علوم رایانش و فناوری اطلاعات
نشریه علمی انجمن کامپیوتر ایران
مجلد ۱۴، شماره ۱، ۱۳۹۵
صفحات ۴۰-۴۸
مقاله عادی

نشریه علوم رایانش و فناوری اطلاعات

طراحی و پیاده‌سازی جمع‌کننده دهدهی افزونه‌ای با توان مصرفی پایین

سعید گرگین لیلا میرمقتدایی

پژوهشکده برق و کامپیوتر، سازمان پژوهش‌های علمی و صنعتی ایران، تهران، ایران

چکیده

در سیستم‌های کامپیوتری عمل جمع جزء اصلی کلیهٔ پردازش‌ها و پایه تمام عملیات حسابی است؛ به طوری که افزایش سرعت در عمل جمع بر کارآیی کل سیستم تاثیر شگرفی دارد. در نظام‌های عددی متعارف، انتشار نقلی باعث وابستگی زمان جمع به طول عملوندها می‌شود که این مسئله در عملیات با دقت بالا مشکل آفرین است و تاخیر زیادی را تحمیل خواهد کرد. حال آنکه با به‌کارگیری نظام‌های عددی نامتعارف افزونه‌ای، عمل جمع بدون انتشار نقلی و در زمان ثابت قابل انجام است. در این مقاله، با توجه به اهمیت روزافزون سیستم‌های محاسباتی مبنای ده، با استفاده از الگوریتم افراز بیت‌های وزن‌دار و خاصیت افزونگی ذاتی موجود در ارقام دهدهی، تقسیم‌بندی جدیدی برای جمع اعداد دهدهی افزونه‌ای ارائه شده است که با حفظ سرعت، توان مصرفی و مساحت روی تراشه را نسبت به روش‌های پیشین کاهش می‌دهد. هم‌چنین خروجی الگوریتم ارائه شده به شکل ارقام علامت‌دار متقارن و بازه [7 ,7-] است اما امکان استفاده از تمام فضای کدینگ (بازه نامتقارن [7 ,8-]) در ورودی وجود دارد.

کلمات کلیدی: جمع دهدهی، نمایش افزونه‌ای، رقم علامت‌دار، انتشار نقلی.

۱- مقدمه

پیشرفت غیرقابل تصور صنعت نیمه هادی، پیاده‌سازی سخت‌افزاری بسیاری از توابع را میسر ساخته است و در این میان به دلایل مختلف از جمله عدم نمایش دقیق برخی از اعداد (مانند ۰/۲) در نظام اعداد دودویی، هم‌چنین به وجود آمدن نیازهای جدید در کاربردهای مالی/تجاری، موجبات ظهور مجدد سیستم‌های محاسباتی سخت‌افزاری دهدهی را فراهم آورده و توجه بسیاری از پژوهشگران را معطوف خود ساخته است [۱]. پژوهش‌های بسیاری در خصوص مسائل و مشکلات استفاده از حساب دودویی [۲-۴]، هم‌چنین مطالعات گسترده‌ای در مورد حجم داده‌های دهدهی در پایگاه‌های اطلاعاتی مختلف [۵] انجام شده است، که برآیند نتایج این تحقیقات موید ضرورت و اهمیت به‌کارگیری سخت‌افزارهای حساب دهدهی است. به دلیل اهمیت این موضوع استاندارد IEEE754، مورد بازنگری قرار گرفت و نمایش دهدهی به عنوان بخشی از استاندارد به آن افزوده شد [۶]. هم‌چنین پردازنده‌های تجاری گوناگونی (z900 eServer [۷]، Power6 [۸]، z10 [۹]) با واحد اختصاصی محاسبات دهدهی روانه بازار شده‌اند.

مدارات حسابی دهدهی با استفاده از روش‌های کدگذاری خاص براساس مدارات دودویی پیاده‌سازی می‌گردد (جدول ۱) که همین مسئله موجب پیچیدگی بیشتر مدارات دهدهی در مقایسه با مدارات مشابه دودویی می‌شود. به طور مثال، در نمایش دهدهی وزن هر رقم نسبت به رقم پرارزش‌تر (یا کم‌ارزش‌تر) خود ده برابر کم‌تر (یا بیشتر) است و همانند اعداد دودویی از توان‌های دو پیروی نمی‌کند، لذا در صورت بروز نقلی در عمل جمع، عملیات تصحیح می‌بایست صورت پذیرد که این عمل تصحیح موجب افزایش پیچیدگی می‌گردد [۱۰]. این در حالی است که عمل جمع جزء اصلی کلیهٔ پردازش‌ها و پایه تمام عملیات حسابی است و افزایش یا کاهش سرعت در عمل جمع بر کارآیی کل سیستم تاثیرگذار است [۱۱].

به طور کلی، درنظام‌های عددی متعارف، انتشار نقلی باعث وابستگی زمان جمع به طول عملوندها می‌شود که این مسئله در عملیات با دقت بالا تاخیر زیادی را تحمیل می‌کند. حال آنکه می‌توان با بهره‌گیری ازنظام‌های عددی نامتعارف افزونه‌ای، عمل جمع را بدون انتشار نقلی و در زمان ثابت پیاده‌سازی نمود [۱۲]. با توجه به اهمیت عمل جمع و نقش کلیدی آن در پیاده‌سازی سایر واحدهای محاسباتی، روش‌های مختلفی برای بهبود عملکرد آن ارائه شده است. در این مقاله

با استفاده از الگوریتم افراز مجموعه بیت‌های وزن‌دار [۱۳] و خاصیت افزونگی ذاتی ارقام دهدهی، تقسیم بندی جدیدی مبتنی بر چهار بیت ارائه خواهد شدکه با توجه به کاهش ورودی‌های توابع به‌کارگرفته شده، در مقایسه با الگوریتم اصلی، توان‌مصرفی و مساحت کم‌تری روی تراشه دارد.

جدول ۱- روش‌های مختلف کدگذاری ارقام دهدهی

ردیف	نام کد یا مرجع	بازه	کدینگ	α	θ	ξ	ρ	γ
۱	۸۴۲۱	$[0,9]$	●●●●	0	9	10	0	6
۲	افزون بر ۳	$[3,12]$	●●●●	3	12	10	0	6
۳	۴۲۲۱	$[0,9]$	●●● ●	0	9	10	0	0
۴	[۱۴]	$[-5,5]$	●●●●●	−5	5	11	1	20
۵	[۱۵]	$[-6,6]$	●●●●●	−6	6	13	3	18
۶	[۱۳و۱۶]	$[-7,7]$	○●●●	−7	7	15	5	1
۷	[۱۷]	$[-9,9]$	●●●● ○○○○	−9	9	19	9	36
۸	[۱۸و۱۹]	$[0,15]$	●●●●	0	15	16	6	0
۹	[۲۰و۲۱]	$[0,10]$	●●●● ●	0	10	11	1	6
۱۰	[۲۲و۲۳]	$[0,18]$	●●● ● ●●● ●	0	18	19	9	0
۱۱	[۲۴]	$[-8,9]$	○●●● ■	−8	9	18	8	0
۱۲	[۲۵]	$[-9,7]$	○●●● ○	−9	7	17	7	0

در ادامه این مقاله، در بخش دوم، کلیاتی در مورد الگوریتم جمع دهدهی افزونه‌ای و هم‌چنین کدینگ‌های مختلف و جمع کننده‌های افزونه‌ای پیشین ارائه می‌شود. در بخش سوم به ارائه الگوریتم افراز مجموعه بیت‌های وزن‌دار خواهیم پرداخت و در بخش چهارم شیوه افراز مبتنی بر چهار بیت مطرح و با چند مثال تشریح خواهد شد. در بخش پنجم، کلیه الگوریتم‌ها از جوانب مختلف مورد مقایسه و ارزیابی قرار خواهد گرفت و در پایان، در بخش ششم نتیجه‌گیری مقاله ارائه می‌شود.

