارد.

۳- مبانی

۳-۱- استاندارد IEEE برای اعداد ممیز-شناور

نمایش ممیز-شناور محدوده‌ی دینامیکی وسیع‌تر و دقت بالاتری را در مقایسه با نمایش ممیز-ثابت فراهم می‌کند، اما مساحت و مصرف توان بیشتری را تحمیل می‌کند. نمایش معمول اعداد ممیز-شناور مطابق استاندارد IEEE-754 در دو فرمت نمایش دقت ساده[۵] و دقت مضاعف[۶] است. فرمت نمایش دقت ساده ۳۲ بیتی است که شامل ۱ بیت علامت، ۸ بیت نما و ۲۳ بیت مانتیس است. در این مقاله، تمرکز روی فرمت نمایش دقت ساده است. شکل ۱ نمایش باینری فرمت دقت ساده را نشان می‌دهد.

$$-1^{Sign} \times 1.\text{Mantissa} \times 2^{Exponent-127}$$

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

شکل ۱: نمایش باینری فرمت دقت ساده

بخش مانتیس یک عدد ممیز-شناور همیشه کوچک‌تر از یک است، اما همواره یک بیت مخفی '1' در سمت چپ نقطه اعشار در نظر گرفته می‌شود و significand را تشکیل می‌دهد. بنابراین، significand یک عدد نرمال شده در محدوده [1,2) است. بیت علامت، علامت significand را نشان می‌دهد، در حالی‌که علامت نما به طور ضمنی در درون آن قرار دارد. مقادیر ممیز-شناور به ازای مقادیر مختلف نما و مانتیس در جدول ۱ نشان داده شده است.

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

E = 0	Zero
0< E < 255	Normal numbers
E = 255 , M = 0	Infinite
E = 255 , M /= 0	Not-a-Number

۳-۲- ضرب‌کننده ممیز-شناور با دقت ساده

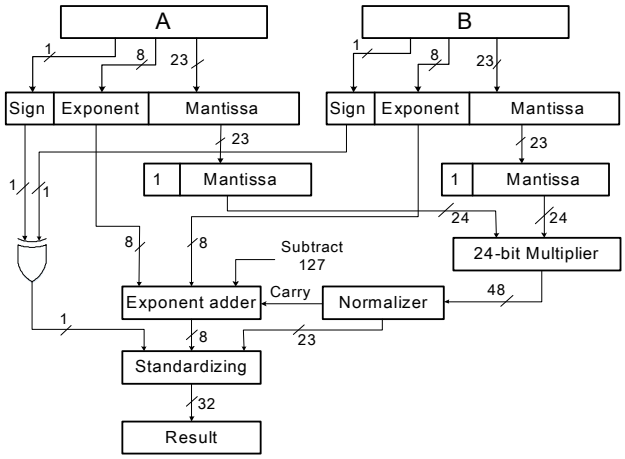
ضرب‌کننده ممیز-شناور ۳۲ بیتی عمل ضرب دو عملوند ورودی با دقت ساده را انجام داده و حاصل‌ضرب را نیز در دقت ساده تولید می‌کند. الگوریتم ضرب دو عدد ممیز-شناور طبق ۶ مرحله زیر انجام می‌شود:

مرحله ۱: ضرب مانتیس‌ها (ضرب‌کننده ۲۴ بیتی)
مرحله ۲: نرمال‌سازی مانتیس نتیجه
مرحله ۳: جمع نماها
مرحله ۴: محاسبه علامت نتیجه
مرحله ۵: بررسی وقوع سرریز (یا زیرریز)
مرحله ۶: استانداردسازی

شکل ۲ طرح کلی ضرب‌کننده ممیز-شناور ۳۲ بیتی را نشان می‌دهد. با توجه به این که در این طرح، بلوک ضرب‌کننده ۲۴ بیتی است، خروجی این ضرب‌کننده عددی ۴۸ بیتی است؛ اما طبق استاندارد برای فرمت با دقت ساده، تنها به ۲۳ بیت از آن برای مانتیس نتیجه نیاز است، بنابراین

[2] Triple Modular Redundancy
[3] Checker Adder
[4] Reduced Precision Checking
[5] Single-Precision
[6] Double-Precision

خروجی ۴۸ بیتی ضرب‌کننده از یک نرمال‌کننده عبور کرده و تغییر احتمالی در مقدار نما را نیز باعث می‌گردد.



