**دانشگاه امیرکبیر**

**دانشکده کامپیوتر**

**گرایش معماری کامپیوتر**

*پروژه کارشناسی ارشد*

**عنوان :**

# مقایسه چهار طرح ضرب کننده RNS

**استاد پروژه**

**دکتر بابک صادقیان**

**ارائه دهنده**

***پیام حمیدی***

پاییز ۱۳۸۵

# فصل اول

# مـــقدمـــه

## ۱- مقدمه

همانطور که می دانیم ضرب پیمانه اي در علم رمزنگاري نقش مهمي ايفا مي کند. از جمله روشهاي رمزنگاري که به ضرب کننده پیمانه اي سریع نیاز دارد، روش رمزنگاري RSA مي باشد که در آن نیاز به توان رساندن اعداد بزرگ در پیمانه هاي بزرگ مي باشد. معمولاً براي نمایش اعداد در این حالات از سیستم باقي مانده (RNS) استفاده مي شود و ضرب (به عنوان هسته توان رساني) در این سیستم به کار مي رود.

در اینجا براي آشنایي بیشتر به توضیح سیستم عددي باقي مانده مي پردازیم و به کاربردها و فواید آن اشاراتي خواهیم داشت.

### ۱-۱ سیستم عددي باقیمانده (Residue Number System (RNS))

در حدود ۱۵۰۰ سال پیش معمایي به صورت شعر توسط یك شاعر چیني به صورت زیر بیان شد. «آن چه عددي است که وقتي بر اعداد ۵،۳و۷ تقسیم مي شود باقیمانده هاي ۳،۲و۲ بدست مي آید؟» این معما یکي از قدیمي ترین نمونه هاي سیستم عددي باقي مانده است.

در RNS یك عدد توسط لیستي از باقیمانده هایش برn عدد صحیح مثبت $m_1$ تا $m_n$ که این اعداد دو به دو نسبت به هم اولند (یعني بزرگترین مقسوم علیه مشترك دوبدوشان یك است) به نمایش در مي آید. به اعداد $m_1$ تا $m_n$ پیمانه (moduli) مي گویند. حاصلضرب این nعدد، تعداد اعدادي

که مي توان با اين پيمانه ها نشان داد را بيان مي كند. هر باقيمانده $x_i$ را به صورت $x_i=X \bmod m_i$ نمايش مي دهند. در مثال بالا عدد مربوطه به صورت $X=(2/3/2)_{RNS(7/5/3)}$ به نمايش در مي آيد كه $X \bmod 7=2$ و $X \bmod 5=3$ و $X \bmod 3=2$. تعداد اعداد قابل نمايش در اين مثال $M=3\times7\times5=105$ مي باشد. مي توان هرمجموعه 105 تايي از اعداد صحيح مثبت يا منفي متوالي را با اين سيستم عددي باقيمانده نمايش داد.

اثبات اين كه هر عدد صحيح موجود در محدوده، نمايش منحصر به فردي در اين سيستم دارد به كمك قضيه باقي‌مانده هاي چيني (Chinese Remainder Theorem (CRT)) امكان پذير است. اين قضيه به صورت زير بيان مي شود:

## ۱-۲ قضيه باقي مانده هاي چيني:

اعداد صحيح مثبت $m_1,m_2,m_3,...,m_n$ را كه نسبت به هم دو به دو اول هستند در نظر بگيريد و M را حاصلضرب $m_1,...,m_n$ فرض كنيد. همچنين اعداد $a,u_1,u_2,...,u_n$ را فرض كنيد. اثبات مي شود كه فقط و فقط يك عدد صحيح U وجود دارد كه شرايط زير دارد:

$$1\le j\le n \quad , \quad u_j = U \bmod m_j \quad , \quad a \le U < a+M$$

كه U برابر است با:

$$U=\left(\sum_{j=1}^{n} u_j.s_j.s_j^{-1}\right) \bmod M \ , \ s_j=\frac{M}{u_j}$$

اعمال رياضي جمع، تفريق و ضرب به راحتي و به صورت زير در اين سيستم انجام مي شود.

$$U=(u_1,u_2,\dots u_n)\ ,\ V=(v_1,v_2,\dots,v_n)$$

$$M=(m_1,m_2,\dots m_n)$$

$$(u_1,u_2,\dots u_n)\otimes(v_1,v_2,\dots v_n)=((u_1\otimes v_1)\bmod m_1,\dots,(u_n\otimes v_n)\bmod m_n)$$

در فرمول بالا به جاي علامت ⊗مي توان هر كدام از علائم +،-،* را قرار داد.

سه عمل رياضي (+،-،*) در اين سيستم عددي راحتتر از سيستم نمايش عادي اعداد انجام مي شود، زيرا هنگام انجام اين عمل در اين سيستم رقم نقلي (carry) بين بخشها رد و بدل نمي شود. در واقع انجام عمليات مربوط به مانده هاي هر پيمانه تاثيري روي ديگر عمل ها ندارد. يعني محاسبه "$u_1\otimes v_1 \bmod m_1$" مي تواند بطور مستقل (و در واقع موازي) انجام شود و نتيجه آن تاثيري در بقيه "$u_i\otimes v_i \bmod m_i$" ها ندارد. بدين ترتيب عمليات رياضي سريعتر (بعلت موازي شدن) و راحت تر (بعلت عدم تاثيرگذاري محاسبات مربوط به هر مانده برهم) انجام مي شود.

### ۱-۳- كاربردهاي RNS

سيستم عددي باقي مانده در چند دهه اخير مورد توجه قرار گرفته، زيرا مي توان بعضي از اعمال رياضي را تحت RNS به صورت چند مجموعه زير عمل رياضي تقسيم كرد. ولي به دليل اينكه اين اعمال فقط شامل ضرب، جمع و تفريق هستند از RNS در محاسبات "خاص منظوره" استفاده مي شود. RNS در پياده سازي سريع مسائلي كه شامل

تصحیح و تشخیص خطا در سیستم های Fault-tolerant و سیستم های پردازش سیگنال هستند کاربرد دارد. کاربردهایی از قبیل تبدیل فوریه سریع، فیلتر دیجیتال و پردازش تصویر از اعمال ریاضی سریع RNS استفاده می کند. RNS راه خود را در کاربردهایی مثل تبدیلات تئوری اعداد و تبدیل فوریه گسسته پیدا کرده است. همچنین مستقل بودن رقم های باقیمانده باعث می شود که رخ دادن خطا در یک رقم به رقم های بعدی منتقل نشوند که این مسأله، باعث ایجاد یک معماری Fault-tolerant خواهد شد. [20],[35]

سیستم عددی RNS در رمزنگاری و به خصوص در روش RSA کاربرد زیادی دارد[35]. البته در RSA از ضرب پیمانه ای جهت عملیات توان رسانی استفاده می‌شود.

در این پروژه سعی می شود که چهار طرح از رویکردهای ضرب RNS را پیاده‌سازی و با هم مورد مقایسه قرار دهیم. این مقایسه براساس حجم و تاخیر طرح ها می‌باشد. در پیاده سازی سعی شده است که از پیشنهادات مقالات جهت عناصر بکار رفته استفاده شود (بخصوص در دو طرح اول) و در مواقعی که پیشنهاد خاصی انجام نشده (مثل طرح های سوم و چهارم) پیشنهاد مناسب از لحاظ خود من انجام شده است.

در ادامه ابتدا به اصول ضرب RNS و روشهای بکار رفته برای اینکار اشاره می‌کنیم. سپس هر یک از چهار طرح را به تفصیل مورد بررسی قرار

مي دهيم و در مورد هر طرح، الگوريتم و سخت افزار بيان خواهد شد و سپس تاخير و مساحت آن را تعيين مي كنيم. در نهايت جمع بندي و مقايسه چهار طرح را انجام مي دهيم. در ضمايم نيز كدهاي VHDL نوشته شده را خواهيد يافت.

## ۲- روشهاي ضرب پيمانه اي

اين روشها را مي توان به دو دسته كلي تقسيم كرد. در دسته اول ابتدا عمل ضرب به صورت كامل انجام مي شود و سپس كاهش به پيمانه روي نتيجه آخر اعمال مي شود. اين روشها را Reduction After Multiplication (RAM) مي نامند. در دسته دوم عمل كاهش به پيمانه در هر مرحله ضرب و با هر حاصلضرب جزئي انجام مي شود كه به اين روشها Reduction During Multiplication (RDM) مي گويند[38]. از ميان طرحهاي مورد نظر ما دو طرح اول به دسته اول و دو طرح بعدي به دسته دوم تعلق دارند.

### ۲-۱- روش مونتگمري

در روش RDM چون روش كاهش به پيمانه به دفعات تكرار مي شود بايد اين عمل را سرعت بخشيد. يكي از تكنيك هاي پر طرفدار براي اينكار كه در طرحهاي ما نيز به كار رفته روش مونتگمري [2] در كاهش پيمانه است.

پيمانه N را در نظر بگيريد. عدد R را كه نسبت به N اول است و $N<R$ را طوري انتخاب كنيد كه محاسبات به پيمانه R آسان باشد. $N'$ را نيز طوري انتخاب كنيد كه $RR^{-1}-NN'=1$ باشد. حال

برای محاسبه $TR^{-1} \bmod N$ به شرطی که $0 \le T < RN$ باشد:

function REDC(T):

$m \leftarrow (T \bmod R)N' \bmod R$

$t \leftarrow (T + mN)/R$

if $t \ge N$ then return $(t - N)$ else return $t$

بدین ترتیب با به کارگیری عدد کمکی R، عمل کاهش T به پیمانه N سریعتر انجام می شود.

## ۲-۲- بررسی اجمالی روشهای موجود پیاده سازی ضرب در RNS

طرحهای ارائه شده را می توان براساس روش پیاده سازی سخت افزاری به سه مجموعه تقسیم کرد.

### مجموعه اول:

از تعداد خاصی از پیمانه ها مثل $2^n$ یا $2^n+1, 2^n-1$ استفاده می کنند. در این مجموعه n می تواند مقادیر کوچك، متوسط و گاهی بزرگ داشته باشد. در پیاده سازی این طرح ها عمدتاً فقط از مدارات منطقی استفاده شده و از ROM استفاده نمی شود. در هر حال این طرحها به پیمانه های خاصی محدود هستند و به همین دلیل کاربرد های محدودی دارند[3]. به طور مثال می توان به طرحهای [11],[12],[13] مراجعه کرد.

### مجموعه دوم:

توانایی کار با هر پیمانه ای را دارند ولی پیاده سازی این گروه، راه حلهایی بر اساس

ROM دارند و معمولاً از مدارات منطقي ديگر استفاده چنداني نمي كنند. اندازه حافظه با افزايش n به سرعت رشد مي كند كه طرح را براي پيمانه‌اي بزرگ غير عملي مي سازد. به طور مثال مي توان طرحهاي [7],[8],[9],[10] را ذكر كرد.

**مجموعه سوم:**

جهت پيمانه هاي متوسط و بزرگ طراحي مي شوند. معمولاً به صورت هيبريد هستند و از عناصر رياضي پايه مثل ضرب و جمع كننده هاي باينري كه به صورت موازي عمل مي كنند به همراه چندين ROM با اندازه كوچك و عناصري منطقي استفاده مي كند. طرحهاي مورد نظر ما در اين دسته قرار مي گيرند. همچنين مي توان به طرحهاي [14],[15],[16],[17],[18] اشاره كرد.

## ۲-۳- نكاتي پيرامون چهار طرح موردنظر

اين طرحها از مقالات [3],[4],[5],[6] انتخاب شده اند. دو طرح با تكنيك RAM كار مي كنند و از تكنيك هاي ابتكاري سود مي جويند و امكان پياده سازي دقيق و در سطح گيت آنها وجود دارد. دو طرح بعدي با هدف بهبود الگوريتم رمزنگاري RSA انتشار يافته اند و طرحهاي ضرب براي RNS به روش RDM پيشنهاد داده اند.

## ۳- طرح اول:

### ۳-۱- مقدمه

این طرح در مرجع [4] یعني مقاله "RNS Arithmatic Multiplier for Medium and large Moduli" بیان شده. در مقدمه آن اشاره شده است که طرحهاي بر اساس ROM براي پیمانه هاي کوچك مفید هستند ولي براي پیمانه هاي متوسط و بزرگ باید از طرحهایي که هم از ROMهاي کوچك و هم از عناصر ریاضي بهره مي برند استفاده کرد که طرح مورد نظر چنین حالتي دارد.

### ۳-۲- بررسي سوابق

براي ضرب پیمانه اي مي توان از ضرب و تبدیل هاي متوالي استفاده کرد [20],[21] که اصلاً روش بهینه اي نیست. بهمین دلیل جستجو براي روشهاي بهتر در جریان است. عمده کارهاي انجام شده بر روي پیمانه هاي خاص مثل $2^n \pm 1$ یا پیمانه هاي کوچك مي باشد [22]-[31]. طرح [22] پیاده سازي ضرب کننده اي است که پیمانه هاي آن اعداد اول بشکل k<n و $(2^n - 2^k - 1)$ مي باشد. مقاله[23] ضرب کننده پیمانه square-law را پیاده کرده که بر اساس ROM است و بر روي اندازه پیمانه ها محدودیت دارد. مقایسه [24] تکنیکي براي ضرب پیمانه‌اي در یك عدد اول را بیان کرده است که طرح همان مشکلات طرح [23] را دارد.

مرجع [25] ضرب کننده با پیمانه هاي بزرگ به شکل $(2^n - 1, 2^n, 2^n + 1)$ را مطرح کرد. [29] ضرب کننده

پیمانه اي را مطرح کرده از Factored decomposition بهره گرفته و در آن پیمانه ها مي توانند اعداد غیر اول باشد.

عمده این روشها بر اساس بکارگیري عناصر ریاضي binary-based هستند و هیچ کدام بجز طرح [16] براي استفاده از محاسبات مانده اي (Residue Arithmatic) طراحي نشده اند.

