ارائه روشی کارآمد و سریع جهت تشخیص سرریز برای مجموعه پیمانه $\{2^n-1, 2^n, 2^n+1\}$

مرضیه سادات امیرشاکرمی[*(۱)] مهدی حسین زاده[(۲)] علی آستانه اصل[(۳)]

(۱) دانشجوی کارشناسی ارشد، دانشگاه آزاد اسلامی، واحد اراک، گروه کامپیوتر، اراک، ایران.

(۲) استادیار،دانشگاه آزاد اسلامی، واحد علوم و تحقیقات تهران، گروه کامپیوتر، تهران، ایران.

(۳) استادیار،دانشگاه آزاد اسلامی، واحد اراک، گروه ریاضی، اراک،ایران.

Provide Efficient and Speed Method for Detecting Overflow for the Moduli Set $\{2^n-1, 2^n, 2^n+1\}$

Marzieyh sadat Amirshakarami [(1)]* *Mehdi Hosseinzadeh [(2)]* *Ali Astaneh Asl [(3)]*

[(1)] Department of Computer Engineering, Islamic Azad University, Arak Branch, Arak, Iran

[(2)] Department of Computer Engineering, Islamic Azad University, Science and Research Branch, Tehran, Iran

[(3)] Department of Mathematic, Islamic Azad University, Arak Branch, Arak, Iran

[*] عهده دار مکاتبات

نشانی: دانشگاه آزاد اسلامی، واحد اراک، گروه کامپیوتر، اراک، ایران.

تلفن: ۰۸۶-۳۳۶۶۳۰۴۱-۳۳۶۶۳۰۴۹ **پست الکترونیکی**: ***marzy.amshka@gmail.com***

ارائه روشی کارآمد و سریع جهت تشخیص سرریز در مجموعه پیمانه $\{2^n-1, 2^n, 2^n+1\}$

چکیده

امروزه با توجه به پیشرفت تکنولوژی و فناوری اطلاعات نیاز به محاسبات سریع داده امری ضروری می باشد. سیستم اعداد مانده‌ای، سیستمی نامتعارف و غیروزنی است که محاسبات به صورت موازی روی باقیمانده‌های تقسیم عدد بر چندین پیمانه، انجام می‌شود. انجام اینگونه محاسبات باعث بالا رفتن سرعت محاسبات و کاهش توان مصرفی می‌گردد. یکی از مشکلات این سیستم انتشار سرریز است که به دلیل انجام عملیات پیمانه‌ای و طبیعت بی وزن بودن اعداد است. برای رفع این مشکل در بسیاری از موارد استفاده از مبدل برعکس و تبدیل اعداد به سیستم وزن‌دار جهت مقایسه و شناسایی سرریز ضروری است. در این مقاله الگوریتم جدیدی به منظور بهبود شناسایی سرریز بر روی مجموعه پیمانه$\{2^n-1, 2^n, 2^n+1\}$ ارائه می دهیم که نیازی به انجام مقایسه کامل و فرآیند تبدیل معکوس ندارد. همچنین روش پیشنهادی در مقایسه با روش های قبلی، علاوه بر مولفه های سخت افزاری کمتر، تاخیر در آن بسیار پایین است.

واژگان کلیدی: سیستم اعداد مانده ای، شناسایی سرریز، مجموعه پیمانه $\{2^n-1, 2^n, 2^n+1\}$، مضرب مشترک.

Provide Efficient and Speed Method for Detecting Overflow in Module Set $\{2^n-1, 2^n, 2^n+1\}$

Abstract

Nowadays, due to advances in technology and information technology fast processing of data is a growing need. Residue Number System is an unconventional and non-Weighted System that parallel computing on remainders of dividing number on several modules is performed. The computations are performed cause to increase the speed of computing and reduce the power consumption.

One of the problems in this system is published overflow that due to the modular operations and nature of non-weighting numbers. In order to solve this problem in many cases using to binary number system converter and convert the number to the weighted numbers to comparison and overflow detection is necessary.

In this article, the novel methods to improve the detection of overflow in moduli set {2^n-1, 2^n, 2^n+1} without doing a full comparison and need the reverse convert process. Also The proposed method is compared with previous approaches, in addition to fewer hardware components, the delay is very low.