شکل ۲: ضرب‌کننده ممیز-شناور ۳۲ بیتی پایه

۴- طرح‌های پیشنهادی

در این مقاله از روش کاهش دقت محاسبات از طریق حذف بیت‌های کم‌ارزش‌تر استفاده می‌شود تا با توجه به سخت‌افزار کاهش یافته بتوان یک سری افزونگی را برای دستیابی به قابلیت اطمینان از نوع تشخیص/تصحیح خطا در مد کاری تحمل‌پذیری اشکال اعمال نمود.

۳-۱- معماری ضرب‌کننده ممیز-شناور تحمل‌پذیر اشکال با قابلیت تشخیص خطا

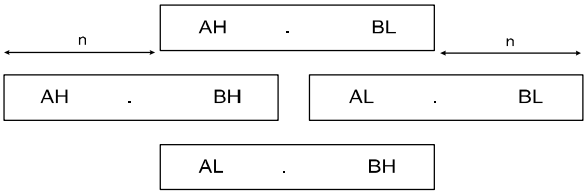
ضرب‌کننده استفاده شده در معماری پیشنهادی، یک ضرب‌کننده آرایه‌ای ماجولار است. این نوع ضرب‌کننده بر اساس جمع حاصل‌ضرب‌های پاره‌ای با ساختار بازگشتی است. اگر $^\circ$ تابع الحاق را نشان دهد، در یک ضرب‌کننده آرایه‌ای ماجولار $2n\times 2n$، هر عملوند به دو نیمه مرتبه بالا و مرتبه پایین تقسیم می‌شود. برای مثال، عملوندها با $2n$ بیت به صورت $A=A_H{}^\circ A_L$ در نظر گرفته می‌شوند. هر دو بخش A_H و A_L، n بیتی هستند. حاصل‌ضرب عملوندهای A و B در شکل ۳ نشان داده شده است. تقسیم کردن هر عملوند به دو نیمه، چهار حاصل‌ضرب پاره‌ای را تولید می‌کند. برای جمع این چهار حاصل‌ضرب پاره‌ای، ابتدا باید آن‌ها را با هم هم‌تراز کرد. به منظور روشن شدن هم‌ترازی حاصل‌ضرب‌های پاره‌ای، ضرب دو عملوند $2n$ بیتی A و B بر اساس ضرب آرایه‌ای ماجولار در شکل ۴ نشان داده شده است. جمع بیش از دو عملوند می‌تواند با روش CSA[۷] انجام شود. شکل ۵ سازماندهی مناسب این چهار حاصل‌ضرب پاره‌ای را برای کاهش تعداد سطوح در استفاده از CSA نشان می‌دهد. در نتیجه، معماری ماجولار ضرب‌کننده با ضرب دو مرحله‌ای مشابه شکل ۵ خواهد بود. در مرحله اول، زیرضرب‌کننده‌ها (ضرب‌کننده‌های n بیتی) حاصل‌ضرب‌های پاره‌ای را تولید می‌کنند. سپس، در مرحله دوم حاصل‌ضرب‌های پاره‌ای با استفاده از معماری جمع‌کننده چهار ورودی با طرح نشان داده شده در شکل ۵ با هم جمع می‌شوند.

$$P = A\times B = A_H{}^\circ A_L \times B_H{}^\circ B_L$$
$$= A_H\times B_H + A_L\times B_H + A_H\times B_L + A_L\times B_L$$