۲- پیش‌زمینه

همان‌طور که در بخش پیشین بدان اشاره شد، برای نمایش اعداد دهدهی با استفاده از مدارات دودویی روش‌های کدگذاری مختلفی استفاده می‌شود. برخی از متداول‌ترین روش‌های مورد استفاده به همراه خصوصیات منحصر به فرد هر کد، در جدول (۱) نشان داده شده است. این خصوصیات عبارتند از:کوچک‌ترین مقدار قابل نمایش (α)، بزرگ‌ترین مقدار قابل نمایش (θ)، تعداد نمادهای مجاز (ξ)، شاخص افزونگی (ρ)، تعداد حالاتی که مورد استفاده قرار نمی‌گیرند (γ) ونمایش نقطه‌ای هر رقم که با عنوان کدینگ در جدول ۱ مشخص شده است.

با مطالعه دقیق جدول ۱ اختلاف اساسی مبنای ده با سایر مبناهای رایج (که عموماً توانی از دو هستند) آشکار می‌شود. در واقع برای نمایش اعداد در نظام‌های افزونه‌ای نیاز به فضای ذخیره‌سازی بیشتری به منظور نمایش بیت‌های افزونه وجود دارد، اما در صورتی که مبنای مورد استفاده توانی از دو نباشد لزوماً این قاعده بر قرار نخواهد بود. برای مثال در حساب دهدهی، حداقل به چهار بیت برای نمایش یک رقم غیرافزونه‌ای که ده مقدار مختلف می‌تواند داشته باشد نیاز است (ردیف‌های ۶ و ۸ در جدول ۱). لذا، شش حالت از شانزده حالت فضای کد گذاری استفاده نمی‌شود و می‌تواند به عنوان نمادهای لازم برای نمایش افزونه‌ای استفاده گردد. به این خاصیت اصطلاحاً «افزونگی ذاتی» می‌گویند. از آن‌جا که کدهای دارای این خاصیت از فضای کدگذاری و سخت‌افزار به شکل بهینه استفاده می‌کنند، به‌کارگیری آن‌ها موجب کاهش حجم سخت‌افزار و توان مصرفی مدارات مربوطه می‌گردد.

۲-۱- جمع دهدهی افزونه‌ای

در نمایش غیرافزونه، چنانچه x و y ارقام ورودی جمع‌کننده و c_{in} به عنوان نقلی در نظر گرفته شود؛ حاصل جمع میانی $p = x + y + c_{in}$ محاسبه می‌شود و مقادیر حاصل جمع‌کو نقلی خروجی c_{out} براساس مجموعه روابط (۱) محاسبه می‌شود:

$$s = |p|_{10} \text{ و } c_{out} = \left\lfloor \frac{p}{10} \right\rfloor \qquad (۱)$$

به طور مشخص در این روش، برای بدست آوردن حاصل جمع هر موقعیت، مقدار نقلی ورودی از موقعیت‌های با ارزش مکانی کم‌تر می‌بایست مشخص گردد (انتشار نقلی). حال آن‌که با استفاده از نمایش افزونه‌ای این انتشار به طور کامل حذف می‌شود. الگوریتم کلی جمع بدون انتشار نقلی برای یک نظام عددی با ارقام علامت‌دار در بازه $[-\alpha, \theta]$ و مبنای r به شکل زیر قابل ارائه است:

الگوریتم ۱ (جمع بدون انتشار نقلی ارقام علامت‌دار)

ورودی‌ها و خروجی: اعداد k رقمی با نمایش ارقام علامت‌دار در مبنای r که ورودی‌های به شکل $X = x_{k-1} \cdots x_0$ و $Y = y_{k-1} \cdots y_0$ و خروجی به شکل $S = s_{k-1} \cdots s_0$ نمایش داده می‌شود [۱۲].

تمام رقم‌های اعداد ورودی و خروجی در بازه $[-\alpha, \theta]$ قرار دارند و سه مرحله زیر روی تمام ارقام به صورت موازی اجرا می‌شود:

۱. به‌دست آوردن حاصل جمع $P = p_{k-1} \cdots p_0$ ($p_i = x_i + y_i$). هر رقم p_i در بازه $[-2\alpha, 2\theta]$ قرار دارند.

۲. تجزیه هر p_i به حاصل‌جمع موقتی w_i و رقم انتقالی t_{i+1} به نحوی که شرایط $p_i = w_i + r \times t_{i+1}$ و $-\alpha + t_{min} \le w_i \le \theta + t_{max}$ برقرار باشند.

۳. به‌دست آوردن حاصل‌جمع نهایی به صورت: $s_i = w_i + t_i$.

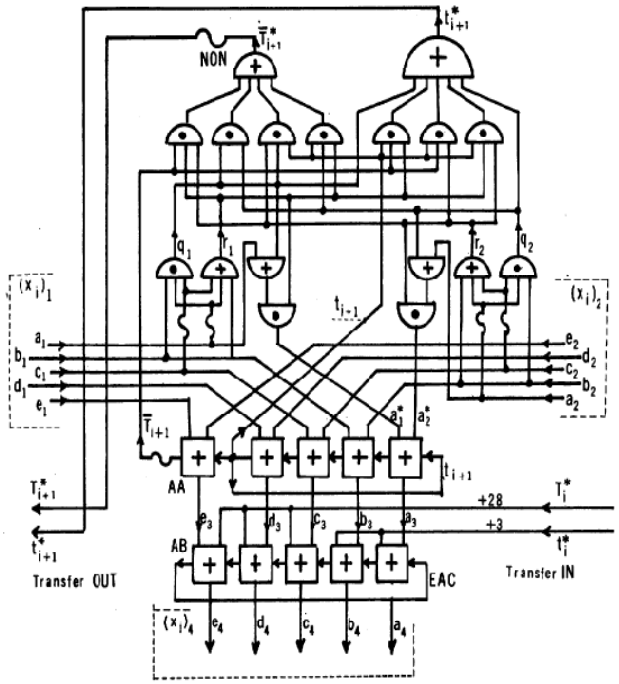
لازم به ذکر است انتخاب بازه رقم انتقالی $[t_{min}, t_{max}]$ باید به نحوی باشد که در مرحله سوم رقم‌انتقالی جدیدی تولید نشود. در نظام اعداد ارقام علامت‌دار متداول $t_{max} = 1$ و $t_{min} = -1$ انتخاب می‌شوند. مقدار r در نمایش دهدهی برابر ۱۰ و در نمایش‌های مبنای دو برابر 2^h که $h > 2$ است، انتخاب می‌شود.

۲-۲- جمع‌کننده‌های دهدهی افزونه‌ای

در حساب دهدهی، به دلیل امکان انتخاب مجموع ارقام مختلف، هم‌چنین روش‌های کدگذاری متفاوت،الگوریتم‌های متنوعی ارائه شده است که در ادامه توضیح مختصری در مورد هر یک عرضه می‌گردد. لازم به ذکر است با توجه به اینکه در الگوریتم ۱ نیاز به سه عمل جمع وجود دارد، در الگوریتم‌های زیر به نحوی سعی بر کم‌کردن و ساده‌سازی الگوریتم اصلی شده است.