## ۳-۳- الگوریتم

دو باقیمانده (Residue) X, Y را در نظر بگیرید هدف محاسبه $|XY|_m$ (حاصلضرب X و Y در پیمانه m) مي باشد. هر دو عدد را مي توان با n بیت نمایش داد:

$$X=\sum_{j=1}^{n} x_j.2^{j-1} \quad , \quad Y=\sum_{y=1}^{n} y_j.2^{j-1}$$

که در آن ها n برابر است:

$$n=\lceil \log_2^m \rceil$$

و $x_j$ و $y_j$ بیت هاي باینري X و Y مي باشند. حاصلضرب X و Y را مي توان به صورت زیر نمایش داد.

$$XY=\sum_{k=1}^{n}\sum_{j=1}^{n} x_j.y_k.2^{(j+k-2)}$$

با بسط عبارت فوق داریم:

$$XY=2^0(x_1y_1)+2^1(x_1y_2+x_2y_1)+2^2(x_1y_3+x_2y_2+x_3y_1)+\ldots$$
$$\ldots+2^{(2n-3)}(x_{(n-1)}y_n+x_ny_{n-1})+2^{(2n-2)}(x_ny_n)$$

که مي توان نوشت:

$$XY=\sum_{j=1}^{n}2^{(j-1)}\sum_{k=1}^{j} x_k.y_{(j+1-k)}+\sum_{j=n+1}^{2n-1}2^{(j-1)}\sum_{k=j+1-n}^{n} x_k.y_{(j+1-k)} \quad (۱-۳)$$

حالا $v_j$ را به صورت زیر تعریف می کنیم:

$$v_j=\begin{cases} v_j=\sum_{k=1}^{j} x_k y_{(j+1-k)}, & j=1,2,\ldots,n \\ v_j=\sum_{k=j+1-n}^{n} x_k y_{(j+1-k)}, & j=n+1,n+2,\ldots,2n-1 \end{cases} \qquad (۲-۳)$$

بنابراین رابطه (۳-۱) به صورت زیر بازنویسی می شود:

$$XY=\sum_{j=1}^{2n-1} 2^{(j-1)}.v_j \qquad (۳-۳)$$

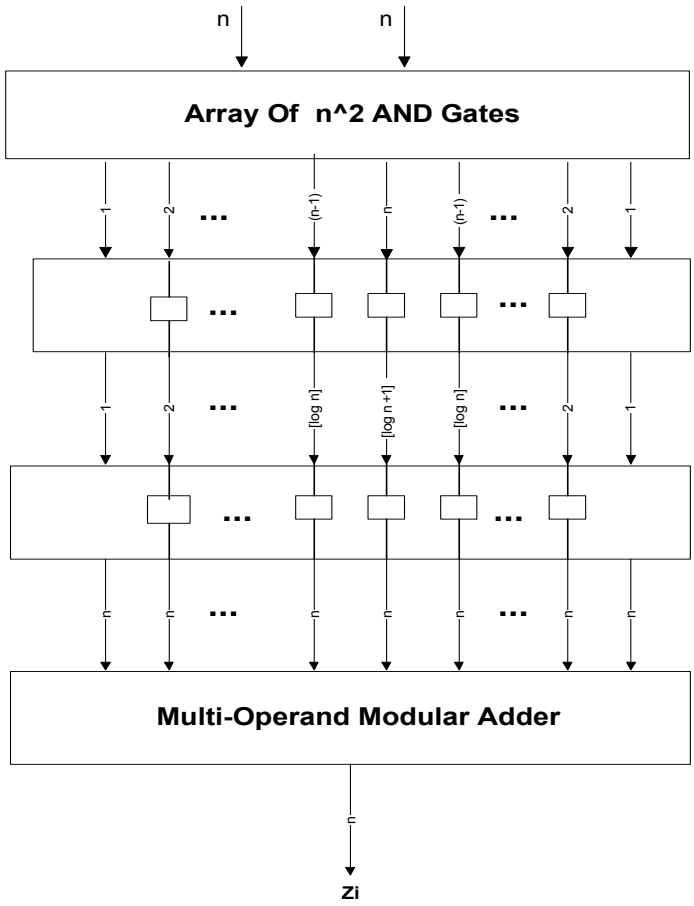
و برای محاسبه $Z=|XY|_m$ معادله بالا به صورت زیر در می آید:

$$Z=\left|\sum_{j=1}^{2n-1} |_2^{(j-1)}.v_j|_m\right|_m \qquad (۴-۳)$$

## ۳-۴- پیاده سازی سخت افزاری

شکل بلوکی سخت افزار این طرح را در شکل (۳-۱) مشاهده می کنید. همانطور که مشاهده می شود، طرح پیشنهادی در چهارطبقه پیاده شده است.

با دقت در رابطه (۳-۴) در می یابیم که برای پیاده سازی این طرح باید تمام $|2^{(j-1)}v_j|_m$ را محاسبه کرده و در انتها به صورت پیمانه ای با هم جمع کنیم. بررسی $v_j$ در رابطه (۳-۲) مشخص می کند که $v_j$ تعداد یك های نمایش باینری عبارت $\sum_k x_k y_{(j+1-k)}$ می باشد.




یت

اد

سود. پس برای هر $v_j$ ، $p_j$ گیت AND مورد نیاز است که :

$$P_j = \begin{cases} j & 1 \le j \le n \\ 2n - j & n < j \le 2n-1 \end{cases}$$

پس کل گیت های AND مورد نیاز در این طبقه برابر است با :

$$\sum_{j=1}^{2n-1} P_j = n^2$$

پس طبقه اول تمام بیت های X و Y را با هم AND می کند.

در مرحله بعد $P_j$ بیت هر $v_j$ را به طبقه کدگزار (Encoder) اعمال می کنیم. هر کدگذار تعداد یک های موجود در ورودی خود را به صورت باینری محاسبه میکند. پس خروجی هر کدگذار به $\lceil \log_2(P_i+1) \rceil$ بیت احتیاج دارد. با توجه به این

که $P_1=1$ و $P_{2n-1}=1$ مي باشند، براي اين دو حالت نيازي به كدگزار نيست. درنتيجه تعداد كل كدگزارهاي مورد نياز 2n-3 مي باشد.

طبقه سوم، آرايه اي است از 2n-1، ROM. ROM موجود در محل زام خروجيهاي j امين كدگزار را دريافت كرده و $\left|2^{j-1}V_j\right|_m$ را ايجاد مي كند. در نتيجه خروجي اين طبقه 2n-1 عدد در پيمانه m مي باشد كه بايد با هم جمع شوند.

عمل جمع چند عدد پيمانه اي در طبقه آخر توسط واحد، (MOMA) Multi-Operand Modular Adder انجام ميشود. جهت پياده سازي اين MOMA از طرح پيشنهادي مقاله [19] استفاده شده است. در ضميمه "هـ" الگوريتم اين مقاله آورده شده است.

همانطور كه در ابتداي الگوريتم نشان داده ايم طرح اول محاسبه حاصلضرب يك جفت از مانده هاي مربوط به يك پيمانه را انجام مي دهد. در واقع براي انجام يك ضرب RNS بايد به تعداد پيمانه ها از اين طرح داشته باشيم. يعني جهت هر عمليات ضرب روي هر جفت پيمانه نياز به يك پياده سازي از طرح را داريم. در نتيجه سخت افزار طرح برابر حاصلضرب مساحت به دست آمده براي طرح اول ضربدر تعداد پيمانه ها خواهد بود. در مورد تاخير، عمليات بطور موازي خواهد بود و كندترين شاخه تاخير كل را مشخص مي كند.

### ۳-۵- محاسبه پیچیدگي مساحت و تأخیر طرح اول

مساحت (A) بر اساس تعداد گیت هاي به کار رفته بیان مي شود. در مورد ROM هم هر سلول حافظه یك گیت فرض مي شود. واضح است که پیچیدگي یك تمام جمع کننده هشت گیت و یك مولتي پلکسر (2به1) سه گیت مي باشد[15],[19].

براي محاسبه تأخیر، تأخیر هر گیت را $T_g$ فرض مي کنیم. براي ROM نیز زمان دسترسي به یك ROM، $2^b.\omega$ بیت $2bT_g$ درنظر گرفته مي شود. تاخیر تمام جمع کننده 2.Tg و تاخیر مولتي پلکس (۲به۱) هم 2.Tg فرض شده است [15],[19].

اولین طبقه شامل $n^2$ گیت AND مي باشد. در نتیجه مساحت آن $n^2$ گیت و تأخیر آن $T_g$ میباشد. طبقه دوم 2n-3 کدگزار با اندازه هاي مختلف دارد. هر کدگزار با $P_j$ بیت ورودي حداکثر به اندازه $P_j$ تمام جمع کننده یا هشت $P_j$ گیت را اشغال مي کند. تأخیر موردنیاز حداکثر $\frac{P_j}{2}$ تأخیر یك تمام جمع کننده یا به طور متناظر $P_jT_g$ مي‌باشد[33]. چون تعداد کل گیت هاي AND، $n^2$ مي باشد در نتیجه در این طبقه $n^2$ تمام جمع کننده یا به طور متناظر $8n^2$ گیت براي طبقه کدگزار مورد نیاز است. حداکثر تأخیر این طبقه مربوط به بزرگترین کدگزار که متناسب با $P_j=n$ است مي باشد در نتیجه تأخیر این مرحله nTg است.

سومین طبقه شامل 2n-1 ROM مي باشد که هر کدام اندازه اي برابر $\left(2^{\lceil \log_2(P_j+1) \rceil}.n\right)$ دارند. پس فضاي حافظه موردنیاز برحسب گیت برابر است؛

$$n\sum_{j=1}^{2n-1}2^{\lceil \log_2 (P_j+1)\rceil}$$

تأخیر این مرحله بر حسب بدترین حالت مربوط به $P_j=n$ میباشد، که برابر است با:

$$2\lceil \log_2^{(n+1)} \rceil Tg$$

طبقه آخر طرح پیشنهادي يك MOMA است. همانطور که درمقاله [8] اثبات شده است، تعداد گیت هاي لازم براي این MOMA برابر است با:

$$2n^2+27n+(2.\lceil \log_2(2n-1)\rceil.n)$$

و تأخیر این مرحله $T_g.(4n+25)$ مي باشد.

پس در جمع:

$$A_{P1}=11n^2+n(27+2\lceil \log_2(2n-1)\rceil+\sum_{j=1}^{2n-1}2^{\lceil \log_2(p_j+1)\rceil})$$

و تأخیر کلي برابر است:

$$T_{P1}=5n+26+2\lceil \log_2(n+1)\rceil T_g$$

همانطور که قبلاً بیان کردیم براي محاسبه مساحت طرح در حالت RNS باید مساحت بدست آمده در بالا را در تعداد پیمانه ها ضرب کنیم و تاخیر نیز، تاخیر مربوط به بزرگترین پیمانه مي باشد. پس در واقع براي مساحت و تاخیر طرح اول اگر تعداد پیمانه ها را size فرض کنیم در حالت RNS داریم.

$$A_{P1_{RNS}}=\text{size}.A_{P1} \qquad (۵-۳)$$

$$T_{P1_{RNS}}=T_{P1}|_n=\lceil \log_2 \text{Max}(m_i)\rceil \qquad (۶-۳)$$

## طرح دوم:

### ٤-١- مقدمه

این طرح در مقاله [3] با عنوان
"New Efficient Structure for a Modular Multiplier for RNS"
پیشنهاد شده.

این طرح فقط از عناصر ریاضي استفاده کرده و براي پیمانه هاي متوسط و بزرگ مناسب است.

### ٤-٢- بررسي سوابق

تمام موارد ذکر شده همان موارد ذکر شده در طرح اول و در بخش بررسي اجمالي است. فقط یك مورد جدید ([15]) مشاهده مي شود که بر اساس pseudo-RNS کار مي کند.

### ٤-٣- الگوریتم

دو باقیمانده Y,X را در نظر بگیرید. که هر کدام به صورت زیر بیان مي شوند.

$$X=\sum_{j=0}^{n-1}x_j 2^j$$

$$Y=\sum_{j=0}^{n-1}y_j 2^j$$

هدف محاسبه $|XY|_m$ حاصلضرب پیمانه اي Y,X در پیمانه m است.