Key words: ***Residue Number System(RNS), Overflow Detection, moduli set {2^n-1, 2^n, 2^n+1},Common Multiple.***

۱-مقدمه

نمایش اعداد می‌تواند اشکال مختلفی داشته باشد. به عنوان مثال می‌توان به سیستم اعداد مانده‌ای[1] اشاره کرد. سیستم اعداد مانده‌ای یکی از مباحث مطرح در حساب کامپیوتری سریع می‌باشد که از نیمه‌های قرن بیستم مورد تحقیق و بررسی بوده است [۱]. این سیستم، یک سیستم غیر وزنی است که با مجموعه ای از پیمانه ها مشخص می شود. در صورتی که تمامی پیمانه ها نسبت به هم دو به دو اول باشند سیستم بهینه است. همچنین می توان سرعت محاسبات را با کم کردن تاخیر افزایش داد و توان مصرفی را نیز بدین صورت کاهش داد[۲].

سیستم اعداد مانده‌ای چون دارای خواص جمع و ضرب بدون رقم نقلی و تفریق بدون رقم قرضی بین پیمانه‌ای است در محاسبات سرعت بالا و در کاربردهای پردازش سیگنال دیجیتال[2][۳]، فیلترهای دیجیتال[3]، پردازش تصویر[4] [٤]، رمزنگاری از جمله در الگوریتم رمزنگاری***RSA***[5] [٥]و ارتباطات دیجیتال، بطور کلی در کاربردهایی که در یک محدوده از اعداد، اعمال جمع و تفریق و ضرب تکرار می‌شوند بسیار کاربرد دارد. علاوه براین، در این سیستم بدلیل اینکه محاسبات روی باقیمانده‌ها بطور جداگانه انجام می‌شود. اگر خطایی روی یکی از این باقیمانده‌ها اتفاق بیفتد تاثیر آن بر دیگر پیمانه‌ها منتقل نمی‌گردد در حالیکه در سیستم های متداول همانند مبنای دودویی یا مبنای دهدهی، عملیات ریاضی نیازمند انتشار رقم نقلی هستند. به عبارت دیگر معماری‌های RNS ذاتاً در مقابل خطا تحمل‌پذیر هستند به همین دلیل می توان آن را در محیط های نویزدار مانند شبکه های حسگر بیسیم[6]به منظور کدینگ ارتباطات استفاده کرد که با افزودن پیمانه های اضافی[7]به سیستم، می توان تشخیص و تصحیح خطا[8]را به آسانی فراهم نمود[٦].

مهمترین پارامترها در انجام عملیات حسابی عبارتند از سرعت، سربار سخت افزاری و توان مصرفی واحدهای حساب.

شناسایی سرریز یکی از موارد اصلی در طراحی مؤثر و کارآمد سیستم‌های اعداد مانده‌ای است. کاربرد شناسایی سرریز در نتایج حاصل از عملیات حسابی است. مطالعات نشان می دهد که یکی از ساده ترین راهها برای تشخیص سرریز در جمع دو عدد دهدهی***X*** و***Y*** در بازه ***[0, M-1]*** زمانی ***X+Y>=M*** است که باشد. بنابراین مسئله تشخیص سرریز در سیستم اعداد وزن دار می تواند با یک مقایسه ساده نیز تشخیص داده شود[۹]. روش عمومی برای شناسایی سرریز در فرآیند جمع دو عدد در سیستم اعداد مانده ای، نیاز به مقایسه اعداد دارد بطوریکه اگر حاصل ***X+Y mod m*** کمتر از ***X*** شود سرریز اتفاق می افتد. از طرفی به دلیل بی‌وزن بودن این سیستم، انجام عمل مقایسه در آن نسبت به سیستم‌های وزن‌دار بسیار پیچیده است. به منظور رفع این مشکل در بسیاری از موارد استفاده از مبدل برعکس و تبدیل اعداد به سیستم وزن‌دار جهت مقایسه و شناسایی سرریز ضروری است. در الگوریتم های ***ROM-base*** قبلی نیز، نیاز به تبدیل اعداد از سیستم مانده‌ای به دودویی است که این امر فرآیندی زمانبر می‌باشد[۷].

