Journal of Electronical & Cyber Defence
Vol. 10, No. 1, Spring 2022, Serial No. 37

The residue number system component design based on OneHot coding

A. A. Emrani Zarandi*, F. Sabbah

*Assistant Professor, Department of Computer Engineering, Faculty of Technology and Engineering, Shahid Bahonar University, Kerman, Iran

(Received: 17/08/2021, Accepted: 13/12/2021)

ABSTRACT

Nowadays by extensive application of Internet and online tools, security has become a major concern for computer system designers. The network security and prevention of the enemy intrusion and access to sensitive information are based on the cryptographic algorithms which require complex processing and fast computations. Therefore, computer arithmetic and finding new methods to improve basic operations such as addition, subtraction and multiplication is of great importance. In digital systems, low-delay computations play an important role, and using a residue number system is one of the ways that can speed up computational operations by parallelizing and reducing the carry propagation chain. OneHot coding, with its special features, is well compatible with the residue number system to help speed up operations by eliminating the carry propagation. In addition, with the limited changes of input in OneHot coding, the rate of circuit switching is reduced, resulting in reducing a dynamic power consumption comparing with a binary coding system. Therefore, in this article, in order to take advantage of the benefits of OneHot coding in the residue number system, the required components are designed, including adder, subtractor, and multiplier of modules with OneHot coding. The innovation of this research is presenting the structure of a forward converter to transform inputs into residues in the OneHot format. The proposed structures have been implemented and their performance has been analyzed and evaluated.

Keywords: : Adder, Subtractor, Multiplier, Converter, OneHot coding, Residue number system.

* Corresponding Author Email: a.emrani@uk.ac.ir

نشریه علمی "پدافند الکترونیکی و سایبری"
سال دهم، شماره ۱، بهار ۱۴۰۱، ص ۱۳۳-۱۱۹

علمی-پژوهشی

طراحی مؤلفه‌های سیستم اعداد مانده‌ای براساس کدینگ OneHot

فریده صبّاح[۱]، آزاده السادات عمرانی زرندی[۲*]

۱- کارشناسی ارشد، ۲- استادیار، دانشکده فنی و مهندسی، دانشگاه شهید باهنر، کرمان، ایران

(دریافت: ۱۴۰۰/۰۵/۲۶، پذیرش: ۱۴۰۰/۰۹/۲۲)

چکیده

با گسترش اینترنت و ابزارهای برخط، حفظ امنیت به دغده‌ی اصلی طراحان سیستم‌های کامپیوتری تبدیل شده است. برای حفظ امنیت، جلوگیری از نفوذ، و دستیابی دشمنان به اطلاعات حساس الگوریتم‌های رمزنگاری مورد استفاده قرار می‌گیرند، که آن‌ها نیز به انجام پردازش‌های پیچیده و محاسبات سریع نیاز دارند. ازاین‌رو، حساب کامپیوتری و ارائه‌ی ترفندهای جدید جهت ارتقای عملیات پایه مثل جمع، تفریق و ضرب از اهمیت شایانی برخوردار است. در سیستم‌های دیجیتال انجام محاسبات با کمترین تأخیر نقش مهمی دارد و استفاده از سیستم اعداد مانده‌ای یکی از راه‌هایی است که با موازی‌سازی و کاهش انتشار بیت نقلی می‌تواند باعث سرعت بخشیدن به عملیات محاسباتی گردد. علاوه‌براین، کدینگ OneHot با ویژگی‌های خاص خود، سازگاری خوبی با سیستم اعداد مانده‌ای دارد تا با حذف انتشار رقم نقلی در عملیات محاسباتی کمک شایانی به سرعت بخشیدن عملیات کند. علاوه‌براین، با محدود بودن تغییرات ورودی‌ها درکدینگ OneHot فعالیت سوئیچینگ مدار کم می‌شود که نتیجه‌ی آن کاهش توان مصرفی دینامیک نسبت به کدینگ باینری در این سیستم است. ازهمین‌رو در این مقاله جهت بهره بردن از مزایای کدینگ OneHot در سیستم اعداد مانده‌ای، طراحی اجزای مورد نیاز شامل جمع‌کننده، تفریق‌کننده و ضرب‌کننده پیمانه‌ای با کدینگ OneHot را انجام دادیم. همچنین ساختار مبدل مستقیم جهت تبدیل ورودی‌ها به باقی‌مانده‌هایی در فرمت کدینگ OneHot برای اولین بار معرفی شده است. ساختارهای پیشنهادی پیاده‌سازی شده‌اند و از لحاظ عملکرد مورد تحلیل و ارزیابی واقع شده‌اند.

کلیدواژه‌ها: جمع‌کننده، تفریق‌کننده، ضرب‌کننده، مبدل، کدینگ وان‌هات ، سیستم اعداد مانده‌ای

۱- مقدمه

در سامانه‌های راداری، پدافندها، و پردازشگرهای سیگنال دیجیتال (DSP[۲]) و مدارهای مجتمع پردازش‌داده (ASIC[۳]) و کاربردهایی مانند رمزنگاری و حفاظت از داده‌ها از ساختارهای محاسباتی استفاده می‌شود. مدار جمع‌کننده به‌عنوان اساسی‌ترین بخش محاسباتی یک پردازشگر است که تأخیر انتشار بیت نقلی از معایب جمع‌کننده‌ی باینری است. در سیستم باینری هر موقعیت دارای وزن بیتی مشخصی است و هنگام جمع دو عدد در این سیستم، بیت نقلی خروجی در بدترین حالت وابسته به تمام موقعیت‌های بیتی پایین‌تر است. تلاش‌های زیادی صورت گرفته تا جمع‌کننده‌های بهینه‌ای برای کاهش تأخیر انتشار بیت نقلی [۲,۱] ارائه شوند؛ اما وابستگی بین موقعیت‌های بیتی از خصوصیت‌های ذاتی سیستم باینری است، بنابراین به‌طور کامل نمی‌توان موازی‌سازی را برای جمع اعداد در سیستم باینری به کار برد و در کاربردهایی مانند رمزنگاری و پردازش سیگنال‌های دیجیتال کارایی سیستم باینری تحت تأثیر تأخیر انتشار بیت نقلی به‌شدت کاهش می‌یابد.

سیستم اعداد مانده‌ای یک سیستم عددی غیروزنی است که محاسبات موازی، انتشار محدود رقم نقلی، توان مصرفی کم و ارتباطات امن را پشتیبانی می‌کند و بیش از نیم قرن است که در محاسبات موازی برای کاربردهای مختلف به‌عنوان جایگزین مناسب سیستم اعداد باینری، در تحقیق‌های گارنر [۴] و محققان دیگر [۶,۵] مورد توجه قرار گرفته است. در سیستم اعداد مانده‌ای، اعداد باینری به‌صورت تعدادی اعداد کوچک در پیمانه‌هایی که دو به دو نسبت به هم اول هستند نمایش داده می‌شود و عملیات روی باقی‌مانده‌های متناظر در پیمانه‌ها به‌صورت جداگانه انجام می‌گیرد. پس حین انجام عملیات حسابی وابستگی بین باقی‌مانده‌ها در پیمانه‌های مختلف وجود ندارد و این ویژگی باعث افزایش سرعت محاسبات می‌شود. بنابراین در انجام عملیات به طور متوالی و گسترده، استفاده از سیستم اعداد مانده‌ای نقش بسیار مؤثری در بهبود کارایی خواهد داشت [۴]. برای استفاده از سیستم اعداد مانده‌ای در کاربردهای گسترده، باید مدارهای پایه آن به‌صورت بهینه طراحی شوند.

استفاده از کدینگ باینری برای سیستم اعداد مانده‌ای باعث انتشار بیت نقلی در هر پیمانه است و برای حل این مشکل می‌توان از کدینگ‌های متفاوت دیگری مانند کدینگ OneHot

* رایانامه نویسنده مسئول: a.emrani@uk.ac.ir

[2] Digital Signal Processor
[3] Application Specific Integrated Circuit

ناشر: دانشگاه جامع امام حسین (ع)

بهره برد. در کدینگ OneHot برای نمایش اعداد 0 تا k به $k+1$ بیت نیاز است. در این نوع کدینگ تنها یکی از موقعیت‌های بیتی، مقدار یک را می‌گیرد و بقیه صفر هستند و مقدار عدد نمایش داده شده برابر با موقعیت بیتی است که مقدار آن برابر با ۱ است (شماره موقعیت‌های بیتی از 0 تا k است) [۸].

استفاده از کدینگ OneHot در سیستم اعداد مانده‌ای باعث افزایش سرعت عملیات و کاهش توان مصرفی می‌گردد؛ ازاین‌رو سیستم اعداد مانده‌ای مبتنی بر این کدینگ در کاربردهایی مانند پردازش تصویر و پردازش سیگنال‌های دیجیتال نیز مورد توجه واقع شده‌اند [۳].

از آنجا که در سیستم اعداد مانده‌ای از باقی‌مانده‌ی اعداد در کانال‌های مختلف استفاده می‌شود، به کار بردن کدینگ OneHot برای این ساختار قابل توجیه است. اما تحقیقاتی کمی در این زمینه صورت گرفته است و حتی در زمینه‌ی طراحی کامل ساختار اعداد مانده‌ای با این کدینگ کاری صورت نگرفته است. با توجه به اینکه در تحقیقات گذشته طراحی کامل اجزا مورد نیاز RNS با کدینگ OneHot صورت نگرفته است و تنها به‌صورت موردی برای یک کاربرد خاص در حد جمع‌کننده بوده است، و سایر مدارات محاسباتی قابل استفاده در سیستم اعداد مانده‌ای با این کدینگ طراحی نشده است، ما بر آن شدیم که به مطالعه‌ی این موضوع بپردازیم. علاوه‌براین، لازمه‌ی استفاده از کدینگ OneHot و انجام محاسبات با این کدینگ در سیستم اعداد مانده‌ای، داشتن مداری برای تبدیل عدد باینری به باقی مانده‌هایی در کدینگ OneHot است، مبدل مستقیم موردنیاز نیز طراحی شده است.

از همین رو، با توجه به کم بودن تعداد تحقیقات انجام شده در زمینه‌ی طراحی مدارات محاسباتی پیمانه‌ای با کدینگ OneHot در سیستم اعداد مانده‌ای، در این مقاله ساختارهای جدید برای مدارهای جمع‌کننده، تفریق‌کننده، جمع‌کننده/تفریق‌کننده مانده‌ای با کدینگ OneHot و دو مدار برای ضرب‌کننده‌ی مانده‌ای با کدینگ OneHot و همچنین ساختاری برای مبدل مستقیم (تبدیل اعداد وزنی به باقی‌مانده‌های OneHot) که تاکنون طراحی نشده است را ارائه دادیم، تا به کمک این ساختارها بتوان از مزایای کدینگ OneHot در سیستم اعداد مانده‌ای بهره برد. هدف از این طراحی‌ها، بالا بردن سرعت محاسبات است، و ساختارهای معرفی شده برای نخستین بار در این مقاله طراحی شده‌اند. از همین رو، ابتدا در قسمت۲ مقاله پیش‌زمینه‌ای از سیستم اعداد مانده‌ای و کدینگ OneHot ارائه می‌شود. در قسمت ۳ طرح پیشنهادی برای مدار جمع‌کننده و تفریق‌کننده وجمع‌کننده/تفریق‌کننده پیمانه‌ای با کدینگ OneHot و در قسمت ۴ طرح پیشنهادی برای مدار مبدل مستقیم سیستم اعداد مانده‌ای با کدینگ OneHot معرفی شده است. قسمت ۵ به ارائه‌ی دو طرح پیشنهادی برای ضرب‌کننده پیمانه‌ای با کدینگ OneHot پرداخته‌ایم. در قسمت ۶، نتایج شبیه‌سازی و پیاده‌سازی مدارهای ارائه شده، بیان شده است و در نهایت در قسمت ۷ نتیجه‌گیری مقاله آورده شده است.