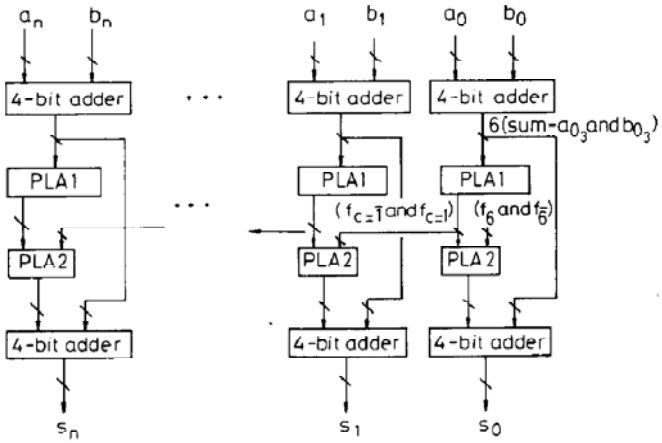
- اولین روش جمع افزونه‌ای برای محاسبات دهدهی در مقاله [۱۵] ارائه گردید. در این روش هر رقم در بازه [6, -6] است و از پنج بیت برای نمایش هر رقم استفاده می‌گردد. در این شیوهٔ کد گذاری هر مقدار با سه برابر خود و در صورت منفی بودن با سه برابر منهای ۳۱، کد می‌شود و به‌دلیل متقارن بودن

بازه و خود مکمل بودن کد، عمل منفی کردن به آسانی قابل انجام است. شمای از این جمع کننده در شکل ۱ نشان داده شده است.



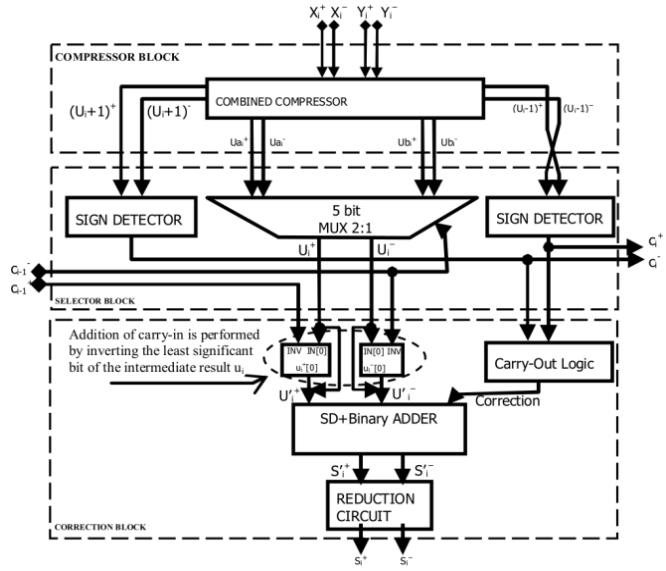
شکل ۱- جمع‌کننده دهدهی افزونه‌ای ارائه شده در [۱۵]

- در مقاله [۱۶]، با استفاده از کدگذاری مکمل دو، از ارقامی در بازه [7 ,7-] برای نمایش اعداد دهدهی افزونه‌ای (RBCD)[1] استفاده شده است. این روش نیز کاملاً مبتنی بر الگوریتم کلی جمع ارقام علامت‌دار بوده و از دو PLA برای پیاده‌سازی دو تابعِ تولید کننده رقم انتقالی و مقدار تصحیح استفاده می‌کند (شکل ۲).



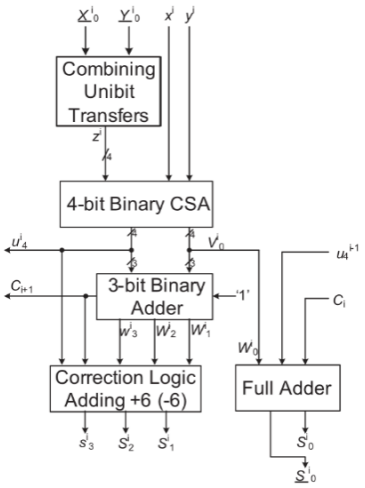
شکل ۲- جمع‌کننده دهدهی افزونه‌ای ارائه شده در [۱۶]

- در مقاله [۱۷]، با استفاده از نمایش اعداد در بازه متقارن [9 ,9-] یک جمع‌کننده جدید افزونه‌ای ارائه شده است. در این روش هر رقم از ترکیب دو رقم BCD یکی با وزن مثبت و دیگری با وزن منفی تشکیل شده است و ارزش هر رقم دهدهی با کم کردن قسمت منفی از قسمت مثبت به‌دست می‌آید $z_i = z_i^+ - z_i^-$. با وجود اینکه این نظام از نظر سخت‌افزاری، سربار بسیار زیادی دارد اما در مواردی همانند منفی کردن و در نتیجهٔ آن تفریق کردن، هم‌چنین در تبدیلات، دارای کارآیی مطلوبی است. همچنین در مقاله [۲۶]، با استفاده از الگوریتم و چارچوب کلی ارائه شده در [۱۷]، سعی در بهبود این ایده شده است. طرح بهبود یافته در شکل ۳ نشان داده شده است.



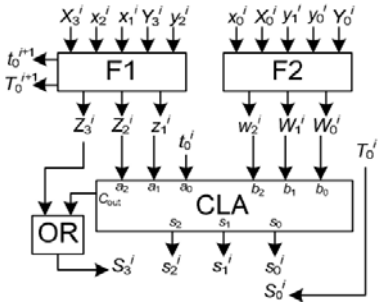
شکل ۳- جمع‌کننده دهدهی افزونه‌ای ارائه شده در [۲۶]

- در مقاله [۱۳] با استفاده از افراز مجموعه بیت‌های وزن‌دار عملیات جمع بر روی عملوندهایی در بازه [7 ,7-] و نمایش مکمل دو، صورت می‌پذیرد. نتایج تحلیل و سنتز صورت گرفته نشان دهنده برتری این الگوریتم در تمامی پارامترهای طراحی است. این الگوریتم در بخش سوم به طور مفصل مورد تجزیه و تحلیل قرار می‌گیرد.
- در [۲۴]، با به کارگیری نمایش یونی‌بیت و استفاده از مجموع ارقامی در بازه [9, -8]، در واقع با یک کدگذاری پنج بیتی، الگوریتمی کارآ برای جمع دهدهی افزونه ارائه شده است (شکل ۴).



شکل ۴- جمع‌کننده دهدهی افزونه‌ای ارائه شده در [۲۴]

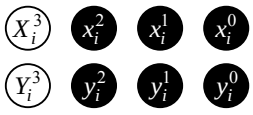
- در مقاله [۲۵] با استفاده از روش ارائه شده در [۱۳] و ارقامی در بازه [7 ,9-]، با استفاده از پنج بیت الگوریتم جدیدی برای جمع افزونه‌ای دهدهی ارائه شده است. به دلیل استفاده از پنج بیت به منظور کدینگ ارقام و توابع بزرگ‌تر در مسیر بحرانی تاخیر این جمع کننده (F2)، تقریباً در تمامی مقایسه‌ها، عملکردی به مراتب کم‌تر از [۱۳] دارد. شمای از این جمع کننده در شکل ۵ نشان داده شده است.



شکل ۵- جمع‌کننده دهدهی افزونه‌ای ارائه شده در [۲۵]

۳- الگوریتم افراز بیت‌های وزن‌دار

در الگوریتم جمع افزونه‌ای با استفاده از افراز بیت‌های وزن‌دار، مرحله اول الگوریتم ۱، بدون هزینه و با استفاده از نمایش ذخیره نقلی انجام می‌شود. این حاصل جمع میانی با استفاده از مجموعه ارقام [7 ,7-]، در شکل ۱ نشان داده شده است. این مجموعه، «ارقام علامت‌دار دهدهی هفت‌تایی»[۲] و به اختصار DSSD نام‌گذاری شده و مشابه روش کدگذاری [۱۶]، $\alpha = 7$ و هر رقم به صورت مکمل‌دو نمایش داده می‌شود. لازم به ذکر است که این نحوه نمایش دارای افزونگی ذاتی است و علاوه بر جمع، امکان تفریق را نیز به آسانی فراهم می‌نماید. در شکل ۶ بیت‌های علامت، که بیت‌هایی با وزن منفی هستند و اصطلاحاً به آن‌ها نگابیت[۳] گفته می‌شود به شکل دوایر توخالی (حروف بزرگ) و سایر بیت‌ها که و اصطلاحاً به آن‌ها پوزی‌بیت[۴] گفته می‌شود با دوایر توپر (حروف کوچک) نشان داده شده است.