در این مقاله روشی جهت تشخیص سرریز در سیستم اعداد مانده ای ارائه می شود که دارای سرعت بیشتر و هزینه سخت افزاری کمتری نسبت به روش های پیشین است تا از این طریق کارایی سیستم اعداد مانده‌ای جهت استفاده در سیستم‌های محاسباتی افزایش یابد. بدین منظور مجموعه پیمانه $\{2^n - 1, 2^n, 2^n + 1\}$را برای رنج اعداد داینامیک زوج در نظر می گیریم. این پیمانه بدین دلیل مورد توجه است که با توجه به خاصیت بازگشت رقم نقلی[9] (***EAC***)، پیچیدگی عملیات حسابی آن نیز نسبت به مجموعه پیمانه های دیگرکم تر می‌باشد.

---

[1] ***Residue Number System***
[2] ***Digital Signal Processing***
[3] ***Digital Filtering***
[4] ***Image Processing***
[5] ***Cryptography***
[6] ***Wireless Sensor Network***
[7] ***Redundant Module***
[8] ***Error Detection and Correction***
[9] ***End-around-carry***

۲- پیش زمینه

سیستم اعداد مانده ای به وسیله یک مجموعه پیمانه همانند $\{m_1, m_2, m_3, k, m_n\}$، که همه پیمانه ها اعداد صحیح و مثبت می باشند مشخص می شود. بزرگترین ناحیه ممکن در محدوده نمایش $[-a, a+M)$ می باشد که a یک عدد صحیح و M برابر است با حاصلضرب همه پیمانه ها در همدیگر.

$$M = \prod_{i=1}^{n} m_i \qquad (۱)$$

برای داشتن ماکزیمم دامنه نمایش اعداد در این سیستم ، این پیمانه ها باید بصورتی انتخاب شوند که دو به دو نسبت به هم اول باشند به عبارت دیگر بزرگترین مقسوم علیه مشترک آن ها یک باشد که معمولا به صورت زیر نوشته می شود.

$$Gcd\left(m_i \text{ و } m_j\right) = 1, for\ i \neq j \qquad (۲)$$

عدد صحیح X که $a \leq X < a + M$ ، نمایش یکتایی، در سیستم اعداد مانده ای دارد که به وسیله مجموعه باقیمانده های (x_1, x_2, x_3, k, x_n) نمایش داده می شود ، بطوریکه[۷]:

$$X \xrightarrow{RNS} (x_1, x_2, \ldots., x_2),\ x_i = X \bmod m_i, i = 1,2,3,\ldots,n \qquad (۳)$$

در سیستم اعداد مانده‌ای تبدیل اعداد مانده‌ای به اعداد باینری معمولا بر اساس روش‌های قضیه باقیمانده چینی یا سیستم اعداد مبنای درهم می‌باشد. هر دوی این روش‌ها بصورت ترتیبی اجرا می‌شوند و به دلیل اینکه تعداد زیادی جمع‌کننده باید بصورت سریال پشت سر هم اجرا شوند در نتیجه تاخیر محاسبه عدد وزنی از معادل مانده‌ای آن بسیار زیاد می‌باشد[۸].

اصول طرح پیشنهادی در سیستم اعداد مانده ای برای کلیه مجموعه پیمانه هایی است که فقط یکی از پیمانه های آن زوج باشد که در مجموعه پیمانه در نظر گرفته شده نیز این شرط برقرار است. محدوده نمایش پویا برای یک سیستم مانده‌ای با مجموعه پیمانه $\{2^n - 1, 2^n, 2^n + 1\}$ به صورت زیر می باشد که این محدوده نمایش تعداد بیت های مورد استفاده برای ROM را نیز نشان می دهد:

$$M = (2^n - 1).(2^n).(2^n + 1) = 2^{3n} - 2^n \qquad (٤)$$

تعریف ۱: سه بیت که بیت های x_4, y_4, P نامیده می شوند، نشانگر زوج یا فرد بودن مضرب ها هستند و طبق فرمول زیر بدست می آیند. اگر مضرب عدد فرد باشد آن ها را متناظر با بیت ۱و اگر زوج باشد بیت ۰ را برای آن ها در نظر می گیریم.

