بررسی تأثیر ریزدانگی در طراحی واحدهای قابل بازپیکربندی حساب دهدهی

سمانه امامی و مهدی صدیقی

چکیده: امروزه استفاده از حساب دهدهی در بسیاری از کاربردها نظیر برنامه‌های مالی مورد توجه قرار گرفته است. شاید بتوان مهم‌ترین عامل تأثیرگذار در این زمینه را نیاز به دقت بیشتر در نمایش اعداد در این کاربردها دانست. در دهه‌های اخیر، مطالعات گسترده‌ای در زمینه طراحی سخت‌افزارهای حسابی دهدهی انجام گرفته که اغلب آنها طراحی‌های خاص‌منظوره و برخی نیز با در نظر گرفتن قابلیت بازپیکربندی بوده‌اند. اما تاکنون تحقیقی بر روی پارامترهای تأثیرگذار در طراحی سخت‌افزارهای قابل بازپیکربندی دهدهی از جمله سطح ریزدانگی و انعطاف‌پذیری آنها انجام نشده است. بررسی‌های ما نشان می‌دهد برای جمع‌کننده‌های دهدهی قابل بازپیکربندی، ریزدانگی سطح بیت مناسب نبوده و بهتر است برای این واحدها، ریزدانگی را در سطح یک یا حتی چند رقم دهدهی افزایش داد. این افزایش سطح ریزدانگی، منجر به حدود ۱۲٪ بهبود در مساحت و ۱۳/٤٪ بهبود در توان مصرفی شده است. اما بر خلاف جمع‌کننده‌ها با افزایش ریزدانگی در واحدهای مورد نیاز ضرب‌کننده‌های دهدهی، مساحت و توان مصرفی حدود ۷۵٪ افزایش می‌یابد.

کلیدواژه: **حساب کامپیوتری، سخت‌افزارهای دهدهی، واحدهای قابل بازپیکربندی، ریزدانگی.**

۱- مقدمه

اغلب کامپیوترهای امروزی با کاربردهای رایج، بر مبنای حساب دودویی طراحی شده‌اند که این امر ناشی از سادگی سخت‌افزارهای دودویی و سرعت بالای مدارهای آن می‌باشد. اما کامپیوترهای اولیه مانند ENIAC [۱]، UNIVAC [۲] و IBM۶۵۰ [۳] از حساب دهدهی استفاده می‌کردند. در سال‌های اخیر نیز، تحقیقات گسترده‌ای بر روی طراحی سخت‌افزارهای حسابی دهدهی برای کامپیوترهای خاص انجام شده است.

یکی از مشکلات اصلی حساب دودویی که باعث افزایش تمایل به استفاده از حساب دهدهی گردیده است، عدم دقت آن در نمایش برخی از اعداد اعشاری مانند ۰/۱ می‌باشد [۴]. پایین‌بودن دقت نمایش این اعداد، موجب بروز خطای قابل ملاحظه‌ای در کاربردهای مالی و تجاری می‌گردد. به عنوان مثال، بررسی انجام‌شده در یک سیستم بزرگ پرداخت صورت‌حساب تلفن نشان داده که استفاده از حساب دودویی به جای حساب دهدهی، در یک سال به طور تقریبی حدود ۵ ملیون دلار خسارت وارد می‌نماید [۵]. از سوی دیگر، بشر همواره به دلیل دارابودن ده انگشت، استفاده از مبنای ۱۰ را در محاسبات خود ترجیح داده است. با توجه به دلایل بیان‌شده، امروزه گرایش به استفاده از حساب دهدهی در کامپیوترها در حال افزایش است تا جایی که مؤسسه استاندارد IEEE در نسخه استاندارد IEEE ۷۵۴-۲۰۰۸ برای حساب ممیز شناور، به تشریح جنبه‌های مختلف حساب ممیز شناور دهدهی شامل شیوه نمایش، عملیات حسابی، حالت‌های مختلف گردکردن و رسیدگی به استثناها پرداخته است [۶].

به طور کلی، پیاده‌سازی حساب دهدهی به دو روش نرم‌افزاری و سخت‌افزاری امکان‌پذیر می‌باشد. منظور از پیاده‌سازی نرم‌افزاری، استفاده از کتابخانه‌های نرم‌افزاری قابل اجرا بر روی سخت‌افزارهای دودویی مانند IBM decNumber [۷]، Intel DFP[1] Math [۸] و انواع ممیز شناور دهدهی از پیش تعریف شده[2] در GCC[3] [۹] است. در مقابل، پیاده‌سازی سخت‌افزاری با وجود واحدهای دهدهی در پردازنده ماشین‌هایی مانند IBM System z۹ [۱۰]، IBM System z۱۰ [۱۱] و IBM z۱۹۶ [۱۲] پشتیبانی می‌شود. بررسی‌ها نشان می‌دهد که پیاده‌سازی نرم‌افزاری عملیات DFP بسیار کندتر از معادل سخت‌افزاری آنها می‌باشد [۱۳] و [۱۴]. بنابراین با پیشرفت فناوری و افزایش قابلیت یکپارچه‌سازی تعداد بسیار زیادی ترانزیستور در یک تراشه، سازندگان تراشه‌ها به طراحی و پیاده‌سازی سخت‌افزاری عملیات DFP تمایل بیشتری پیدا کرده‌اند تا بتوانند کارایی کاربردهای مالی و تجاری را افزایش دهند.

با توجه به پدیدآمدن تجهیزات منطقی قابل بازپیکربندی، طراحی و پیاده‌سازی سیستم‌های دیجیتال، بدون نیاز به طی‌کردن مراحل مربوط به ساخت فیزیکی تراشه‌های سفارشی ممکن گردیده است. این سخت‌افزارها در واقع پوشاننده شکاف میان پیاده‌سازی نرم‌افزاری و سخت‌افزاری هستند، زیرا از یک سو کارایی بسیار بیشتری نسبت به پیاده‌سازی نرم‌افزاری داشته و از سوی دیگر، انعطاف‌پذیری بالاتری نسبت به سخت‌افزارهای خاص‌منظوره دارند [۱۵] و [۱۶]. به علاوه با به وجود آمدن کاربردهای مختلف و بالابودن هزینه ساخت سخت‌افزارهای خاص‌منظوره، تمایل به استفاده از سخت‌افزارهای قابل بازپیکربندی افزایش یافته است. این نوع سخت‌افزارها، نه تنها برای کاربردهای نمونه‌سازی[4] بلکه برای تولید محدود محصول نهایی نیز بسیار مناسب و مقرون به صرفه می‌باشند.

در حوزه حساب دهدهی نیز کارهایی در رابطه با استفاده از سخت‌افزارهای قابل بازپیکربندی و به ویژه FPGA[5]ها برای پیاده‌سازی مدارهای دهدهی انجام گرفته که در بخش کارهای گذشته به مرور آنها

این مقاله در تاریخ ۱۴ بهمن ماه ۱۳۹۴ دریافت و در تاریخ ۱۸ مهر ماه ۱۳۹۵ بازنگری شد.

سمانه امامی، دانشکده مهندسی کامپیوتر و فناوری اطلاعات، دانشگاه صنعتی امیرکبیر، تهران، (email: s.emami@aut.ac.ir).

مهدی صدیقی، دانشکده مهندسی کامپیوتر و فناوری اطلاعات، دانشگاه صنعتی امیرکبیر، تهران، (email: msedighi@aut.ac.ir).