در معادلات بالا $n=1+\lceil \log_2^m \rceil$. با فرض $m\neq 2^n$ مي توان نوشت:

$$m=2^n-a\ ,\ 1\le a<2^{n-1}$$

عدد a را مي توان با یك عدد باینري k بیتي نمایش داد، که

$$k=1+\lfloor \log_2 a \rfloor\ ,\ 1\le k\le n-1$$

در نتیجه m در محدوده $m\in(2^{n-1},2^n)$ قرار دارد:

اگر حاصلضرب X,Y را با Z نمایش دهیم داریم:

$$Z=\sum_{i=0}^{2n-1}z_i 2^i \qquad (۱-٤)$$

n-1 بیت کم ارزش Z را با A نمایش می دهیم و n+1 بیت با ارزش Z را به سه گروه D,C,B به شرح زیر تقسیم می کنیم:

$$\begin{cases} A=\sum_{i=0}^{n-2}z_i 2^i \\ B=z_{n-1} \\ C=\sum_{i=n}^{2n-2-k}z_i 2^{i-n} \\ D=\sum_{i=2n-1-k}^{2n-1}z_1 2^{i-(2n-1-k)} \end{cases}$$

در نتیجه رابطه (۱-٤) را می توان به صورت زیر بازنویسی کرد:

$$Z=A+B.2^{n-1}+C.2^n+D.2^{(2n-1-k)}$$

هدف محاسبه $|Z|_m$ است. پس داریم:

$$|Z|_m=\left|A+B.2^{n-1}+C.2^n+D.2^{(2n-1-k)}\right|_m$$

چون A یک عدد n-1 بیتی است داریم:

$$A_{max}=2^{n-1}-1=\frac{m+a}{2}-1\Rightarrow|A|_m=A$$

# فصل دوم

# روشهای ضرب پیمانه‌ای

همچنین داریم:

$$\left|B.2^{n-1}\right|_m = B.2^{n-1}$$

و نیز چون $2^n = m + a$ و C یک عدد (n-1-k) بیتی است پس:

$$|2^n|_m = a \quad , \quad |C|_m = C \qquad (۲-٤)$$

$$(aC)_{max} = 2^{n-1} - 2^k = \frac{m+a}{2} - 2^k \qquad (۳-٤)$$

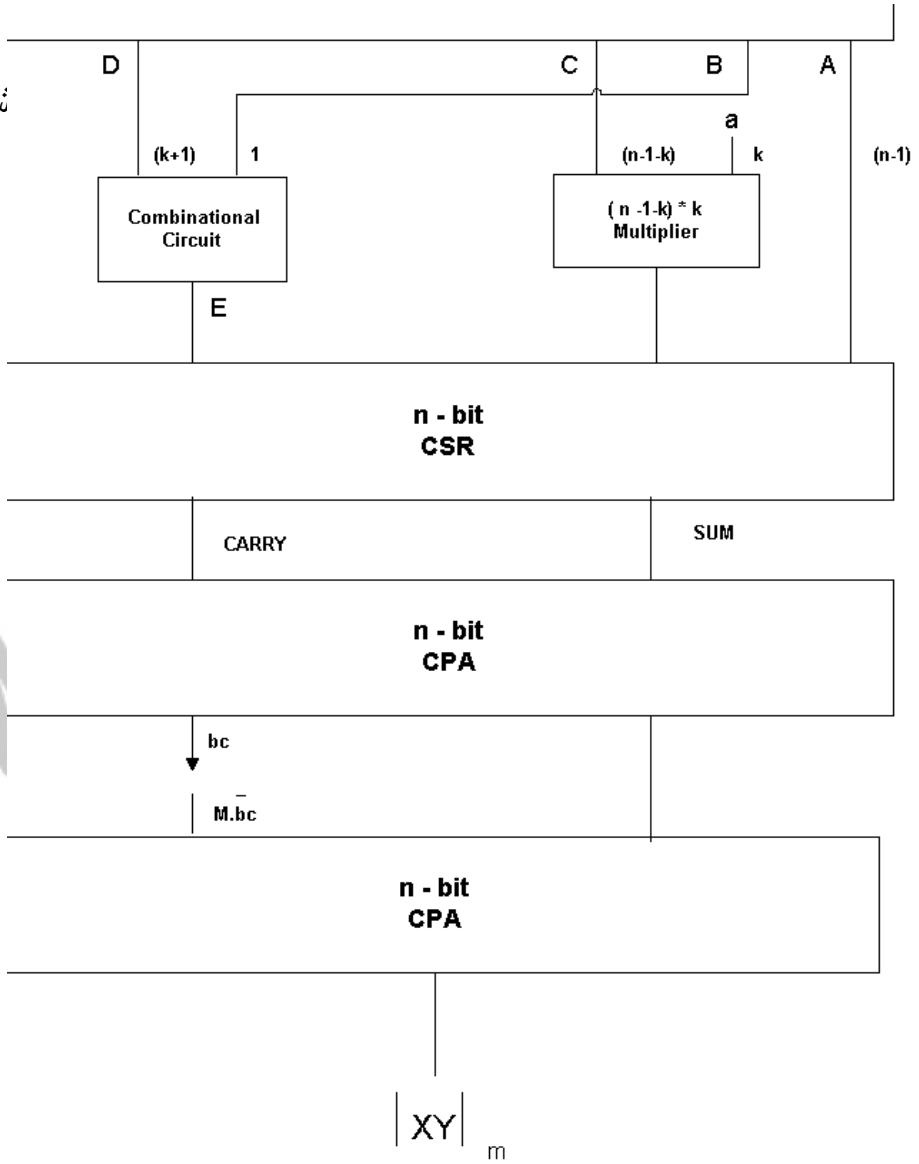
با دقت در رابطه (۲-٤) و (۳-٤) می توان نتیجه گرفت:

$$(A + aC)_{max} = m + a - 2^k - 1$$

و چون $a < 2^k + 1$ است:

$$\left|A + aC\right|_m = A + aC$$

$$|Z|_m = |A + aC + |D.2^{(2n-1-k)} + B2^{n-1}|_m|_m$$




شکل ٤-٤ نمودار بلوکی سخت افزار طرح دوم (٤-١)

شکل ٤-١: نماي بلوکي سخت افزار طرح دوم

دو باقیمانده Y,X ابتدا به یك ضرب كننده (n*n) بیتي باینري اعمال مي شود. حاصلضرب ایجاد شده عدد باینري 2n بیتي Z خواهد بود. تقسیم بندي انجام شده در الگوریتم به طور متناظر در شكل دیده مي شود. عدد n-k-1 بیتي C توسط یك ضرب كننده باینري (n-1-k).k بیتي در عدد ثابت k بیتي a ضرب مي شود همچنین یك مدار تركیبي وظیفه محاسبه عبارت زیر را دارد:

$$E=\left|D2^{(2n-1-k)}+B2^{n-1}\right|_m$$

این مدار تركیبي كه مي تواند یك PAL یا FPGA با سرعت بالا باشد بیت هاي D,B كه k+2 بیت مي باشد دریافت كرده و E را كه حداكثر n بیت است تولید مي‌كند.

در مرحله سوم یك n(CSA) Carry Save Adder بیتي وظیفه جمع E,aC,A را دارد. جمع حاصله به صورت SUM و CARRY بیان مي شود.

در مرحله چهارم، یك جمع كننده n بیتي CARRY, SUM را هم جمع مي كنیم carry خروجي این مرحله $b_c$ نشان دهنده منفي بودن عدد خروجي است، در صورت منفي بودن خروجي مربوطه، در مرحله پنجم حاصل را با m و در غیر اینصورت با 0 جمع مي كنیم.

طرح دوم نیز همانند طرح اول فقط روي یك جفت مانده از یك دسته عدد RNS عملیات ضرب پیمانه اي انجام مي دهد و همانند طرح اول براي انجام ضرب RNS بطور كامل باید به تعداد پیمانه ها سخت افزار دوم را تكرار كنیم. همچنین مساحت

طرح در حالت کامل حاصلضرب تعداد پیمانه ها در یك واحد ضرب به شکل طرح اول خواهد بود و براي تاخیر باید تاخیر واحد مربوط به بزرگ‌ترین پیمانه را در نظر بگیریم.

### ٤-٥- محاسبه پیچیدگي مساحت و تأخیر طرح دوم

این طرح کلاً دو عدد جمع کننده باینري، یك جمع کننده Carry save Adder و ١/٢٥ ضرب کننده باینري n*n را در خود دارد. جمع کننده باینري با پیچیدگي حجم $O(n^2)$، جمع کننده CSA با پیچیدگي حجم O(logn) و ضرب کننده با پیچیدگي حجم $O(n^2 \log n)$ [3] در این طرح بکار رفته اند.

در مورد عدد ضرب کننده ها باید دقت کنیم که ضرب کننده (n-1-k).k در بدترین حالت داراي پیچیدگي به ازاي k=(n-1)/2 است که ١/٤ پیچیدگي مساحت یك ضرب کننده معمولي را به خود اختصاص مي دهد. نتیجه فوق به سادگي با محاسبه حداکثر مقدار (n-1-k).k (با مشتق گیري) بدست مي آید. پیچیدگي کلي را برحسب n به صورت زیر مي توان نوشت.

$$A_{P2} = O(n^2 \log n) \times 1.25 + 2.O(n^2) + O(\log n)$$

درمورد تأخیر باید گفت که تأخیر ضرب کننده ها $6\lfloor \log n \rfloor$ و تأخیر جمع کننده ها برابر $2\lceil \log n \rceil + 2$ در نتیجه تأخیر کلي را مي توان به صورت زیر نوشت:

$$T_{P2} = 8\lfloor \log n \rfloor + 2$$

براي تاخیر مساحت در حالت RNS با فرض داشتن size عدد پیمانه مي توان نوشت:

$$A_{P2_{RNS}} = size.A_{P2} \qquad (٤-٤)$$

$$T_{P2_{RNS}} = T_{P2}|_n = 1 + \lfloor \log_2 Max(m_i) \rfloor \qquad (٤-٥)$$

**٥- طرح سوم:**

اين طرح در مقاله [5] توسط آقاي پيرسون مطرح شده است. در اين مقاله بعد از ذكر اهميت عمليات RSA به طور سريع، بيان شده است كه در مقاله فوق الگوريتم جديدي براي ضرب پيمانه اي با به كارگيري سيستم عددي RNS و نوعي از روش مونتگمري ارائه شده است. قلب اين الگوريتم روش جديد تغيير از يك RNS به RNS ديگر مي باشد.

در اين مقاله روشهايي كه براي تسريع ضرب RSA ديده شده استفاده از يكي از چهار روش زير است.

١- الگوريتم سلسله مراتبي سريعتر
٢- نرخ كلاك سريعتر
٣- سخت افزار با منظور خاص
٤- الگوريتم ها و كامپيوترهاي موازي

الگوريتم ارائه شده در اين طرح سعي در بكارگيري روش چهارم داشته است. بقيه روش ها چندان مورد نظر نيستند. الگوريتم هاي ترتيبي در حال پيشرفت مي باشند ولي سرعت اين پيشرفت كم است. نرخ كلاك سريعتر هم مي تواند بسيار مفيد باشد ولي مشكلات پياده سازي آن بسيار است. سخت افزارهاي با منظور خاص روش ديگري براي بدست آوردن موازي سازي هستند همه پياده سازي هاي سخت افزاري كنوني نوعي از موازي

سازي را به كار مي گيرند. الگوريتم مورد نظر مطمئناً روي سخت افزاري با منظور خاص بسيار سريعتر اجرا مي شود ولي تمايل به چنين راه حلي كم است. البته پياده سازي ما به صورت سخت افزاري با منظور خاص مي باشد. در اينجا الگوريتم RSA را به صورت زير يادآوري مي كنيم. فرض كنيد كه پيغام t پيمانه m و يك توان e موجود باشد. الگوريتم توان رساني پايه كه توسط حلقه RSA براي هر بيت $e_i$ انجام مي شود به صورت زير است:

1. x: = t
2. v: = 1
3. for i=1 to n
4. if $e_i$=1 then v = (vx) mod m
5. x: = $x^2$ (mod m)

مقدار نهايي v نتيجه مورد نظر يعني $t^e \bmod m$ است. روشي سريعتر از توان رساني پيمانه اي با تكرار مربع كردن عدد مورد نظر وجود ندارد. چهار روش مشخص براي بدست آوردن موازي سازي وجود دارد:

الف- به كارگيري پروسسورهاي مختلف جهت رمز كردن جرياني از اطلاعات. اين روش براي يك عمليات كليد خصوصي چندان مناسب نيست ولي ميزان گذردهي كلي را تسريع مي بخشد.

ب- مي توان توان رسايي مرحله ٥ و ضرب مرحله ٤ از يك تكرار قبل را به طور موازي انجام داد كه البته اين عمل در صورتي ممكن است كه حلقه از بيت هاي كم ارزش تر به سمت بيت هاي پر

ارزش تر حرکت کند. البته معمول است که این عمل از بیت هاي بالاتر به سمت بیت هاي پایین تر انجام مي شود ولي این کار خواص ذاتي موازي شدن را از بین مي برد.

ج- براي عملیات کلید خصوصي مي توان فرض کرد فاکتورهاي q,p پیمانه m موجود هستند. مي توان توان رساني پیمانه اي را به طور مجزا و موازي به پیمانه q,p انجام داد و با کمك قضیه باقیمانده هاي چیني این دو را با هم ترکیب کرد. این عمل اغلب در پیاده سازي هاي نرم افزاري سریع انجام مي شود.

د- در آخر ضرب هاي دقت بالاي مراحل ٤و٥ عملیاتي کند ($O(n^2)$) مي باشند. مثلاً با به کارگیري یك مدول ضرب ٣٢ بیتي و یك پیمانه ٥١٢ بیتي، ١٦ کلمه در هر عدد وجود دارد و نیاز به ٢٥٦ عدد ضرب ٣٢*٣٢ مي باشد. مي توان تمام این ضرب ها را به طور موازي انجام داد و نتیجه را به طور موازي با هم جمع کرد. برخلاف روشهاي دیگر ضرب موازي نیاز به معماري دارد که موازي سازي Fine-grained را ممکن مي سازد که این روش اگر سر بار عملیات بین پروسسوري خیلي بیشتر از زمان ضرب باشد عملي نیست.

یك پیاده سازي خوب از RSA سه ایده اول را به کار مي گیرد. در ادامه گونه اي بهتر از ایده چهارم را مورد بررسي قرار مي دهیم.

در قسمت سوم ضرب پیمانه اي بررسي مي شود. در عملیات RSA یك ضرب ساده با دقت چندگانه

نیست بلکه ضربی است پیمانه ای. پیاده سازی ترتیبی این عملیات معمولاً نیاز به دو برابر زمان یك ضرب معمولی دارد. زیرا که هر مرحله ضرب n*۱ یك کاهش به پیمانه m، n*۱ را به دنبال دارد. این نگرش به ضرب پیمانه ای هیچگونه مشابه موازی جهت پیاده سازی ندارد.

این الگوریتم به شکل زیر بیان میشود:

$$\begin{cases} s := xy \\ q := m^{-1}s \\ r := s-qm \end{cases}$$

مقادیر میانی s و qm نیاز به دو برابر فضای اشغالی توسط پیمانه ها دارد. خارج قسمت q به طور تقریبی محاسبه شده و معمولاً نیاز به تصحیح دارد. پیچیدگی این الگوریتم از همان مرتبه پیچیدگی ضرب است ولی سه برابر از آن کندتر است زیرا که هر مرحله نیاز به یك ضرب با دقت بالا دارد. به علاوه هر ضرب به حاصلضرب ضرب قبلی وابسته است و در نتیجه باید پشت سر هم انجام شود. امکانات به کار رفته برای پیاده سازی این الگوریتم روش مونتگمری و سیستم عددی RNS می باشد. هر کدام از این دو تکنیك به تنهایی کمکی نمی کند ولی وقتی با هم ترکیب می شوند و روش مناسبی برای تبدیل از یك سیستم عددی به سیستم عددی دیگر به کار گرفته شود، پیشرفت قابل ملاحظه ای حاصل می شود.