۲- پیش‌زمینه

در سیستم اعداد مانده‌ای اعداد با ارقام باقی‌مانده‌ای که طول بیتی کوچک‌تری نسبت به باینری، نمایش داده می‌شوند؛ نمایش غیروزنی در سیستم اعداد مانده‌ای، این امکان را فراهم می‌سازد که انتشار بیت نقلی، از یک رقم باقی‌مانده‌ای به رقم باقی‌مانده‌ای دیگر وجود نداشته باشد؛ بنابراین می‌توان پردازش‌ها را به‌صورت موازی روی ارقام باقی‌مانده انجام داد. در سیستم اعداد مانده‌ای مجموعه‌ای از پیمانه‌ها وجود دارد که دو به دو نسبت به هم اول هستند و تعداد و مقدار پیمانه‌ها با توجه به شرط اول بودن نسبت به یکدیگر، اختیاری است و با توجه به کاربرد مورد نظر تعیین می‌گردد و حاصل ضرب پیمانه‌ها را دینامیک‌رنج یا همان رنج اعداد قابل نمایش در سیستم می‌نامند که آن را با M نمایش می‌دهند و جهت افزایش دینامیک‌رنج می‌توان مقادیر پیمانه‌ها را افزایش یا تعداد پیمانه‌ها را توسعه داد. اگر در یک سیستم اعداد مانده‌ای، مجموعه‌ی پیمانه‌ها به‌صورت $\{3,4,5,7\}$ باشد، در این حالت حاصل ضرب پیمانه‌ها یا همان داینامیک‌رنج برابر با 420 است. بنابراین می‌توان گفت سیستم اعداد مانده‌ای از سه بخش کلی تشکیل می‌شود: ۱) مبدل مستقیم ۲) کانال‌های محاسباتی که عملیات پیمانه‌ای به‌صورت موازی در پیمانه‌های مختلف در آن‌ها انجام می‌شود ۳) مبدل معکوس.

شکل (۱) معماری کلی یک سیستم اعداد مانده‌ای را نشان می‌دهد که در قسمت اول، اعداد باینری به باقی‌مانده‌های باینری اولیه در پیمانه‌های سیستم اعداد مانده‌ای تبدیل می‌شوند و در قسمت دوم هر کانال متناظر با یک پیمانه است و مدارات مربوط به عملیات پیمانه‌ای جمع، ضرب و تفریق در این بخش قرار می‌گیرند. پس از انجام عملیات پیمانه‌ای مربوطه (که به‌صورت مجزا و موازی است)،در قسمت سوم برای تبدیل نمایش پیمانه‌ای به عدد صحیح باینری از مبدل معکوس استفاده می‌شود.

شکل (۱): معماری یک سیستم اعداد مانده‌ای

انتشار بیت نقلی به دلیل باینری بودن سیستم اعداد مانده‌ای است و برای حل این مشکل کدینگ‌های متفاوتی ازجمله OneHot معرفی شده است؛ در کدینگ وان‌هات برای نمایش اعداد 0 تا k به $k+1$ بیت نیاز است تا مقدار مربوط به صفر را نیز رمزگذاری کند. در این نوع کدینگ تنها یکی از موقعیت‌های بیتی، مقدار یک را می‌گیرد و بقیه صفر هستند و مقدار عدد نمایش داده شده برابر با موقعیت بیتی است که مقدار آن برابر با ۱ است (شماره‌ی موقعیت‌های بیتی از 0 تا k است).

نمایش اعداد صفر تا ۷ با استفاده از این نوع کدینگ به‌صورت رابطه (۱) خواهد بود:

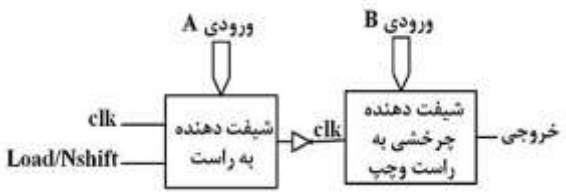
$$\begin{aligned}
&0 \rightarrow 00000001\\
&1 \rightarrow 00000010\\
&2 \rightarrow 00000100\\
&3 \rightarrow 00001000\\
&4 \rightarrow 00010000\\
&5 \rightarrow 00100000\\
&6 \rightarrow 01000000\\
&7 \rightarrow 10000000
\end{aligned} \qquad (۱)$$

۳- طرح پیشنهادی برای جمع‌کننده/تفریق‌کننده پیمانه‌ای با کدینگ OneHot

محاسبات باقی‌مانده‌ای بر مبنای کدینگ وان‌هات در سال ۱۹۹۵ مورد توجه واقع شد [۸]. مدارهای جمع‌کننده، تفریق‌کننده و ضرب‌کننده ارائه شده در [۸] بر مبنای شیفت چرخشی نشان‌دهنده، کاهش بیش از ۸۵ درصد PDP[1] مدارات طراحی شده با این کدینگ در مقایسه با طرح‌های متداول در سیستم اعداد مانده‌ای می‌گردد که مدار [۱۱] نیز ساختار جدیدی بر مبنای طرح ارائه شده در [۸] است. از کدینگ OneHot برای طراحی جمع‌کننده پیمانه‌ای، جهت افزایش سرعت عملیات پیمانه‌ای در کاربردهای با سایز کوچک داده می‌توان استفاده کرد و زمانی‌که این کدینگ برای نمایش مقدار باقی‌مانده‌ها در یک پیمانه استفاده شود آن را OHR[2] می‌نامند [۹]. ویژگی خوب استفاده از کدینگ OneHotدر طراحی اجزا سیستم مانده‌ای این است که، وابستگی به پیمانه ندارد و می‌توان طراحی‌های انجام‌شده را برای مجموعه‌پیمانه‌های مختلف استفاده کرد. با توجه به اهمیت ساختار جمع‌کننده در قسمت محاسبات سیستم اعداد مانده‌ای در این بخش ساختار جدید و سریع برای جمع‌کننده/تفریق‌کننده پیمانه‌ای بر اساس کدینگ OneHot پیشنهاد می‌دهیم. انتشار بیت نقلی محدودییت اصلی است که باعث کاهش سرعت عملیات جمع و تفریق می‌گردد. در طرح پیشنهادی برای جمع‌کننده/تفریق‌کننده پیمانه‌ای با کدینگ OneHot دو مقدار A و B (A و B دو مقدار OneHot در پیمانه m است)، را در نظر می‌گیریم که برای نمایش هر عدد در پیمانه m با کدینگ OneHot به m بیت نیاز داریم.

با توجه به طرح‌های ارائه شده در مقالات [۷] و [۱۰] و در نظر گرفتن این موضوع که در کدینگ OneHot در هر زمان تنها یک بیت از m بیت می‌تواند مقدار ۱ را داشته‌باشد، می‌توان با انجام شیفت چرخشی به چپ و راست مقدار داده را عوض کرد [۱۴,۱۳]. با شیفت دادن داده به سمت چپ موقعیت بیتی، بیت با مقدار یک افزایش می‌یابد و ارزش آن بالا می‌رود پس برای جمع می‌توانیم داده را به سمت چپ شیفت بدهیم، با انجام شیفت چرخشی به راست، می‌توان مقدار داده را کم کرد؛ چون بیت‌های کم ارزش داده در سمت راست قرار دارند و هر چه به سمت چپ می‌رویم ارزش بیت‌های داده بیشتر می‌شود؛ با شیفت دادن داده به سمت راست موقعیت بیتی، بیت با مقدار یک کاهش می‌یابد و می‌توان ارزش داده را کم کرد. اساس طراحی ساختار جمع‌کننده/تفریق‌کننده پیشنهادی همین نکته است.

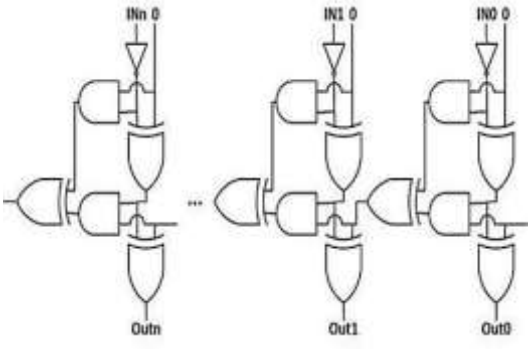
بنابراین اگر در یک عملیات محاسباتی تفریق عدد اول با کدینگ OneHot را به اندازه ارزش عدد دوم به راست و در عملیات محاسباتی جمع عدد اول با کدینگ OneHot را به اندازه ارزش عدد دوم به چپ شیفت بدهیم حاصل در هر پیمانه‌ای و با کدینگ OneHot بدست می‌آید. در نتیجه می‌توان با قرار دادن ورودی‌ها در شیفت رجیسترهای متشکل از فلیپ فلاپ، عمل تفریق و جمع در هر پیمانه‌ای با کدینگ OneHot را انجام داد. حال برای طراحی مدار با استفاده از این روش عدد B را در شیفت رجیستر راست قرار می‌دهیم که با هر شیفت به سمت راست بیت کم ارزش به‌عنوان کلاک برای شیفت رجیستر دیگر استفاده شود (این شیفت رجیستر برای جمع به سمت چپ و برای تفریق به سمت راست شیفت چرخشی می‌دهد و ورودی A به این شیفت رجیستر داده می‌شود) تا به تعداد کافی شیفت انجام شود. نکته مهم در این طراحی این است، که با توجه به نحوه انجام محاسبات پیمانه ای، استفاده از شیفت، سازگاری کامل با انجام جمع و تفریق به صورت پیمانه‌ای دارد. شکل (۲) ساختارکلی جمع‌کننده/تفریق‌کننده پیمانه‌ای پیشنهادی را به‌ازای پیمانه کلی m نشان می‌دهد.



شکل (۲): جمع‌کننده/تفریق‌کننده پیمانه‌ای پیشنهادی به‌ازای پیمانه‌ی کلی m

[1] Power-Delay Product
[2] One Hot Residue

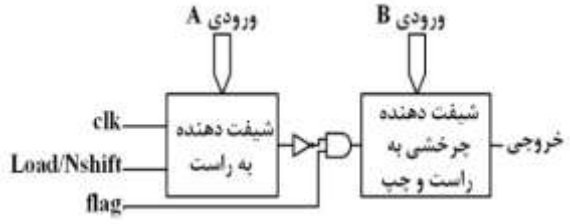
برای آنکه تعداد شیفت‌های عدد A مشخص شود می‌توان از دو روش مکمل دو کردن ورودی B و یا در نظر گرفتن بیت flag استفاده کرد. در روش اول ابتدا با استفاده از مدار مکمل در شکل (۳)، ورودی B را مکمل دو می‌کنیم. ورودی B به‌صورت کدینگ OneHot است مکمل دو باعث می‌شود شکل ظاهری عدد OneHot ما به گونه‌ای شود که تمامی بیت‌های با ارزش مکانی بیشتر از بیت با مقدار یک، همگی یک شوند.



شکل (۳): مدار مکمل دو به‌ازای ورودی n بیتی[۱۲]

از آنجا که تعداد گیت‌های مدار مکمل دو با توجه به تعداد بیت‌های عدد OneHot ورودی تغییر می‌کند و با افزایش تعداد بیت‌ها سخت‌افزار مدار نیز افزایش می‌یابد و باعث افزایش تأخیر، توان و ناحیه مصرفی مدار می‌شود، جهت رفع این مشکل سراغ روش جایگزین می‌رویم.