شکل ۳: ضرب عملوندهای A و B با تقسیم به حاصل‌ضرب‌های پاره‌ای

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

شکل ۴: ضرب A×B براساس ضرب آرایه‌ای ماجولار



شکل ۵: سازماندهی مناسب حاصل‌ضرب‌های پاره‌ای n بیتی برای انجام ضرب ۲n بیتی

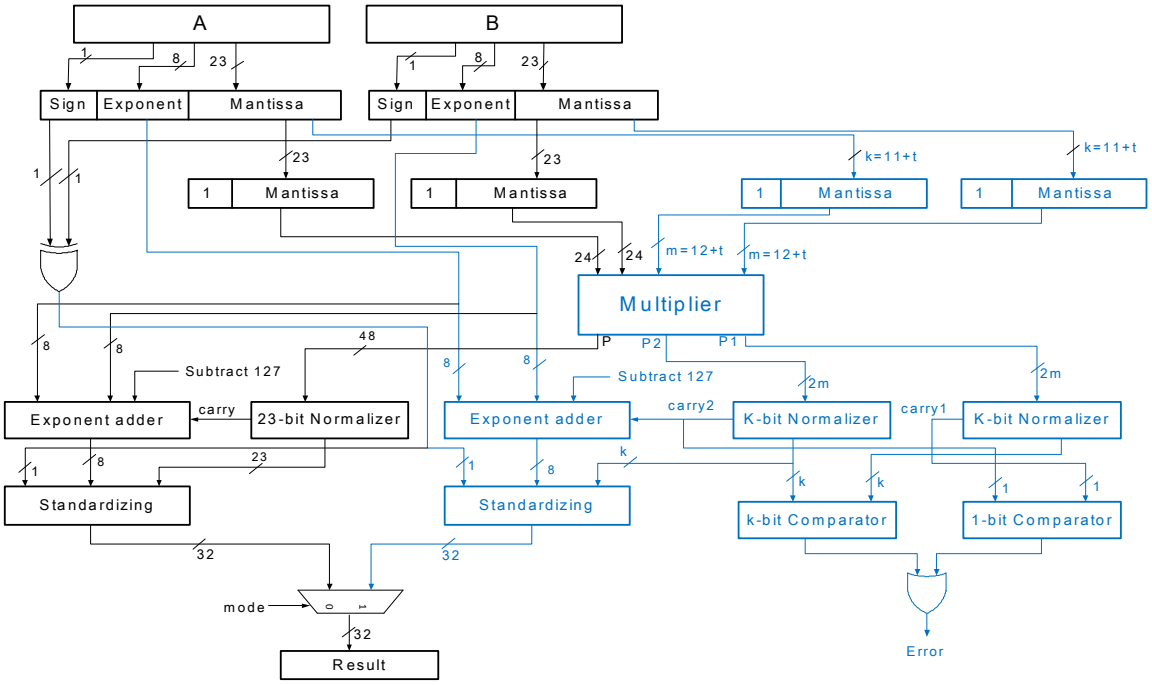
همان‌طور که در شکل ۲ نشان داده شده است، ضرب‌کننده ممیز-شناور توصیف‌شده در بخش ۳-۲ شامل یک ضرب‌کننده ۲۴ بیتی برای ضرب دو significand عملوندهای ورودی است. طبق آن‌چه که در بالا گفته شد، این ضرب‌کننده‌ی ۲۴ بیتی می‌تواند با استفاده از چهار ضرب‌کننده ۱۲ بیتی مطابق شکل ۵ پیاده‌سازی شود.