$$x_i = X \bmod m_i \leftrightarrow X = k.m_i + x_i \qquad (٥)$$

۳-شناسایی سرریز در مجموعه پیمانه $\{2^n - 1, 2^n, 2^n + 1\}$

در مرحله اول با در نظر گرفتن مجموعه پیمانه مورد نظر مضرب های پیمانه 2^n را محاسبه می کنیم. با در نظر گرفتن دو عدد X و Y که در بازه $[0, M-1]$ هستند بررسی می کنیم که این اعداد در کدام مجموعه از مضرب ها هستند.

$$\begin{cases} X = k_1 2^n + r_1 \\ Y = k_2 2^n + r_2 \end{cases} \qquad (٦)$$

برای $i = 1, 2$ داریم :
$$\left.\begin{matrix} k_i \text{ زوج} \rightarrow 0 \\ k_i \text{ فرد} \rightarrow 1 \end{matrix}\right\}$$

بیت حاصل در این مرحله را برای X با x_4 و برای Y با y_4 در نظر می گیریم.

حاصل جمع X و Y را به پیمانه های (m_1, m_2, m_3)بدست می آوریم.

$$Z = X + Y \equiv (z_1, z_2, z_3) \qquad (٧)$$

تعریف ۲: برای آنکه بررسی کنیم $x_2 + y_2 \geq 2^n$ هست یا نه از متغیری به نام A استفاده می کنیم که $A = Carry\ out(x_2 + y_2)$است. به عبارت دیگر اگر مجموع دو عدد X و Y دارای Carry − out باشد بدین معنی است که $x_2 + y_2$ بزرگتر یا مساوی پیمانه 2^n هست و A

برابر یک را برای آن در نظر می گیریم در غیر اینصورت بیت ۰ برای آن در نظر گرفته می شود.

بر اساس مقادیر x_4, y_4, A اگر سرریزی رخ ندهد انتظار داریم که z_4 برابر مقدار زیر باشد. این مقادیر براساس جدول درستی (۱) بررسی شده است.

جدول (۱): بیت های مورد انتظار در صورت عدم وجود سرریزی

x_4	y_4	A	Z_4 مورد انتظار در صورت عدم وجود سرریزی
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

بر اساس این جدول برای بدست آمدن نتیجه z_4 رابطه ایی به صورت زیر خواهیم داشت:

$$z_4 = x_4 \oplus y_4 \oplus carry\ out(x_2 + y_2) = x_4 \oplus y_4 \oplus A \quad (۸)$$

اثبات: بر این اساس به بررسی چند حالت از این جدول می پردازیم.

حالت اول: اگر حالتی را در نظر بگیریم که $\begin{matrix} 0 = x_4 \\ 0 = y_4 \end{matrix}\}$ یعنی در رابطه های زیر مضرب های k_1 و k_2 نسبت به پیمانه 2^n زوج هستند که جمع این دو عدد به صورت زیر نشان داده می شود:

$$X = 2k_1 2^n + r_1$$

$$Y = 2k_2 2^n + r_2$$

$$X + Y = \big(2(k_1 + k_2)\big) * 2^n + r_1 + r_2 = 2k_3 2^n + r_3 \quad (۹)$$

و اگر در جمع این دو عدد رقم نقلی (Carry – out) نداشته باشیم پس بیت A = 0 است بنابراین اگر سرریزی رخ نداده باشد $z_4 = 0$ می باشد و اگر بیت $A = 1$ باشد در صورتی که سرریزی وجود نداشته باشد $z_4 = 1$ می باشد.