برای طراحی مداری با کارایی بهتر، به جای استفاده از مدار مکمل دو، از یک بیت کنترلی به اسم *flag* استفاده کردیم؛ به‌صورتی که مقدار بیت کنترلی صفر است، و پس از هر شیفت به راست اگر مقدار بیت کم ارزش عدد A یک شود مقدار بیت کنترلی نیز یک می‌شود و دیگر تغییر نمی‌کند. مدار ساده شده شکل (۴) ساختار کلی جمع‌کننده پیمانه‌ای پیشنهادی را به‌ازای پیمانه کلی m نشان می‌دهد که نسبت به ساختار قبلی با حذف مدار مکمل دو، دارای توان و ناحیه‌ی مصرفی خیلی کمتری است. از لحاظ توان مصرفی این مدار برابر با توان دینامیکی مدار ارائه شده از طریق [۱۰] تقریباً برابر است اما در کنار آن، سرعت محاسبه بالاتر می‌رود، و تأخیر کاهش می‌یابد.



شکل (۴): ساختار کلی جمع‌کننده/تفریق‌کننده پیمانه‌ای پیشنهادی به‌ازای پیمانه کلی m با کدینگ OneHot

همچنین باید در نظر داشته باشیم که با توجه به اینکه تفریق را در چه پیمانه‌ای (m) انجام می‌شود، از شیفت رجیسترهای m بیتی استفاده می‌گردد. مدار طراحی شده برای کلیه پیمانه‌ها قابل استفاده است، ولی با توجه به نوع نمایش اعداد در کدینگ OneHot بهتر است از پیمانه‌های کوچک با تعداد بیشتر استفاده شود. پس می‌توان برای هر مجموعه پیمانه‌ای با در نظر گرفتن n کوچک بهترین کارایی و سرعت را داشت.

۴- مبدل مستقیم پیشنهادی با کدینگ OneHot

سی معکوس تشکیل می‌شود. مبدل مستقیم و مبدل معکوس، برای ارتباط سیستم اعداد مانده‌ای، با دیگر مدارهای دیجیتال مورد نیاز هستند. اعداد ورودی اولیه یا خروجی نهایی سیستم اعداد مانده‌ای، معمولاً در فرم سیستم عددی متداول هستند. وظیفه‌ی مبدل مستقیم این است که فرم ده‌دهی یا باینری را به فرم مانده‌ای تبدیل کند. طراحی مبدل‌های مستقیم وابسته به پیمانه‌ها و ممکن است بر پایه lookup table (سایز جداول به اندازه‌ی پیمانه‌ها وابسته است)، مدارات منطقی-ترکیبی یا ترکیبی از هر دو باشد.

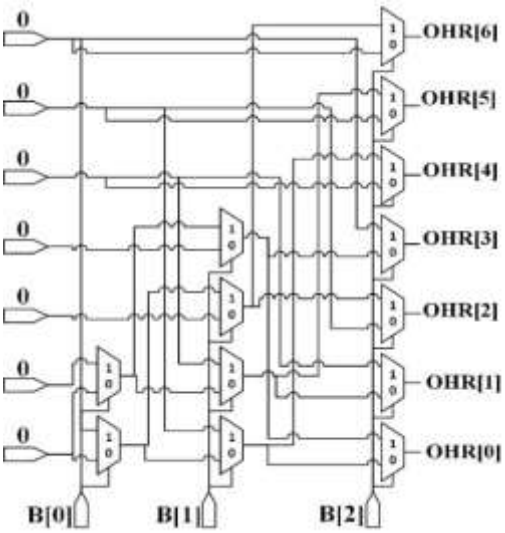
در سیستم اعداد مانده‌ای با کدینگ باینری، مبدل مستقیم عدد باینری را به عدد باینری در پیمانهm تبدیل می‌کند. اما برای استفاده کردن از کدینگ OneHot در قسمتی که عملیات پیمانه‌ای انجام می‌شود، لازم است مبدل مستقیمی طراحی شود که عدد باینری را به عدد OneHot در پیمانه‌ی m تبدیل کند، تا بتوان محاسبات خود را در بخش محاسباتی سیستم اعداد مانده‌ای با کدینگ OneHot انجام داد.

در حالت کلی برای تبدیل کردن یک عدد وزنی به عدد مانده‌ای باید عدد وزنی را بر پیمانه‌های سیستم تقسیم کرد. سپس باقی‌مانده‌های به دست آمده، ارقام عدد مانده‌ای را تشکیل می‌دهند. این فرایند در ظاهر به دلیل استفاده از عمل تقسیم،کند به نظر می‌آید. اما در طراحی‌ها، واقعاً عمل تقسیم انجام نمی‌شود و مبدل مستقیم با استفاده از جمع‌کننده‌های ماژولار پیاده‌سازی می‌شوند و دارای سربار کمی است.

از آنجا که اعداد ورودی به‌صورت باینری هستند، بنابراین تبدیل از باینری به عدد مانده‌ای در کدینگ OneHot را مورد بررسی قرار می‌دهیم. در صورتی که باقی‌مانده عدد در پیمانه‌ی m را داشته باشیم برای تبدیل ورودی باینری به کدینگ OneHot کافی است از یک دیکدر استفاده کنیم؛ اما در صورتی که بخواهیم عدد باینری را به عدد OneHot در پیمانه‌ی m تبدیل کنیم؛ اولین ایده، استفاده از طرح ارائه‌شده از طریق [۷] است.

طبق طرح [۷]، باتوجه به اینکه کدینگ یکی از ورودی‌ها به‌صورت باینری و کدینگ ورودی دیگر به‌صورت OneHot است؛

می‌توان با در نظر گرفتن مقدار صفر برای ورودی OneHot (برای مثال ورودی برابر با مقدار ۰۰۰۰۰۰۱ و m=7) و در نظر گرفتن مقدار دلخواه برای ورودی باینری، عدد باینری ورودی را به معادل پیمانه‌ای با کدینگ OneHot تبدیل کرد. بدین صورت که ورودی باینری با صفر جمع می‌شود و در نهایت خروجی برابر با مقدار معادل پیمانه‌ای با کدینگ OneHot عدد بایری ورودی است. با ثابت در نظر گرفتن مقدار ورودی OneHot (صفر در نظر گرفتن ورودی OneHot برای تبدیل ورودی باینری به معادل پیمانه‌ای با کدینگ OneHot) مدار ارائه شده در [۷] ساده‌تر می‌شود و برای پیمانه ۷ با ورودی باینری ۳ بیتی به شکل (۵) در می‌آید.



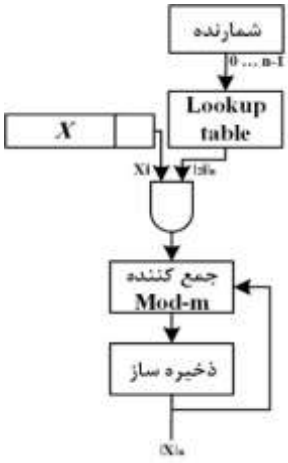
شکل (۵): مبدل مستقیم از عدد باینری به باقی‌مانده‌های OneHot در پیمانه ۷ و ورودی باینری ۳ بیتی

عیب این روش این است که با بزرگ شدن محدوده دینامیکی سیستم اعداد مانده‌ای یا بازه‌ی اعداد قابل نمایش و با افزایش تعداد بیت‌های ورودی باینری، تعداد سطوح استفاده از مالتی‌پلکسر افزایش می‌یابد و با بزرگ شدن پیمانه‌های مورد استفاده نیز تعداد مالتی‌پلکسرهای مورد نیاز برای طرحی مدار افزایش می‌یابد. در نتیجه مقدار مساحت و توان افزایش و میزان سرعت کاهش می‌یابد. ضمن اینکه هر سطح مالتی پلکسر وابسته به سطح قبلی است. بنابراین در این پژوهش با بررسی مبدل مستقیم در سیستم اعدادمانده‌ای با کدینگ باینری و با در نظر گرفتن مجموعه پیمانه‌ی$\{ 2^n-1 , 2^n , 2^n+1 \}$ برای اولین بار مدارهایی طراحی کردیم که با در نظر گرفتن ورودی باینری خروجی OneHot در پیمانه‌ی مربوطه را به ما بدهند و از نظر سخت‌افزار بهینه باشند.

با توجه به تعداد بیت‌های مورد نیاز برای نمایش عدد OneHot، استفاده از این کدینگ در پیمانه‌های کوچک مناسب‌تر است؛ زیرا با افزایش مقدار پیمانه‌ها برای نمایش اعداد مانده‌ای به تعداد بیت‌های بیشتری نیاز داریم. یکی از مهم‌ترین ملاحظات در زمان طراحی سیستم اعداد مانده‌ای، انتخاب مجموعه پیمانه مناسب است؛ زیرا بازه‌ی اعداد قابل نمایش، سرعت و پیچیدگی سخت‌افزار سیستم بستگی به تعداد و فرم پیمانه‌های انتخاب شده دارد. انتخاب پیمانه‌ها بر روی پیچیدگی مبدل‌های مستقیم و معکوس و مدارات حساب مانده‌ای مؤثر است. مبدل مستقیم و واحد حسابی برای برخی مجوعه‌های پیمانه‌ای از ساختار نسبتاً مشابهی برخوردار هستند و از آنجایی که مدارهای طراحی شده برای مجموعه‌پیمانه‌ی$\{ 2^n-1 , 2^n , 2^n+1 \}$ قابل استفاده در مدار سایر پیمانه‌ها است، از مجموعه پیمانه فوق برای طراحی استفاده کردیم. پس فقط کافی است که برای سه پیمانه‌ی مشهور، یک مدار مخصوص تبدیل مستقیم و نیز مدار جمع‌کننده و مدار ضرب‌کننده طراحی کنیم؛ بنابراین با در نظر گرفتن مجموعه پیمانه $\{ 2^n - 1 , 2^n , 2^n + 1 \}$ مبدل مستقیم را برای تک‌تک این پیمانه‌ها طراحی می‌کنیم.

۴-۱- طراحی مبدل مستقیم برای $\{2n-1,2n,2n+1\}$

برخی از مجموعه‌پیمانه‌های خاص درRNS به‌عنوان پیمانه‌های Low-cost شناخته می‌شود. به این دلیل که تبدیل به یا از این پیمانه‌ها می‌تواند به آسانی و بدون عملیات پیچیده انجام شود. از جمله عملیات، می‌توان به پیدا کردن معکوس ضربی و ضرب اشاره کرد. در شکل (۶) ساختار کلی مبدل مستقیم برای پیمانه‌ی کلی m با کدینگ باینری را نشان دادیم که با استفاده از جدول look-up مقدار 2^i در پیمانه m را با مقدار بیت iام عدد باینری X، and می‌کنیم و سپس حاصل را به‌ازای iهای متفاوت از طریق جمع‌کننده باینری در پیمانه m با یکدیگر جمع می‌کنیم تا حاصل به‌صورت مانده‌ای در پیمانه m به دست آید.