در این بخش یک ضرب‌کننده ممیز-شناور ارائه می‌شود که می‌تواند در دو مد کاری عادی (با دقت کامل و شامل یک ضرب‌کننده ۲۴ بیتی) و تحمل‌پذیری اشکال (با دقت کاهش‌یافته و قابلیت تشخیص خطا شامل دو ضرب‌کننده m بیتی) کار کند. در مد تحمل‌پذیری اشکال با توجه به حذف تعدادی از بیت‌های کم‌ارزش، برخی از زیرضرب‌کننده‌ها آزاد می‌شوند که می‌توانند برای انجام محاسبات افزونه با هدف تشخیص یا تصحیح خطا به‌کار روند. بنابراین، برای دستیابی به قابلیت تشخیص خطا در ضرب‌کننده ممیز-شناور در مد تحمل‌پذیری خرابی، به جای یک ضرب‌کننده ۲۴ بیتی از دو ضرب‌کننده با دقت کاهش یافته m بیتی ($m = ۱۲+t$ و $t \le ۶$) طبق مفهوم روش دوتایی‌کردن و مقایسه استفاده می‌شود. بنابراین، هر مانتیس ۲۳ بیتی از عملوندهای ورودی به یک مانتیس k بیتی جدید ($k = ۱۱+t$) تبدیل می‌شود (یعنی خذف k-۲۳ بیت کم‌ارزش‌تر مانتیس) این مانتیس به همراه یک بیت ضمنی، یک significand m بیتی ($m = k+۱$) را تشکیل می‌دهد. در این روش به جای یک ضرب‌کننده ۲۴ بیتی با خروجی ۴۸ بیتی، دو ضرب‌کننده با دقت کاهش‌یافته m بیتی استفاده می‌شود تا دو

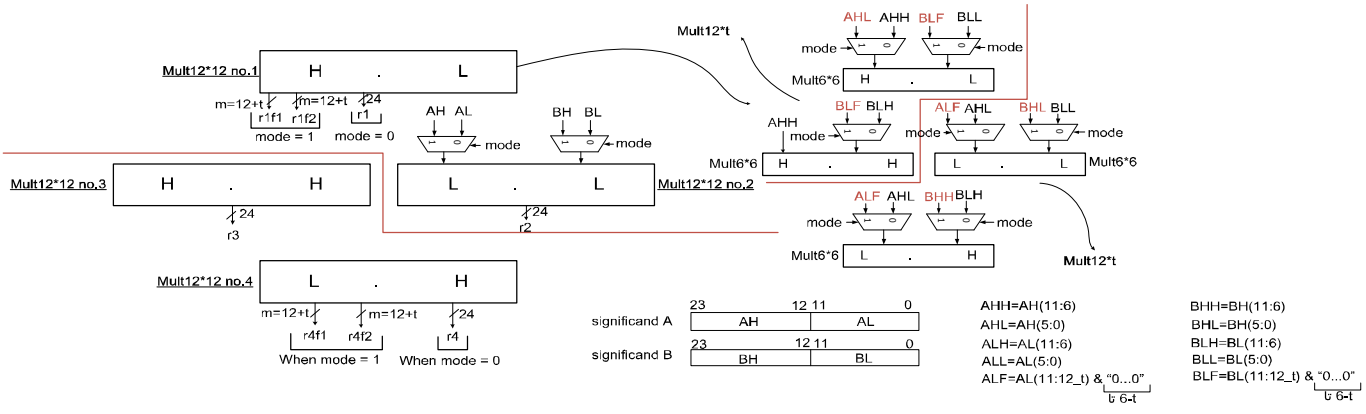
[7] Carry Save Adder

خروجی ۲m بیتی تولید شده بعد از عبور از نرمال‌کننده توسط مقایسه‌کننده با هم مقایسه شوند تا خطای احتمالی تشخیص داده شود.