1. Decimal Floating-Point
2. Built-in
3. GNU Compiler Collection
4. Prototyping
5. Field-Programmable Gate Array

پرداخته خواهد شد. اما ریزدانگی[1] FPGAها پایین و در سطح بیت بوده و هزینه بازپیکربندی آنها برای مدارهای پیچیده مانند حساب دهدهی، بسیار زیاد می‌باشد. با توجه به این که در طراحی SOC[2]ها، هدف از یک سو به دست آوردن کارایی نزدیک به ASIC[3] و از سوی دیگر انعطاف‌پذیری، زمان طراحی و هزینه پایین مشابه FPGA می‌باشد، معماری‌های قابل بازپیکربندی درشت‌دانه (CGRAs)[4] راه حل مناسبی برای نیل به این اهداف خواهند بود. زیرا این دسته از معماری‌ها، کارایی محاسباتی بهتری نسبت به FPGAها و انعطاف‌پذیری بیشتر و ساخت ساده‌تری نسبت به ASICها دارند. اما نکته مهم در طراحی این ساختارها، توجه به میزان انعطاف‌پذیری و سطح ریزدانگی واحدهای تشکیل‌دهنده آن می‌باشد. از این رو در این مقاله، چندین طراحی مختلف برای واحدهای پایه‌ای حساب دهدهی ارائه گردیده و تأثیر پارامترهای بیان‌شده در طراحی سخت‌افزارهای قابل بازپیکربندی دهدهی درشت‌دانه بررسی شده است. نتایج به دست آمده نشان می‌دهد انتخاب نوع طراحی، بستگی مستقیمی به هدف و محدودیت‌های موجود داشته و بالابردن ریزدانگی یا انعطاف‌پذیری طراحی، لزوماً بر تمام ویژگی‌های آن مانند تأخیر و مساحت، تأثیر یکسانی نخواهد داشت.

ساختار مقاله به این شرح است که در بخش دوم، به مرور کارهای گذشته در زمینه حساب دهدهی و استفاده از قابلیت بازپیکربندی در این کاربرد پرداخته خواهد شد. در بخش سوم، واحدهای مورد نیاز برای عملیات پایه دهدهی مانند جمع و ضرب مورد بررسی قرار گرفته و طراحی‌های مختلف قابل بازپیکربندی از نظر سطح ریزدانگی و میزان انعطاف‌پذیری برای این واحدها ارائه می‌گردد. بخش چهارم به مقایسه و ارزیابی طراحی‌های انجام‌شده پرداخته و در نهایت در بخش پنجم نتیجه‌گیری کار ارائه خواهد شد.

۲- مروری بر کارهای گذشته

با وجود این که تحقیقات گسترده‌ای در زمینه پیاده‌سازی سخت‌افزاری حساب دهدهی و طراحی مدارهای خاص‌منظوره (ASIC) برای عملیات حسابی مختلف دهدهی به ویژه عملیات پایه مانند جمع، تفریق و ضرب به عمل آمده است (از جمله [۱۷] تا [۲۰]) اما تحقیقات انجام‌شده در زمینه به کارگیری و استفاده از قابلیت بازپیکربندی در مدارهای حسابی دهدهی بسیار محدود می‌باشد. با این وجود، مطالعات صورت‌گرفته در این حوزه به چند دسته تقسیم می‌شوند:

دسته اول شامل کارهای انجام‌شده در زمینه طراحی مدارهای دهدهی با عملوندها و عملگرهای مختلف می‌باشد. برخی از کارهایی که در این دسته قرار می‌گیرند به طراحی مدارهای حسابی ترکیبی دودویی و دهدهی پرداخته‌اند. به عنوان مثال در [۲۱] یک واحد جمع‌کننده/ تفریق‌کننده ارائه شده که می‌تواند ورودی‌های دودویی و دهدهی بپذیرد. این واحد عملیاتی، توانایی انجام محاسبات بر روی نمایش‌های unsigned، signed magnitude و قالب‌های مختلف مکمل را دارا می‌باشد. از کارهای مشابه در این زمینه می‌توان به [۲۲] تا [۲۶] اشاره نمود.

برخی دیگر از تحقیقات انجام‌شده در حوزه دهدهی، قابلیت بازپیکربندی را در امکان انجام عملیات بر روی ورودی‌هایی با اندازه‌های مختلف تعریف نموده‌اند. به عنوان نمونه در [۲۷] یک جمع‌کننده موازی پیشوندی Ling ارائه شده که می‌تواند عملیات جمع را برای ورودی‌هایی با اندازه ۴، ۸ یا ۱۶ بیت انجام دهد.

دسته دیگر، شامل کارهای انجام‌شده در زمینه پیاده‌سازی مناسب مدارهای دهدهی بر روی FPGAهای موجود می‌باشد. به عنوان مثال [۲۸] نشان داده که FPGAها می‌توانند برای پیاده‌سازی کارای بعضی پردازنده‌های با کاربرد خاص (مانند کاربردهای حساب دهدهی) به کار روند. در این راستا، محک[5] telco [۵] مورد مطالعه قرار گرفته و بهبود کارایی نزدیک به ده برابری گزارش شده است. در [۲۹] روشی برای پیاده‌سازی جمع دهدهی چند عملوندی بر روی Xilinx Virtex-۵/۶ [۳۰] ارائه شده که از نظر کارایی با جمع‌کننده‌های درختی دودویی برابری می‌کند. در [۳۱] نیز یک ضرب‌کننده دهدهی موازی پیشنهاد گردیده که بر روی FPGAها پیاده‌سازی می‌شود و می‌تواند در یک واحد MAC نیز مورد استفاده قرار گیرد. از کارهای مشابه در این زمینه می‌توان [۳۲] تا [۳۴] را نام برد.

در برخی از پژوهش‌های مرتبط با پیاده‌سازی مدارهای دهدهی بر روی ساختارهای قابل بازپیکربندی موجود مانند FPGAها، ایده استفاده از بسترهای خاص‌منظوره مانند زنجیره‌های نقلی نیز مطرح شده و تأثیر آنها در بهبود کارایی مدارها گزارش شده است [۲۹] و [۳۱]. ارائه چنین ساختارهایی نشان‌دهنده اهمیت در نظر گرفتن کاربرد در طراحی ساختارهای قابل بازپیکربندی می‌باشد.

همان گونه که اشاره گردید از یک سو

- تحقیقات انجام‌گرفته در زمینه به کارگیری قابلیت بازپیکربندی در مدارهای حسابی دهدهی، محدود به مطالعات موردی می‌باشد.
- جهت‌گیری ساختارهای قابل بازپیکربندی در صنعت، در راستای ایجاد بسترهای خاص‌منظوره در طراحی (مانند زنجیره نقلی) است.
- ریزدانگی FPGAها پایین و هزینه بازپیکربندی آنها برای مدارهای دهدهی بسیار زیاد می‌باشد.
- یک بررسی کل‌نگرانه در مورد امکان و چگونگی افزودن قابلیت بازپیکربندی به مدارهای دهدهی وجود ندارد.

و از سوی دیگر برای ارائه یک ساختار قابل بازپیکربندی درشت‌دانه، نکته قابل توجه، میزان ریز یا درشت‌دانگی واحدهای تشکیل‌دهنده آن می‌باشد. زیرا افزایش ریزدانگی، به معنی حرکت به سوی طراحی با انعطاف‌پذیری کمتر و کارایی بالاتر می‌باشد. از این رو در این مقاله به بررسی این موضوع برای عملیات پایه دهدهی (جمع، تفریق و ضرب) پرداخته خواهد شد.

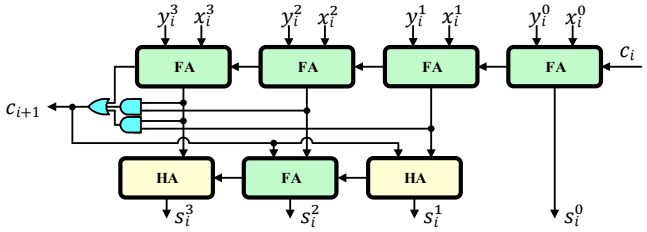
۳- سخت‌افزارهای حسابی دهدهی قابل بازپیکربندی

در این بخش، واحدهای قابل بازپیکربندی مورد نیاز برای عملیات حسابی پایه یعنی جمع، تفریق و ضرب دهدهی با میزان ریزدانگی‌های متفاوت، طراحی و مقایسه خواهند شد.