شکل(۶): طراحی مبدل مستقیم برای پیمانه‌ی کلی m با کدینگ باینری [۱۲]

مبدل مستقیم برای پیمانه 2^n: حداکثر تعداد بیت داده در محدوده‌ی دینامیکی RNS با مجموعه پیمانه $\{ 2^n + 1 , 2^n , 2^n - \}$

{ 1 برابر $3n$ بیت است. در مبدل مستقیم سیستم اعداد مانده‌ای با کدینگ باینری برای پیمانه 2^n کافی است n بیت کم ارزش عدد(B_3) را طبق اثبات رابطه (۲) تا (۵) جدا کنیم. عدد $3n$ بیتی X را به قسمت‌های n بیتی مشابه رابطه (۱) تقسیم می‌کنیم. باقی‌مانده قسمت‌هایی که ضریب 2^n و 2^{2n} دارد نسبت به 2^n صفر است[۱۲]:

$$X = B_1 2^{2n} + B_2 2^n + B_3 \quad (۲)$$

$$|2^n|_{2^n} = 0 \quad (۳)$$

$$|2^{2n}|_{2^n} = |2^n \times 2^n|_{2^n} = 0 \quad (۴)$$

$$\begin{aligned} r2 = |X|_{2^n} &= |B_1 \times 2^{2n} + B_2 \times 2^n \\ &\quad + B_3|_{2^n} \\ &= ||B_1 \times 2^{2n}|_{2^n} \\ &\quad + |B_2 \times 2^n|_{2^n} \\ &\quad + |B_3|_{2^n}|_{2^n} = B_3 \end{aligned} \quad (۵)$$

از آنجایی که 2^n بر پیمانه 2^nبخش‌پذیر است و 2^{2n} نیز ضریبی از 2^n است، در نتیجه باقی‌مانده‌ی آن‌ها به پیمانه 2^n برابر صفر است. اگر r_2 مانده‌ی عدد X در پیمانه 2^n باشد و باقی‌مانده‌ی دو قسمت $B_2 2^n$ و $B_1 2^{2n}$ در X در پیمانه 2^n برابر صفر می‌شود. بنابراین این پیمانه نیاز به سخت‌افزاری ندارد و برای به دست آوردن r_2 کافی است n بیت اول عدد را جدا کنیم.

برای طراحی مبدل مستقیم سیستم اعداد مانده‌ای با کدینگ OneHot و تبدیل باینری به عدد مانده‌ای با کدینگ OneHot، می‌توانیم از این ویژگی استفاده کنیم و تنها کافی است n بیت کم‌ارزش عدد را جدا کنیم و برای تبدیل آن به OneHot کافی است n بیت را به یک دیکدر n در 2^n بدهیم تا خروجی مورد نظر هم در پیمانه 2^nو با کدینگ OneHot به دست آید. پس برای مبدل مستقیم تبدیل باینری به عدد مانده‌ای OneHot در این پیمانه تنها به یک دیکدر نیاز داریم.

مبدل مستقیم برای پیمانه 2^n-1 : در مبدل مستقیم سیستم اعداد مانده‌ای با کدینگ باینری برای پیمانه 2^n-1طبق اثبات زیر کافی است عدد را به سه دسته n بیتی تقسیم کنیم و این دسته‌ها را مطابق رابطه (۶) جمع پیمانه‌ای کنیم [۱۲]:

$$|X|_{2^n-1} = |B_1 + B_2 + B_3|_{2^n-1} \quad (۶)$$

با در نظر گرفتن X، رابطه (۷) را داریم:

$$X = B_1 2^{2n} + B_2 2^n + B \quad (۷)$$

اگر $r3$ مانده‌ی عدد X در پیمانه 2^n-1 باشد با توجه به اینکه باقی‌مانده‌ی 2^n در پیمانه‌ی 2^n-1 برابر با یک است پس قسمت $B_2 2^n$ برابر با B_2 می‌شود و چون باقی‌مانده‌ی 2^{2n} در پیمانه 2^n-1 طبق اثبات زیر برابر با یک است. پس قسمت $B_1 2^{2n}$ برابر با B_1 می‌شود؛ و باقی‌مانده‌های هر عدد دودویی X نسبت به پیمانه 2^n-1 به‌صورت رابطه (۸) تا (۱۰) محاسبه می‌شود:

$$|2^n|_{2^n-1} = 1 \quad (۸)$$

$$|2^{2n}|_{2^n-1} = |2^n \times 2^n|_{2^n-1} = |1 \times 1|_{2^n-1} = 1 \quad (۹)$$

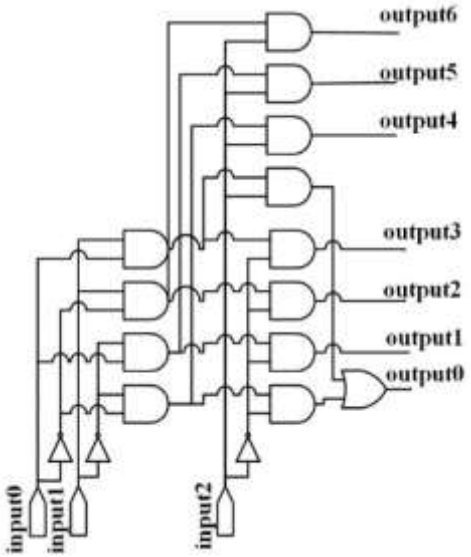
$$\begin{aligned} r3 = |X|_{2^n-1} &= |B_1 \times 2^{2n} + B_2 \times 2^n \\ &\quad + B_3|_{2^n-1} \\ &= ||B_1 \times 2^{2n}|_{2^n-1} \\ &\quad + |B_2 \times 2^n|_{2^n-1} \\ &\quad + |B_3|_{2^n-1}|_{2^n-1} \\ &= |B_1 + B_2 + B_3|_{2^n-1} \end{aligned} \quad (۱۰)$$

بر اساس این اثبات برای پیمانه2^n -1اگر بتوانیم هر n قسمت از عدد باینری را به عدد OneHot معادل در پیمانه‌ی 2^n -1 تبدیل کنیم می‌توان با جمع پیمانه‌ای این سه دسته با کدینگ OneHot عدد نهایی در پیمانه 2^n -1و با کدینگ OneHot را به دست آورد. پس به جای یک عدد $3n$ بیتی باینری اگر بتوانیم سه عدد n بیتی باینری را به OneHot تبدیل کنیم می‌توانیم با جمع عدد نهایی در پیمانه2^n -1و به‌صورت OneHot را به دست بیاوریم.

بدین منظور ضمن در نظر گرفتن چند سیگنال ثابت به‌عنوان عدد OneHot شروع که همان نمایش صفر در کدینگ OneHot (۰۰۰۰۰۱) است و $n=3$ مداری بر پایه‌ی گیت طراحی می‌کنیم، که با استفاده از این مدار بتوان هر عدد باینری $3n$ بیتی را به ۳ عدد m بیتی OneHot تبدیل کرد. با استفاده از یک متغییر برای ذخیره‌سازی حاصل جمع‌های این سه m بیت و یک مدار جمع‌کننده OneHot که در قسمت قبل بیان کردیم، حاصل نهایی را به‌صورت یک عدد m بیتی OneHot داشته باشیم. می‌توان گفت این مدار ساده‌ترین مبدل مستقیم سیستم اعداد مانده‌ای با کدینگ OneHot است. ساختار این مدار به‌گونه‌ای است که با استفاده از داده‌ی باینری موقعیت سیگنال‌های ثابت عوض می‌شوند تا خروجی به‌صورت OneHot نمایش داده شود. در سطح اول درصورتی که کم ارزش‌ترین بیت ورودی باینری یک باشد، یعنی سیگنال با مقدار یک، باید یک واحد به سمت بالا برود. به عبارتی سیگنال ۰ با سیگنال ۱ جا به جا شود. در سطح دوم درصورتی که دومین بیت ورودی باینری یک باشد، یعنی سیگنال با مقدار یک باید دو واحد به سمت بالا برود، چراکه بیت دوم باینری دارای ارزش مکانی ۲ است.

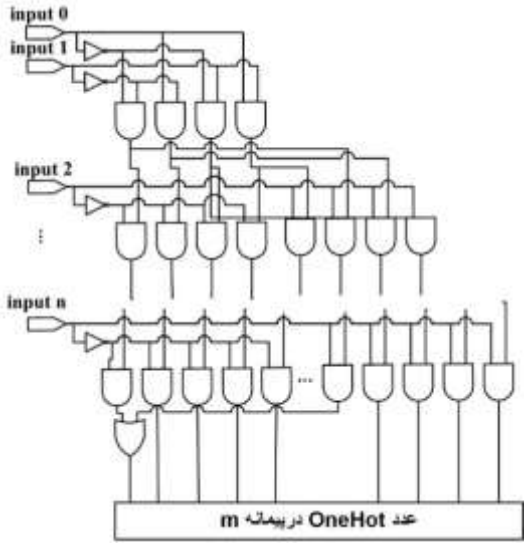
در سطح سوم، درصورتی که سومین بیت ورودی باینری یک باشد، یعنی سیگنال با مقدار یک باید ۴ واحد (چون بیت سوم باینری دارای ارزش مکانی ۴ است) به سمت بالا برود. به عبارتی یا سیگنال ۰ با سیگنال ۴ جا به جا شود که در این صورت بیت اول و دوم ورودی باینری صفر بوده و یا سیگنال ۱ با سیگنال ۵ جا به جا شود، که در این صورت بیت اول ورودی باینری یک و بیت دوم ورودی باینری مقدار صفر را داشتند، یا سیگنال ۲ با سیگنال ۶ جا به جا شود که در این صورت بیت اول ورودی باینری مقدار صفر و بیت دوم ورودی باینری دارای مقدار یک بوده است، یا سیگنال ۳ با سیگنال ۱ جا به جا شود که در این صورت بیت اول و دوم ورودی باینری یک بودند.

با توجه به توضیحات داده شده وضعیت مدار در هر سطح وابسته به بیت‌های قبلی است و از آنجایی که تمامی بیت‌ها در ورودی داده شده و سیگنال‌های استفاده شده هم دارای مقادیر ثابتی هستند وابستگی داده‌ای زیادی در مدار وجود ندارد و تأخیر حداقل است. برای ساده‌سازی مدار بیان شده سیگنال‌های ثابت را حذف کردیم و با توجه به داده باینری ورودی خروجی OneHot شده را به دست آوردیم. شکل (۷) مدار طراحی شده پیشنهادی را به‌ازای $n=3$ و پیمانه ۷ نشان می‌دهد. از آنجایی که مداری برای این کاربرد تاکنون طراحی نشده می‌توان گفت مدار پیشنهادی اولین مدار در این کاربرد است.



شکل (۷): مدار مبدل مستقیم طراحی شده پیشنهادی برای ورودی ۳ بیتی$n=3$ و پیمانه ۷

با گسترش تعداد بیت‌های ورودی باینری و نیز افزایش پیمانه، مدار فوق به شکل (۸) برای پیمانه m و ورودی n بیتی در می‌آید. ضمن اینکه پیمانه ما برابر با $m = 2^n - 1$ است.



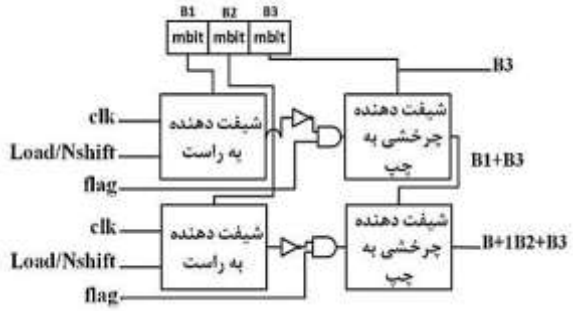
شکل (۸): مدار مبدل مستقیم تبدیل عدد n بیتی باینری به عدد مانده‌ای با کدینگ OneHot در پیمانه $m= 2^n-1$

در شکل (۸) معکوس بیت nام، به 2^{n-1}امین گیت and سمت چپ وارد می‌شود و از جواب گیت and $2^{n-1}+1$ام، برای ورودی گیت or در سطح آخر استفاده می‌کنیم؛ با توجه به اینکه حداکثر مقدار n بیت ورودی باینری همواره یکی بیشتر از پیمانه m است، در نتیجه چرخش داده را حداکثر برای یک بیت در مدار داریم به همین دلیل در سطح آخر یک گیت or استفاده می‌شود.