در ادامه، روش مونتگمری را به سبك خودش بیان می کند. مقاله بیان می کند که کاهش پیمانه ای بعد از هر ضرب به دلیل جلوگیری از

رشد نمايي اندازه نتايج ميانی مي باشد كه با افزودن مضربي از پيمانه جهت صفر كردن بيت هاي مرتبه بالاتر حاصلضرب انجام مي شود. مونتگمري سعي كرد كه بيت هاي مرتبه پايين حاصلضرب را صفر كرده و باقيمانده بيت ها را به سمت پايين شيفت دهد. البته نياز به ضرب در يك عدد ثابت جهت بدست آوردن نمايش صحيحي از نتيجه در مرحله آخر وجود دارد. الگوريتم مونتگمري را به صورت زير بيان مي كند.

فرض كنيد $\omega = 2^n$ باشد و $\omega \geq 4 \times m$. $m'$ را به گونه اي انتخاب مي كنيم كه $m' \times m = -1 \bmod \omega$. حال الگوريتم به شكل زير نشان داده مي شود.

$$\begin{cases} s := xy \\ q := m's \pmod{\omega} \\ r := (s+qm)/\omega \end{cases}$$

هنوز هم مقادير ميانی s و qm دو برابر فضاي r,y,x,m را اشغال مي كنند به دليل اينكه qm=-s $\bmod\omega$ مي باشد، s+qm همواره مضربي از $\omega$ است. در اينجا مقاله ذكر مي كند كه كاهش به پيمانه $\omega$ و تقسيم بر $\omega$ عملياتي وقت گير براي اعدادي باينري دقت چندگانه هستند و به نظر مي رسد كه پيشرفت خاصي حاصل نشده زيرا كه همچنان به سه ضرب مثل روش اول نياز است به همين دليل روش مونتگمري را با نمايش جديدش يعني در سيستم عددي RNS بيان مي كند.

فرض كنيد $v_1, v_2, \ldots, v_n, \omega_1, \omega_2, \ldots, \omega_n$ دو به دو نسبت به هم اول باشند و فرض كنيد

$\omega = \omega_1, \omega_2, \ldots, \omega_n, v = v_1, v_2, \ldots, v_n$ . عدد $x < \omega < v$ را مي توان با باقيمانده هايش به شكل زير بيان كرد:

$$[x_{\omega_1}, x_{\omega_2}, \ldots, x_{\omega_n}]$$

كه در آن $x_{\omega_i} = x \bmod \omega_i$ مي باشد (و همينطور در مورد $x_{v_i} = x \bmod v_i$ ، $v$). $\omega, v$ را به گونه اي انتخاب مي كنيم كه از $4m$ بزرگتر باشد. حالا سه عمل ضرب به عمليات ساده تري تبديل مي شود ولي هنوز دو عمليات مشكل ديگر وجود دارد: كاهش به پيمانه $\omega$ در مرحله دوم و تقسيم بر $\omega$ در مرحله آخر كه اين عمل را مي توان با تبديل از يك سيستم RNS $(\omega_j)$ به ديگري $(v_i)$ انجام داد. اين عمليات را به شكل زير نشان مي دهند:

$$[x_{v_1}, x_{v_2}, \ldots, x_{v_n}] \leftarrow [x_{\omega_1}, x_{\omega_2}, \ldots x_{\omega_n}]$$

همچنين فرض كنيد كه مقادير از پيش محاسبه شده نمايش $m'$ در سيستم عددي RNS، $\omega_j$ يعني $m' = [m'_{\omega_1}, m'_{\omega_2}, \ldots, m'_{\omega_n}]$ و به طور مشابه نمايش $m$ و $\omega^{-1}$ را در سيستم عددي $v_i$ داشته باشيم. در اينجا مقاله يادآور ميشود كه به دليل دو بدو نسبت به هم اول بودن $\omega$ و $v$ معكوس $\omega \bmod v_i$ وجود دارد. قدم هاي ضرب پيمانه اي دو عدد كه در پيمانه هاي $\omega_j$ نمايش داده شده اند با حاصلي در پيمانه $v_i$ به شكل زير بيان مي شود.

$$\text{input}: [x_{\omega_1}, x_{\omega_2}, \ldots, x_{\omega_n}], [y_{\omega_1}, \ldots, y_{\omega_n}]$$

$$\text{output}: [r_{v_1}, \ldots, r_{v_n}], r = xy \pmod m$$

1. $[x_{v_1}, x_{v_2}, \ldots, x_{v_n}] \leftarrow [x_{\omega_1}, x_{\omega_2}, \ldots, x_{\omega_n}]$

2. $[y_{v_1}, y_{v_2}, \ldots, y_{v_n}] \leftarrow [y_{\omega_1}, y_{\omega_2}, \ldots, y_{\omega_n}]$

3. $s_{\omega_j} := x_{\omega_j}.y_{\omega_j} \pmod{\omega_j}$

4. $s_{v_j} := x_{v_j}.y_{v_j} \pmod{v_j}$

5. $q_{\omega_j} := s_{\omega_j}.m'_{\omega_j} \pmod{\omega_j}$

6. $[q_{v_1}, q_{v_2}, \ldots, q_{v_n}] \leftarrow [q_{\omega_1}, q_{\omega_2}, \ldots, q_{\omega_n}]$

7. $r_{v_i} := (s_{v_i} + q_{v_i} m_{v_i})\omega_{v_i}^{-1} \pmod{v_i}$

مرحله ۳ نمایش s را در پیمانه ω و قدم چهارم s به پیمانه v را محاسبه می کند. واضح است که مرحله ۵ و ۶ مقدار q را در روش مونتگمري استاندارد محاسبه میکند. عدد r که با $r_{v_i}$ که در مرحله ۷ محاسبه می شود داراي خواص زیر است:

1- $r < v$

2- $r\omega = s + qm \bmod v$

3- $r\omega = 0 \bmod \omega$

چون $s + qm = 0 \bmod \omega$ است، از شرایط ۲ و ۳ مذکور نتیجه زیر حاصل می شود. $r\omega = s + qm$ و در نتیجه r خارج قسمت صحیح $(s+qm)/\omega$ است. مشکلي که در نگاه اول در الگوریتم مشاهده مي شود آغاز شدن آن با ورودیهاي به پیمانه $\omega_j$ و اتمام آن با نتیجه اي به پیمانه $v_j$ است ولي براي تکرارهاي عملیات توان رساني لازم نیست که به پیمانه قبلي بازگردیم. در واقع مي توان عملیات توان رساني را در همان پیمانه کمکي ادامه داد و فقط نتیجه نهایي را به پیمانه اصلي بازگرداند. مسائل دیگري که در الگوریتم وجود دارد دو تبدیل RNS قدمهاي ۱و۲ هستند که اگر

هدف توان رساني باشد هر دو آنها با يك تبديل جايگزين مي شوند. با اينكه حتي در اين حال هم دو تبديل RNS لازم است اين دو تبديل از هم مستقل و قابل انجام به صورت موازي مي باشند. در آخر بايد ذكر شود كه الگوريتم تبديل RNS به ما اجازه مي دهد كه اعمال ضرب مراحل ۵و۷ را درداخل عمليات تبديل بگنجانيم. اين مسئله در هنگام بيان روش تبديل RNS دقيقتر بررسي مي شود.

## ۵-۱- تبديل سيستم RNS (Residue Conversion)

تبديل از يك سيستم عددي RNS به سيستمي ديگر در ابتدا توسط Mitz و kaltofen [32] در مبحث تقسيم RNSمورد بررسي قرار گرفت متأسفانه الگوريتم آنها نسبتاً پيچيده و از لحاظ محاسباتي هزينه بر مي باشد. الگوريتم پيشنهادي اين دو نفر مقدار دقيق نتيجه را محاسبه مي كند. در اين مقاله روش سريعتر و ساده تري پيشنهاد شده است كه به شرح زير مي باشد.

نقطه شروع كار به كارگيري قضيه باقيمانده چيني براي ايجاد عدد صحيح بزرگي است كه توسط پيمانه ها نمايش داده شده ولي همانطور كه خواهيد ديد در اين روش اين عدد صحيح را به پيمانه هاي سيستم عددي خروجي كاهش مي دهيم و تمام محاسبات ما بر روي اعداد صحيح كوچكي انجام مي شود. عدد $[x_{\omega_1}, x_{\omega_2}, \ldots, x_{\omega_n}]$ در سيستم عددي $[\omega_1, \omega_2, \ldots, \omega_n]$ نمايش عددي است كه توسط قضيه

باقیمانده چیني به صورت زیر قابل دستیابي است.

$$x=\left(\sum_{j=1}^{n} C_{\omega_j} x_{\omega_j}\right) \bmod \omega \qquad (۱-۵)$$

که روش محاسبه عدد ثابت $C_{\omega_j}$ قبلاً بیان شده است.

اگر از کاهش به پیمانه $\omega$ صرفنظر کرده و مجموع کاهش نیافته را $x'$ بنامیم، مي توانیم $x' \bmod v_i$ را با یك ضرب داخلي ساده با اعدادي کوچك و صحیح به پیمانه اعداد صحیح کوچك محاسبه کنیم.

$$x'(\bmod v_i)=\left(\sum_{j=1}^{n}(C_{\omega_j}(\bmod v_i))x_{\omega_j}\right) \bmod v_i$$

ولي مسلم است که نمي توان از mod $\omega$ صرفنظر کرد براي محاسبه آن باید $\omega\lfloor x'/\omega \rfloor$ را از $x'$ کم کنیم و یا اینکه همین مقدار $\omega\lfloor x'/\omega \rfloor$ به پیمانه $v_i$ را از هر $x_{v_i}$ بکاهیم.

$$x'-\omega\left\lfloor \frac{x'}{\omega}\right\rfloor = x'(\bmod \omega)$$

$$x' \bmod v_i-(\omega.\left\lfloor \frac{x'}{\omega}\right\rfloor) \bmod v_i = x'(\bmod \omega)(\bmod v_i)$$

پس مي توان نوشت

$$x'(\bmod \omega)(\bmod v_i)=(\sum_{j=1}^{n} C_{\omega_j} x_{\omega_j}) \bmod v_i-\omega\left\lfloor \sum_{j=1}^{n} C_{\omega_j} x_{\omega_j}/\omega\right\rfloor$$

توجه کنید که $\lfloor x'/\omega \rfloor$ عدد بزرگي نیست، زیرا که به جمع $\omega_j$ ها محدود است. در نتیجه تصحیح مورد نیاز هم یك عدد صحیح کوچك است. با اینکه محاسبه مقدار دقیق $x'/\omega$ ممکن است مشکل باشد

مي توانيم با يك حاصلضرب داخلي مثل حاصلضرب رابطه (۵-۱) تقريب مناسبي از مقدار آن بدست آوريم.

مي توانيم بعضي از ضرب ها در اعداد ثابت را نيز در اين الگوريتم تبديل بگذجانيم. اگر بخواهيم x را در عدد ثابت k در پيمانه $\omega$ قبل از تبديل به RNS دوم ضرب كنيم مي توانيم به سادگي به جاي ضرايب $C_{\omega_j}$ از ضرايب $kC_{\omega_j}.\bmod\omega$ استفاده كنيم. براي ضرب در يك عدد ثابت مثل $\ell$ در پيمانه v بعد از تبديل به سيستم RNS دوم مي‌توانيم $\ell \bmod v_i$ را در هر ضريبي كه براي محاسبه $x_{v_i}$ به كار رفته ضرب كنيم تا $x_{v_i}$ بدست آيد و همچنين اين ضريب را در ضريب تصحيح $\omega \bmod v_i$ كه در هر $\lfloor x'/\omega \rfloor$ ضرب مي شود نيز اعمال كنيم. فرمول كلي براي تبديل x در نمايش RNS، $\omega$ به $u = \ell(kx(\bmod\omega))(\bmod v)$ در سيستم RNS دوم عبارت است از:

$$a = \left\lfloor \sum_{j=1}^{n} x_{\omega_j} \frac{kC_{\omega_j}(\bmod\omega)}{\omega} \right\rfloor$$

$$u_{v_i} = \left(\sum x_{\omega_j}(kC_{\omega_j}(\bmod\omega))\right) - \ell a(\bmod v_i)$$

با اين كار مي توان ضرب هاي انجام شده در مراحل 5 و 7 الگوريتم پيشنهادي را در عمل تبديل گنجاند.

## ۵-۲- پياده سازي سخت افزاري

قبل از شروع اين بخش بايد متذكر شوم كه مقاله روش خاصي را جهت پياده سازي الگوريتم

پیشنهاد نکرده و خود من روشي را پیشنهاد مي کنم.

با دقت در الگوریتم درمي یابیم که در واقع دو عمل اصلي در کل الگوریتم مورد انجام است. یکي قسمتهاي تبدیل از یك سیستم عددي RNS به RNS دیگر

(که شامل تبدیل هاي استاندارد یعني بدون ضرب قبل و بعد از تبدیل و تبدیل با اصلاحات مربوط به ضربهاي قبل و بعد از تبدیل مي باشد) و دیگري بقیه اعمال مربوط به خود روش مونتمگري مي باشد.

نکته دیگر که در پیاده سازي در نظر گرفته شده دسته بندي کلي عناصر مورد لزوم است. اجزا مورد نظر را مي توان به دو دسته کلي تقسیم کرد:

# فصل سوم

# طرح اول

دسته اول شامل مجموعه عملیاتي است که با کمك ROM قابل پیاده سازي هستند مثل کاهش به پیمانه و اعمالي که یك (یا همه بجز یك) از عملوندها عددي ثابت است. دسته دوم عناصر ریاضي ساده باینري هستند.

یعني در کل طرح عناصر یا بکمك ROM پیاده شده اند و یا عناصري معمولي مثل جمع کننده و ضرب کننده هستند. در انتخاب عناصر دسته دوم سعي شده است که از معماري هایي با بالاترین سرعت استفاده شود.