۳-۱ جمع و تفریق دهدهی

مهم‌ترین و اساسی‌ترین عملگر در عملیات حسابی، جمع می‌باشد و به طور کلی جمع‌کننده‌ها را می‌توان به دو دسته جمع‌کننده‌های انتشاری

1. Granularity
2. System-On-Chip
3. Application-Specific Integrated Circuit
4. Coarse-Grain Reconfigurable Architectures
5. Benchmark



شکل ۱: یک جمع‌کننده تک‌رقمی BCD (پیاده‌سازی سطح بیت).

(مانند CRA[1]، CSKA[2] و CSLA[3]) و جمع‌کننده‌های موازی (مانند L-F[4]، B-K[5] و H-C[6]) تقسیم نمود. در جمع‌کننده‌های انتشاری دهدهی، دهدهی، کوچک‌ترین واحد تشکیل‌دهنده، یک جمع‌کننده تک‌رقمی می‌باشد.

شکل ۱ پیاده‌سازی یک جمع‌کننده تک‌رقمی BCD[7] را با استفاده از جمع‌کننده‌های دودویی و دروازه‌های[8] منطقی پایه نمایش می‌دهد.

هر چند می‌توان با کنار هم قرار دادن تعداد دلخواهی از این جمع‌کننده‌های تک‌رقمی، یک CRA دهدهی تشکیل داد. اما در مورد جمع‌کننده‌های انتشاری دیگر مانند CSKA و CSLA مسئله کمی متفاوت است. در این جمع‌کننده‌ها به دلیل تعریف سیگنال‌های تولید و انتشار رقم نقلی[9] و همچنین ماهیت بلاکی‌بودن آنها، اندازه واحدهای سازنده جمع‌کننده، بزرگ‌تر شده و ریزدانگی طراحی افزایش می‌یابد. اگر $X=\sum_{i=0}^{k-1} X_i\times 10^i$ و $Y=\sum_{i=0}^{k-1} Y_i\times 10^i$ دو عدد k رقمی BCD باشند، سیگنال‌های تولید (G_i) و انتشار (P_i) رقم نقلی برای هر رقم BCD از (۱) به دست می‌آیند که در این معادلات α_i^j نشان‌دهنده بیت j ام از i امین رقم BCD می‌باشد

$$\begin{aligned} p_i^j &= x_i^j + y_i^j g_i^j = x_i^j y_i^j \\ P_i &= p_i^3 p_i^2 + g_i^2 p_i^2 + p_i^2 g_i^1 p_i^0 \\ G_i &= g_i^3 + p_i^3 p_i^2 + p_i^3 p_i^1 + p_i^3 g_i^0 + g_i^2 p_i^1 + g_i^2 g_i^0 + p_i^2 g_i^1 g_i^0 \end{aligned} \qquad (۱)$$

تفریق دهدهی نیز از جمع‌کردن مکمل ۱۰ عامل دوم تفریق با عامل اول آن به دست می‌آید. مکمل ۱۰ عدد Y که با Y^{10c} نمایش داده می‌شود، از مکمل‌کردن بیت‌های حاصل جمع هر رقم دهدهی به اضافه ۶ (به پیمانه ۱۶) و اضافه‌کردن یک واحد به صورت رقم نقلی ورودی جمع حاصل می‌شود، یعنی

$$\begin{aligned} Y_i^{10c} &= Y_i^{9c} + 1 \\ Y_i^{9c} &= 9 - Y_i = 15 - (Y_i + 6) = \overline{Y_i + 6} \end{aligned} \qquad (۲)$$

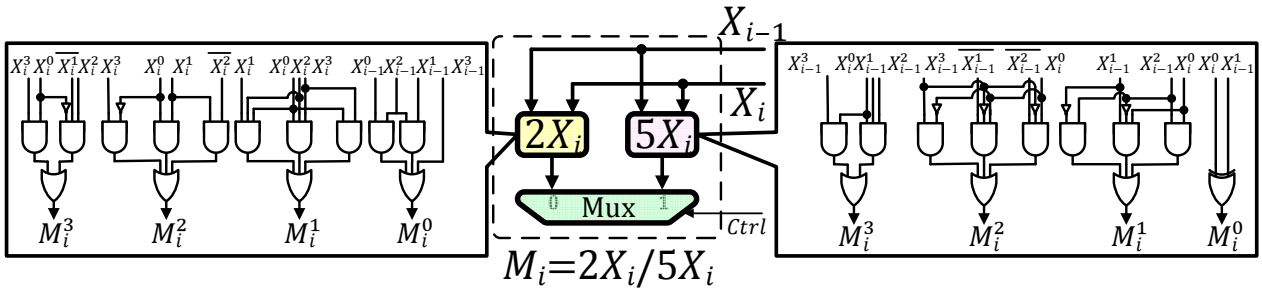
با توجه به مطالب بیان‌شده حداقل ریزدانگی در یک جمع‌کننده دهدهی، یک رقم BCD است. اما برای ساختن یک جمع‌کننده قابل بازپیکربندی که بتواند CSKA و CSLA را نیز پیاده‌سازی نماید، به دلیل ماهیت بلاکی‌بودن این جمع‌کننده‌ها، اندازه واحدهای سازنده جمع‌کننده بزرگ‌تر شده و ریزدانگی طراحی افزایش می‌یابد. در این صورت با داشتن چند دروازه منطقی پایه (برای محاسبه سیگنال skip یا مکمل ۱۰ یک عدد) و قراردادن اتصالات قابل بازپیکربندی (شامل multiplexerها) بین بلاک‌های با اندازه مناسب، می‌توان هر یک از جمع‌کننده‌های بیان‌شده را پیاده‌سازی نمود.

در CSKA دودویی اگر اندازه بلاک‌های جمع‌کننده ثابت در نظر گرفته شود، اندازه مناسب برای آنها تقریباً برابر $\sqrt{n/2}$ است که n تعداد بیت‌های عملوندهای ورودی جمع می‌باشد [۳۵]. حال با توجه به این که استاندارد IEEE برای عملوندهای دهدهی، ۱۶ رقم (۶۴ بیت) است [۶]، باید دید اندازه مناسب برای بلاک‌های یک جمع‌کننده ۱۶رقمی CSKA دهدهی چقدر است. اگر فرمول $\sqrt{n/2}$ مربوط به جمع‌کننده دودویی را برای $n=16$ رقم دهدهی نیز در نظر بگیریم، اندازه مناسب بلوک‌ها $\sqrt{8}$ خواهد بود اما اگر به جای تعداد رقم‌ها، تعداد بیت‌های عملوندهای ورودی جمع در نظر گرفته شود، برای $n=64$ بیت ورودی، اندازه بلاک‌ها $\sqrt{32}$ تخمین زده می‌شود. هر چند در این حالت، باید به مضرب ۴ بودن اندازه بلاک‌ها دقت شود، زیرا نمی‌توان یک رقم BCD را به دو نیم تقسیم کرد. از سوی دیگر به نظر می‌رسد بلاک‌های با اندازه ۴ رقم دهدهی نیز به دلیل تشکیل ساختاری منظم در جمع‌کننده ۱۶ رقمی، گزینه مناسبی برای طراحی این جمع‌کننده باشند. اما در طراحی این جمع‌کننده‌ها بهترین نتیجه (از نظر تأخیر) زمانی حاصل می‌شود که اندازه بلاک‌ها یکسان نبوده و روندی افزایشی- کاهشی داشته باشد [۳۵]. بنابراین انتظار می‌رود یک جمع‌کننده ۱۶ رقمی دهدهی که از بلاک‌های با اندازه‌های ۱، ۲، ۳، ۴، ۳، ۲، ۱ تشکیل می‌گردد، تأخیر کمتری نسبت به هر یک از جمع‌کننده‌های با بلاک‌های هم‌اندازه داشته باشد. از این رو طراحی‌های مختلف برای جمع‌کننده CSKA دهدهی با اندازه بلاک‌های ۲، ۳ یا ۴ رقم دهدهی انجام شده و در بخش مقایسه و ارزیابی، به تحلیل دقیق‌تر آنها پرداخته خواهد شد.