مدار کلی ضرب‌کننده ممیز-شناور پیشنهادی در دو مد کاری عادی و تحمل‌پذیر اشکال با قابلیت تشخیص خطا در شکل ۶ الف و معماری پیشنهادی برای تجزیه ضرب‌کننده اولیه به ضرب‌کننده‌های کوچک‌تر برای استفاده در دو مد کاری در شکل ۶ ب نشان داده شده است. ایده اصلی معماری پیشنهادی این است که ضرب‌کننده ۲۴ بیتی اصلی به چهار ضرب‌کننده با طول یکسان ۱۲ بیتی تقسیم شود تا در مد تحمل‌پذیری اشکال بتوان با کاهش دقت محاسبات و استفاده از دو ضرب‌کننده با دقت کاهش‌یافته، خطاهای احتمالی را تشخیص داد. زمانی که سیگنال ورودی mode برابر '1' باشد که به معنای مد تحمل‌پذیری اشکال با قابلیت تشخیص خطا است، از چهار زیرضرب‌کننده ۱۲ بیتی برای انجام دو ضرب $m = ۱۲+t$ بیتی استفاده می‌شود. در شکل ۷ نحوه انجام ضرب $m = ۱۲+t$ بیتی نشان داده شده است. واضح است که می‌توان با استفاده از دو زیرضرب‌کننده‌ی ۱۲ بیتی در ضرب‌کننده ۲۴ بیتی اولیه، حاصل‌ضرب پاره‌ای ۱۲×۱۲ و دو حاصل‌ضرب پاره‌ای $۱۲\times t$ مربوط به ضرب $۱۲+t$ بیتی نشان داده شده در شکل ۷ را تولید کرد. برای این کار، با توجه به شکل ۶ ب، ضرب‌کننده ۱۲ بیتی شماره ۲، حاصل‌ضرب ۱۲ بیتی $A_H\times B_H$ را تولید می‌کند ($r_۲$). ضرب $۱۲\times t$ بیتی $A_H\times B_{LF}$ و $B_H\times A_{LF}$، به دو ضرب‌کننده ۶ بیتی نیاز دارد. بنابراین، ضرب‌کننده ۱۲ بیتی شماره ۱ به چهار زیرضرب‌کننده ۶ بیتی تقسیم می‌شود و از این زیرضرب‌کننده‌های ۶ بیتی برای تولید دو حاصل‌ضرب $۱۲\times t$ بیتی استفاده می‌شود. اما یک ضرب‌کننده $t\times t$ بیتی برای تولید حاصل‌ضرب پاره‌ای $A_{LF}\times B_{LF}$ به عنوان سربار برای انجام یک ضرب $۱۲+t$ بیتی اضافه می‌شود. در نتیجه، همه حاصل‌ضرب‌های



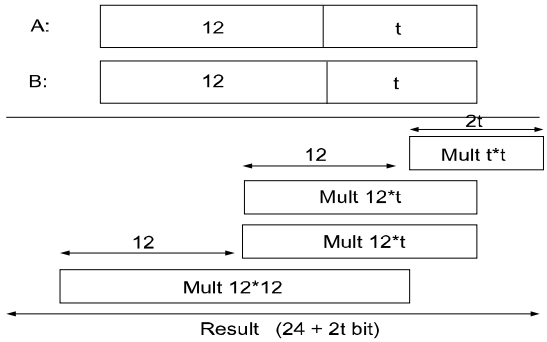
الف.



ب.

شکل ۶: الف) طرح ضرب‌کننده پیشنهادی ممیز-شناور در دو مد کاری عادی و تحمل‌پذیر اشکال با قابلیت تشخیص خطا. ب) بخش ضرب‌کننده طرح پیشنهادی