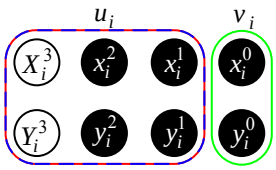
شکل ۶- نمایش p_i به شکل ذخیره‌نقلی مکمل‌دو

در مرحله دوم الگوریتم ۱، از مقایسه حاصل جمع میانی با بازه مجاز ارقام، حاصل‌جمع موقتی w_i و رقم انتقالی t_{i+1} (با در نظر گرفتن شرایط مربوطه) استخراج می‌شود. با توجه به اینکه عمل مقایسه به نوعی پیچیدگی عمل جمع را دارد و موجب افزایش زمان محاسبه می‌شود، راهکارهای مختلفی برای اجتناب از مقایسه به کارگرفته می‌شود که در ادامه روش افراز، به عنوان روش ارائه شده در [۱۳] و بهبود یافته در این مقاله، مورد مطالعه قرار گرفته است.

۳-۱- افراز مبتنی بر شش بیت

از نظر تئوری امکان استخراج مقدار رقم انتقالی t_{i+1} با استفاده مجموعه بیت‌های p_i وجود دارد، اما در صورت استفاده از تمام بیت‌های این مجموعه، با یک تابع منطقی هشت ورودی روبه‌رو خواهیم بود که در عمل از نظر پیاده‌سازی شرایط مطلوبی را فراهم نخواهد ساخت. لذا با افراز مجموعه بیت‌های p_i سعی در کم کردن تعداد ورودی‌های تابع فوق‌الذکر خواهیم نمود. در شکل ۷ افراز مجموعه بیت‌های p_i به دو مجموعه u_i و v_i نشان داده شده است؛ ($\|p_i\| = \|u_i\| + \|v_i\|$) به‌طوری‌که u_i شامل سه بیت پرارزش عملوندها و v_i شامل دو بیت باقیمانده است.

بـــــــا ایـــــــن روش افـــــــراز مقـــــــدار $\|v_i\| \in [0,2]$ و $\|u_i\| \in \{-16,-14,\dots,-2,0,2,\dots,10,12\}$ خواهد بود. با استفاده از u_i، رقم‌انتقالی دهدهی $t_{i+1} \in \{-1,0,1\}$ استخراج و مقدار باقیمانده با توجه به رابطه $\|z_i\| = \|u_i\| - 10 \times t_{i+1}$ به شکل $\|z_i\| \in \{-6,-4,-2,0,2,4\}$ مشخص می‌گردد. در مرحله بعد مقدار حاصل‌جمع میانی $w_i = \|z_i\| + \|v_i\|$ با استفاده از مقادیر v_i و z_i به‌دست می‌آید. این مقدار در بازه $[-6,4] + [0,2] = [-6,6]$ قرار دارد که در نهایت با اضافه شدن مقدار رقم‌انتقالی موقعیت کم‌ارزش‌تر، حاصل‌جمع نهایی $s_i \in [-7,7]$ حاصل می‌گردد.

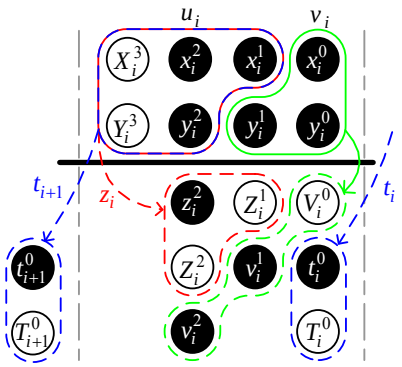


شکل ۷- افراز p_i با استفاده از شش بیت

در افراز مبتنی بر شش بیت، تعداد ورودی‌های تابع تولید کننده رقم انتقالی و همچنین مقدار باقیمانده از هشت به شش کاهش می‌یابد که این امر علاوه بر افزایش سرعت، موجب کاهش مساحت بر تراشه و توان مصرفی نیز می‌گردد. در بخش بعد سعی بر کاهش بیشتر تعداد ورودی‌های تابع فوق‌الذکر خواهد بود.

۳-۲- افراز مبتنی بر پنج بیت

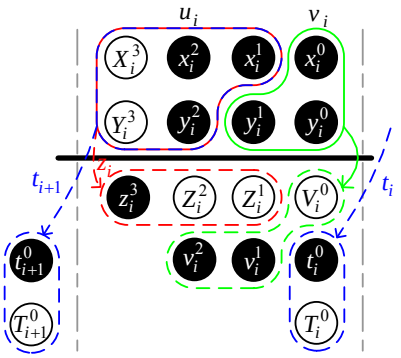
چنانچه افرازمجموعه بیت‌های p_i همانند شکل ۸ صورت پذیرد، تعداد ورودی‌های تابع تولید کننده رقم انتقالی و مقدار باقیمانده پنج بیت خواهد بود. در شکل ۸ جزئیات بیشتری از نحوه استخراج z_i و t_{i+1} به همراه کدینگ مورد استفاده نشان داده شده است.



شکل ۸- افراز p_i و استخراج t_{i+1} و z_i با کدینگ $Z_i^1 Z_i^2 z_i^2$ [۱۳]

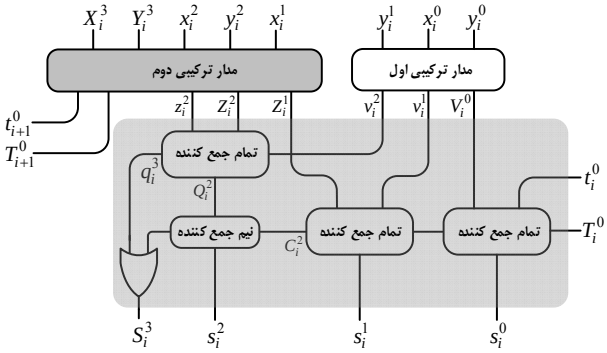
با افراز پنج بیتی، مقدار v_i افزایش می‌یابد ($\|v_i\| \in [0,4]$) و مقدار دوازده از بازه u_i خارج می‌شود. در این صورت، $\|z_i\| \in \{-6,-4,-2,0,2\}$ و $t_{i+1} \in \{-1,0,1\}$ خواهد شد که همانند افراز شش بیتی حاصل جمع میانی در بازه $[-6,6]$ به‌دست می‌آید و در نهایت حاصل‌جمع نهایی در بازه مطلوب $s_i \in [-7,7]$ حاصل می‌گردد.در عمل، همان‌طور که در شکل ۸ نشان داده شده است، بوسیله دو مدار ترکیبی مقدار v_i به $v_i^2 v_i^1 V_i^0$ تبدیل و مقادیر $(Z_i^1 Z_i^2 z_i^2) z_i$ و $(t_{i+1}^0 T_{i+1}^0) t_{i+1}$ از u_i به‌دست می‌آیند. نحوه کدینگ به گونه‌ای در نظر گرفته شده است که علاوه بر حفظ درستی مقدار نمایش داده شده، مقدار

خروجی نیز در بازه و کدینگ مطلوب حاصل شود ($s_i^0 s_i^1 s_i^2 S_i^3$). به طور مثال به جای شکل ۸ می‌توان از شکل ۹ نیز استفاده نمود. در این افراز نحوه کدینگ مقدار v_i به $v_i^2 v_i^1 V_i^0$ تبدیل و مقدار t_{i+1} ($t_{i+1}^0 T_{i+1}^0$) مانند حالت قبل از u_i می‌باشد و تفاوت در کدینگ z_i می‌باشد. z_i را در این کدینگ به صورت $z_i^3 Z_i^2 Z_i^1$ کدگذاری می‌کنیم و در این حالت نیزنحوه کدینگ به گونه‌ای در نظر گرفته شده است که علاوه بر حفظ درستی مقدار نمایش داده شده، مقدار خروجی نیز در بازه و کدینگ مطلوب حاصل شود ($s_i^0 s_i^1 s_i^2 S_i^3$).