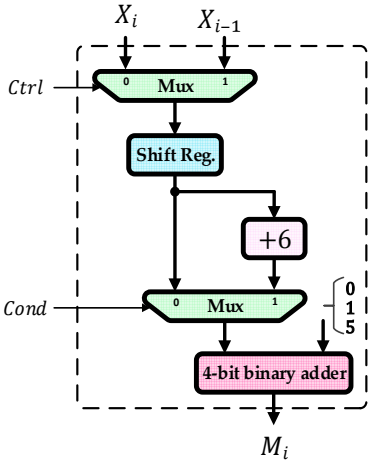
۳-۲ ضرب دهدهی

ضرب دهدهی می‌تواند به یکی از سه روش ضرب سریالی، آرایه‌ای یا موازی انجام شود [۳۶] تا [۳۸]. در تمامی این روش‌ها، عملیات ضرب شامل سه مرحله تولید حاصل‌ضرب‌های جزئی (PPG)[10]، کاهش حاصل‌ضرب‌های جزئی (PPR)[11] و جمع نهایی[12] می‌باشد. در مراحل دوم و سوم با توجه به روش پیاده‌سازی ضرب دهدهی از یکی از انواع جمع‌کننده‌های دهدهی استفاده می‌شود. اما برای تولید حاصل‌ضرب‌های جزئی، در طراحی‌های ASIC موجود، راه حل‌های مختلفی پیشنهاد شده است. یکی از راه‌های ممکن، استفاده از یک واحد ترکیبی خاص است که دو رقم BCD (۸ بیت) را به عنوان ورودی دریافت کرده و حاصل‌ضرب آنها را در ۸ بیت به عنوان خروجی ارائه دهد [۳۹]. البته این روش به دلیل استفاده از مداری خاص‌منظوره با پیچیدگی بالا، حتی در طراحی‌های ASIC نیز کمتر مورد استفاده قرار می‌گیرد. بنابراین به کار بردن آن در یک سخت‌افزار قابل بازپیکربندی با توجه به انعطاف‌پذیر نبودن آن برای استفاده در کاربردهای دیگر، منطقی به نظر نمی‌رسد. راه حل دیگر برای تولید حاصل‌ضرب‌های جزئی، استفاده از جدول‌های جستجو (LUT)[13] با ۸ ورودی می‌باشد که بتوانند جدول ضرب ۱۰×۱۰ را پیاده‌سازی نمایند [۴۰]. هر چند جدول‌های جستجو انعطاف‌پذیری بالایی در پیاده‌سازی

1. Carry Ripple Adder
2. Carry Skip Adder
3. Carry Lookahead Adder
4. Ladner-Fischer
5. Brent-Kung
6. Han-Carlson
7. Binary Coded Decimal
8. Gates
9. Carry Propagate and Generate Signals
10. Partial Product Generation
11. Partial Product Reduction
12. Final Addition
13. Look Up Table



شکل ۲: واحد قابل بازپیکربندی تولید مضارب ۲ و ۵ یک عدد دهدهی (پیاده‌سازی سطح بیت).



شکل ۳: واحد قابل بازپیکربندی تولید مضارب ۲ و ۵ یک عدد دهدهی (پیاده‌سازی سطح یک رقم).

مدارهای مختلف داشته و ممکن است انتخاب خوبی برای استفاده در سخت‌افزارهای قابل بازپیکربندی به نظر برسد، اما جدول جستجوی با ۸ ورودی، مساحت و توان زیادی مصرف کرده و به همین دلیل گزینه مناسبی نمی‌باشد. اما رایج‌ترین روش موجود برای تولید PP[1]ها، استفاده از مضارب آسان[2] می‌باشد به این معنی که در ابتدای انجام عملیات ضرب، برخی از مضارب عملوند اول ضرب (که مجموعه مضارب آسان نامیده می‌شود)، تولید شده و بقیه مضارب مورد نیاز حین عمل ضرب با استفاده از این مجموعه به دست می‌آیند. در میان مجموعه‌های مختلف ارائه‌شده برای مضارب آسان، مضارب ۲ و ۵ یک عدد دهدهی به دلیل تولید سریع و بدون انتشار رقم نقلی[3] همواره مورد توجه قرار داشته‌اند. این این بدان معنی است که در این مضارب، حاصل‌ضرب تولیدشده برای موقعیت i، تنها به رقم iام و $i-1$ام عدد دهدهی وابسته است [۳۶]. این ویژگی، مستقل از نوع طراحی انجام‌شده (طراحی ASIC یا قابل بازپیکربندی) است و به همین دلیل به نظر می‌رسد وجود یک واحد ترکیبی برای تولید مضارب ۲ و ۵ در سخت‌افزار قابل بازپیکربندی حسابی دهدهی به بالارفتن کارایی آن کمک کند. بنابراین در این بخش به ارائه طراحی‌های مختلف این واحد ترکیبی و مقایسه آنها خواهیم پرداخت.

فرض کنید $X=X_{n-1}X_{n-2}\ldots X_1X_0$ یک عدد nرقمی دهدهی BCD بوده و هدف، به دست آوردن عدد $n+1$ رقمی M می‌باشد به طوری که

$$M=\begin{cases}2X & \text{if } ctrl=0\\ 5X & \text{if } ctrl=1\end{cases} \qquad (۳)$$

این واحد می‌تواند در پایین‌ترین سطح منطقی با استفاده از دروازه‌های منطقی پایه پیاده‌سازی گردد. معادلات (۴) چگونگی این پیاده‌سازی را نمایش می‌دهند. در این معادلات α_i^j نشان‌دهنده بیت jام از iامین رقم BCD می‌باشد. با وجود این که این طراحی بدیهی‌ترین و ساده‌ترین سطح قابلیت بازپیکربندی را نشان می‌دهد، اما استفاده از آن در مواردی که کارایی بالاتر و تأخیر کمتر در طراحی دارای اهمیت می‌باشد، مفید خواهد بود. شکل ۲ پیاده‌سازی واحد قابل بازپیکربندی توصیف شده را نشان می‌دهد

$$\begin{aligned}
M_i^0 &= \overline{ctrl}[X_{i-1}^3+X_{i-1}^0X_{i-1}^2+X_{i-1}^1X_{i-1}^2]+\\
&\quad ctrl[X_{i-1}^1\overline{X_i^0}+\overline{X_{i-1}^1}X_i^0]\\
M_i^1 &= \overline{ctrl}[\overline{X_i^0}X_i^2+X_i^0\overline{X_i^2}\,\overline{X_i^3}+\overline{X_i^0}X_i^1X_i^2]\\
M_i^2 &= \overline{ctrl}[X_i^1\overline{X_i^2}+X_i^0X_i^1+\overline{X_i^0}X_i^2]+\\
&\quad ctrl[X_{i-1}^3\overline{X_i^0}+X_i^0\overline{X_{i-1}^1}\,\overline{X_{i-1}^3}+X_i^0\overline{X_{i-1}^2}\,\overline{X_{i-1}^3}]\\
M_i^3 &= \overline{ctrl}[X_i^0X_i^2+\overline{X_i^0}X_i^1X_i^2]+\\
&\quad ctrl[X_{i-1}^3X_i^0+X_{i-1}^1X_{i-1}^2X_i^0]
\end{aligned} \qquad (۴)$$