حالت دوم :در اینجا حالتی را در نظر می گیریم که یکی از پیمانه ها عددی فرد و پیمانه دیگر عددی زوج باشد پس حاصل نتیجه این دو بیت به صورت $\begin{matrix} 1 = x_4 \\ 0 = y_4 \end{matrix}\}$ یا $\begin{matrix} 0 = x_4 \\ 1 = y_4 \end{matrix}\}$ خواهد بود. بدین معنی است که بر اساس بیت های ۰ یا ۱ در رابطه زیر به ترتیب مضرب های k_1 و k_2 زوج یا فرد هستند. پس اگر مضرب عدد X را زوج و مضرب عدد Y را فرد فرض کنیم، جمع این دو عدد را به صورت زیر داریم:

$$X = 2k_1 2^n + r_1$$

$$Y = 2(k_2+1)2^n + r_2$$

$$X + Y = (2(k_1 + k_2) + 1) \cdot 2^n + r_1 + r_2 = 2(k_3 + 1)2^n + r_3 \quad (۱۰)$$

اگر مقدار بیت A = 0 باشد در صورتی که سرریزی رخ نداده باشد $z_4 = 1$ است و اگر بیت A = 1 باشد به معنای وجود رقم نقلی است بنابراین اگر بیت $z_4 = 0$ باشد یعنی سرریزی رخ نداده است.

به همین صورت برای بقیه ی حالات نیز این روابط برقرار است.

پس از بدست آوردن حاصل نتیجه z_4، مضرب 2^n را درZ بدست می آوریم که اگر مضرب بدست آمده زوج یا فرد بود بیت های متناظر ۰ و ۱ را برای آن در نظر می گیریم و با P نمایش می دهیم. برای ذخیره نتیجه حاصل جمع از یک ROM استفاده می کنیم.

در آخر برای اطمینان از وجود یا عدم وجود سرریز در جمع دو عدد X و Y نتایج بیتهای z_4 وP را با گیت ***XOR*** مقایسه می کنیم. نتیجه نهایی با سیگنال خروجی سرریز مشخص می شود. اگر نتیجه این مقایسه یک بود سرریز نداریم در غیر اینصورت جمع دو عدد X و Y با سرریز همراه است. به عبارت دیگر خروجی معادل بیت ۱ به معنای وجود سرریز و

خروجی معادل بیت ۰ به معنای عدم وجود سرریز است. لذا می توان تمام این موارد را به صورت الگوریتم زیر بیان کرد:

مجموعه پیمانه : $\{2^n - 1, 2^n, 2^n + 1\}$

$$M = (2^n - 1).(2^n).(2^n + 1)$$

$$0 \le X \le M - 1$$
$$0 \le Y \le M - 1$$
$$X \xrightarrow{RNS} (x_1, x_2, x_3, x_4)$$
$$Y \xrightarrow{RNS} (y_1, y_2, y_3, y_4)$$
$$X = a * x_2 + r$$
$$Y = b * y_2 + r$$
$$Z = X + Y$$
$$0 \le Z \le M - 1$$
$$Z \xrightarrow{RNS} (z_1, z_2, z_3)$$
$$Z_i = (x_i + y_i) \bmod m_i \quad , i = 1,2,3,4$$
$$z_4 = x_4 \ominus y_4 \oplus Carry_{out} \bmod 2^n \qquad (11)$$

```
Cout << " overflow ";
Else
Cout << " NO overflow ";
```

همانطور که مشاهده می‌شود، در فرآیند جمع دو عدد ط X و Y تابع شناسایی سرریز در محدوده نمایش M عبارتست از:

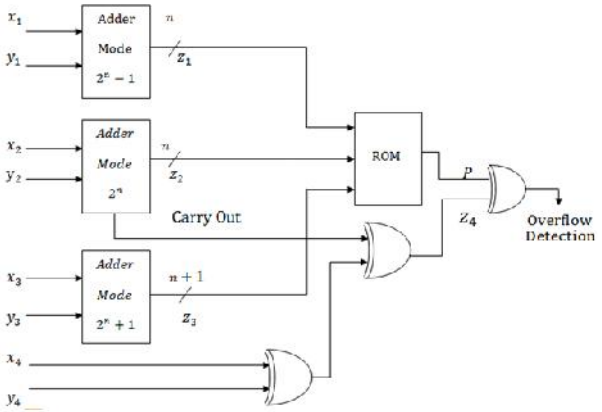
$$overflow = P \oplus x_4 \oplus y_4 \boxminus Carry\_out \bmod 2^n \qquad (12)$$

در رابطه بالا علامت $\boxminus$ بیانگر علامت XOR منطقی است.