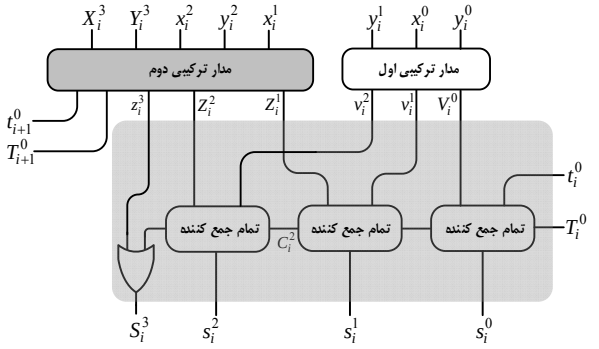
شکل ۹- افراز p_i و استخراج t_{i+1} و z_i با کدینگ $z_i^3 Z_i^2 Z_i^1$

چنانچه z_i مطابق با شکل ۸ استخراج گردد، شمای کلی مدار جمع دهدهی دو رقم در بازه $[-7,7]$، همانند شکل ۱۰ خواهد بود.



شکل ۱۰- مدار جمع‌کننده دو رقم در بازه $[-7,7]$ با کدینگ $Z_i^1 Z_i^2 z_i^2$

حال اگر z_i مطابق با شکل ۹ استخراج گردد، شمای کلی مدار جمع دهدهی دو رقم در بازه $[-7,7]$، همانند شکل ۱۱ می‌گردد.



شکل ۱۱- مدار جمع‌کننده دو رقم در بازه $[-7,7]$ با کدینگ $z_i^3 Z_i^2 Z_i^1$

با مقایسه شکل ۱۰ و ۱۱، سادگی بخش جمع‌کننده در شکل ۱۱ به دلیل استفاده از کدینگ $z_i^3 Z_i^2 Z_i^1$ قابل مشاهد است (یک نیم جمع‌کننده کمتر استفاده شده است). لازم به ذکر است در پیاده‌سازی واقعی هر دو شکل از جمع‌کننده پیش‌بینی نقلی استفاده می‌شود.

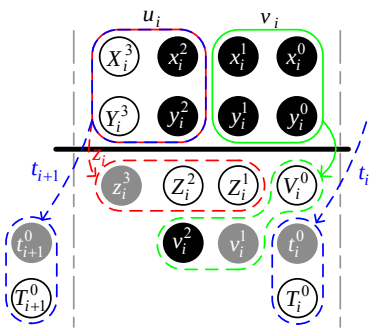
از آن‌جایی که عملوندها در نمایش DSSD به شکل مکمل دو کد می‌شوند، به آسانی با معکوس کردن بیت‌های مفروق و افزودن مقدار یک به شکل نقلی ورودی عمل تفریق قابل انجام است. لازم به ذکر است که برای راحتی کار و عدم نیاز به گیت‌های معکوس‌کننده، نگابیت‌ها به شکل معکوس کدگذاری شده‌اند (مقدار صفر منطقی نشان دهنده ارزش حسابی منفی یک و مقدار یک منطقی نشان دهنده مقدار حسابی صفر است) [۲۷].

۴- الگوریتم افراز مجموعه بیت‌های وزن‌دار با تقسیم‌بندی چهار بیتی

در بخش ۳-۲ افراز پنج بیتی مجموعه بیت‌های p_i نشان داده شد. چنانچه عمل افراز به شکل چهار بیتی انجام شود، مقدار v_i در بازه $[0,6]$ قرار می‌گیرد و مقدار u_i به بازه $\{-16,-12,-8,-4,0,4,8\}$ محدود می‌شود. در این افراز، اگر $\|u_i\| = -8$ باشد؛ برای استخراج مقدار صحیح رقم انتقالی، مقدار v_i تعیین کننده خواهد بود. لذا همان‌طور که در [۱۹] نشان داده شده است، از نظر تئوری امکان افراز با چهار بیت وجود ندارد مگر اینکه حالات خاص ایجاد شده به نوعی با کم‌ترین تاثیر بر کارآیی مدیریت شود.

اگر به‌طور پیش فرض در حالت $\|u_i\| = -8$ مقدار $t_{i+1} = -1$ و $\|z_i\| = 2$ در نظر گرفته شود، چنانچه $\|v_i\| > 4$ باشد، با در نظر گرفتن رقم انتقالی ورودی امکان تولید مقداری خارج از بازه $[-7,7]$ وجود خواهد داشت. از آنجا که تنها حالت مشکل ساز، در $\|u_i\| = -8$ و $\|v_i\| > 4$ حادث می‌شود، این حالت به عنوان یک استثنا شناسایی شده و اقدام لازم برای رفع آن صورت می‌پذیرد.

به طور کلی، در حالت $\|u_i\| = -8$ مقدار $t_{i+1} = -1$ با یک نگابیت با کدگذاری معکوس $T_{i+1}^0 = 0$ و یک پوزی بیت $t_{i+1}^0 = 0$، هم‌چنین $\|z_i\| = 2$ با $Z_i^2 = 0.Z_i^1 = 0$ و $z_i^3 = 1$ نشان داده می‌شود. حال اگر در این شرایط $\|v_i\| > 4$ باشد، در واقع حالت استثنا رخ داده است می‌بایست مقادیر t_{i+1} و z_i به $t_{i+1} = 0$ و $\|z_i\| = -8$ تصحیح گردد. برای انجام تصحیح رقم انتقالی، کافیست مقدار $t_{i+1}^0 = 1$ شود. اما برای تصحیح مقدار z_i، علاوه بر صفر کردن مقدار z_i^3، مقدار v_i^1 که در صورت بروز استثنا همواره یک است می‌بایست صفر گردد. بیت‌های تحت تاثیر استثنا در شکل ۱۲ با رنگ خاکستری نشان داده شده است.



شکل ۱۲- افراز چهار بیتی p_i و استخراج t_{i+1} و z_i

در ادامه جزئیات فرآیند فوق‌الذکر همراه با روابط منطقی لازم برای پیاده‌سازی مدار مربوطه در روابط (۲) تا (۶) آمده است. در واقع بعد از تشکیل p_i و افراز آن به دو مجموعه u_i و v_i، چهار مرحله زیر برای رسیدن به حاصل‌جمع نهایی متصور است. قسمت‌های مختلف این مراحل دارای هم‌پوشانی زمانی هستند و معادلات لازم برای پیاده‌سازی توابع منطقی در هر مرحله ذکر شده است. تمام روابط با استفاده از جدول درستی به راحتی قابل استخراج هستند.