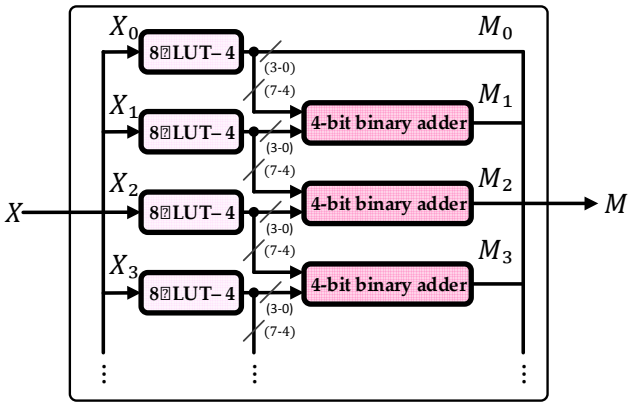
همان طور که در معادلات و شکل فوق مشاهده می‌شود، پیاده‌سازی سطح بیت دارای جزئیات زیاد و انعطاف‌پذیری پایینی بوده و به همین دلیل برای طراحی واحدهای قابل بازپیکربندی مناسب به نظر نمی‌رسد. بنابراین سطح ریزدانگی افزایش یافته و پیاده‌سازی در سطح یک رقم دهدهی مورد توجه و بررسی قرار گرفت. معادلات (۵) چگونگی این پیاده‌سازی را نمایش می‌دهند

$$\begin{aligned}
A_i &= \begin{cases} left\ shift(X_i) & \text{if } ctrl=0\\ right\ shift(X_{i-1}) & \text{else}\end{cases}\\
B_i &= \begin{cases} A_i+6 & \text{if } (ctrl=0 \text{ and } X_i\geq 5)\\ A_i & \text{else}\end{cases}\\
M_i &= \begin{cases} B_i+1 & \text{if } (ctrl=0 \text{ and } X_{i-1}\geq 5)\\ B_i+5 & \text{if } (ctrl=1 \text{ and } X_i:odd)\\ B_i & \text{else}\end{cases}
\end{aligned} \qquad (۵)$$

شکل ۳ پیاده‌سازی واحد قابل بازپیکربندی توصیف‌شده را نمایش می‌دهد که در این شکل، سیگنال $Cond$ برای تولید B_i صحیح از شرط $(ctrl=0 \text{ and } X_i\geq 5)$ حاصل می‌گردد.

همان گونه که می‌دانیم هر چه واحدهای طراحی‌شده، انعطاف‌پذیری بالاتری داشته باشند برای استفاده در سخت‌افزارهای قابل بازپیکربندی مناسب‌تر هستند. اما در اینجا منظور از انعطاف‌پذیری، توجه به دو جنبه می‌باشد: اول، امکان پیاده‌سازی چندین عملیات مختلف حسابی بر روی واحد مورد نظر وجود داشته باشد و دوم، اجزای تشکیل‌دهنده آن بتوانند در اجرای عملیات حسابی مختلفی شرکت کنند. به عبارت دیگر، اجزای

1. Partial Product
2. Easy Multiples
3. Carry-Free



شکل ۴: واحد قابل بازپیکربندی تولید مضارب ۲ و ۵ یک عدد دهدهی (پیاده‌سازی با استفاده از LUT-۴).

خاص‌منظوره کمتری داشته باشد. طراحی ارائه‌شده در شکل ۳ از جنبه دوم، نسبت به شکل ۲ انعطاف‌پذیری بیشتری دارد، زیرا دارای اجزای عام‌منظوره‌تری بوده که می‌توانند به صورت مستقل در سایر عملیات حسابی نیز مورد استفاده قرار بگیرند. بنابراین با افزایش ریزدانگی در این طراحی، انعطاف‌پذیری بالاتری حاصل گردید.

در طراحی واحدهای قابل بازپیکربندی، معمول این است که از اجزای عام‌منظوره‌تری مانند LUTها استفاده می‌شود. با این وجود، همان طور که قبلاً بیان گردید، استفاده از LUT-۸ برای تولید حاصل‌ضرب‌های جزئی به دلیل مساحت و توان مصرفی بالا مناسب نیست. به همین دلیل تلاش شد تا طراحی جدیدی بر اساس LUT با اندازه کوچک‌تر و در نتیجه مساحت و توان کمتر ارائه گردد. شکل ۴ حاصل تلاش برای پیاده‌سازی واحد تولید مضارب ۲ و ۵ یک عدد دهدهی با استفاده از LUT-۴ می‌باشد. در این طراحی در ابتدا n سطر از LUTهای ۴ ورودی (در هر سطر هشت LUT-۴) و سپس $n-1$ سطر جمع‌کننده‌های ۴بیتی ساده دودویی به کار رفته است.

در این نوع پیاده‌سازی، LUTها می‌توانند با توجه به چهار بیت مربوط به هر رقم BCD برای تولید مضارب ۲ یا ۵ یک عدد دهدهی پیکربندی شوند. در بخش بعد مقایسه و ارزیابی دقیق‌تری از پیاده‌سازی‌های انجام‌شده ارائه می‌گردد.

۳-۳ ترکیب با حساب دودویی

اگرچه تمرکز اصلی این مقاله بر روی طراحی واحدهای حسابی دهدهی قابل بازپیکربندی و تأثیر پارامترهای مختلف مانند ریزدانگی و انعطاف‌پذیری در آنها می‌باشد، اما می‌توان به سادگی و با تغییر جزئی، از همین واحدها برای انجام عملیات حساب دودویی نیز استفاده کرد. این کار در جمع‌کننده تک‌رقمی BCD با افزودن یک سیگنال کنترلی انجام می‌گردد که به عنوان ورودی به دروازه‌های منطقی AND در این مدار داده می‌شود. در حالتی که این سیگنال کنترلی مقدار ۱ داشته باشد، نیمه پایینی مدار فعال بوده و جمع دهدهی انجام می‌شود. اما در صورت صفربودن این سیگنال، نیمه پایینی که مربوط به تصحیح جمع دهدهی است از مدار خارج شده و عملیات جمع دودویی انجام می‌گردد. از آنجا که جمع‌کننده‌های انتشاری مطرح‌شده در این مقاله مانند CRA، CSKA و CSLA از کنار هم قرار دادن تعدادی جمع‌کننده تک‌رقمی ایجاد می‌شوند، این نوع پیاده‌سازی ترکیبی حساب دودویی و دهدهی به تمامی آنها قابل تعمیم می‌باشد.

٤- مقایسه و ارزیابی

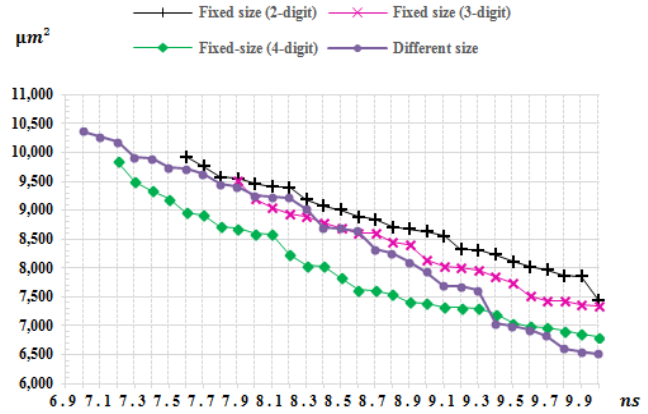
به منظور ارزیابی بهتر طراحی‌های انجام‌شده، کد VHDL مربوط به تمام پیاده‌سازی‌های انجام‌شده، توسعه داده شده و مورد آزمون قرار گرفت. سپس با استفاده از ابزار Synopsys Design Compiler با تکنولوژی TSMC ۱۳۰ nm CMOS در شرایط نرمال (ولتاژ هسته ۱٫۲ ولت و دمای عملیاتی ۲۵ درجه سانتی‌گراد) سنتز شده و نتایج مربوط به مساحت و تأخیر گزارش گردید. همچنین نتایج مربوط به توان مصرفی کل، توسط ابزار Power Compiler گزارش شده است.