با توجه به رابطه ٤-٨ لازم است ابتدا مضرب های دو عدد مورد نظر و $Carry$ حاصل از پیمانه 2^n تعیین شده، سپس بر اساس مقایسه ای که بین آن ها با بیت در نظر گرفته شده برای مضرب حاصل جمع انجام می شود، و با استفاده از مدار پیشنهادی تشخیص داده می‌شود که آیا سرریز اتفاق افتاده یا خیر. بنابراین مسئله مهم تعیین زوج و فرد بودن مضرب اعداد مانده ای و حاصل $Carry\_out$ نسبت به پیمانه 2^n است.

به دلیل انجام مقایسه ای بهتر با دیگر مقالات فرض براین شده است که برای تشخیص سرریز مجموع دو عدد X و Y به ما داده شده است و از تاخیر و مساحت جمع کننده های پیمانه ای صرفه نظر می شود.

پیاده سازی مدار پیشنهادی جهت شناسایی سرریز در شکل ۱ نشان داده شده است. بر این اساس واحد تشخیص سرریز شامل گیت های XOR و ROM است.



شکل(۱): مدار شناسایی سرریز در سیستم اعداد مانده ای

پارامترهای مساحت[1] و زمان[2] (AT) مدار پیشنهادی، جهت شناسایی سرریز در مجموعه پیمانه $\{2^n - 1, 2^n, 2^n + 1\}$ از سیستم اعداد مانده‌ای با استفاده از مدل استاندارد گیت یکتا برآورد شده است [۱۰]. در این مدل، هر گیت پایه دو ورودی همچون ***AND***، ***OR***، ***NOR*** و ***NAND*** مساحتی برابر ***A=1*** و تأخیری برابر با ***T=1*** دارند. همچنین در مورد هر گیت دو ورودی ***XOR / XNOR*** می‌توان گفت ***A=T= 2***. بنابراین پارامترهای مساحت و زمان (AT) مدار پیشنهادی با استفاده از مدل استاندار گیت پایه عبارتند از:

$$A_{ODU} = 3A_{XOR} + A_{ROM} = 6 + A_{ROM}$$
$$T_{ODU} = \tau_{XOR} + \tau_{ROM} = 2 + T_{ROM} \qquad (13)$$

۳-۱-بررسی مثال

در این بخش برای روشن تر شدن مطالب ارائه شده در قسمت های پیشین به ارائه یک مثال می پردازیم. با فرض اینکه مجموعه پیمانه $M = (3,4,5)$ را داریم. دو عدد X و Y را در بازه ی $[0 - 59]$ در نظر می گیریم.

---
[1] ***Area***
[2] ***Time***

جدول(۲): محاسبه گروهی از اعداد برای $RNS\{۳,٤,٥\}$

X	X_{RNS}	Y	Y_{RNS}	Z	Z_{RNS}	P	$Result$
۱٥	(۰,۳,۰,۱)	٥٥	(۱,۳,۰,۱)	۱۰	(۱,۲,۰,۱)	۰	۱
۱۹	(۱,۳,٤,۰)	٤٤	(۲,۰,٤,۱)	۳	(۰,۳,۳,۱)	۰	۱
٤	(۱,۰,٤,۱)	۲۳	(۲,۳,۳,۱)	۲۷	(۰,۳,۲,۰)	۰	۰
۱۲	(۰,۰,۲,۱)	۲۷	(۰,۳,۲,۰)	۳۹	(۰,۳,٤,۱)	۱	۰

می خواهیم وجود یا عدم وجود سرریز را در جمع دو عدد X = ۱٥ و Y = ٥٥ بر اساس الگوریتم پیشنهادی بررسی کنیم.

اگر مضرب عدد فرد باشد بیت ۱و اگر زوج باشد بیت ۰ را برای آن در نظر می گیریم. این بیت های ۰ و۱ معادل x_4 و y_4 در سیستم اعداد مانده ای هستند.

$X = ۱٥ = ۳ \square ٤ + ۳ \rightarrow$ مضرب عدد ۳ $= 1 \equiv$ (فرد)
x_4

$Y = ٥٥ = ۱۳ \square ٤ + ۳ \rightarrow$ مضرب عدد ۳ $= 1 \equiv$ (فرد)
y_4

در اینجا با توجه به فرد بودن هر دو مضرب پس برای دو بیت x_4 و y_4 بیت ۱ در نظر گرفته می شود.