۱- برای تشخیص حالت استثنا، در چهار بیت کم‌ارزش رخ دادن مقدار پنج یا شش ($\|v_i\| > 4$) بررسی می‌شود (E_i^l) و در چهار بیت پرارزش منفی هشت ($\|u_i\| = -8$) بررسی می‌گردد (E_i^h). در صورت مثبت بودن هر دو، استثنا رخ داده است ($E_i = E_i^l E_i^h$).

$$\begin{aligned} E_i^l &= x_i^1 y_i^1 (x_i^0 \vee y_i^0) \\ (E_i^h &= (X_i^3 \oplus Y_i^3) x_i^2 y_i^2 \vee \overline{X_i^3 Y_i^3} x_i^2 y_i^2 \end{aligned} \qquad (۲)$$

۲- استخراج t_{i+1} و z_i از u_i با توجه به رخ داد یا عدم رخ داد استثنا و در نظر گرفتن $u_i = 10\|t_{i+1}\| + \|z_i\|$. لازم به ذکر است که مقدار بیت‌های T_{i+1}^0، Z_i^1 و Z_i^2 مستقل از استثنا محاسبه می‌شوند و فقط بیت‌های t_{i+1}^0، z_i^3 و v_i^1 از رخ داد استثنا تاثیر می‌پذیرند (در شکل ۱۲ نشان داده شده است).

$$\begin{aligned} t_{i+1}^0 &= (X_i^3 \vee Y_i^3)(x_i^2 \vee y_i^2) \vee X_i^3 Y_i^3 \vee \\ &\quad E_i^l (X_i^3 \vee Y_i^3 \vee x_i^2 y_i^2) \\ T_{i+1}^0 &= X_i^3 Y_i^3 (x_i^2 \vee y_i^2) \end{aligned} \qquad (۳)$$

$$\begin{aligned} z_i^3 &= E_i^h \overline{E_i^l} \\ Z_i^2 &= (X_i^3 \vee Y_i^3) x_i^2 y_i^2 \vee (x_i^2 \oplus y_i^2) \overline{X_i^3 Y_i^3} \vee \\ &\quad X_i^3 Y_i^3 \overline{x_i^2 y_i^2} \\ Z_i^1 &= (X_i^3 \oplus Y_i^3)(x_i^2 \vee y_i^2) \vee X_i^3 Y_i^3 \overline{x_i^2 y_i^2} \end{aligned} \qquad (۴)$$

۳- تبدیل v_i به $v_i^2 v_i^1 V_i^0$، با در نظر گرفتن حالت استثنا برای v_i^1.

$$\begin{aligned} V_i^0 &= \overline{x_i^0 \oplus y_i^0} \\ v_i^1 &= \overline{E_i} (y_i^1 \oplus (x_i^0 \vee y_i^0)) \\ v_i^2 &= (x_i^0 \vee y_i^0)(x_i^1 \vee y_i^1) \vee x_i^1 y_i^1 \end{aligned} \qquad (۵)$$

۴- تمام مقادیر تولید شده به وسیله یک جمع‌کننده سه بیتی جمع شده و بیت‌های s_i^0، s_i^1 و s_i^2 را تولید می‌کنند و نقلی خروجی جمع‌کننده به وسیله «یای» منطقی با z_i^3 بیت S_i^3 را تولید می‌نماید.در روابط نشان داده شده مقادیر نقلی به شکل ساده و با فرض استفاده از جمع کننده انتشار نقلی موجی نشان داده شده است که در عمل، از جمع‌کننده پیش‌بینی نقلی استفاده می‌شود.

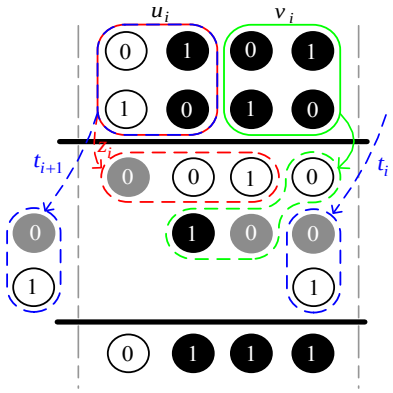
$$\begin{aligned} s_i^0 &= (V_i^0 \oplus t_i^0) \oplus T_i^0 C_i^1 = \text{carry}\,(V_i^0, T_i^0, t_i^0) \\ s_i^1 &= (v_i^1 \oplus Z_i^1) \oplus C_i^1 C_i^2 = \text{carry}\,(Z_i^1, C_i^1, v_i^1) \\ s_i^2 &= (v_i^2 \oplus Z_i^2) \oplus C_i^2 C_i^3 = \text{carry}\,(Z_i^2, C_i^2, v_i^2) \\ S_i^3 &= z_i^3 \vee C_i^3 \end{aligned} \qquad (۶)$$

برای یک‌پارچه ساختن واحد جمع‌کننده و تفریق‌کننده می‌توان از یک سیگنال کنترلی مانند Sub استفاده نمود. با استفاده از این سیگنال کنترلی در صورت نیاز به انجام عمل تفریق بیت‌های مفروق به وسیله گیت XOR معکوس می‌شود و با در نظر گرفتن سیگنال Sub در روابط (۵) مقدار $v_i^2 v_i^1 V_i^0$ یک واحد افزایش می‌یابد. لازم به ذکر است برای پیاده سازی تفریق فقط تاخیر گیت XOR به مسیر بحرانی تاخیر افزوده می‌شود.

برای روشن‌تر شدن افراز چهار بیتی دو مثال در حالت‌های عدم رخ داد استثنا و رخ داد استثنا در ادامه مورد بررسی قرار می‌گیرد.

مثال ۱ (جمع دو عدد افزونه‌ای با استفاده از افزار چهار بیتی- حالت عدم رخ داد استثنا)

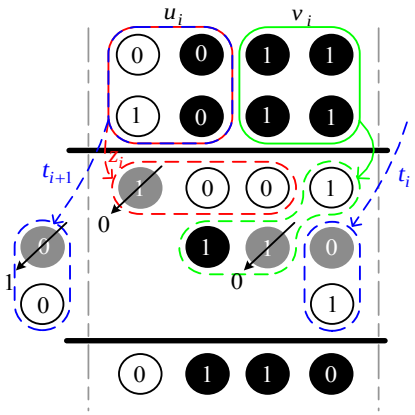
فرآیند جمع دو عدد $\bar{3}$ و 2 در شکل ۱۳ نشان داده شده است.دو عدد بر اساس افراز چهار بیتی، تقسیم بندی شده و پس از محاسبه بیت‌های $z_i^3 Z_i^2 Z_i^1$، $v_i^2 v_i^1 V_i^0$، $t_{i+1}^0 T_{i+1}^0$ و E_i و با توجه به عدم رخ داد استثنا جمع نهایی صورت می‌گیرد. در این مثال رقم نقلی ورودی صفر در نظر گرفته شده است. به دست می‌آید ($\bar{3} + 2 = \bar{1}$).



شکل ۱۳- عدم رخ داد استثنا در افراز چهار بیتی

مثال ۲ (جمع دو عدد افزونه‌ای با استفاده از افزار چهار بیتی- حالت رخ داد استثنا)

در این مثال، دو عدد $\bar{5}$ و 3 برای جمع انتخاب شده است. برخلاف مثال قبل برای جمع این دو عدد حالت استثنا رخ می‌دهد و قبل از جمع نهایی، همان‌طور که در شکل ۱۴ نشان داده شده است تصحیح صورت می‌پذیرد.



شکل ۱۴- رخ داد استثنا در افراز چهار بیتی

لازم به ذکر است در محاسبه بیت‌های t_{i+1}^0، z_i^3 و v_i^1، حالت استثنا لحاظ می‌گردد و این بیت‌ها به شکل صحیح بدست می‌آید، در شکل ۱۴ برای درک بهتر مقدار قبل و بعد از تصحیح نشان داده شده است.