اگر بخواهیم به جای تقسیم عدد $3n$ بیتی باینری به سه قسمت n بیتی هر $3n$ بیت را با گسترش این مدار به OneHot تبدیل کنیم به‌ازای هر سطح که اضافه می‌شود، 2^n 1- گیت and و به دلیل چرخش‌های هر سطح، در سطح بعدی ۲ گیت or و در سطح بعد ۳ گیت or و برای تعداد سطح‌های اضافه شده بیشتر از پیمانه m به 2^n 1- گیت or، برای پیمانه 2^n 1- (2^n+1) گیت and و از سطح دوم به بعد 2^n 1+ گیت or برای پیمانه (2^n+1) نیاز داریم. زیرا با اضافه شدن تعداد سطوح بیشتر از n سطح تعداد جا به جایی‌های سیگنال‌ها به 2^n 1- می‌رسد. به عبارتی با اینکه همه بیت‌ها جا به جا نمی‌شوند اما در تصمیم‌گیری دخیل هستند که این نیازمند استفاده از گیت و مساحت بیشتری است. شکل (۹) نشانگر مبدل مستقیم طراحی شده پیشنهادی برای ورودی $3n$ بیتی و پیمانه m است؛ بنابراین تبدیل هر عدد باینری به سه عدد OneHot و سپس استفاده از جمع بین آن‌ها مدار بهینه‌تری برای مبدل مستقیم سیستم اعداد مانده‌ای با کدینگ OneHot است. این طراحی دارای سرعت بالاتری نسبت به حالتی است که در آن ابتدا عدد باینری $3n$ بیتی را به سه عدد n بیتی تقسیم کنیم و ابتدا آن‌ها را با هم جمع باینری کنیم و سپس اصل را با انکدر به OneHot تبدیل کنیم. با استفاده از روش اول دیگر انتشار بیت نقلی نداریم.

برای جمع مورد نیاز در این قسمت می‌توانیم از مدار جمع‌کننده/تفریق‌کننده پیشنهادی با کدینگ OneHot استفاده کرد که در این صورت این مدار به دلیل کاهش گیت‌های منطقی و استفاده از فلیپ فلاپ، سرعت بیشتری نسبت به طراحی‌های پیشین دارد و نیز می‌توان از مدار طراحی شده در [۱۰] استفاده کرد که از نظر مساحت و توان به‌صرفه‌تر است.

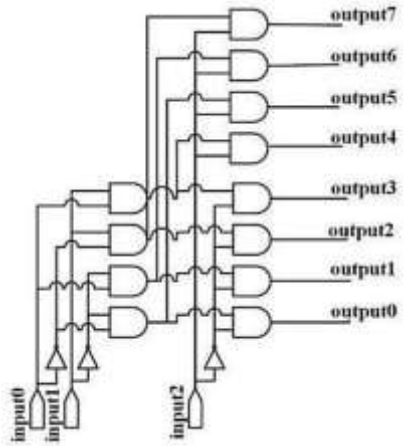
در شکل (۹) پیاده‌سازی سخت‌افزاری مبدل مستقیم در پیمانه 2^n–1 با کدینگ OneHot نشان داده شده است. همان‌گونه در شکل مشاهده می‌شود، در ابتدا X به سه قسمت m بیتی OneHot طبق شکل ۹ تبدیل شده است.



شکل (۹): مبدل مستقیم نسبت به مجموعه پیمانه2^n-1 و 2^n با کدینگ OneHot

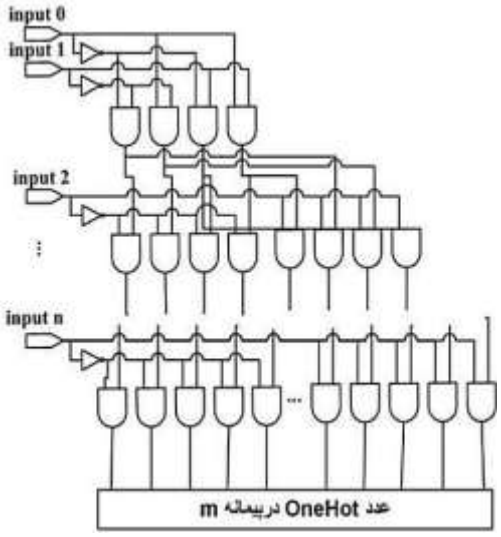
در مدار شکل (۹) پس از تبدیل سه قسمت عدد باینری n بیتی به‌صورت m بیت باکدینگ OneHot معادل در پیمانه2^n - 1 دو مقدار B_1 و B_3 را ابتدا با هم جمع می‌کنیم. سپس حاصل را به‌اندازه‌ی عدد B_2 به سمت چپ شیفت چرخشی می‌دهیم تا با این کار مقدار B_2 را به مقدار B_1+B_3 اضافه کرده باشیم.

مبدل مستقیم برای پیمانه2^n +1: شکل (۱۰) مبدل مستقیم طراحی شده برای تبدیل اعداد باینری n بیتی به OneHot در پیمانه2^n +1 را به‌ازای $n=3$ و پیمانه ۹ نشان می‌دهد. در شکل (۱۰)، خروجی $out8$ برابر با صفر است زیرا با سه بیت باینری فقط تا عدد ۷ را می‌توان نشان داد.



شکل (۱۰): مدار پیشنهادی برای تبدیل اعداد باینری n بیتی به OneHot به‌ازای n=3 و پیمانه 9

پس به جای یک عدد بیتی باینری می‌توانیم ۳ تا n بیت باینری را به OneHot تبدیل کنیم و با جمع و تفریق آن‌ها عدد نهایی در پیمانه 2^n +1 و به‌صورت OneHot را به دست بیاوریم. مدار تبدیل عدد n بیتی با پیمانه m به‌صورت شکل (۱۱) است.



شکل (۱۱): مدار تبدیل n بیت باینری به m بیت OneHot در $m=2^n+1$

باتوجه به اینکه پیمانه m است و ورودی باینری n بیتی است مقدارm برابر با 2^n+1 است. پس همواره مقدار ورودی باینری یکی کمتر از تعداد m است و ما چرخش نداریم. موارد فوق را می‌توان به‌صورت زیر خلاصه کرد:

عدد $3n$ بیتی X را می‌توان به‌صورت رابطه (۱) نوشت که قسمت‌های B_1 و B_2 و B_3 هر کدام دارای n بیت و $B_1,B_2,B_3 \leq 2^n-1$ هستند. باقی‌مانده‌ی هر عدد باینری X را نسبت به مجموعه پیمانه‌های $(2^n+1 , 2^n , 2^n-1)$، به‌صورت رابطه‌ی (۱۱) تا (۱۳) می‌توان محاسبه کرد:

$$|X|_{2^n-1} = |B_1 + B_2 + B_3|_{2^n-1} \quad (۱۱)$$

$$|X|_{2^n} = B_3 \quad (۱۲)$$

$$|X|_{2^n+1} = |B_1 - B_2 + B_3|_{2^n+1} \quad (۱۳)$$

در حالت کلی، اگر عدد X را به‌صورت m قسمت n بیتی در نظر بگیریم آنگاه X را می‌توان به‌صورت رابطه‌ی (۱۴) نوشت:

$$X = k_{m-1}2^{(m-1)n} + \cdots + k_2 2^{2n} + k_1 2^n + k_0 \quad (۱۴)$$

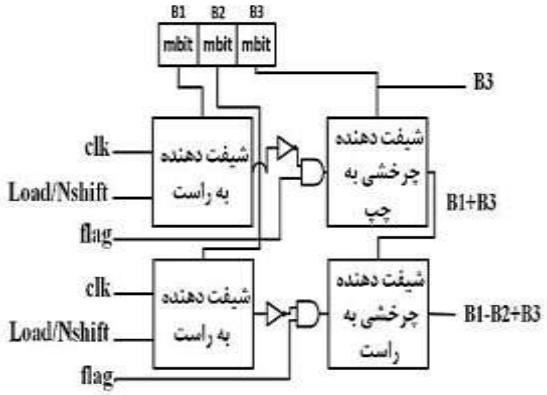
حال مانده‌های هر عدد دودویی X را نسبت به مجموعه پیمانه‌های$\{2^n-1 , 2^n , 2^n+1\}$، طبق رابطه‌ی (۱۵) تا (۱۷) به دست می‌آید.

$$|X|_{2^n-1} = |k_{1m-1} + \cdots + k_2 + k_1 + k_0|_{2^n-1} \quad (۱۵)$$

$$|X|_{2^n} = k_0 \quad (۱۶)$$

$$|X|_{2^n+1} = |k_{1m-1} - \cdots + k_2 - k_1 + k_0|_{2^n+1} \quad (۱۷)$$

پیاده‌سازی سخت‌افزاری مبدل مستقیم نسبت به پیمانه 2^n+1با کدینگ OneHot نشان داده شده است. که در ابتدا X به سه قسمت m بیتی OneHot طبق شکل (۱۲) تبدیل شده است.



شکل (۱۲). مبدل مستقیم نسبت به مجموعه پیمانه2^n+1 و 2^n با کدینگ OneHot

در مدار شکل (۱۲) پس از تبدیل سه قسمت عدد باینری n بیتی به‌صورت m بیت باکدینگ OneHot معادل در پیمانه$2^n + 1$ دو مقدار B_1 و B_3 را ابتدا با هم جمع می‌کنیم. سپس حاصل را به مقدار عدد B_2 به سمت راست شیفت چرخشی می‌دهیم تا با این کار مقدار B_2 را از B_1+B_3 تفریق کرده باشیم.

۵- ضرب‌کننده OneHot پیمانه‌ای پیشنهادی

در این بخش ساختار جدید برای ضرب‌کننده‌ی پیمانه‌ای بر اساس کدینگ OneHot پیشنهاد می‌دهیم. در انجام عمل ضرب با کدینگ OneHot، با در نظر گرفتن حافظه و استفاده از آن به‌عنوان ذخیره‌سازی اطلاعات lookup table مداری ساده و بهینه طراحی کردیم.

اگر دو عدد A و B را به‌عنوان دو عدد مانده‌ای در پیمانه m و با کدینگ OneHot در نظر بگیریم حاصل‌ضرب آن‌ها (Z) به‌صورت رابطه (۱۸) محاسبه می‌شود:

$$Z = |A \times B|_m \quad (۱۸)$$

بنابراین حاصل ضرب نیز در محدوده‌ی صفر تا m-1 قرار دارد؛ با ذخیره‌سازی حاصل‌ضرب (Z) در جدول و فعال کردن خانه‌هایlookup table ،از طریق مقدار دو ورودی A و B مدار ضرب‌کننده را طراحی کردیم؛ بدین منظور با در نظر داشتن دو ورودی A و B به‌عنوان دو عدد مانده‌ای در پیمانه‌ی m، به یک جدول $m\times m$ نیاز داریم که ورودی A را به‌عنوان فعال‌ساز سطر و ورودی B را به‌عنوان فعال‌ساز ستون قرار می‌دهیم.

از آنجایی که مقدار عدد A و B با کدینگ OneHot است پس فقط یک بیت مقدار یک را دارد در نتیجه با در نظر گرفتن ورودی‌ها به‌عنوان فعال‌ساز، فقط یک سطر و ستون از جدول مقدار یک را دارند، در این صورت در هر ضرب فقط یک خانه از جدول دارای مقدار سطر و ستون برابر با یک است؛ و ما حاصل $A\times B$ را در این خانه ذخیره می‌کنیم.