حال به جزئیات پیاده سازي مي پردازیم:

### ۵-۲-۱- پیاده سازي تبدیل RNS

با توجه به الگوریتم این تبدیل در ابتدا باید $\left\lfloor \frac{x'}{\omega} \right\rfloor$ را با توجه به فرمول زیر محاسبه کنیم:

$$\left\lfloor \frac{x'}{\omega} \right\rfloor = \sum_{j=1}^{n} \frac{c_{\omega_j}}{\omega} . x_{\omega_j}$$

$\frac{c_{\omega_j}}{\omega}$ اعدادي ثابت هستند و مي توان ضرب آن ها در $x_{\omega j}$ را با ROM انجام داد و سپس جمع این مقادیر میاني را با یك جمع کننده چند عملوندي (MOA) انجام داد. باید توجه کرد که براي نمایش مقادیر $\frac{c_{\omega_j}}{\omega}$ که اعدادي اعشاري هستند آنها را در $2^{16}$ (که مقادیر بدست آمده بدون خطا بدست آمدند) ضرب مي کنیم تا تقریبي از نمایش آنها در سیستم دیجیتال داشته باشیم و باید در

نهایت در هنگام محاسبه نهایي مقادیر بدست آمده را به شکل اولیه خود (با تقسیم به $2^{16}$ یا همان شیفت به راست) تبدیل کنیم. در ادامه به مقادیر $(c\omega_j/\omega).2^{16}$ نام Cωerr داده ایم.

حالا باید $x' \bmod v_i$ را محاسبه کنیم. در واقع باید هر $(c_{\omega_j} \bmod v_i).(x_{\omega_j} \bmod v_i)$ را محاسبه کرده و با هم جمع کنیم. با توجه به ثابت بودن $c_{\omega_j} \bmod v_i$ ها مي توان از یك ROM براي محاسبه حاصلضرب پیمانه اي استفاده کرد و براي جمع نهایي از طرح MOMA مرجع [۱۹] بکار رفته در طرح اول استفاده کرد.

حال باید براي بدست آوردن نتیجه نهایي مقادیر $(\omega \bmod v_j)\left\lfloor \frac{x'}{\omega} \right\rfloor (\bmod v_j)$ را از هر $x' \bmod v_i$ کم کرد. در واقع باید نتیجه خروجي MOMA را بعد از شیفت به راست با کمك یك ROM به پیمانه $v_j$ کاهش دهیم و براي ضرب پیمانه اي در ω به پیمانه $v_j$ نیز از ROM استفاده مي کنیم. براي محاسبه تفریق باید از یك تفریق کننده مناسب استفاده کنیم و در نهایت نتیجه نهایي با عدد نتیجه تفریق از یك ROM و کاهش به پیمانه $v_j$ بدست مي آید. در شکلهاي (۵-۱) و (۵-۲) بلوك دیاگرام تبدیل RNS استاندارد و سریع بکار رفته در طرح سوم را مشاهده مي کنید. در شکل (۵-۱) بلوك دیاگرام تبدیل RNS استاندارد که در بالا توضیح داده شده را مشاهده مي کنیم. براي فهم شکل باید مطالب زیر را متوجه شوید.

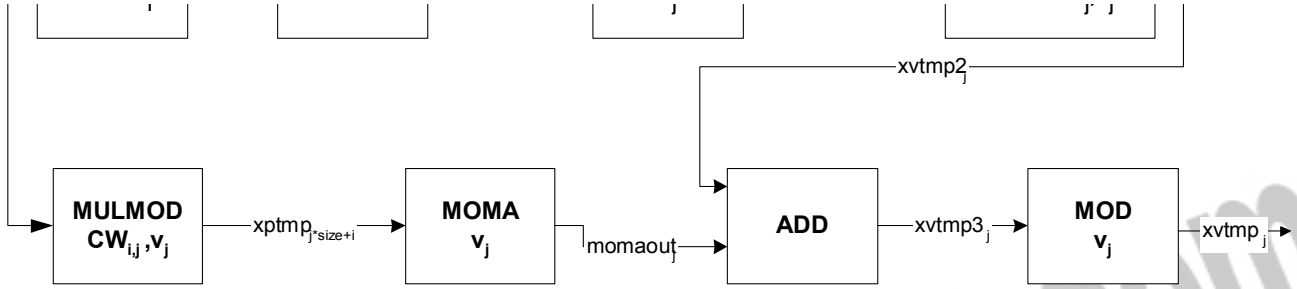
در مجموع دو مسیر وجود دارد که به یك جمع کننده (ADD) ختم مي شوند و بعد از آن فقط یك مسیر مشترك را مي‌بینید.

مسیر اول (بالا) در ابتدا هر $x\omega_i$ را با یك ROM در $C\omega err_i$ ها ضرب مي كند. نتایج این مرحله ($xerr_i$) توسط یك MOA با هم جمع مي شوند و نتیجه حاصله (xerrsum) با یك ROM به پیمانه $v_j$ كاهش مي یابد. در قدم بعدي این مقدار ($xerrsummod_j$) با كمك یك ROM در $\omega$ به پیمانه $v_j$ ضرب شده و حاصل به پیمانه $v_j$ كاهش مي یابد.

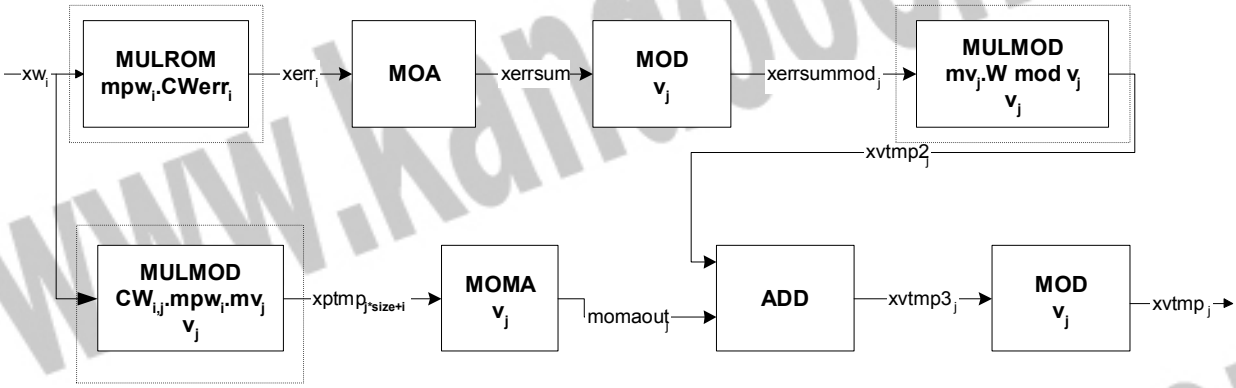
مسیر دوم (پایین) ابتدا با یك ROM هر $x\omega_i$ را در $C\omega_{i,j}$ كه همان $C\omega_i \bmod v_j$ است ضرب كرده و به پیمانه $v_j$ كاهش مي دهد سپس نتایج حاصله ($xptmp_{(j.size+i)}$) كه در آن size تعداد پیمانه هاست را با همدیگر توسط MOMA جمع پیمانه اي مي‌كند. نتیجه بدست آمده ($momaout_j$) و نتیجه بدست آمده از مسیر اول به ورودي ADD داده مي شود و عملیات تفریق انجام مي گردد. در نهایت این مقدار ($xvtmp3_j$) را به پیمانه $v_j$ كاهش مي دهیم تا $xv_i$ بدست آید.

تبدیل RNS سریع همانگونه كه قبلاً گفتیم براي تسریع عملیات، ضرب هاي در عدد ثابت مراحل ٥ و ٦ الگوریتم را به داخل الگوریتم تبدیل RNS مي آوریم در واقع تفاوت این تبدیل با تبدیل هاي اصلي از لحاظ سخت افزار در سه عدد از بلوكها كه در شكل ٥-٢ با نقطه چین مشخص شده اند، مي باشد و همان روش قبلي در این حالت نیز بكار

مي رود. در شكل (٥-٢) بلوك دياگرام اين تبديل RNS سريع را مشاهده ميكنيد توضيحات مربوط به آن (با دقت در تغييرات در بلوكهاي مشخص شده) مانند شكل (٥-١) مي باشد.




شکل ۵-۱: شکل بلوکي تبديل RNS استاندارد



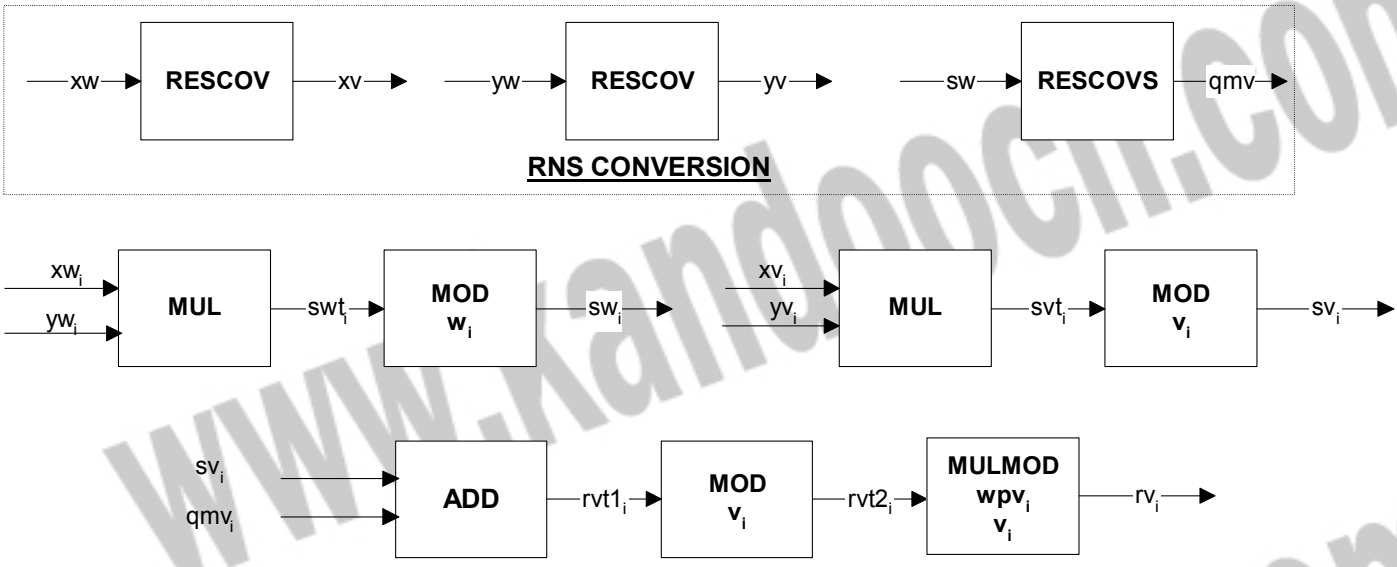

شکل ۵-۲: شکل بلوکي تبديل RNS سريع

## ۵-۲-۲- پياده سازي بخش اصلي الگوريتم (الگوريتم مونتگمري با RNS)

حال پياده سازي خود الگوريتم را بررسي مي كنيم. شكل الگوريتم بسيار گوياست. در واقع اعمال مورد نظر ضرب $x_{\omega_i} \times y_{\omega_i}$ و ضرب $x_{v_i} \times y_{v_i}$ است كه محاسبه $x_{v_i}, y_{v_i}$ ها با كمك بلوك هاي تبديل RNS استاندارد كه در بخش قبلي بررسي شد انجام مي‌شود. بعد از اين دو ضرب بايد $r_{v_i}$ ها را محاسبه كنيم. جهت اين عمل بايد $s_{v_i}$ ها را با $q_{m_{v_i}}$ كه تبديل يافته $s_{\omega_i}$ با تبديل اصلاح شده كه در بخش قبل بررسي شده جمع پيمانه اي كرده و نهايت در معكوس $\omega_i$ ضرب كنيم.

بلوك دياگرام مربوط در شكل (٥-٣) مشاهده مي شود.

مطالب گفته شده در بالا در شكل مشهود است. ضرب $x_{\omega_i} \times y_{\omega_i}$ و $x_{v_i} \times y_{v_i}$ توسط بلوك MUL و كاهش به پيمانه نتايج حاصله توسط ROM صورت مي گيرد. در بالاي شكل نيز عمليات تبديل RNS استاندارد (RESCOV) و تبديل RNS سريع (RESCOVS) مشخصه مي شود. در پايين شكل عمليات محاسبه $rv_i$ كه شامل جمع $sv_i$ و $q_{mv_i}$ كاهش به پيمانه $v_i$ با ROM و ضرب پيمانه اي در معكوس $\omega$به پيمانه $v_i$ توسط ROM را مشاهده مي كنيد.




**شكل ٥-٣: شكل بلوكي طرح سوم**

### ۵-۳- محاسبه پيچيدگي مساحت و تأخير طرح سوم:

قبلاً گفتيم كه عناصر پياده سازي ما كلاً به دو دسته اصلي تقسيم مي شوند، يكي اعمالي كه با كمك ROM انجام مي شوند و ديگري اعمالي كه با عناصر رياضي معمولي مثلاً ضرب كننده و ... انجام مي شوند. پس ابتدا تك تك عناصر مورد استفاده را از لحاظ تأخير و مساحت مورد بررسي قرار مي دهيم و در نهايت تأخير ومساحت كل طرح را بررسي مي كنيم.

### ۵-۳-۱- عناصر وابسته به ROM:

اين عناصر شامل، كاهش پيمانه اي، ضرب پيمانه اي و ضرب در عدد ثابت مي باشند. تأخير و مساحت همه اينها را با فرمولهاي زير (كه قبلاً هم ديده ايم) بيان مي‌كنيم.

$T = 2\,b$ , b= number of input bits

$A = 2^b.n$ , n = number of output bits

### ۵-۳-۲- عناصر رياضي:

**جمع كننده :**

براي عمل جمع از جمع كننده Parallel-perfix (Brent-kung) مرجع [33] استفاده شده كه داراي پيچيدگي زير مي باشد.