۵- مقایسه و ارزیابی

در این بخش به ارزیابی الگوریتم پیشنهادی و مقایسه آن با الگوریتم‌های ارائه شده برای جمع دهدهی افزونه‌ای خواهیم پرداخت. با یک بررسی ساده و استفاده از تعداد گیت‌های تشکیل دهنده مسیر بحرانی تاخیر (با در نظر گرفتن پیاده‌سازی با گیت‌های دو ورودی Δ_g)، مشخص می‌شود که تاخیر این مدار $10\ \Delta_g$ است در حالی که بهترین طرح ارائه شده قبلی دارای تاخیر $12\ \Delta_g$ است. به منظور ارزیابی دقیق‌تر، در جدول ۲ کلیه روش‌های جمع افزونه‌ای دهدهی، شامل جمع‌کننده‌های افزونه‌ای دهدهی با ارقام علامت‌دار هم‌چنین جمع‌کننده‌هایی که در کاهش حاصل‌ضرب‌های جزئی در ضرب‌کننده‌های دهدهی مورد استفاده قرار گرفته‌اند (در واقع جمع‌کننده‌های افزونه‌ای با ارقام بدون علامت)، با استفاده از تلاش منطقی بررسی شده است. در این جدول برای طرح‌هایی که فاصله تاخیر آن‌ها نزدیک به‌هم است، این نتایج با در نظر گرفتن دو فرض حداقل انشعاب و انشعاب بدون محدودیت گزارش شده است.

جدول ۲- مقایسه تاخیر و مساحت جمع‌کننده‌های افزونه‌ای دهدهی

مرجع	تاخیر (FO4)	نسبت تاخیر	مساحت (NAND2)	نسبت مساحت
[۱۳]	8.56-9.96	1.13	184	1.82
[۱۵]	16.11	2.14	142	1.4
[۱۶]	13.56	1.8	122	1.21
[۱۷]	14.74	1.95	428	4.24
[۱۹]	7.86-9.20	1.04	103	1.02
[۲۰]	8.89-10.45	1.18	109	1.08
[۲۳]	9.00-12.17	1.19	194	1.92
[۲۴]	9.06-10.44	1.2	150	1.48
[۲۵]	8.87-9.98	1.18	186	1.84
طرح پیشنهادی	**7.54-9.05**	**1**	**101**	**1**

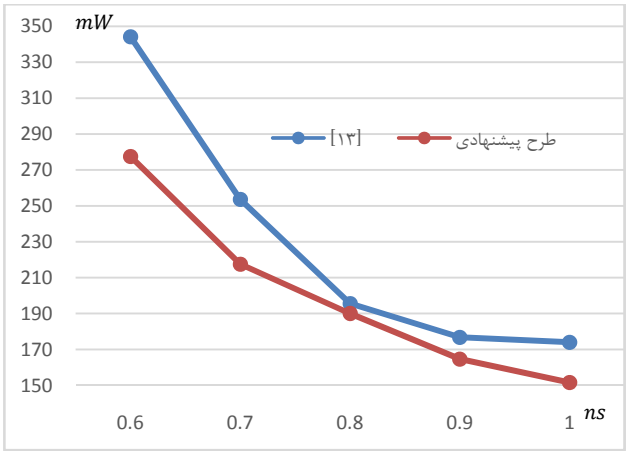
همان‌طور که مشاهده می‌شود طرح پیشنهادی در مقایسه با جمع‌کننده‌های ارقام علامت‌دار ([۱۳]، [۱۵-۱۷] و [۲۴، ۲۵])، در حدود ۱۳ درصد از سریع‌ترین جمع‌کننده تاخیر کم‌تری نشان می‌دهد. هم‌چنین این جمع‌کننده کم‌ترین مقدار مساحت را دارد (101 NAND2).

در جدول ۳، طرح‌های ذکر شده در جدول۲، با استفاده از نتایج سنتز مورد بررسی مجدد قرار گرفته و مقایسه کاملی از تاخیر، مساحت و توان مصرفی ارائه شده است. برای سنتز از کتابخانه استاندارد ۱۳۰ نانومتر TSMC استفاده شده است و شرایط برای تمامی طرح‌های یکسان در نظر گرفته شده است.

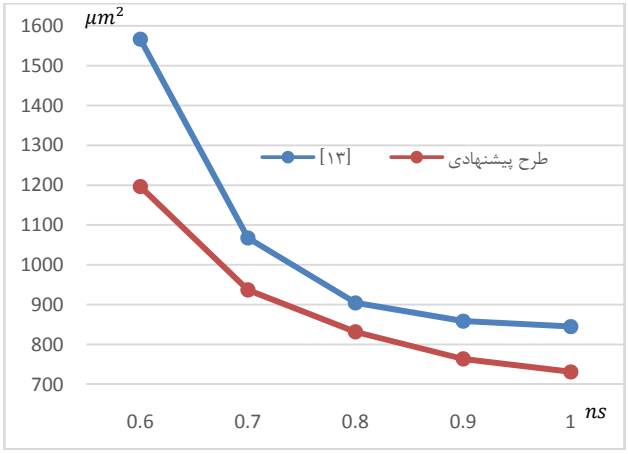
نتایج سنتز بدست آمده، موید تحلیل‌های اولیه و نتایج جدول ۲ است. در شکل‌های ۱۵ و ۱۶ نتایج سنتز طرح پیشنهادی و طرح [۱۳] برای توان مصرفی و مساحت در فاصله زمانی بین ۰/۶ تا ۱ نانو ثانیه نشان داده شده است. طرح پیشنهادی بدون افزایش تاخیر، مساحت و توان مصرفی کمتری دارد. با توجه به اینکه جمع‌کننده‌های افزونه‌ای، اجزای تشکیل دهنده واحدهای محاسباتی بزرگ‌تر مانند ضرب‌کننده‌ها هستند، حتی بهبودهای کوچک نیز تاثیر به‌سزایی در کارآیی کل واحد محاسباتی دارد.

جدول ۳- نتایج سنتز جمع‌کننده‌های افزونه‌ای دهدهی

مرجع	تاخیر (ns)	توان مصرفی پویا (mW)	توان مصرفی ایستا (pW)	مساحت (μm^2)
[۱۳]	0.69	1.17	37.24	1100
[۱۵]	1.16	2.13	70.98	1901
[۱۶]	0.87	1.65	44.34	1480
[۱۷]	1.10	4.04	95.79	3139
[۱۹]	0.63	1.22	32.31	1133
[۲۰]	0.75	1.48	37.87	1354
[۲۳]	0.77	1.54	45.78	1390
[۲۴]	0.77	1.52	46.77	1289
[۲۵]	0.70	3.16	40.19	1310
طرح پیشنهادی	**0.63**	**1.09**	**49.20**	**1095**



شکل ۱۵- مقایسه توان مصرفی



شکل ۱۶- مقایسه مساحت

۶- نتیجه‌گیری

در این مقاله، با توجه به اهمیت و نقش کلیدی عمل جمع، با به‌کارگیری نظام عددی نامتعارف افزونه‌ای، یک الگوریتم جمع افزونه‌ای دهدهی مبتنی بر افراز بیت‌های وزن‌دار ارائه گردید. در طرح پیشنهادی از خاصیت افزونگی ذاتی موجود در ارقام دهدهی استفاده شده و ارقام ورودی به شکل ارقام علامت‌دار متقارن و بازه [7,7−] است. با توجه به افراز چهار بیتی صورت گرفته، ورودی توابع داخلی جمع‌کننده به چهار محدود شده و امکان پیاده‌سازی با تعداد گیت‌های کم‌تر فراهم گردید. برای مقایسه، بررسی جامعی روی تمامی جمع‌کننده‌های افزونه‌ای دهدهی صورت پذیرفت و به دو شکل تحلیلی و مبتنی بر سنتز، طرح پیشنهادی مورد بررسی قرار گرفت. نتایج نشان دهنده کاهش توان مصرفی و مساحت بر تراشه بدون از دست دادن کارآیی می‌باشد.