نحوه‌ی ذخیره‌سازی در جدول و استفاده از کدینگ‌های متفاوت باعث ایجاد مدارهای متفاوتی می‌شود؛ اگر از کدینگ باینری برای ذخیره‌سازی استفاده کنیم، پس از خواندن داده، برای استفاده و انجام محاسبات دیگر در سیستم اعداد مانده‌ای باید آن را به کدینگ OneHot تبدیل کنیم که باعث اضافه شدن مدار تبدیل مستقیم، در این قسمت طراحی می‌شود؛ اما از نظر ذخیره‌سازی چون تعداد بیت‌های کمتری در نمایش با کدینگ باینری استفاده می‌شود فضای ذخیره‌سازی کمتر است.

اگر از کدینگ OneHot برای ذخیره‌سازی استفاده کنیم، تعداد بیت‌های بیشتری برای نمایش OneHot مورد نیاز است. درنتیجه فضای ذخیره‌سازی بزرگ‌تری داریم. هر چه پیمانه ما بزرگ‌تر باشد فضای ذخیره‌سازی بزرگ‌تری مورد نیاز است که برای پیمانه‌های بزرگ‌تر به‌صرفه نیست.

در این طراحی ما برای ذخیره‌سازی اطلاعات از اعداد ده‌دهی استفاده کردیم و پس از خواندن مقدار هر خانه در جدول مقدار بیت در موقعیت عدد خوانده شده را برابر با یک قرار می‌دهیم. جدول ذخیره‌سازی یا lookup table برای پیمانه ۷ به صورت شکل (۱۳) است.

$\times_7$	1	2	3	4	5	6	7
1	0	0	0	0	0	0	0
2	0	1	2	3	4	5	6
3	0	2	4	6	1	3	5
4	0	3	6	2	5	1	4
5	0	4	1	5	2	6	3
6	0	5	3	1	6	4	2
7	0	6	5	4	3	2	1

شکل (۱۳): lookup table مدار ضرب‌کننده‌ی پیمانه ۷

برای مثال ضرب دو عدد مانده‌ای ۴ و ۶ در پیمانه ۷ طبق رابطه‌ی (۱۹) برابرست با:

$$|4|_7 \times |6|_7 = |24|_7 = |3|_7 \qquad (۱۹)$$

در جدول شکل بالا اگر عدد اول را ۰۰۱۰۰۰۰ (برابر با عدد ۴) در نظر بگیریم و آن را به سطر بدهیم و عدد دوم را ۱۰۰۰۰۰۰ (برابر با عدد ۶) در نظر بگیریم و آن را به ستون بدهیم (عدد اول از بیت کم ارزش تا بیت با ارزش به ترتیب از بالاترین سطر تا پایین‌ترین سطر وارد می‌شود و عدد دوم از بیت کم‌ارزش تا بیت با ارزش به ترتیب از چپ به راست روی ستون‌ها تأثیر می‌گذارند). طبق شکل (۱۴) با فعال شدن سطر ۴ و ستون ۶ حاصل خانه ای که در جدول فعال می‌شود برابر با ۳ است.

×۷	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0
0	0	1	2	3	4	5	6
0	0	2	4	6	1	3	5
0	0	3	6	2	5	1	4
1	0	4	1	5	2	6	3
0	0	5	3	1	6	4	2
0	0	6	5	4	3	2	1

شکل (۱۴): lookup table مدار ضرب‌کننده پیمانه ۷ با ورودی ۶ و ۴

مقدار ذخیره شده در خانه‌ای از جدول که دارای سطر و ستون با مقدار یک است برابر با ۳ است؛ پس از خواندن آن از جدول مقدار بیت سوم حاصل را برابر با یک قرار می‌دهیم. جدول ذخیره‌سازی یا lookup table برای پیمانه ۷ به صورت شکل (۱۵) است.

×۹	0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8
2	0	2	4	6	8	1	3	5	7
3	0	3	6	0	3	6	0	3	6
4	0	4	8	3	7	2	6	1	5
5	0	5	1	6	2	7	3	8	4
6	0	6	3	0	6	3	0	6	3
7	0	7	5	3	1	8	6	4	2
8	0	8	7	6	5	4	3	2	1

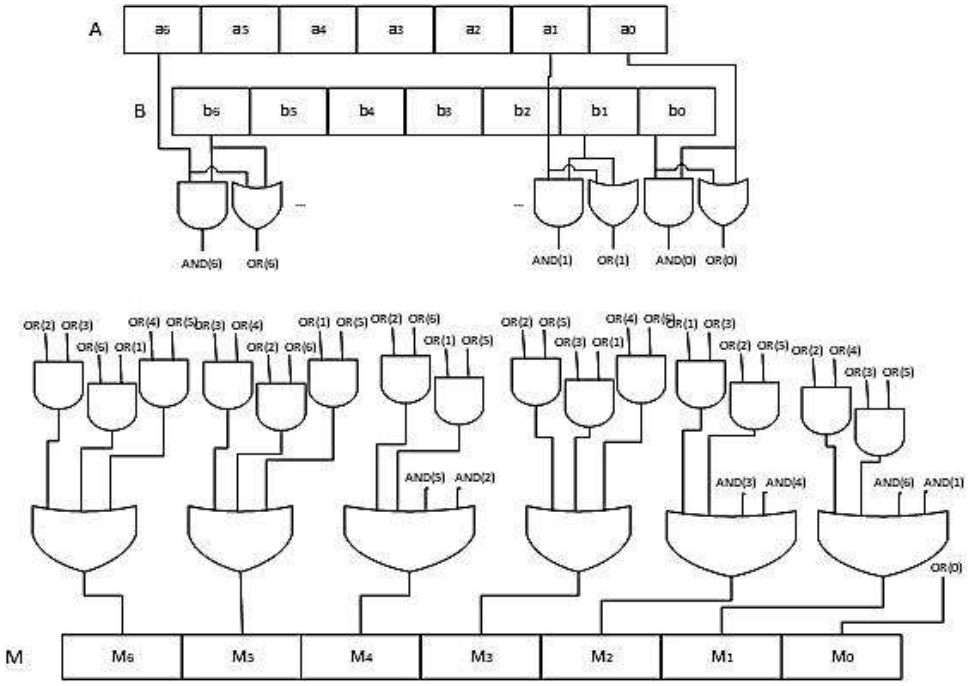
شکل (۱۵): lookup table مدار ضرب‌کننده‌ی پیمانه ۹

طراحی بعدی که برای ساختار ضرب‌کننده با کدینگ OneHot انجام دادیم با استفاده از گیت‌های پایه and و or است؛ فرض کنید A مقدار k را داشته باشد(a_k=1)، آنگاه مقدار مشخصی از B می‌تواند بیت i ام در خروجی ضرب‌کننده را برابر با ۱ کند (جواب ضرب‌کننده برابر با i شود). k می‌تواند مقادیر 0 تا m-1 را داشته باشد؛ زمانی که A=B باشد (A و B ورودی با کدینگ OneHot هستند) می‌توانیم با AND کردن بیت‌های متناظر در A و B، شماره بیتی را که در A و B مقدار ۱ را داشته است تعیین کنیم (در این حالت تنها خروجی یکی از گیت‌های AND برابر با مقدار ۱ می‌شود).

زمانی که A≠B باشد، با OR کردن بیت‌های متناظر در A وB می‌توانیم دو موقعیت بیتی که مقدار ۱ را دارند، مشخص کنیم (یک از این موقعیت‌های بیتی در A دیگری در B است). از آنجایی که عملیات ضرب است، تفاوتی نمی‌کند کدام یک از این دو موقعیت بیتی در A و کدام یک در B بوده است. اگر این دو بیت در موقعیت‌های بیتی iام و jام بوده باشند، دو حالت امکان پذیر است: ۱) ai=1 و bj=0 ۲) ai=0 و bj=1. بنابراین به m گیت OR دو ورودی در سطح اول احتیاج داریم.

زمانی که A=B باشد، با AND کردن بیت‌های متناظر در A و B می‌توانیم موقعیت بیتی که در هر دو ورودی مقدار ۱ را دارد، مشخص کنیم؛ بنابراین برای تشخیص حالتی که A=B است به m گیت AND دو ورودی در سطح دوم نیاز است.

با در نظر داشتن حاصل این دو سطح می‌توانیم با OR کردن حالت‌هایی که باعث به وجود آمدن خروجی یکسان می‌شوند، مدار را کامل کرد. شکل (۱۶) ساختار ضرب‌کننده‌ی پیمانه‌ای پیشنهادی به‌ازای پیمانه ۷ را نشان می‌دهد. ساختار این ضرب‌کننده نسبت به طرح ارائه شده از سوی وان و همکاران[۷] ساده‌تر و دارای تأخیر کمتری است. علاوه بر این در طرح هر دو اپرند از نوع OneHot است و تنها با استفاده از گیت‌های منطقی پیاده‌سازی شده که قبلاً مورد مشابه‌ای طراحی نشده است.



شکل (۱۶): ساختار ضرب‌کننده‌ی پیمانه‌ای پیشنهادی به‌ازای پیمانه ۷

ساختار کلی سیستم اعداد مانده‌ای با استفاده از کدینگ OneHot مطابق شکل ۱۷ خواهد بود.

شکل (۱۷): ساختار کلی سیستم اعداد مانده‌ای با استفاده از کدینگ OneHot

۶: نتایج شبیه‌سازی و پیاده‌سازی

برای مقایسه عملی ابتدا جمع‌کننده/تفریق‌کننده پیمانه‌ای پیشنهادی و طرح‌های معرفی شده اخیر را با استفاده از زبان برنامه نویسی VHDL[1] توصیف و صحت عملکرد آن‌ها را با نرم افزار ModelSim تست کرده‌ایم؛ سپس برای سنتز و پیاده‌سازی از ابزارهایDesign suite[2] ISE و Synopsys Design vision استفاده کردیم. جدول (۱) تا (۵) نتایج شبیه‌سازی جمع‌کننده‌های پیمانه‌ای پیشین و جمع‌کننده/تفریق‌کننده پیشنهادی با کدینگ OneHot را نشان می‌دهد. با توجه به نتایج شبیه‌سازی جمع‌کننده/تفریق‌کننده پیمانه‌ای پیشنهادی ضمن در نظر داشتن اینکه در شبیه‌سازی شماتیک پایه‌های کنترلی به مالتی‌پلکسر تشبیه شد و در نتیجه کد و شبیه‌سازی مدار طراحی شده با احتساب پایه shift-load و clk دارای مساحت و تأخیر بیشتری است. اما باز متوسط سرعت و زمان اجرا نسبت به طراحی‌های پیشین برای پیمانه‌های بزرگ‌تر از ۷ بیشتر است و نیز مدار پیشنهادی دو عملیات جمع و تفریق را محاسبه می‌کند.

با توجه به جدول (۱)، تأخیر طرح ارائه شده در این مقاله حدود ۱۰ درصد نسبت به طرح ارائه شده از طریق [۱۰] در پیمانه ۸ بهتر شده است که با افزایش پیمانه، زمان اجرا نیز افزایش می‌یابد. به طوری که در پیمانه ۹ حدود ۱۱درصد و در پیمانه ۱۱ حدود ۱۵ درصد تأخیر بهتر شده است و تأخیر طرح ارائه شده نسبت به طرح [۷] تیز به ازای پیمانه‌های بیشتر از ۵ همواره کمتر است. البته باید به این نکته هم توجه کرد که در طرح پیشنهادی در [۷] یکی از اپرندها از نوع باینری است و شبیه‌سازی این طرح نیز با فرض اینکه یک اپرند آن از نوع باینری و اپرند دیگر از نوع OneHot است، انجام گرفته است؛ در واقع اگر برای این طرح مداری در نظر گرفته شود که اپرند با کدینگ OneHot را به باینری تبدیل کند، توان و انرژی آن افزایش خواهد یافت.