در مرحله بعد حاصل جمع دو عدد X و Y را نیز محاسبه می کنیم و بعد از محاسبه مضرب آن در صورت زوج یا فرد بودن بیت های ۰ یا ۱ را معادل آن در می گذاریم که آن را با P نشان می دهیم.

$Z = X + Y = ۷۰ \equiv ۱۰ = ۲ \square ٤ + ۲ \rightarrow$
مضرب عدد ۲ $= 0 \equiv P$ (زوج)

مضرب بدست آمده از حاصل جمع عددی زوج است که معادل بیت ۰ است.

$$X = ۱٥ = (۰,۳,۰)$$
$$+ \ Y = ٥٥ = (۱,۳,۰)$$
$$Z = ۱۰ = (۱,۲,۰)$$

بر اساس حاصل جمع بدست آمده بیت $carry-out$ در پیمانه $۲^{n}=٤$ را داریم پس این بیت را معادل بیت ۱ قرار می دهیم. حاصل سه بیت x_4 ، y_4 و $carry\_out = A$ را به صورت زیر باهم XOR می کنیم و نتیجه z_4 بدست می آید.

$z_4 = x_4 \oplus y_4 \oplus (carry\_out \bmod ٤) = 1 \square 1 \square 1 = 1$

با یک شدن حاصل z_4 انتظار داریم که مجموع دو عدد در نظر گرفته شده همراه با سرریز باشد. پس برای اطمینان از این مطلب بیت z_4 را با بیت P مقایسه می کنیم، این عمل مقایسه را با گیت XOR انجام می دهیم. اگر نتیجه این XOR بیت ۰ شد به معنی این است که دو بیت یکسان هستند و سرریز رخ نمی دهد در غیر اینصورت سرریز خواهیم داشت.

$Result = z_4 \oplus P = 1 \square 0 = 1$

در این مثال بعد از انجام XOR بین حاصل z_4 و P می بینیم که نتیجه بدست آمده معادل بیت ۱ است پس در جمع دو عدد X و Y سرریز اتفاق می افتد.

٤- مقایسه

شناسایی سرریز علاوه بر روش هایی که مبتنی بر مبدل برعکس می باشد با استفاده از مقایسه مانده ای نیز می تواند تشخیص داده شود. از آنجائیکه طرح پیشنهادی نیازی به عملیات مذکور ندارد، از اینرو در این بخش سخت‌افزار و تأخیر روش پیشنهادی را با مقاله[۱۱] در جدول۳ و با استفاده از مدل واحد گیت در جدول ٤ مقایسه می کنیم. همانطور که ملاحظه می شود تاخیر مقایسه در الگوریتم پیشنهادی $\tau_{XOR}+\tau_{ROM}$ است و تنها سخت افزار لازم جهت پیاده سازی این الگوریتم سه گیت XOR دو ورودی و حافظه ROM برای ذخیره حاصل جمع می باشد.

نتایج مقایسه‌های صورت گرفته بهبودهایی است که از نظر سرعت و سخت‌افزار موردنیاز در جدول ۳ نشان داده شده است. روشی که در [۱۱] برای شناسایی سرریز ارائه شده است برای بدست آوردن بیت های علامت سه بار مراجعه به حافظه ROM نیاز دارد اما طرح پیشنهادی فقط یک مراجعه به حافظه ROM را دارد. روش ارائه شده در این بخش از

لحاظ سخت‌افزاری مقرون به صرفه است چون در مقایسه با روش قبلی به مساحت کمتری نیاز دارد.اگر پارامترهای مساحت و زمان (AT) مدار پیشنهادی را با مرجع [۱۱] با استفاده ازمدل استاندارد گیت پایه مقایسه کنیم، نتیجه این مقایسه به صورت جدول٤ می باشد.