مراجع

[1] M. F. Cowlishaw, and et. al., "Decimal Floating-Point Specification," *Proc. 15th IEEE Symposium on Computer Arithmetic*, pp. 147-154, Jun. 2001.

[2] IBM Corporation, "The 'telco' Benchmark," http://www2.hursley.ibm.com/decimal/telcoSpec.html, Mar. 2003.

[3] K. Quinn, "Ever Had Problems Rounding off Figures? This Stock Exchange Has," *Wall Street Journal*, Nov. 1983.

[4] M. F. Cowlishaw, "Decimal Arithmetic FAQ," http:/speleotrove.com/decimal/decifaq.

[5] A. Tsang, and M. Olschanowsky, "A Study of Database 2 Customer Queries," *IBM Technical report 03.413*, IBM, San Jose, CA, Apr. 1991.

[6] Institute of Electrical and Electronics Engineers, *IEEE Standard for Floating-Point Arithmetic, IEEE Std 754-2008*, Aug. 2008.

[7] F. Y. Busaba, and et. al., "The IBM z900 Decimal Arithmetic Unit," *Asilomar Conference on Signals, Systems, and Computers*, vol. 2, pp. 1335-1339, Nov. 2001.

[8] S. Shankland, "IBM's POWER6 Gets Help with Math, Multimedia," ZDNet News, Oct. 2006.

[9] C. F. Webb, "IBM z10: The Next-Generation Mainframe Microprocessor," *IEEE Micro*, vol. 28, Issue 2, pp. 19-29, 2008.

[10] R. K. Richards, "Arithmetic Operations in Digital Computers," Van Nostrand, New York, 1955.

[11] B. Parhami, *Computer Arithmetic: Algorithms and Hardware Designs*, Oxford, 2nd ed., 2010.

[12] B. Parhami, "Generalized Signed-Digit Number Systems: A Unifying Framework for Redundant Number Representations," *IEEE Transactions on Computers*, vol. 39, pp. 89–98, Jan. 1990.

[13] S. Gorgin, and G. Jaberipur, "Fully Redundant Decimal Arithmetic," *Proceedings of the 19th IEEE Symposium on Computer Arithmetic, Portland, USA*, pp. 145–152, Jun. 2009.

[14] M. A. Erle, E. M. Schwartz, and M. J. Schulte, "Decimal Multiplication with Efficient Partial Product Generation," *17th IEEE Symposium on Computer Arithmetic*, pp. 21-28, Jun. 2005.

[15] A. Svoboda, "Decimal Adder with Signed Digit Arithmetic," *IEEE Transactions on Computers*, vol. C-18, no. 3, pp. 212-215, Mar. 1969.

[16] B. Shirazi, D. Y. Yun, and C. N. Zhang, "RBCD: Redundant Binary Coded Decimal Adder," *IEE Proceedings Computer & Digital Techniques (CDT)*, vol. 36, no. 2, Mar. 1989.

[17] H. Nikmehr, B. J. Phillips, and C. C. Lim, "A Decimal Carry-free Adder," *Proc. SPIE Conf. Smart Mater., Nano-, Micro-Smart Syst.*, pp. 786–797, Dec. 2004.

[18] R. D. Kenney, M. J. Schulte, and M. A. Erle, "A High-frequency Decimal Multiplier," *IEEE Int. Conf. on computer Design: VLSI in Computers and Processors* (ICCD), pp. 26-29, Oct. 2004.

[19] S. Gorgin, and G. Jaberipur, "A Fully Redundant Decimal Adder and Its Application in Parallel Decimal Multipliers," *Microelectronics Journal*, vol. 40, Issue 10, pp. 1471-1481, Oct. 2009.

[20] M. A. Erle, and M. J. Schulte, "Decimal Multiplication via Carry-save Addition," *Conference on Application-Specific Systems, Architectures, and Processors*, pp. 348-358, Jun. 2003.

[21] T. Lang, and A. Nannarelli, "A Radix-10 Combinational Multiplier," *Proc. 40th Asilomar Conference on Signals, Systems, and Computers*, pp. 313-317, Nov. 2006.

[22] A. Vazquez, E. Antelo, and P. Montuschi, "A New Family of High-Performance Parallel Decimal Multipliers," *Proc. 18th IEEE Symposium on Computer Arithmetic*, pp. 195-204, Jun. 2007.

[23] I. D. Castellanos, and J. E. Stine, "Compressor Trees for Decimal Partial Product Reduction," *ACM Great Lakes Symposium on VLSI*, pp. 107-110, May 2008.

[24] A. Kaivani, and G. Jaberipur, "Fully Redundant Decimal Addition and Subtraction Using Stored-unibit encoding," *Integration the VLSI journal*, 2009.

[25] A. Kaivani, and S-B. Ko, "Decimal Signed Digit Addition Using Stored Transfer encoding," *26th Annual IEEE Canadian Conference on Electrical and Computer Engineering* (CCECE), pp. 1-4, May 2013.

[26] J. Moskal, E. Oruklu, and J. Saniie, "Design and Synthesis of a Carry-Free Signed-Digit Decimal Adder," *Proc. IEEE Int. Symposium on Circuits and Systems* (ISCAS'07), pp. 1089-1092, May 2007.

[27] G. Jaberipur, and B. Parhami, "Posibits, Negabits, and Their Mixed Use in Efficient Realization of Arithmetic Algorithms," *15th Symposium on Computer Architecture and Digital Systems*, pp. 3-9, Sep. 2010.

سعید گرگین مدرک کارشناسی و کارشناسی‌ارشد خود را در سال‌های ۱۳۸۱ و ۱۳۸۴ از دانشگاه آزاد اسلامی اخذ نمود و در سال ۱۳۸۹ موفق به اخذ مدرک دکتری از دانشگاه شهید بهشتی گردید. در حال حاضر او استادیار پژوهشکده برق و فناوری اطلاعات سازمان پژوهش‌های عملی و صنعتی ایران است. همچنین به عنوان محقق در پژوهشکده کامپیوتر پژوهشگاه دانش‌های بنیادی مشغول به فعالیت می‌باشد. در کنار موضوعات تحقیقات کاربردی در حوزه فناوری اطلاعات، سایر زمینه‌های مورد علاقه او عبارتند از: حساب کامپیوتری، طراحی VLSI، سیستم‌های پردازش سریع و پردازش موازی.

آدرس پست‌الکترونیکی ایشان عبارت است از:

gorgin@irost.org

لیلی میرمقتدایی مدرک کارشناسی خود را از دانشگاه آزاد اسلامی در سال ۱۳۸۷ و کارشناسی‌ارشد خود را از دانشگاه شهید بهشتی در سال ۱۳۹۵ اخذ نموده است. تحقیقات کارشناسی‌ارشد او در حوزه حساب کامپیوتری متمرکز بوده و موضوع پایان‌نامه او طراحی و پیاده‌سازی جمع‌کننده‌های ممیزشناور افزونه‌ای دهدهی است. سایر موضوعات مورد علاقه او سخت‌افزارهای با قابلیت پیکربندی مجدد، سیستم‌های نهفته و تراشه‌های زیستی است.

آدرس پست‌الکترونیکی ایشان عبارت است از:

leileemirmoghtadaei@gmail.com

اطلاعات بررسی مقاله:

تاریخ ارسال: ۱۳۹۵/۰۳/۰۳

تاریخ اصلاح: ۱۳۹۵/۰۵/۱۰

تاریخ قبول شدن: ۱۳۹۵/۰۶/۲۳

نویسنده مرتبط: دکتر سعید گرگین، پژوهشکده برق و کامپیوتر، سازمان پژوهش‌های علمی و صنعتی ایران، تهران، ایران.

[1] Redundant BCD
[2] Decimal Septa Signed Digit (DSSD)
[3] Negabit
[4] Posibit