[1] VHSIC Hardware Description Language

[2] Integrated Synthesis Environment

جدول (۱): نتایج شبیه‌سازی تأخیر جمع‌کننده/تفریق‌کننده پیمانه‌ای OneHot پیشنهادی

تاخیر (ns)			
جمع‌کننده [۱۰]	جمع‌کننده [۷]	جمع‌کننده/تفریق‌کننده پیشنهادی	پیمانه (m)
۲.۳۰	۳.۲۴	۲.۸۱	۵
۲.۷۷	۳.۴۱	۲.۸۷	۷
۳.۰۳	۳.۴۱	۲.۷۵	۸
۳.۲۸	۳.۴۹	۲.۹۲	۹
۳.۳۷	۳.۴۹	۳.۲۳	۱۱

با توجه به جدول (۲)، مساحت مدار پیشنهادی به ازای پیمانه‌های بزرگ‌تر از ۸ نسبت به طرح ارائه شده از طریق [۷] بهتر شده است، البته با در نظر داشتن اینکه در شبیه‌سازی شماتیک، پایه‌های کنترلی به مالتی‌پلکسر تشبیه شد؛ در نتیجه کد و شبیه‌سازی مدار طراحی شده با احتساب پایه shift-load و clk دارای مساحت بیشتری است.

جدول (۲): مقایسه‌ی مساحت جمع‌کننده/تفریق‌کننده پیمانه‌ای OneHot پیشنهادی

مساحت (um^2)			
جمع‌کننده [۱۰]	جمع‌کننده [۷]	جمع‌کننده/تفریق‌کننده پیشنهادی	پیمانه (m)
۶۱۴۶.۲۸۶	۷۷۷۰.۳۹۷	۸۱۹۲.۹۴۷	۵
۸۵۸۳.۱۰۹	۱۰۶۹۶.۰۳۶	۱۱۰۶۸.۳۰۰	۷
۹۹۵۱.۴۸۲	۱۲۱۶۲.۱۸۱	۱۳۵۰۴.۳۱۴	۸
۱۲۹۵۴.۹۳۵	۱۸۵۹۳.۶۴۱	۱۴۰۲۶.۹۱۱	۹
۱۷۵۵۶.۷۶۹	۲۳۶۵۶.۱۶۴	۱۷۰۶۸.۸۷۷	۱۱

جدول (۳)، نتیجه‌ی شبیه‌سازی و مقایسه مساحت× تأخیر مدار پیشنهادی با طرح‌های گذشته را نشان می‌دهد که طبق نتایج، توان مدار پیشنهادی نسبت به طرح ارائه شده از طریق[۷] به ازای پیمانه‌های بزرگ‌تر از ۵ و نسبت به طرح ارائه شده توسط [۱۰] به‌ازای پیمانه‌های بزرگ‌تر از ۹ بهتر شده است، البته با در نظر داشتن اینکه در شبیه‌سازی شماتیک پایه‌های کنترلی به مالتی‌پلکسر تشبیه شد؛ می‌توان مساحت×تأخیر مدار پیشنهادی را نزدیک به توان طرح‌های پیشین دانست و باید در نظر گرفت که با افزایش مقدار پیمانه فاصله مساحت×تأخیر دو مدار کمتر می‌شود.

جدول (۳): نتایج مساحت×تأخیر جمع‌کننده/تفریق‌کننده پیمانه‌ای OneHot پیشنهادی

مساحت×تاخیر			
جمع‌کننده [۱۰]	جمع‌کننده [۷]	جمع‌کننده/تفریق‌کننده پیشنهادی	پیمانه (m)
۱۳۵۲۱.۸۲۹۲	۲۵۱۷۶.۰۸۶	۲۳۰۲۲.۱۸۱	۵
۲۳۷۷۵.۲۱۱	۳۶۴۷۳.۴۸۲	۳۱۷۶۶.۰۲۱۰	۷
۳۰۱۵۲.۹۹۰	۴۱۴۷۳.۰۳۷	۳۴۳۸۶.۸۶۳	۸
۴۲۴۹۲.۱۸۶	۶۴۸۸۸.۳۱۷	۴۰۹۵۸.۵۸۰	۹
۵۹۱۶۶.۳۱۱	۸۲۵۶۰.۱۲۳	۵۵۱۳۲.۴۷۳	۱۱

جدول (۴)، مقدار توان×تأخیر را در مدار پیشنهادی و طرح ارائه شده در[۱۰] و [۷] را نشان می‌دهد. که طرح ارائه شده [۱۰] مقدار توان×تأخیر کمتری نسبت به بقیه طرح‌ها داشته است، در پیمانه‌های بیشتر از ۷ مقدار توان×تأخیر در مدار پیشنهادی، تنها نسبت به طرح ارائه در [۷] بهتر شده‌است.

جدول (۴): نتایج توان×تأخیر جمع‌کننده/تفریق‌کننده پیشنهادی OneHot

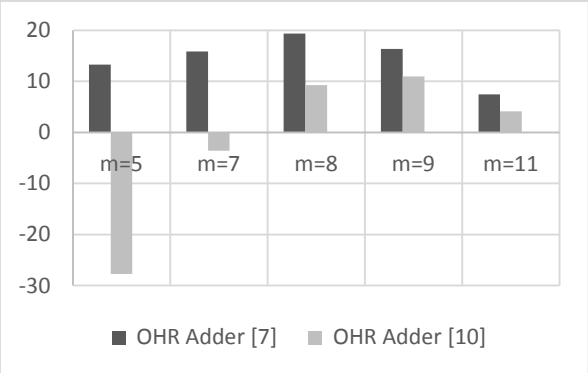
توان×تاخیر			
جمع‌کننده [۱۰]	جمع‌کننده [۷]	جمع‌کننده/تفریق‌کننده پیشنهادی	پیمانه (m)
۰.۹۰	۱.۵	۱.۸۲۳	۵
۱.۵۷	۲.۱	۲.۵۶۵	۷
۲.۰۹	۲.۳۶۳	۲.۸۱۳	۸
۲.۴۷	۳.۵۱	۳.۳۴۶	۹
۳.۲۳	۴.۶۱۷	۴.۶۲۸	۱۱

جدول (۵)، مقدار تأخیر به توان ۲ ×مساحت را در مدار پیشنهادی و طرح ارائه شده در [۱۰] و [۷] نشان می‌دهد. که در طرح پیشنهادی ارائه شده، به‌ازای تمام پیمانه‌ها مقدار تأخیر به توان ۲ ×مساحت بیشتر از طرح ارائه شده در [۷] و به ازای پیمانه‌های بیشتر از ۸، نسبت به طرح ارائه شده در [۱۰] بیشتر است.

جدول (۵): مقایسه‌ی تأخیر به توان ۲ ×مساحت جمع‌کننده/تفریق‌کننده پیشنهادی

تاخیر^2 × مساحت			
جمع‌کننده [۱۰]	جمع‌کننده [۷]	جمع‌کننده/تفریق‌کننده پیشنهادی	پیمانه (m)
۲۹۷۴۸.۰۲۴	۸۱۵۷۰.۵۱۹	۶۴۶۹۲.۳۳۱	۵
۶۵۸۵۷.۳۳۷	۱۳۴۳۷۴.۵۷۶	۹۱۱۶۸.۴۸۶	۷
۹۱۳۶۳.۵۶۱	۱۴۱۴۲۳.۰۵۶	۹۴۵۶۳.۸۷۵	۸
۱۳۹۳۷۴.۳۷۲	۲۳۶۴۶۰.۲۲۶	۱۱۹۵۹۹.۰۵۶۵	۹
۱۹۹۳۹۰.۴۶۹	۲۸۸۱۳۴.۴۴۳	۱۷۸۰۷۷.۸۸۹	۱۱

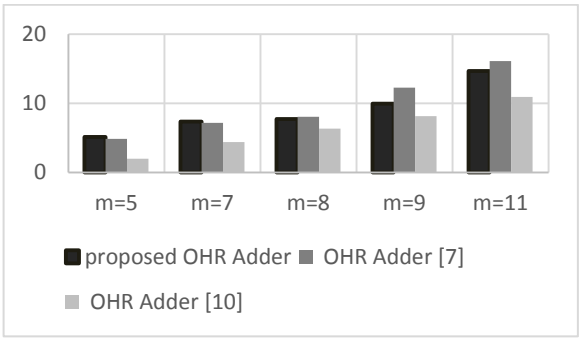
شکل (۱۸) میزان بهبود تأخیر جمع‌کننده/تفریق‌کننده پیمانه‌ای پیشنهادی یک بار نسبت به طرح ارائه شده در [۱۰] و یک بار دیگر نسبت به [۷] را نشان می‌دهد. همان گونه در شکل مشاهده می‌کنید، تأخیر طرح پیشنهادی نسبت به [۷] همواره به‌ازای پیمانه‌های ۵، ۷، ۸، ۹، ۱۱ بهبود داشته است. همچنین تأخیر طرح پیشنهادی نسبت به [۱۰] به جز در پیمانه‌های ۵ و ۷، در بقیه‌ی موارد کاهش خوبی داشته است.



شکل (۱۸): بهبود تأخیر جمع‌کننده/تفریق‌کننده پیشنهادی نسبت به طرح ارائه شده در [۱۰] و [۷]

در زمینه‌ی توان مصرفی مدارهای پیشنهادی باید به آن توجه کرد این است که ورودی مدارها در کدینگ OneHot تنها رنج کمی از اعداد باینری را شامل می‌شوند. به‌عنوان مثال در کدینگ OneHot هر عدد ورودی ۷ بیتی تنها به‌ازای ۷ مقدار از ۲۷ ترکیب باینری معتبر است؛ بنابراین با تغییر در مقادیر ورودی در هر زمان توان مصرفی و تعداد سوئیچینگ‌های مدار نسبت به کدینگ باینری کاهش می‌یابد.

شکل (۱۹) میزان انرژی×تأخیر و شکل (۲۰) مقدار انرژی را برای جمع‌کننده/تفریق‌کننده با کدینگ OneHot به ترتیب برای طرح پیشنهادی و طرح ارائه شده در [۱۰] و [۷] در پیمانه ۵، ۷، ۸، ۹، ۱۱ نشان می‌دهد.



شکل (۱۹): میزان انرژی×تأخیر جمع‌کننده/تفریق‌کننده پیشنهادی،جمع‌کننده ارائه شده در [۱۰] و [۷]

شکل (۲۰): مقدار انرژی جمع‌کننده/تفریق‌کننده پیشنهادی،جمع‌کننده ارائه‌شده در [۱۰] و [۷]

انرژی جمع‌کننده/تفریق‌کننده پیمانه‌ای پیشنهادی OneHot به ترتیب ۴٪، ۲٪ به‌ازای پیمانه‌های ۹ و ۱۱ نسبت به جمع‌کننده پیمانه‌ای ارائه شده در [۱۰] کاهش یافته است؛ این درحالی است که این جمع‌کننده/تفریق‌کننده پیشنهادی توانسته مقدار انرژی×تأخیر را نیز به ترتیب ۲۰٪، ۹٪ به‌ازای پیمانه‌های ۹ و ۱۱ نسبت به جمع‌کننده پیمانه‌ای ارائه شده در [۷] کاهش دهد.