$T = 4 \log n$

$A = 10\,n$

**ضرب كننده:**

براي ضرب از ضرب كننده «column compression tree» مرجع [34] استفاده شده كه پيچيدگي هاي آن

$$t = 3\lceil \log_2^n \rceil$$

$$A = n^2$$

**جمع کننده چند عملوندي (Multi operand Adder) :**

براي اين نوع از عمليات‌ها از طرح Tree-Adder مرجع [33] استفاده شده كه تاخير و مساحت به شرح زير مي باشد.

$$T = \log m + \log n \qquad m = \text{number of operands}$$

$$A = mn + n \log n$$

**جمع كننده پيمانه اي چند عملوندي (Multi-operand Modular-Adder)**

اين عمليات را قبلاً نيز در طرح اول مورد استفاده قرار داديم كه در اينجا يكبار ديگر پيچيدگي هاي آن را ذكر مي كنيم :

$$T = 2(\text{size} + \lceil \log v_i \rceil - 2) + 2.\lceil \log(\text{size}(v_i - 1)) \rceil$$

$$A = (\text{size} - 1).\lceil \log v_i \rceil - \lceil \log(\text{size}(v_i - 1)) \rceil - 1 + 2^{\lceil \log(\text{size}(v_i - 1)) \rceil}$$

## ۵-۳-۳- تأخير و مساحت تبديل كننده RNS استاندارد

### تأخير تبديل كننده RNS استاندارد

حال به محاسبه تأخير تبديل كننده RNS استاندارد مي پردازيم. با دقت در بلوك دياگرام اين تبديل كننده، شكل (۵-۱)، متوجه دو مسير متفاوت مي شويم كه به يك مسير مشترك (ADDER) ختم مي شوند. پس تأخير دومسير و تأخير مسير مشترك را با كمك فرمولهايي كه در بخش قبل آمده محاسبه مي كنيم. لازم به ذكر است كه در تمام فرمولهاي لگاريتم، پايه ۲ مي باشد و همچنين پارامتر Size تعداد مانده‌ها مي‌باشد و

تعداد بیت $\omega_i$ یعني $\lceil \log_2 \omega_i \rceil$ و $v_i$ یعني $\lceil \log_2 v_i \rceil$ برابر n فرض شده است. در واقع فرض کردیم که تعداد بیت $\omega_i$ و $v_i$ برابر باشد. البته احتمال تفاوت تعداد بیت وجود دارد ولي ما مجبور به این فرض براي ساده سازي هستیم.

۱- محاسبه تأخیر مسیر منتهي به ورودي اول ADDER:

الف- تاخیر MULROM

عملیات MULROM ضرب در عدد ثابت توسط ROM است. تاخیر ROM در بخش ۵-۳-۱ بیان شده و با توجه به اینکه تعداد بیت هاي ورودي به این ROM حداکثر $\lceil \log \omega_i \rceil$ مي باشد. پس خواهیم داشت:

$$MULROM_{delay} = 2\lceil \log \omega_i \rceil = 2n$$

ب- تاخیر MOA

با توجه به رابطه تاخیر MOA در بخش ۵-۳-۲ نیازمند تعداد اعداد ورودي و حداکثر بیت هر عدد هستیم. تعداد اعداد ورودي size مي باشد و حداکثر تعداد بیت با توجه به مرحله قبل $\lceil \log(\omega_i . C\omega err_i) \rceil$ است پس مي توان نوشت:

$$MOA_{delay} = \log(size) + \log(\lceil \log(\omega_i . C\omega err_i) \rceil)$$

همانطور که در بخش ۵-۲-۱ گفته شده $C\omega err_i$ از ضرب یك عدد کوچکتر از یك در $2^{16}$ بدست آمده و در نتیجه حداکثر ۱۶ بیت دارد تعداد بیت عدد $\omega_i$ هم که n فرض شده پس خواهیم داشت:

$$MOA_{delay} = \log(size) + \log(16 + n)$$

ج- تاخیر MOD

عملیات کاهش به پیمانه نیز با کمک ROM انجام می گیرد. با توجه به مرحله قبل تعداد بیت های ورودی حداکثر برابر $\lceil \log(size.C\omega err_i.\omega_i) \rceil$ خواهد بود یعنی تعداد بیت عدد حاصل از جمع size عدد (16+n) بیتی. پس خواهیم داشت:

$$MOD_{delay}=2\lceil \log(size.2^{(n+16)}) \rceil$$

د- تاخیر MULMOD

MULMOD نشان دهنده عملیات ضرب پیمانه ای دریک عدد ثابت با ROM است. خروجی مرحله قبل حداکثر $\lceil \log v_i \rceil$ بیت دارد. پس خواهیم داشت:

$$MULMOD_{delay} = 2\lceil \log v_i \rceil = 2n$$

پس در مجموع تاخیر مسیر یک (بالایی) برابر است با:

$$First\ Path_{delay}=4n+\log(size)+2\lceil \log(size.2^{(n+16)}) \rceil+\log(16+n)$$

۲- محاسبه تأخیر مسیر منتهی به ورودی دوم ADDER:

الف- تاخیر MULMOD

عملیات با ROM است و تعداد بیت ورودی حداکثر $\lceil \log \omega_i \rceil$ است. پس:

$$MULMOD_{delay} = 2\lceil \log \omega_i \rceil = 2n$$

ب- تاخیر MOMA

باید با توجه به فرمول مربوط به MOMA در بخش ۵-۳-۲ عمل کنیم با توجه به بلوک دیاگرام تعداد عملوندها size و حداکثر تعداد بیت هر کدام $\lceil \log v_i \rceil$ می‌باشد.

$$MOMA_{delay}= 2(size+n-2)+2\lceil \log(size(2^n-1)) \rceil$$

پس در مجموع تاخیر مسیر دوم به صورت زیر خواهد بود:

$$\text{Second Path}_{\text{delay}} = 2n + 2(\text{size} + n - 2) + 2\lceil \log(\text{size}(2^n - 1)) \rceil$$

۳- محاسبه تأخیر مسیر مشترک:

الف- تاخیر ADDER

براي محاسبه اين تاخير از فرمول مربوطه در بخش ۵-۳-۲ استفاده مي كنيم تعداد بيت ورودي ها حداكثر $\lceil \log v_i \rceil$ مي باشد.

$$\text{ADDER}_{\text{delay}} = 4\log(\lceil \log v_i \rceil) = 4\log(n)$$

ب- تاخير MOD

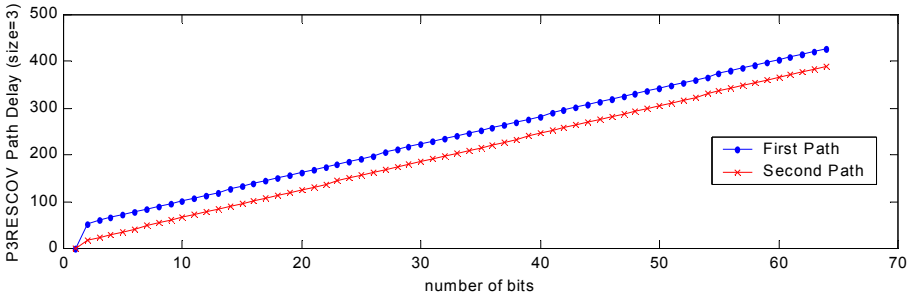
اين عمليات با ROM صورت مي گيرد كه تعداد بيت ورودي حداكثر $\lceil \log(2v_i) \rceil$ است .

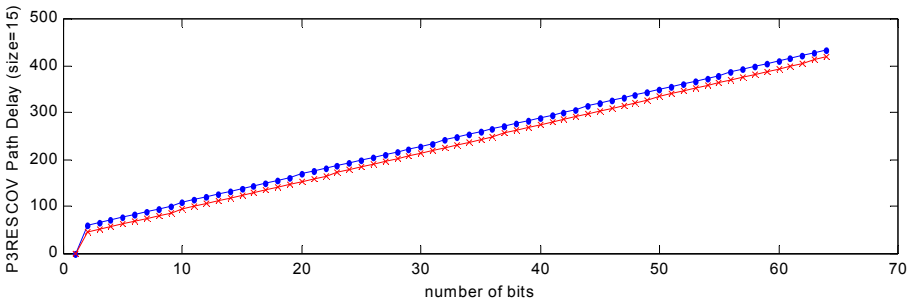
$$\text{MOD}_{\text{delay}} = 2\lceil \log(2v_i) \rceil = 2(1+n)$$

و تاخير كل مسير مشترك برابر است با:

$$\text{Common Path}_{\text{delay}} = 4\log(n) + 2(n+1)$$

بررسي تاخير مسير اول و دوم با كمك MATLAB نشان داده كه براي size<22 با هر تعداد بيت تاخير مسير اول از مسير دوم بيشتر است. به دليل اينكه احتمال انتخاب size با اندازه اي بيشتر از ۲۲ خيلي ضعيف است اين فرض كه مسير اول بعنوان مسير بحراني انتخاب شود فرض مناسبي است. در شكل (۵-٤) منحني تاخير برحسب بيت براي مقادير size برابر با سه و ۱۵ را ببينيد.




شـ

پـ

استاندارد براي مقادير size<22 بصورت زير قابل بيان است:

$$RESCOV_{delay}=First\ Path_{delay}+Common\ Path_{delay}$$
$$=4n+\log(size)+\log(16+n)+2\lceil\log(size.2^{(n+16)})\rceil$$
$$+4\log(n)+2(n+1)$$

**- مساحت تبديل كننده RNS استاندارد:**

۱) مساحت $MULROM\,(C\omega err_i)$

همانطور كه قبلاً ديديم عمليات ضرب در عدد ثابت با ROM انجام مي‌شود. با توجه به فرمولهاي بخش ۵-۳-۱ و اينكه به اندازه تعداد پيمانه ها يعني size از اينگونه ROM داريم (زيرا اين عمل روي هر مانده صورت مي گيرد) مساحت اشغال شده توسط اين بخش بصورت زير خواهد بود.

$$size.\left(2^{\lceil\log(\omega_i)\rceil}.\lceil\log(C\omega err_i.\omega_i)\rceil\right)$$

همانطور که قبلاً گفتیم $C\omega err_i$ از ضرب یک عدد کوچکتر از یک در $2^{16}$ بدست می آید پس ۱۶ بیت دارد پس می توان نوشت:

$$size(2^n.\log\lceil 2^{16}.2^n\rceil)$$
$$= size.2^n.(16+n)$$

۲- مساحت MOA

با توجه به فرمول مساحت MOA در بخش ۵-۳-۲ و با توجه به اینکه تعداد عملوندها size می باشد و اعداد ورودي به MOA در بدترین حالت $C\omega err_i.\omega_i$ می‌باشد می توان مساحت را بصورت زیر نوشت:

$$size.\lceil\log(C\omega err_i.\omega_i)\rceil+\lceil\log(C\omega err_i.\omega_i)\rceil+\log\lceil\log(C\omega err_i.\omega_i)\rceil$$

که با توجه به مباحث بالا خواهیم داشت:

$$(16+n).(size+\log(16+n))$$

۳- مساحت $MULMOD(C\omega_{i,j}, v_j)$

بازهم براي عملیات ضرب در عدد ثابت بصورت پیمانه اي از ROM استفاده می‌شود.

با توجه به اینکه تعداد بیتهاي ورودي آن $\lceil\log\omega_i\rceil$ و تعداد بیت هاي خروجي $\lceil\log v_i\rceil$ می باشد و به تعداد size از اینگونه ROM ها داریم می توان براي مساحت این بخش نوشت.

$$size.2^{\lceil\log\omega_i\rceil}.\lceil\log v_i\rceil = size.2^n.n$$

۴- مساحت MOMA

تعداد عملوندهاي این MOMA، size می باشد و هر ورودي حداکثر $\lceil\log v_i\rceil$ بیت خواهد داشت. با توجه به فرمول مربوطه داریم:

$$(size-1)n-\lceil\log((size)(2^n-1))\rceil+2^{\lceil\log((size)(2^n-1))\rceil}$$

۵- مساحت ADD:

هر ورودي اين جمع كننده حداكثر $\lceil \log v_i \rceil$ و با توجه به فرمول مساحت كننده در بخش ۵-۳-۲ براي مساحت اين جمع كننده خواهيم داشت:

$$10.\lceil \log v_i \rceil = 10n$$

۶- مساحت $MOD(v_j)$ مسير اول

تعداد بيت هاي ورودي حداكثر $\lceil \log(2^{(16+n)}.size) \rceil$ خواهد بود كه ورودي اين مرحله خروجي MOA است كه اين واحد به تعداد size عدد 16+n بيتي را با هم جمع مي كند. تعداد بيت هاي خروجي هم حداكثر $\lceil \log v_i \rceil$ خواهد بود پس مساحت با توجه به تكرار شده اين ROM به تعداد size بشرح زير است:

$$size.2^{\lceil \log(2^{16+n}.size) \rceil}.n$$

۷- مساحت $MOD(v_j)$ در مسير مشترك

عمليات با ROM است. تعداد بيت هاي ورودي آن با توجه به اينكه حداكثر مقدار ورودي به آن حداكثر مقدار خروجي از ADDER يعني $v_j$ خواهد بود مقدار $\lceil \log v_i \rceil$ را خواهد داشت و تعداد بيت هاي خروجي آن هم $\lceil \log v_i \rceil$ خواهد بود و از هر واحد به تعداد size خواهيم داشت پس مساحت بشكل زير است:

$$size.2^n.n$$

۸- مساحت MULMOD $(\omega \bmod v_j, v_j)$

عملیات فوق با ROM صورت می گیرد. ورودي حداکثر مقدار خروجي MOD یعني $\lceil \log v_i \rceil$ خواهد بود و تعداد بیت هاي خروجي آن نیز $\lceil \log v_i \rceil$ خواهد بود و به تعداد size تکرار مي شود پس داریم:

$size.2^n.n$

پس مساحت کل RNS استاندارد بصورت زیر خواهد بود:

$$RESCOV_{Area} = size.2^n(16+n) + (16+n)(size + \log(16+n)) + 3n.size.$$
$$2^n + n(size-1) - \lceil \log(size(2^n-1)) \rceil + 2 + 10n + n.$$
$$size.2^{\lceil \log(size 2^{16+n}) \rceil}$$

## ٤-٣-٥- محاسبه مساحت و تأخیر تبدیل کننده RNS سریع

### ۱- تأخیر تبدیل کننده RNS سریع

الف- تاخیر MULROM $(mp\omega_i.C\omega err_i)$

این عملیات با ROM صورت مي گیرد. تعداد بیت ورودي حداکثر $\lceil \log \omega_i \rceil$ خواهد بود پس تاخیر این بخش بصورت زیر است:

$$MULROM_{delay} = 2\lceil \log \omega_i \rceil = 2n$$

ب- تاخیر MOA

با توجه به رابطه تاخیر MOA در بخش ۵-۳-۲ نیازمند تعداد اعداد ورودي و حداکثر بیت هر عدد هستیم. تعداد اعداد ورودي size مي باشد و حداکثر تعداد بیت با توجه به مرحله قبل $\lceil \log(\omega_i.mp\omega_i.C\omega err_i) \rceil$ است پس مي توان نوشت:

$$MOA_{delay} = \log(size) + \log(\lceil \log(\omega_i.mp\omega_i.C\omega err_i) \rceil)$$

همانطور که قبلاً گفتیم $C\omega err_i$ عددي ۱۶ بيتي است و حداکثر $mp\omega_i$ هم $\omega_i$ مي‌باشد زيرا عددي است در پيمانه $\omega_i$. پس مي توان نوشت:

$$MOA_{delay} = \log(size) + \log(16 + 2n)$$

ج- تاخير MOD

عمليات با ROM صورت مي گيرد. با توجه به مرحله قبل تعداد بيت هاي ورودي حداکثر برابر $\lceil \log(\omega_i.mp\omega_i.size.C\omega err_i.\omega_i) \rceil$ خواهد بود. پس خواهيم داشت:

$$2\lceil \log(size.mp\omega_i.C\omega err_i.\omega_i) \rceil$$

که مي توان نوشت:

$$2\lceil \log(size.2^n.2^{16}.2^n) \rceil$$

که در نهايت داريم:

$$MOD_{delay}=2\lceil \log(size.2^{(2n+16)}) \rceil$$

د- تاخير MULMOD

عمليات با ROM صورت مي گيرد و با توجه به شکل بلوکي و نوع عمليات يعني کاهش پيمانه اي تعداد بيت هاي ورودي و خروجي $\lceil \log v_i \rceil$ است. پس خواهيم داشت:

$$MULMOD_{delay} = 2\lceil \log v_i \rceil = 2n$$

پس در مجموع تاخير مسير يك (بالايي) برابر است با:

$$First\ Path_{delay}=4n+\log(size) + 2\lceil \log(size.2^{(2n+16)}) \rceil + \log(16 + 2n)$$

۲- محاسبه تأخير مسير منتهي به ورودي دوم ADDER:

الف- تاخير MULMOD

عملیات با ROM است و تعداد بیت ورودي حداکثر $\lceil \log \omega_i \rceil$ است. پس داریم:

$$MULMOD_{delay} = 2\lceil \log \omega_i \rceil = 2n$$

ب- تاخیر MOMA

باید با توجه به فرمول مربوط به MOMA در بخش ۵-۳-۲ عمل کنیم با توجه به بلوك دیاگرام تعداد عملوندها size و حداکثر تعداد بیت هر کدام $\lceil \log v_i \rceil$ مي‌باشد.

$$MOMA_{delay} = 2(size + n - 2) + 2\lceil \log(size(2^n - 1)) \rceil$$

پس در مجموع تاخیر مسیر دوم به صورت زیر خواهد بود:

$$Second\ Path_{delay} = 2n + 2(size + n - 2) + 2\lceil \log(size(2^n - 1)) \rceil$$

۳- محاسبه تأخیر مسیر مشترك:

الف- تاخیر ADDER

براي محاسبه این تاخیر از فرمول مربوطه در بخش ۵-۳-۲ استفاده مي کنیم تعداد بیت ورودي ها حداکثر $\lceil \log v_i \rceil$ مي باشد.

$$ADDER_{delay} = 4\log(\lceil \log v_i \rceil) = 4\log(n)$$

ب- تاخیر MOD

این عملیات با ROM صورت مي گیرد که تعداد بیت ورودي حداکثر $\lceil \log(2v_i) \rceil$ است .

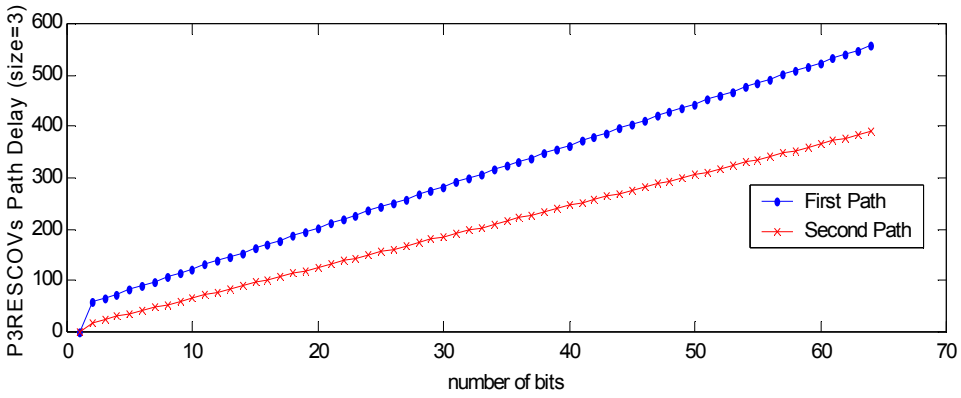
$$MOD_{delay} = 2\lceil \log(2v_i) \rceil = 2(1 + n)$$

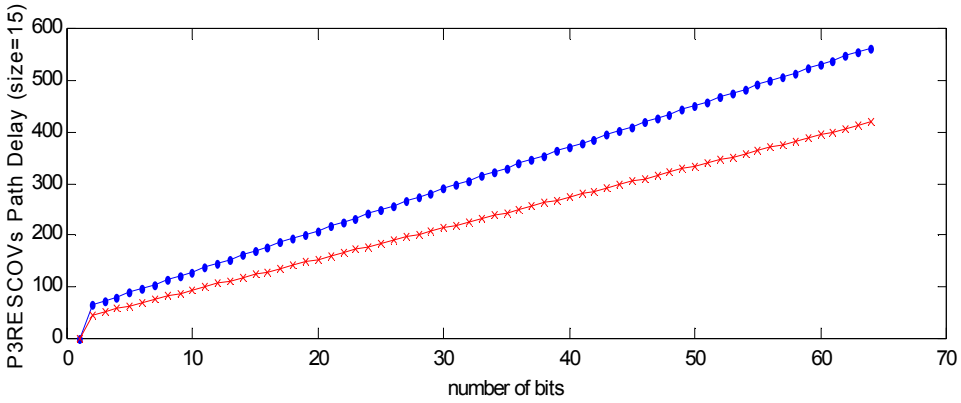
و تاخیر کل مسیر مشترك برابر است با:

$$Common\ Path_{delay} = 4\log(n) + 2(n+1)$$

بررسي تاخیر مسیر اول و دوم با کمك MATLAB نشان داده که براي size<24 با هر تعداد بیت تاخیر مسیر اول از مسیر دوم بیشتر است. به

دلیل اینکه احتمال انتخاب size با اندازه اي بیشتر از ۲٤ خیلي ضعیف است این فرض که مسیر اول بعنوان مسیر بحراني انتخاب شود فرض مناسبي است. در شکل (٥-٥) منحني تاخیر برحسب بیت براي مقادیر size برابر با سه و ۱٥ را ببینید.




$$RESCOVS_{delay}=First\ Path_{delay}+Common\ Path_{delay}$$
$$= 4n + \log(size) + \log(16+2n) + 2\lceil \log(size.2^{(2n+16)}) \rceil + 4\log(n) + 2(n+1)$$

## مساحت تبدیل کننده RNS سریع

۱) مساحت $MULROM\ (mp\omega_i.C\omega err_i)$

روش محاسبه مساحت در RNS استاندارد بیان شده که مساحت این بخش بصورت زیر می باشد:

$$size.\left(2^{\lceil \log(\omega_i) \rceil}.\lceil \log(C\omega err_i.mp\omega_i.\omega_i) \rceil\right)$$

همانطور که قبلاً گفتیم $C\omega err_i$ از ضرب یك عدد کوچکتر از یك در $2^{16}$ بدست می آید پس ۱۶ بیت دارد پس می توان نوشت:

$$size.(2^n.(16+2n))$$

۲- مساحت MOA

روش محاسبه همانند همین بخش در RNS استاندارد است و بصورت زیر نتیجه می شود:

$$size.\lceil \log(mp\omega_i.C\omega err_i.\omega_i) \rceil + \lceil \log(mp\omega_i.C\omega err_i.\omega_i) \rceil + \log\lceil \log(mp\omega_i.C\omega err_i.\omega_i) \rceil$$

که به صورت زیر ساده می شود:

$$(16+2n).(size+\log(16+2n))$$

۳- مساحت $MULMOD\left(C\omega_{i,j}.mp\omega_i.mv_j, v_j\right)$

برای این واحد نیز همانگونه که قبلاً در RNS استاندارد دیدیم عمل می کنیم. تعداد بیت های ورودی $\lceil \log \omega_i \rceil$ و تعداد بیت های خروجی $\lceil \log v_i \rceil$ می باشد و به تعداد size از اینگونه ROM ها داریم می توان برای مساحت این بخش نوشت.

$$size.2^{\lceil \log \omega_i \rceil}.\lceil \log v_i \rceil = size.2^n.n$$

٤- مساحت MOMA

روش محاسبه همان فرمول بخش ۵-۳-۲ است. تعداد ورودیها size و هر ورودی حداکثر $\lceil \log v_i \rceil$ بیت دارد. با توجه به فرمول مربوطه داریم:

$$(size-1)n - \lceil \log((size)(2^n-1)) \rceil + 2^{\lceil \log((size)(2^n-1)) \rceil}$$

۵- مساحت ADDER:

با توجه به فرمول مساحت جمع کننده در بخش ۵-۳-۲ و این که هر ورودي حداکثر $\lceil \log v_i \rceil$ بیت است مساحت بصورت زیر است:

$$10.\lceil \log v_i \rceil = 10n$$

۶- مساحت $MOD(v_j)$ مسیر اول

تعداد بیت هاي ورودي حداکثر $\lceil \log(2^{(16+2n)}.size) \rceil$ خواهد بود که ورودي این مرحله خروجي MOA است که این واحد به تعداد size عدد 16+2n بیتي را با هم جمع مي کند. تعداد بیت هاي خروجي هم حداکثر $\lceil \log v_i \rceil$ خواهد بود پس مساحت با توجه به تکرار شده این ROM به تعداد size بشرح زیر است:

$$size.2^{\lceil \log(2^{16+2n}.size) \rceil}.n$$

۷- مساحت $MOD(v_j)$ در مسیر مشترک

با توجه به وجود ADDER در مرحله قبل ورودي این بخش حداکثر $v_j$ خواهد بود و خروجي آن $\omega_j$ خواهد بود پس همانند تبدیل RNS استاندارد خواهیم داشت:

$$size.2^n.n$$

۸- مساحت $MULMOD\ (mv_j.\omega \bmod v_j, v_j)$

عملیات فوق با ROM صورت مي گیرد. ورودي حداکثر مقدار خروجي MOD یعني $\lceil \log v_i \rceil$ خواهد بود و تعداد بیت هاي خروجي آن نیز $\lceil \log v_i \rceil$ خواهد بود و به تعداد size تکرار مي شود پس داریم:

$$size.2^n.n$$

پس مساحت کل RNS سریع بصورت زیر خواهد بود:

$$RESCOVS_{Area}=$$

$$size.2^{n}(16+2n)+(16+2n)(size+\log(16+2n))+3n.size.$$

$$2^{n}+n(size-1)-\lceil\log(size(2^{n}-1))\rceil+2+10n+n.$$

$$size.2^{\lceil\log(size\,2^{16+2n})\rceil}$$

## ۵-۳-۵- مساحت و تأخیر طرح سوم

حال به محاسبه مساحت و تأخیر کل طرح سوم می پردازیم. لازم به ذکر است که در این بخش فرض بر این است که محاسبات مربوط به تبدیل RNS استاندارد (RESCONV) و تبدیل RNS سریع (RESCONVS) را داریم.

- **محاسبه تأخیر در طرح سوم**

با بررسي بلوك دياگرام طرح سوم در مي يابيم كه در اينجا هم كلاً دو مسير جداگانه داريم به ورودي ADD مي آيند و از آن به بعد مسير سيگنالها يكسان مي‌شود.