پاره‌ای برای انجام یک ضرب m بیتی تولید می‌شوند که طبق شکل ۷ با روش CSA با هم جمع شده و یک حاصل ۲m بیتی را تولید می‌کنند. زیرضرب‌کننده‌های ۱۲ بیتی شماره ۳ و ۴ نیز به همین ترتیب ضرب m بیتی دوم را انجام می‌دهند و ضرب m بیتی دوم به موازات ضرب m بیتی اول انجام می‌شود و دو حاصل‌ضرب ۲m بیتی تولید می‌شود. سپس، این دو حاصل‌ضرب ۲m بیتی از نرمال‌کننده‌های مربوطه عبور کرده و حاصل را در k بیت نرمال می‌کنند. اگر نتایج دو ضرب‌کننده m بیتی بعد از عبور از نرمال‌کننده‌ها با هم برابر نباشند یا رقم‌های نقلی تولید شده توسط نرمال‌کننده‌ها با هم برابر نباشند، سیگنال خطا برابر یک می‌شود و بروز خطا را اعلام می‌کند. در طرح پیشنهادی فرض بر این است که اگر دو عملوند ورودی یک مقایسه‌کننده با هم برابر نباشند، خروجی آن برابر یک می‌شود. در شکل ۶ ب، زمانی که سیگنال ورودی mode برابر '0' باشد، یعنی مد کاری عادی است و عملیات ضرب ۲۴ بیتی برای عملوندهای ورودی انجام می‌شود. در واقع در این حالت، ضرب‌کننده‌های ۱۲ بیتی شماره ۲ و ۳ به ترتیب حاصل‌ضرب‌های $A_L \times B_L$ و $A_H \times B_H$ را تولید می‌کنند. ضرب‌کننده‌های شماره ۱ و ۴ نیز با استفاده از چهار زیرضرب‌کننده ۶ بیتی خود به ترتیب حاصل‌ضرب‌های $A_H \times B_L$ و $A_L \times B_H$ را تولید می‌کنند. با جمع این چهار حاصل‌ضرب پاره‌ای طبق شکل ۵، یک ضرب ۲۴ بیتی انجام شده و حاصل ۴۸ بیتی آن از نرمال‌کننده مربوطه عبور می‌کند تا حاصل را در ۲۳ بیت نرمال کند.



شکل ۷: نحوه انجام ضرب t+۱۲ بیتی

۳-۲- معماری ضرب‌کننده ممیز-شناور تحمل‌پذیر اشکال با قابلیت تصحیح خطا

معماری کلی پیشنهادی برای ضرب‌کننده ممیز-شناور در دو مد کاری عادی و تحمل‌پذیر اشکال با قابلیت تصحیح خطا در شکل ۸ نشان داده شده است. در این طرح، برای دست‌یافتن به قابلیت تصحیح خطا در مد تحمل‌پذیر اشکال، به جای یک ضرب‌کننده ۲۴ بیتی بزرگ با خروجی ۴۸ بیتی، از سه ضرب‌کننده با دقت کاهش‌یافته n بیتی ($n = ۱۲+t$ و $t \leq ۶$) استفاده می‌شود تا سه خروجی ۲n بیتی تولید شده بعد از عبور از نرمال‌کننده‌ها توسط رأی‌دهنده با هم مقایسه شده و خطا احتمالی تصحیح یا پوشانده شود. در واقع، هر مانتیس ۲۳ بیتی به یک مانتیس k بیتی جدید ($k = ۱۱+t$) تبدیل می‌شود که به همراه یک بیت ضمنی، یک significand n بیتی ($n = k+۱$) را تشکیل می‌دهد. در مد تحمل‌پذیر اشکال، از ضرب‌کننده ۲۴ بیتی برای تولید سه حاصل‌ضرب با طول مساوی استفاده می‌شود. اما با توجه به اندازه ضرب‌کننده‌های کوچک‌تر، ممکن است به زیرضرب کننده‌های افزونه کوچک‌تری احتیاج باشد. بنابراین، بخش ضرب‌کننده significand های عملوندهای ورودی مشابه شکل ۶ ب می‌باشد.

همان طور که در شکل ۸ مشاهده می‌شود، بلوک نرمال‌کننده سه‌تایی شده است تا قابلیت تصحیح خطا در کل طرح افزایش یابد. همچنین، از رأی‌دهنده‌هایی استفاده شده تا پس از ارسال نتیجه به بلوک‌های بعدی، نتیجه نهایی با فرمت دقت ساده ۳۲ بیتی تولید شود. نحوه عملکرد این مدار در مد عادی مشابه طرح قبلی است.