ابتدا چهار جمع‌کننده ۱۶رقمی دهدهی شامل جمع‌کننده‌های دارای بلاک‌های با اندازه ثابت و متغیر با یکدیگر مقایسه شدند. برای جمع‌کننده‌های دارای بلاک‌های با اندازه ثابت، از بلاک‌های دورقمی (۸ بلاک)، سه‌رقمی (۵ بلاک + یک بلاک تک‌رقمی) و چهاررقمی (۴ بلاک) استفاده شد. برای جمع‌کننده‌های دارای بلاک‌های با اندازه متفاوت نیز از چیدمان بلاک‌های با اندازه ۱-۲-۳-۴-۳-۲-۱ استفاده گردید. برای مقایسه بهتر این جمع‌کننده‌ها، آنها را در محدودیت‌های زمانی[1] مختلف سنتز کرده و نتایج به دست آمده در شکل‌های ۵ تا ۷ آورده شده است. در این نمودارها، هر خط مربوط به یک جمع‌کننده مجزا بوده و نقطه شروع آن، نشان‌دهنده اولین محدودیت زمانی است که آن جمع‌کننده می‌تواند به درستی کار کند.

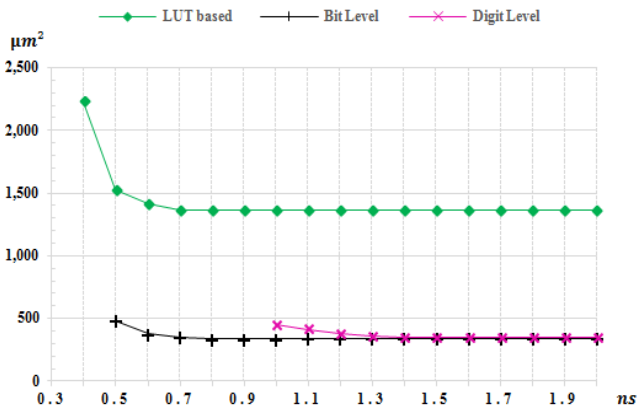
همان طور که مشاهده می‌شود، بررسی‌ها نشان می‌دهد که بهترین نتیجه در میان بلاک‌های با اندازه ثابت، مربوط به بلاک‌های ۴رقمی می‌باشد که نسبت به بلاک‌های ۲رقمی در محدودیت‌های زمانی مختلف، به طور متوسط ۱۲٪ بهبود مساحت و ۱۳٫۴٪ بهبود توان مصرفی را در پی داشته است. این نتیجه به دلیل ریزدانگی بالاتر بلاک‌ها و نظم بیشتر در طراحی حاصل شده که منطبق بر پیش‌بینی‌های اولیه نیز می‌باشد. به علاوه همان طور که انتظار می‌رفت، استفاده از بلاک‌های با اندازه متفاوت در طراحی، موجب شده تا این جمع‌کننده در فرکانس‌های بالاتری (محدودیت‌های زمانی کمتری) نسبت به سایر جمع‌کننده‌ها بتواند به درستی کار کند. در عین حال، جمع‌کننده دارای بلاک‌های متفاوت به دلیل عدم وجود نظم ساختاری، دارای مساحت و توان مصرفی بالاتری بوده و تنها در فرکانس‌های پایین، عملکرد بهتری از خود نشان داده است.

هر چند پژوهش‌های گذشته در این حوزه بر پیاده‌سازی مدارهای دهدهی با استفاده از FPGAهای موجود که بستری متفاوت با طراحی‌های انجام‌شده در این مقاله می‌باشند، تمرکز داشته‌اند اما برای مشاهده نتایج دقیق‌تر و درصد بهبود طرح پیشنهادی نسبت به کارهای گذشته، می‌توان با در نظر گرفتن فرضیات مختلف، تخمینی از مساحت و تأخیر پیاده‌سازی‌های انجام‌شده را به دست آورد. در جدول ۱ نتایج گزارش‌شده در [۲۹] و [۳۲] برای پیاده‌سازی جمع‌کننده‌های دهدهی ۱۶رقمی با دو طراحی انجام‌شده در این مقاله مقایسه شده‌اند. با توجه به این که واحد مساحت در کارهای پیشین، تعداد LUT-۶های استفاده‌شده برای پیاده‌سازی جمع‌کننده‌ها بوده است، به منظور یکسان‌سازی واحدها، کد VHDL مربوط به یک LUT-۶ توسعه داده شد و پس از سنتز، مساحت آن حدود ۸۱۹ میکرومتر مربع به دست آمد. همچنین از آنجا که نتایج تأخیر بیان‌شده در کارهای پیشین، مربوط به اجزای منطقی بوده و از تأخیر اتصالات صرف نظر شده است، برای طراحی‌های جدید نیز، تنها تأخیر این بخش از مدار گزارش شده است. همان طور که در این جدول

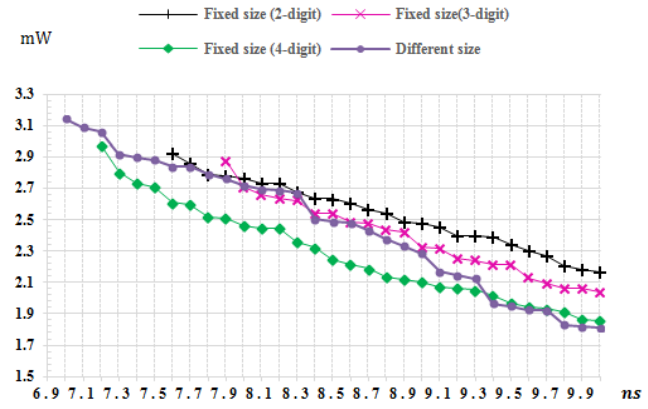
1. Time Constraint



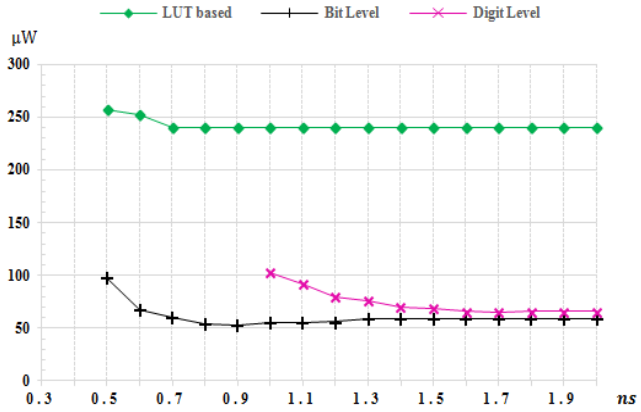
شکل ۵: مقایسه مساحت در جمع‌کننده‌های ۱۶رقمی دهدهی با ریزدانگی متفاوت.



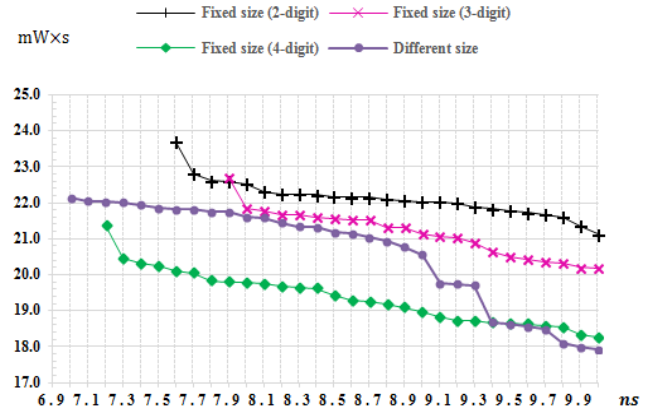
شکل ۸: مقایسه مساحت در پیاده‌سازی‌های مختلف واحد تولیدکننده مضارب ۲ و ۵.