بر اساس نتایج شبیه‌سازی درصد بهبود سرعت جمع‌کننده پیمانه‌ای پیشنهادی با کدینگ OneHot نسبت به طرح ارائه شده در [۱۰] به‌ازای پیمانه‌های ۸، ۹ و ۱۱ به ترتیب برابر ۵٪، ۱۱٪، ۱۶٪ شده‌است.

جمع‌کننده‌ی پیمانه‌ای باکدینگ OneHot، مساحت کمتر و سرعت بیشتری نسبت به جمع‌کننده‌های پیمانه‌ای با کدینگ باینری دارد. جمع‌کننده‌ی پیمانه‌ای باکدینگ OneHot ارائه شده در [۷] مساحت کمتری نسبت به جمع‌کننده پیمانه‌ای باینری دارد؛ از آنجایی که جمع‌کننده/تفریق‌کننده پیمانه‌ای با کدینگ OneHot ارائه شده در این پژوهش از نظر سرعت و مساحت بهینه‌تر از طرح ارائه شده در [۷] است، می‌توان به‌صورت مشابه نتیجه گرفت جمع‌کننده طراحی شده پیشنهادی از نظر سرعت و مساحت بسیار بهتر از جمع‌کننده‌های پیمانه‌ای با کدینگ باینری است. از آنجایی که جمع‌کننده/تفریق‌کننده طراحی شده پیشنهادی از نظر سرعت بهینه‌تر از طرح ارائه شده در [۱۰] است

و با در نظر گرفتن این موضوع که طرح ارائه شده در [۱۰] آخرین طرح ارائه شده در این زمینه است که طبق ادعای ایشان این طرح بهترین مدار جمع‌کننده طراحی‌شده تا سال ۲۰۱۸، از نظر مساحت، توان و سرعت است؛ می‌توان نتیجه گرفت که طرح ارائه شده پیشنهادی سریع‌ترین مدار جمع‌کننده/تفریق‌کننده پیمانه‌ای با کدینگ OneHot ارائه شده است.

۶-۴- نتایج شبیه‌سازی ضرب‌کننده و مبدل مستقیم پیشنهادی

در این بخش تأخیر، سرعت و توان ضرب‌کننده‌ی پیمانه‌ای OneHot و مبدل مستقیم باینری به OneHot مورد بررسی قرار گرفته است و لازم به ذکر است چون مدار ضرب‌کننده‌ی پیمانه‌ای OneHot طراحی شده از طریق گیت‌های منطقی تاکنون ارائه نشده است. پارامترهای ذکر شده در جدول (۶) قابل مقایسه با طرح دیگری نبودند.

جدول (۶): نتایج شبیه‌سازی ضرب‌کننده‌ی پیمانه‌ای و مبدل مستقیم پیشنهادی

پیمانه m=7	ضرب‌کننده‌ی پیمانه‌ای OneHot پیشنهادی با استفاده از lookup table	ضرب‌کننده‌ی پیمانه‌ای OneHot پیشنهادی با استفاده از گیت	مبدل مستقیم تبدیل باینری به OneHot پیشنهادی
تاخیر(ns)	۳.۰۱	۱.۷۹	۰.۸۲
مساحت (um2)	۹۰۹۰۰.۱۰۹	۵۶۱۸.۷۲۱	۱۷۷۸.۳۷۶
توان (mW)	۰.۰۰۵۲۴	۰.۲۱۹۰۳۵	۰.۰۴۲۳۵

طبق نتایج به دست آمده مقدار توان مصرفی در ضرب‌کننده پیمانه‌ای OneHot پیشنهادی با استفاده از lookup table نسبت به ضرب‌کننده پیمانه‌ای OneHot پیشنهادی با استفاده از گیت کمتر است. اما سرعت ضرب‌کننده‌ی پیمانه‌ای OneHot پیشنهادی با استفاده از گیت بیشتر و دارای مساحت کمتری نسبت به ضرب‌کننده‌ی پیمانه‌ای OneHot پیشنهادی با استفاده از lookup table است.

۷- نتیجه‌گیری

انجام محاسبات سریع یک نیاز ضروری سیستم‌های دیجیتال و هوشمند امروزی است و در این زمینه تحقیقات بسیاری صورت گرفته است و واحد محاسبه و منطق تأثیر بسزایی در عملکرد سیستم‌ها دارد. در چنین کاربردهایی می‌توان از سیستم اعداد مانده‌ای استفاده کرد و بهبود کارایی محاسبات می‌تواند منجر به بهبود عملکرد سیستم گردد. با توجه به عدم انتشار بیت نقلی در کدینگ OneHot، جمع‌کننده‌های طراحی شده با این کدینگ سرعت بیشتری نسبت به جمع‌کننده‌های باینری طراحی شده در این سیستم دارند. علاوه‌براین چون نمایش اعداد در این کدینگ برای داده n بیتی مجاز است (تعداد محدودی از تمام حالات 2^n)، در نتیجه تغییرات ورودی‌های جمع‌کننده محدود و فعالیت سوئیچینگ مدار کمتر است که منجر به توان مصرفی دینامیک کمتر نسبت به کدینگ باینری خواهد شد [۱۰]. در این مقاله سعی شد مداراتی برای بهبود سیستم اعداد مانده‌ای معرفی شود که سریع باشند. با توجه به استفاده از کدینگ OneHot مدارهای برای انجام عملیات محاسباتی جمع، ضرب و تفریق معرفی کردیم و نیز ساختاری برای تبدیل کدینگ باینری به عدد پیمانه‌ای با کدینگ OneHot معرفی کردیم.

در کلیه طراحی‌های انجام شده در این پژوهش از کدینگ OneHot استفاده کردیم. که مهم‌ترین مزیت استفاده از این کدینگ برای مثال در جمع‌کننده نسبت به نوع باینری، حذف تأخیر حاصل از انتشار بیت نقلی است. علاوه‌براین طراحی جمع‌کننده باینری پیمانه‌ای وابسته به پیمانه است و اگر بخواهیم آن را به‌صورت پیمانه کلی m طراحی کنیم، تأخیر آن برابر با $(+Delay_{Mux}2\times n+1)Delay_{full-adder}$ می‌شود که در آن n برابر تعداد بیت‌ها و $Delay_{Mux}$ تأخیر مالتی‌پلکسر است. درحالی‌که جمع‌کننده/ تفریق‌کننده پیشنهادی با کدینگ OneHot حداکثر پس از تأخیر m کلاک، خروجی آن آماده است. بنابراین استفاده از این کدینگ باعث افزایش سرعت و کاهش تأخیر بر حسب نانوثانیه، در انجام عملیات می‌گردد.

در این پژوهش، جمع‌کننده/تفریق‌کننده پیمانه‌ای برای داده‌های کوچک، براساس کدینگ OneHot جهت افزایش سرعت عملیات پیمانه‌ای طراحی شده است که نسبت به مدارهای جمع‌کننده پیشین، دارای تأخیر کمتری است و علاوه‌براین قادر به انجام عملیات جمع و تفریق است و برای انجام عملیات تفریق به مدار مجرایی نیاز نداریم. انرژی جمع‌کننده/تفریق‌کننده پیمانه‌ای پیشنهادی OneHot در پیمانه‌های ۹ و ۱۱ نسبت به جمع‌کننده پیمانه‌ای ارائه شده در [۱۰] کاهش یافته است. علاوه‌براین، به علت نیاز بسیاری از کاربردها به ضرب‌کننده، برای تکمیل کار خود، ضرب‌کننده‌ی مناسب سیستم در دو طرح متفاوت نیز طراحی شده است که یک طرح بر مبنای گیت‌های پایه و طرح دیگر با استفاده از lookup table. طراحی شده است و هر دو ورودی نیز کدینگ OneHot هستند. در بخش دیگری برای اولین بار روشی بهینه برای طراحی مدار مبدل مستقیم در سیستم اعدادمانده‌ای با کدینگ OneHot ارائه دادیم، که عدد باینری را به عدد مانده‌ای با کدینگ OneHot تبدیل می‌کند. استفاده از مبدل مستقیم برای انجام محاسبات در سیستم اعداد مانده‌ای ضروری است و چون برای انجام محاسبات مانده‌ای با

کدینگ OneHot نیاز به مبدلی برای تبدیل عدد باینری بـه عـدد مانده‌ای با کدینگ OneHot داریم بـرای اولـین بـار ایـن مـدار را طراحی کردیم. عملکرد همه طرح‌های ارائـه شـده بـا اسـتفاده از شبیه‌سازی تحلیل و بررسی شد.

۸- مراجع

[1] Cui, X., Liu, W., Wang, S. et al. Design of High-Speed Wide-Word Hybrid Parallel-Prefix/Carry-Select and Skip Adders. J Sign Process Syst 90, 409–419 (2018). https://doi.org/10.1007/s11265-017-1249-3

[2] Ebrahim, A., Gebali, F., "Optimized structures of hybrid ripple carry and hierarchical carry lookahead adders", Microelectronics Journal, Vol. 46, No. 9, 2015.

[3] Jayashree, H, V., Vuggirala, Jaswanth, Patil, Ghanshyam, N., "Design of High Speed Area Efficient OHRNS Data Path Subsystems for FFT Implementation", 4th International Conference on Electronics and Communication Systems (ICECS), 2017.

[4] Garner, H. L., "The Residue Number System," in IRE Transactions on Electronic Computers, vol. EC-8, no. 2, pp. 140-147, June 1959.

[5] Chang, Chip, H., Sabbagh, M., Amir, Emrani, Z., Azadeh, A., Tay, Tian, F., "Residue Number Systems: A New Paradigm to Datapath Optimization for Low-Power and High-Performance Digital Signal Processing Applications", IEEE Circuits and Systems Magazine, Vol. 15, No. 4, 2015.

[6] Schinianakis, D., Stouraitis, T., "Multifunction residue architectures for cryptograph", IEEE Trans. Circuits Syst, Vol. 61, No. 4, 2014.

[7] Vun, C. H., Premkumar, A. B., & Zhang, W,. "A new RNS based da approach for inner product computation", IEEE Transactions on Circuits and Systems I: Regular Papers, Vol. 60, No. 8, 2013.

[8] Chren, W. A., "Low delay-power product CMOS design using one-hot residue coding", International Symposium on Low Power Electronics and Design (ISLPED), 1995.

[9] Chren, W., A., "One-hot residue coding for low delay-power product CMOS design", IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, Vol. 45, No. 3, 1998.

[10] Jaefarzadeh, F., Sabbagh, Molahosseini., A., Emrani, Z., Azadeh, A., Sousa, Leonel, "Efficient Modular Adder Designs Based on Thermometer and One-Hot Coding", IEEE Transactions on Very Large-Scale Integration (VLSI) Systems, Vol. 27, No. 9, 2019.

[11] Mohammadi, Gh., M. , Ghavami, B., Pedram, H., "An Adder/Multiplier Circute for One Hot Residue Number System", International Journal of Engineering Research & Technology, Vol. 3, No. 11, 2014.

[12] Omondi, A., Premkumar, B., "Residue Number System: Theory and implementations", Imperial College Press, 2007.

[13] Morris Mano, M., "digital design (Logic Circuit)", Pearson, Chapter 1, Section 1-5, 1927.

[۱۴] فریده صباح، آزاده السادات عمرانی زرندی، «بررسی ساختار جمع‌کننده‌های با کدینگ باینری و وان‌هات و استفاده از آن‌ها در ساختار سیستم اعداد مانده‌ای»، کنفرانس ملی محاسبات توزیعی و پردازش داده‌های بزرگ، ۱۳۹۹.