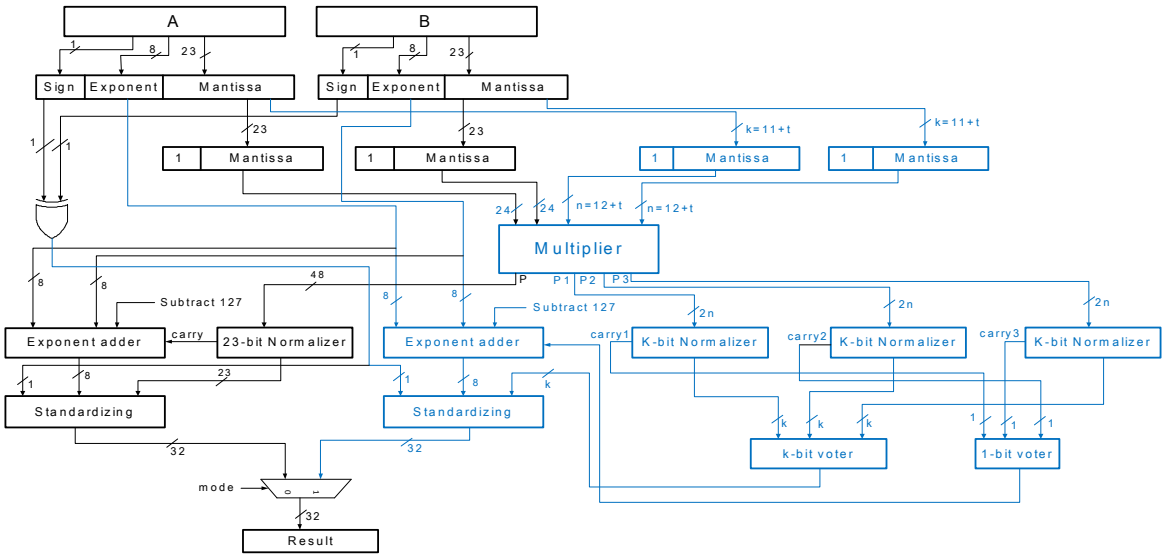
۴- نتایج پیاده‌سازی

طرح پایه و طرح‌های پیشنهادی با استفاده از زبان VHDL پیاده‌سازی شده و سپس سنتز آن‌ها توسط ابزار synopsys انجام شده است تا مساحت و توان مورد نیاز آن‌ها بدست آید. همچنین، مساحت و توان مصرفی طرح‌های پیشنهادی به ازای عرض بیت‌های مختلف مانتیس (k) اندازه‌گیری شده است. نتایج شبیه‌سازی طبق کتابخانه استاندارد CMOS 65nm LPLVT STMicroelectronics (با ولتاژ تغذیه 1.2V و دمای ۲۵ °C) در جدول‌های ۲ و ۳ آورده شده است. جدول‌های ۲ و ۳ تأخیر، مساحت، توان و سربارهای مربوط به آن‌ها را در ضرب‌کننده‌های پیشنهادی به ازای عرض بیت‌های مختلف مانتیس، در مقایسه با ضرب‌کننده پایه نشان می‌دهند. با توجه به این جدول‌ها می‌توان دریافت که انتخاب عرض مانتیس برابر با ۱۱ سربار ناچیزی را تحمیل می‌کند در حالی که طبق [۸] برای این اندازه مانتیس بسیاری از برنامه‌های کاربردی دارای دقت کافی در خروجی خود خواهند بود. همچنین، در هر دو طرح پیشنهادی برای تمام عرض مانتیس‌های ذکر شده، درجه قابلیت اطمینان از ٪۸۱ تا ٪۸۶ تغییر می‌کند. در این‌جا منظور از درجه قابلیت اطمینان، نسبت مساحت دارای قابلیت اطمینان به کل مساحت مصرفی معماری پیشنهادی است. در طرح‌های پیشنهادی، مساحت کل ضرب‌کننده اصلی و اجزای آن به همراه بخش‌های تکرار شده دارای قابلیت اطمینان در برابر اشکال هستند.