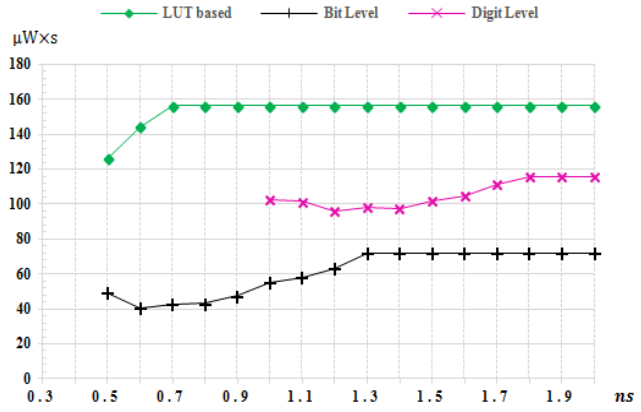
شکل ۶: مقایسه توان در جمع‌کننده‌های ۱۶رقمی دهدهی با ریزدانگی متفاوت.



شکل ۹: مقایسه توان مصرفی در پیاده‌سازی‌های مختلف واحد تولیدکننده مضارب ۲ و ۵.



شکل ۷: مقایسه حاصل ضرب توان در تأخیر (PDP) در جمع‌کننده‌های ۱۶رقمی دهدهی با ریزدانگی متفاوت.



شکل ۱۰: مقایسه حاصل ضرب توان در تأخیر (PDP) در پیاده‌سازی‌های مختلف واحد تولیدکننده مضارب ۲ و ۵.

جدول ۱: مقایسه طراحی‌های پیشنهادی با پیاده‌سازی جمع‌کننده‌های دهدهی پیشین بر روی FPGA.

Adder	Area (LUT – ۶)	Delay (ns)	Area (μm^2)
BCD carry-chain (Low area) [۳۲]	۱۲۸	۲٫۶۶	۱۰۴۸۳۲
BCD carry-chain (Fast version) [۳۲]	۱۶۰	۲٫۵۱	۱۳۱۰۴۰
BCD carry ripple [۲۹]	۸۰	۲٫۱۷	۶۵۵۲۰
Proposed (۴ – digit blocks)	–	۱٫۹۱	۹۸۴۰
Proposed (different size blocks)	–	۲٫۰۲	۱۰۲۷۰

نیز مشاهده می‌شود، بهترین طراحی پیشنهادی نسبت به بهترین پژوهش پیشین (جمع‌کننده با بلوک‌های ۴رقمی نسبت به [۲۹]) حدود ۱۲٪ در تأخیر و بیش از ۵۰۰٪ در مساحت بهبود داشته است.

برای مقایسه بهتر پیاده‌سازی‌های مختلف واحد تولید مضارب ۲ و ۵ یک عدد دهدهی، هر سه طراحی انجام‌شده (سطح بیت، سطح یک رقم دهدهی و مبتنی بر LUT) در محدودیت‌های زمانی مختلف سنتز شده و نتایج به دست آمده در شکل‌های ۸ تا ۱۰ نشان داده شده است. در این نمودارها نیز هر خط مربوط به یک طراحی مجزا بوده و نقطه شروع آن،

نشان‌دهنده اولین محدودیت زمانی است که طراحی مورد نظر می‌تواند به درستی کار کند.

با توجه به تعریف بیان‌شده برای انعطاف‌پذیری یک واحد حسابی، پیاده‌سازی سطح بیت دارای کمترین میزان انعطاف‌پذیری و پیاده‌سازی با استفاده از LUT دارای بیشترین میزان انعطاف‌پذیری می‌باشد. بنابراین با توجه به نتایج به دست آمده، با افزایش انعطاف‌پذیری در طراحی این واحد، مساحت و توان مصرفی افزایش می‌یابد و هر چه بلوک‌های عام‌منظوره‌تری در طراحی به کار گرفته شود (استفاده از LUT در مقابل پیاده‌سازی سطح یک رقم و پیاده‌سازی سطح یک رقم در مقابل سطح بیت)، سربار مساحت و توان مصرفی بیشتر خواهد بود. بیشترین مقدار این هزینه سربار مربوط به پیاده‌سازی مبتنی بر LUT در مقایسه با پیاده‌سازی سطح بیت می‌باشد که در محدودیت‌های زمانی مختلف به طور متوسط منجر به ۷۵٪ افزایش مساحت و توان مصرفی شده است.

۵- نتیجه‌گیری

در این مقاله، تأثیر میزان ریزدانگی و انعطاف‌پذیری در طراحی سخت‌افزارهای قابل بازپیکربندی دهدهی بررسی گردید. نتایج به دست آمده از پیاده‌سازی جمع‌کننده‌های دهدهی مختلف با ریزدانگی متفاوت نشان داد که انتخاب اندازه مناسب برای بلاک‌های سازنده یک واحد حسابی، تأثیر مستقیمی در کارایی، مساحت و توان مصرفی آن خواهد داشت. در این بررسی، بهترین نتیجه از نظر مساحت و توان مصرفی در فرکانس‌های بالا، مربوط به جمع‌کننده با بلاک‌های ۴رقمی و در فرکانس‌های پایین، مربوط به جمع‌کننده دارای بلاک‌های متفاوت بوده است. به علاوه، جمع‌کننده دارای بلاک‌های متفاوت در محدودیت‌های زمانی کمتری نسبت به سایر جمع‌کننده‌ها توانسته به درستی کار کند.

برای بررسی تأثیر میزان انعطاف‌پذیری واحدها در طراحی سخت‌افزارهای قابل بازپیکربندی دهدهی، پیاده‌سازی‌های مختلف واحد تولید مضارب ۲ و ۵ یک عدد دهدهی با یکدیگر مقایسه گردید. نتایج حاصل از این بررسی نشان داد که بالابردن انعطاف‌پذیری (استفاده از LUT)، تأخیری قابل رقابت با طراحی‌های خاص‌منظوره ایجاد کرده است، اما در مقابل مساحت و توان مصرفی را به میزان قابل توجهی افزایش می‌دهد.

مراجع

[1] H. H. Goldstine and A. Goldstine, "The electronic numerical integrator and computer (ENIAC)," *IEEE Annals of the History of Computing*, vol. 18, no. 1, pp. 10-16, Spring 1996.

[2] G. Gray, "UNIVAC I instruction set," *Unisys History Newsletter*, vol. 5, no. 3, pp. 1-3, Jun. 2001.

[3] F. E. Hamilton and E. C. Kubie, "The IBM magnetic drum calculator type 650," *J. of ACM*, vol. 1, no. 1, pp. 13-20, Jan. 1954.

[4] M. F. Cowlishaw, "Decimal floating-point: algorism for computers," in *Proc. of the 16th IEEE Symp. on Computer Arithmetic*, vol. 1, pp. 104-111, Jun. 2003.

[5] IBM Corporation, *The Telco Benchmark*, http://speleotrove.com/decimal/telcoSpec.html, Accessed on 26 Apr. 2017.

[6] IEEE Standards Committee, "754-2008 IEEE standard for floating-point arithmetic," *IEEE Computer Society Standard*, pp. 1-58, Aug. 2008.

[7] N. Chainani, Decfloat: The Data Type of the Future, http://www.ibm.com/developerworks/data/library/techarticle/dm0801chainani/, Accessed on 26 Apr. 2017.

[8] M. Cornea, *Intel Decimal Floating-Ppoint Math Library*, software.intel.com/en-us/blogs/2008/03/06/inteldecimal-floating-point-math-library, Accessed on 26 Apr. 2017.

[9] GCC, *The GNU Compiler Collection*, http://www.gcc.gnu.org, Accessed on 26 Apr. 2017.

[10] A. Y. Duale, M. H. Decker, H. G. Zipperer, M. Aharoni, and T. J. Bohizic, "Decimal floating-point in Z9: an implementation and testing perspective," *IBM J. Research and Development*, vol. 51, no. 1, pp. 217-227, Jan. 2007.

[11] C. F. Webb, "IBM z10-the next-generation mainframe microprocessor," *IEEE Micro*, vol. 28, no. 2, pp. 19-29, May 2008.

[12] S. Carlough, A. Collura, S. Mueller, and M. Kroener, "The IBM zEnterprise-196 decimal floating-point accelerator," in *Proc. of the 20th IEEE Symp. on Computer Arithmetic*, pp. 139-146, Jul. 2011.