جدول (۳): مقایسه سخت افزار و تاخیر الگوریتم پیشنهادی با مرجع[۱۱]

طراحی	تاخیر	تعداد بیت استفاده شده در ROM	تعداد مراجعات به حافظه ROM	*HA/FA*	*MUX*	*XOR/XNOR*	*AND/OR*
[۱۱]	$t_{mux}+2t_{XOR}+t_{AND}+3t_{ROM}$	$2^{3n}-2^{n}$	*3*	–	*1*	*2*	*1*
طرح پیشنهادی	$t_{XOR}+t_{ROM}$	$2^{3n}-2^{n}$	*1*	–	–	*3*	–

جدول(٤): مقایسه سخت افزار و تاخیر الگوریتم پیشنهادی با مرجع [۱۱] با استفاده از مدل واحد گیت

طراحی	*Unit gate Area*	*Unit gate Delay*
[۱۱]	$5+A_{2:1MUX}=8$	$3+T_{2:1MUX}=5$
طرح پیشنهادی	$3A_{XOR}=6$	$T_{XOR}=2$

۵- نتیجه گیری

تشخیص سرریز از موضوعات اصلی در طراحی کارآمد سیستم های ***RNS*** است. در این مقاله الگوریتمی کارا و مبتنی بر محاسبات ریاضی به منظور شناسایی سرریز بر روی مجموعه سه پیمانه ای $\{2^n-1, 2^n, 2^n+1\}$ در سیستم اعداد مانده‌ای ارائه شده است. الگوریتم روش پیشنهادی که برای شناسایی سرریز ارائه شده است، بجای انجام مقایسه و بدون نیاز به تبدیل اعداد مانده‌ای به اعداد دودویی از یک مدار بسیار ساده که از گیت های ***XOR*** و ***ROM*** تشکیل شده است، استفاده می‌کند. بعلاوه روش پیشنهادی کلی است و برای همه ی مجموعه پیمانه هایی که یک پیمانه زوج دارند هم استفاده می شود. این روش با تعداد محدودی مؤلفه سخت‌افزاری قابل پیاده‌سازی بوده و سریعتر از الگوریتم‌های قبلی است.

مراجع:

1) Garner H., "The Residue Number System", IRE Trans. Electronic Computer, 1959,Vol. EC8, pp. 14 0-147.

2) Hosseinzadeh Mehdi, Sabbagh Amir and Navi Keivan, "An Improved Reverse Converter for the Moduli set {2^n-1, 2^n, 2^n+1, 2^{n+1}-1}", IEICE Electronic Express, 2008, Vol. 5, No. 17, pp. 672-677.

3) Hosseinzadeh Mehdi," High-speed Low-Power VLSI Design for Capability of High Rate Processing Using the Multiple Valued Logic Residue Number Systems", ph.D Dissertation, Islamic Azad University, Science and Research Branch,1387(In Persian).

4) Jaberipour Ghassem, "A One-step Modulo 2^n+1 Adder Based on Double-lsb Representation of Residues", Submitted, 2008.

5) Bajard Jean-Claude and Imbert Laurent, "Brief contributions: A Full RNS Implementation of RSA," IEEE Transactions on Computer, 2004, Vol. 53, No. 6, pp. 769-774.

6) Askarzadeh Majid, Hosseinzadeh Mehdi, and Navi Keivan,"A New Approach to Overflow Detection in Moduli Set {2^n-3, 2^n-1, 2^n+1, 2^n+3}", Second International Conference on Computer and Electrical Engineering, 2009, pp. 439-442.

7) Parhami Behrooz, "Computer arithmetic: algorithms and hardware designs."Oxford University Press, New York, 2000.

8) Mi Lu ,"Arithmetic and Logic in Computer Systems", John Wiley & Sons, Texas A&M University, 2004.

9) Siewobr H. and Gbolagade K.A.,"Overflow Detection in Residue Number Systems Addition before Forward Conversion", International Journal of Computational Intelligence and Information Security, 2011,Vol. 2, No. 9.

10) Zimmerman Reto, "Efficient VLSI Implementation of Modulo (2n ±1) Addition and Multiplication", in Proc. 14th IEEE Symp. Computer Arithmetic, Adelaide, Australia, 1999, pp. 158-167.

11) Rouhifar Mehrin, Hosseinzadeh Mehdi and Teshnehlab Mohammad, "A New Approach to Overflow Detection in moduli set {2^n-1,2^n,2^n+1}", International Journal of Computational Intelligence and Information Securit, 2011,Vol. 2, No.3, pp. 35- 43.