در مقایسه با کارهای گذشته، تنها کار مشابه از نوع تشخیص خطا در [۱۰] ارائه شده که 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 به ٪۱۰/۹ سربار احتیاج دارد.

۵- نتیجه‌گیری

در این مقاله، دو معماری برای ضرب‌کننده ممیز-شناور با قابلیت انجام کار در دو مد کاری عادی و تحمل‌پذیر اشکال پیشنهاد گردید. در طرح‌های پیشنهادی 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 با استفاده از تجزیه سخت‌افزار ضرب‌کننده درون مدار به ضرب‌کننده‌های کوچک‌تر پس از



شکل ۸: طرح ضرب‌کننده پیشنهادی ممیز-شناور در دو مد کاری عادی و تحمل‌پذیر اشکال با قابلیت تصحیح خطا.

جدول ۲: نتایج پیاده‌سازی طرح ضرب‌کننده ممیز-شناور پیشنهادی در دو مد معمولی 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 (میلی وات)	سربار توان	درجه قابلیت اطمینان
k=۲۳ (طرح پایه)	۸/۷۰	NA	۱۱۲۹۰/۲	NA	۵/۶۲	NA	NA
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 در دو مد معمولی 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 (میلی وات)	سربار توان	درجه قابلیت اطمینان
k=۲۳ (طرح پایه)	۸/۷۰	NA	۱۱۲۹۰/۲	NA	۵/۶۲	NA	NA
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٪

حذف بیت‌های کم‌ارزش‌تر به منظور انجام محاسبات افزونه، به قابلیت‌های اشاره شده می‌رسند. نتایج پیاده‌سازی نشان می‌دهد که با سربارهای معقول می‌توان به ساختارهایی رسید که دارای درجه بالایی از قابلیت اطمینان در برابر خطاهای منفرد باشند.

فهرست منابع

[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 Journal of Solid-State Circuits*, vol. 42, no. 10, pp. 2257-2269, Oct. 2007.

[2] L. Huang, M. Lai, K. Dai, and H. Yue, "Hardware Support for Arithmetic Units of Processor with Multimedia Extension," *IEEE Intl. Conf. Multimedia and Ubiquitous Engineering*, pp. 633-637, 2007.

[3] 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, 2006.

[4] T. Karnik, and P. Hazucha, "Characterization of Soft Errors Caused by Single Event Upsets in (CMOS) Processes," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 2, pp. 128-143, 2004.

[5] S.C. Lai, S.L Lu, K. Lai, and J.K Peir, "Ditto Processor," *IEEE Intl. Conf. Dependable Systems and Networks*, pp. 525-534, 2002.

[6] D. Lipetz, and E. Schwarz, "Self Checking in current Floating-Point Units," *20th IEEE Symp.Computer Arithmetic,* pp.73–76, 2011.

[7] M. Maniatakos, Y. Makris, P. Kudva, and B. Fleischer, "Exponent Monitoring for Low-Cost Concurrent Error Detection in FPU Control Logic," *IEEE VLSI Test Symposium*, pp.235–240, 2011.

[8] 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, 2002.

[9] P.J. Eibl, A.D. Cook, and D.J. Sorin, "Reduced Precision Checking for a Floating Point Adder," *24th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, pp. 145-152, Oct. 2009.

[10] K. Seetharam, L.C.T. Keh, R. Nathan, and D.J. Sorin., "Applying Reduced Precision Arithmetic to Detect Errors in Floating Point Multiplication," *IEEE Pacific Rim International Symposium on Dependable Computing (PRDC),* pp. 232-235, Dec 2013.