[13] L. K. Wang, C. Tsen, M. J. Schulte, and D. Jhalani, "Benchmarks and performance analysis of decimal floating-point applications," in *Proc. of 25th Int. Conf. on Computer Design*, pp. 164-170, Oct. 2007.

[14] M. Anderson, C. Tsen, L. K. Wang, K. Compton, and M. J. Schulte, "Performance analysis of decimal floating-point libraries and its impact on decimal hardware and software solutions," in *Proc. of 27th Int. Conf. on Computer Design*, pp. 465-471, Oct. 2009.

[15] D. A. Buell and K. L. Pocek, "Custom computing machines: an introduction," *J. of Supercomputing*, vol. 9, no. 3, pp. 219-230, Sept. 1995.

[16] S. A. Hauck, T. W. Fry, M. M. Hosler, and J. P. Kao, "The chimaera reconfigurable functional unit," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 2, pp. 206-217, Feb. 2004.

[17] R. D. Kenney and M. J. Schulte, "High-speed multioperand decimal adders," *IEEE Trans. on Computers*, vol. 54, no. 8, pp. 953-963, Aug. 2005.

[18] A. Vazquez and E. Antelo, "Conditional speculative decimal addition," in *Proc. of the 7th Conf. on Real Numbers and Computers*, pp. 47-57, Jul. 2006.

[19] S. Gorgin and G. Jaberipur, "A fully redundant decimal adder and its application in parallel decimal multipliers," *Microelectronics J.*, vol. 40, no. 10, pp. 1471-1481, Oct. 2009.

[20] A. Vazquez, E. Antelo, and J. D. Bruguera, "Fast radix-10 multiplication using redundant BCD codes," *IEEE Trans. on Computers*, vol. 63, no. 8, pp. 1902-1914, Aug. 2014.

[21] P. Chawla, P. Palsodkar, and S. Sheikh, "Hardware efficient reconfigurable arithmetic unit," in *Proc. of the Int. Conf. on Advances in Computer Science and Electronics Engineering*, pp. 159-163, Jan. 2012.

[22] W. Haller, W. H. Li, M. R. Kelly, and H. Wetter, *Highly Parallel Structure for Fast Cycle Binary and Decimal Adder Unit*, International Business Machines Corporation, US Patent 2006/0031289, 8 pp., 2006.

[23] H. Calderon, G. Gaydadjiev, and S. Vassiliadis, "Reconfigurable universal adder," in *Proc. of IEEE Int. Conf. on Application-Specific Systems, Architecture Processors*, pp. 186-191, Jul. 2007.

[24] S. Veeramachaneni, M. Kirthi Krishna, V. Prateek, S. Subroto, S. Bharat, and M. B. Srinivas, "A novel carry-look ahead approach to a unified BCD and binary adder/subtractor," in *Proc. 21st Int. Conf. on VLSI Design*, pp. 547-552, Jan. 2008.

[25] M. Tyagi, A. Vaishist, and K. Khare, "A novel hardware efficient reconfigurable 32-bit arithmetic unit for binary, BCD and floating point operands," *International J. of Engineering Science and Technology*, vol. 3, no. 5, pp. 4449-4464, May 2011.

[26] M. Dorrigiv and G. Jaberipur, "Low area/power decimal addition with carry-select correction and carry-select sum-digits," *Integration, the VLSI J.*, vol. 47, no. 4, pp. 443-451, Sept. 2014.

[27] S. Ganguly, A. Mittal, and S. E. Ahmed, "A reconfigurable parallel prefix ling adder with modified enhanced flagged binary logic," in *Proc. Asia Pacific Conf. on Postgraduate Research in Microelectronics and Electronics, PrimeAsia'12*, 7 pp., Dec. 2012.

[28] A. Nannarelli, "FPGA based acceleration of decimal operations," in *Proc. Int. Conf. on Reconfigurable Computing and FPGAs*, pp. 146-151, Dec. 2011.

[29] A. Vazquez and F. de Dinechin, *Multi-Operand Decimal Tree Adders for FPGAs*, INRIA, Research Report, 2010.

[30] Xilinx Inc, Virtex-6 User Guide, http://www.xilinx.com/, Accessed on 26 Apr. 2017.

[31] M. Baesler and T. Teufel, "FPGA implementation of a decimal floating-point accurate scalar product unit with a parallel fixed-point multiplier," in *Proc. of Int. Conf. on Reconfigurable Computing and FPGAs*, pp. 6-11, Dec. 2009.

[32] G. Biaul, M. Vazquez, J. P. Deschamps, and G. Sutter, "Decimal addition in FPGA," in *Proc. of 5th Southern Conf. on Programmable Logic*, pp. 101-108, Apr. 2009.

[33] G. Sutter, E. Todorovich, G. Bioul, M. Vazquez, and G. P. Deschamps, "FPGA implementations of BCD multipliers," in *Proc. of IEEE Int. Conf. on Reconfigurable Computing and FPGAs*, pp. 36-41, Dec. 2009.

[34] M. Vazquez, G. Sutter, J. P. Deschamps, and G. Bioul, " Decimal adders/subtractors in FPGA: efficient 6-input LUT implementations," in *Proc. of Int. Conf. on Reconfigurable Computing and FPGAs*, pp. 42-47, Dec. 2009.

[35] I. Koren, *Computer Arithmetic Algorithms*, A. K. Peters, Natic, Massachusetts, pp. 116-119, 2002.

[36] M. A. Erle and M. J. Schulte, "Decimal multiplication via carry-save addition," in *Proc. of IEEE Int. Conf. on Application-Specific Systems, Architecture Processors*, pp. 348-359, Jun. 2003.

[37] B. Hickmann, A. Krioukov, M. Schulte, and M. Erle, "A parallel IEEE 754 decimal floating-point multiplier," in *Proc. of 25th Int. Conf. on Computer Design*, pp. 296-303, Oct. 2007.

[38] S. Gorgin, G. Jaberipur, and B. Parhami, "Design and evaluation of decimal array multipliers," in *Proc. 43rd Asilomar Conf. on Signals, Systems and Computers*, pp. 1782-1786, Nov. 2009.

[39] S. Gorgin, G. Jaberipur, and R. Hashemi-Asl, "Efficient ASIC and FPGA implementation of binary coded decimal digit multipliers," *Circuits, Systems and Signal Processing,* vol. 33, no. 12, pp. 3883-3899, Dec. 2014.

[40] T. Ueda, *Decimal Multiplying Assembly and Multiply M*odule, US Patent 5379245, 1995.

سمانه امامی مدارک کارشناسی و کارشناسی ارشد خود را در رشته مهندسی کامپیوتر به ترتیب در سال های ۱۳۸۷ و ۱۳۸۹ از دانشگاه شهید بهشتی دریافت نمود. نام‌برده از سال ۱۳۹۰ تحصیلات خود را در مقطع دکتری معماری کامپیوتر در دانشگاه صنعتی امیرکبیر آغاز کرده و هم اکنون دانشجوی دکتری این دانشگاه می‌باشد. زمینه‌های تحقیقاتی مورد علاقه ایشان عبارتند از: حساب کامپیوتری، سنتز سطح بالا و طراحی سخت‌افزارهای قابل بازپیکربندی.

مهدی صدیقی در سال ۱۳۶۹ مدرک کارشناسی خود را در رشته مهندسی کامپیوتر و برق از دانشگاه صنعتی شریف و مدارک کارشناسی ارشد و دکتری خود را در همان رشته از دانشگاه کلرادو در آمریکا به ترتیب در سال های ۱۳۷۳ و ۱۳۷۷دریافت نمود. نام‌برده از سال ۱۳۸۰ در دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر مشغول به فعالیت گردید و اینک نیز عضو هیأت علمی این دانشکده است. زمینه‌های علمی مورد علاقه ایشان شامل موضوعاتی مانند طراحی VLSI، سنتز مدارهای حسابی، سیستم‌های نهفته و محاسبات کوانتومی می‌باشد.