**- محاسبه تأخیر مسیر اول**

۱) تأخیر ضرب کننده

در این محاسبه هم از فرمول بخش ۵-۳-۲ در مورد ضرب کننده استفاده می کنیم با توجه به اینکه هر ورودی حداکثر $\omega_i$ خواهد بود در مورد تاخیر داریم:

$$MUL_{delay}=3\lceil \log(\lceil \log \omega_i \rceil)\rceil = 3\lceil \log n \rceil$$

۲) تأخیر $MOD\ (\omega_j)$

همانطور که دیدیم این عملیات با ROM صورت می گیرد و با توجه به وجود ضرب کننده در مرحله قبل حداکثر تعداد بیت ورودی $\lceil \log \omega_i^2 \rceil$ خواهد بود. پس:

$$MOD(\omega_j)_{delay}=\lceil \log \omega_i^2 \rceil=4n$$

۳) تأخیر تبدیل RNS سریع روی Sω:

قبلاً محاسبه این تاخیر انجام شد $(RESCOVS_{delay})$ و تاخیر مسیر اول برابر خواهد بود.

$$First\ Path_{delay}= RESCOVS_{delay}+4n+3\lceil \log n \rceil$$

**- محاسبه تأخیر مسیر دوم:**

۱) تأخیر عملیات تبدیل RNS استاندارد روی $y\omega_i, x\omega_i$

قبلاً محاسبه شده

۲) تأخیر ضرب کننده

ورودی های ضرب کننده حداکثر مقدار $v_i$ را دارند و در نتیجه تاخیر بصورت زیر خواهد بود:

$$MUL_{delay}=3\lceil \log n \rceil$$

۳) تأخیر $MOD\ (v_i)$

عملیات با ROM صورت گرفته و حداکثر مقدار ورودي $v_i^2$ است و در نتیجه

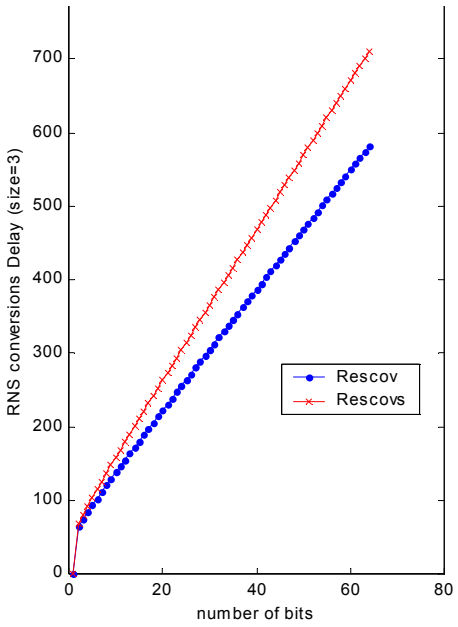
$$MOD\ (v_i)_{delay}=4n$$

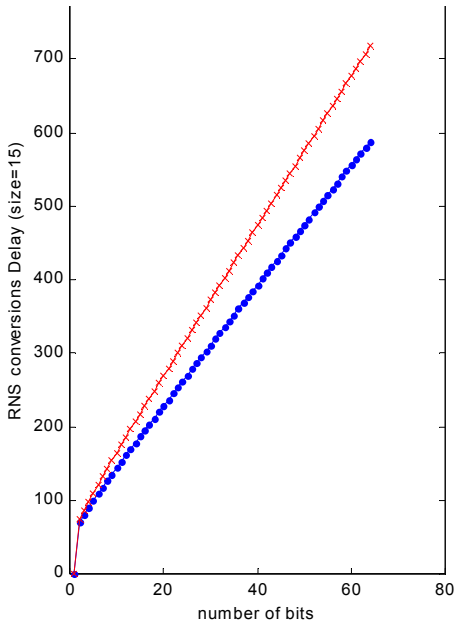
و در نتیجه تاخیر کل مسیر دوم برابر خواهد بود با:

$$Second\ Path_{delay}= RESCOVS_{delay}+4n+3\lceil \log n \rceil$$

با توجه به فرمولهاي مسير اول و دوم مشخص است که مبناي تعيين مسير براي مقايسه تاخير تبديل RNS استاندارد ($RESCOV_{delay}$) با تاخير تبديل RNS سريع ($RESCOVS_{delay}$) مي باشد.

بررسي فرمول هاي مربوط به تاخير اين دو واحد نشان مي دهد که وجود 2n در فرمول $RESCOVS_{delay}$ بجاي n در فرمول مربوط به $RESCOV_{delay}$ باعث بزرگتر شدن تاخير RNS سريع خواهد شد. در واقع منظور ما از RNS سريع، افزايش سرعت RNS نبوده و بلکه منظور افزايش سرعت کل عمليات (با توجه به حذف دو عمليات ضرب در عدد ثابت) بوده است منحني مربوط به اين دو تاخير براي ۳ و ۱۵ پيمانه در شکل (۵-۶) تاييديه بر مطالب فوق است پس با توجه به مطالب فوق نتيجه مي شود که تاخير مسير اول بيشتر خواهد بود.




۱) تاخیر جمع کننده

در اینجا با توجه به فرمول جمع کننده در بخش ۵-۳-۲ عمل می کنیم و در نظر میگیریم که ورودی حداکثر $v_i$ خواهد بود پس خواهیم داشت:

$$ADD_{delay}=4\lceil \log \lceil \log v_i \rceil \rceil = 4 \log n$$

۲) تأخیر $MOD(v_i)$

جمع کننده، مرحله قبلی این عملیات است و ورودی به این مرحله حداکثر $2v_i$ خواهد بود. پس:

$$MOD(v_i)_{delay}=2\lceil \log(2v_i) \rceil = 2(n+1)$$

۳) تأخیر $\text{MULMOD}(mpv_i, v_i)$

در این بخش حداکثر ورودي $v_i$ خواهد بود و در نتیجه

$$\text{MULMOD}(mpv_i, v_i)_{delay}=2n$$

و تاخیر کل مسیر مشترك برابر است با

$$\text{Common Path}_{delay}= 4(\log n)+2(n+1)+2n$$

و در نتیجه تاخیر طرح سوم را مي توان بصورت زیر نوشت:

$$P3_{delay}=\text{First Path}_{delay}+\text{Common Path}_{delay}$$
$$=RESCOVS_{delay}+4n+3\lceil \log n \rceil+4n+4(\log n)+2$$
$$=10n+\log(size)+\log(16+2n)+8\log(n)+4+3\lceil \log n \rceil$$
$$+2\lceil \log(size.2^{(2n+16)}) \rceil$$

**- محاسبه مساحت طرح سوم**

۱- مساحت ضرب کننده ها

با توجه به فرمول مساحت ضرب کننده در بخش ۵-۳-۲ و اینکه ورودي ضرب کننده ها به ترتیب حداکثر $v_i, \omega_i$ مي باشد و از هر ضرب کننده به تعداد پیمانه ها لازم است. براي هر ضرب کننده خواهیم داشت:

a) $\lceil \log \omega_i \rceil^2.size = n^2.size$

b) $\lceil \log v_i \rceil^2.size = n^2.size$

۲- مساحت MOD در هر کدام از مسیرهاي اول و دوم

با استفاده از فرمول بخش ۵-۳-۱ در مورد عملیات با ROM و اینکه ورودي هر MOD به ترتیب، حداکثر $v_i^2, \omega_i^2$ مي باشد و خروجي آن ها حداکثر $v_i, \omega_i$ خواهند بود براي مساحت این دو داریم:

a) $2^{\lceil \log \omega_i^2 \rceil}.\lceil \log \omega_i \rceil.size = 2^{2n}.n.size$

b) $2^{\lceil \log v_i^2 \rceil}.\lceil \log v_i \rceil.size = 2^{2n}.n.size$

۳- مساحت ADDER

هر ورودي اين بخش حداكثر مقدار $v_i$ دارد و با توجه به بخش ۵-۳-۲ و اينكه به تعداد size از اين واحد لازم داريم.

$ADDER_{delay}=10\lceil \log v_i \rceil.size=10n.size$

٤- مساحت $MULMOD(\omega pv_i, v_i)$

حداكثر مقدار ورودي و خروجي اين ROM، $v_i$ است و بتعداد پيمانه ها از اين ROM خواهيم داشت. پس:

$MULMOD_{delay}=2^{\lceil \log v_i \rceil}.\lceil \log v_i \rceil.size=2^n.n.size$

٥- مساحت $MOD(v_i)$ در مسير مشترك:

با توجه به وجود ADDER در مرحله قبل ورودي اين ROM حداكثر مقدار $2v_i$ را خواهد داشت و خروجي آن نيز حداكثر $v_i$ خواهد بود و به تعداد size از اين واحد خواهيم داشت:

$MOD(v_i)_{delay}=2^{\lceil \log 2v_i \rceil}.\lceil \log v_i \rceil.size = 2^{(n+1)}.n.size$

٦- مساحت تبديل RNS استاندارد

از اين واحد به دو مبدل براي تبديل $y_\omega, x_\omega$ به $y_v, x_v$ نياز داريم و مساحت آن ($RESCOV_{Area}$) قبلاً محاسبه شده است.

۷-مساحت تبديل RNS سريع

اين واحد را براي تبديل $S_\omega$ يعني حاصلضرب $y_\omega, x_\omega$ به $q_{m_v}$ بكار خواهيم گرفت و مساحت آنها ($RESCOVS_{Area}$) قبلاً محاسبه كرديم.

مساحت كل طرح سوم بصورت زير قابل محاسبه است:

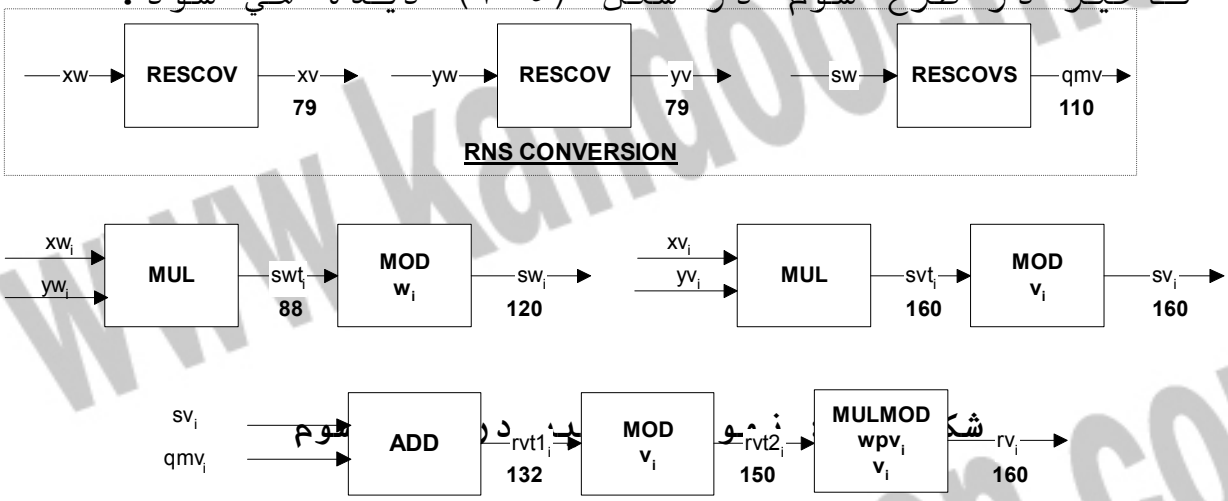
$$P3_{Area} = RESCOVS_{Area} + 2.RESCOV_{Area} + 2n^2.size + 2^{2n+1}.n.size + 10n.size + 2^n.n.size + 2^{(n+1)}.n.size$$

## ٥-٤- نتايج پياده سازي در طرح سوم:

براي آزمايش طرح سوم و چگونگي بدست آوردن نتيجه ضرب به روش مونتگمري در RNS ضرب عدد يك در عدد يك به پيمانه m= 55261 مورد بررسي قرار گرفت. RNS اصلي w=(37,41,43) انتخاب شده كه در نتيجه W=65231 است. همينطور RNS كمكي V=(29,31,251) انتخاب شده. با توجه به تئوري الگوريتم مونتگمري حاصل مورد نظر به صورت زير است.

$$1\times1\times(W^{-1}\bmod m)=38073$$

نتيجه شبيه سازي به VHDL نشان مي دهد كه حاصلضرب منتجه به شكل (25,5,172) در RNS پيمانه V به دست آمده كه نمايشگر عدد ۳۸۰۷۳ در اين RNS است و در نتيجه عمليات صحيح صورت گرفته. مساحت كل طرح بر حسب گيت ٦٢٥٧٨٩ است و نتايج تأخير در طرح سوم در شكل (٥-٧) ديده مي شود.



## ٦- طرح چهارم:

مرجع مورد نظر ما به عنوان طرح چهارم جهت ضرب پيمانه اي در RNS، مرجع [6] يعني مقاله "Residue number systems: a key to parllelism in public key cryptography" است. بامطالعه ابتدايي مقاله مشاهده مي شود كه مقاله ازمرجع ديگري براي بيان كامل عمليات ضرب و كاهش پيمانه اي نام مي برد. به همين دليل ادامه و جزئيات كار ما نيز در آن مقاله يعني مرجع [35] مي باشد. پس در ادامه به سراغ مقاله مذكور مي رويم. درمقدمه مقاله اشاره شده است كه روشهاي كاهش پيمانه اي براي مسائلي كه شامل اعداد صحيح بسيار بزرگي هستند (مانند مسأله امضا ديجيتال و كليد عمومي [36]) نياز به روشهاي موثري جهت كاهش پيمانه اي دارند. در اغلب اوقات همينگونه مسائل تمايل به تغيير شكل به شكل پياده سازيهاي سريع در هنگام به كارگيري سيستم عددي RNS دارند. محور اصلي اين مقاله تركيب سيستم عددي RNS با روشهاي كاهش پيمانه اي موثر مثل روش مونتگمري [2] يا روش [37] مي باشد. مقايسه [2] و [37] نشان مي دهد كه روش [2] بهينه است و به همين دليل در اين مقاله مورد استفاده قرار مي گيرد. كاربرد عمليات بيان شده در اين مقاله دريافتن پياده سازي سريعي جهت الگوريتم RSA است. در ادامه جزئيات يك كاهش پيمانه اي سريع مورد بررسي قرار گرفته كه با اينكه مطالبي مشترك با

مباحث قبلي دارد براي تكميل بودن مبحث به طور كامل مورد بررسي قرار مي گيرند.

## ٦-١- بيان مقاله در مورد سيستم RNS

فرض كنيد مجموعه اعداد دوبدو نسبت به هم اولي به شكل زير داشته باشيم.

$$B=\{p_1,p_2,p_3,\dots p_n\}$$

B پايه سيستم عددي باقيمانده ناميده مي شود. در اين RNS تمام اعداد x در كلاس مانده $z_p$ كه P برابر است با $\Pi P_i$ نمايشي منحصر به فرد به كمك مانده هاي خود به شكل زير دارند:

$(X_1,X_2,X_3,\dots,X_n)$ كه $X_i=x \bmod p_i$ با كمك قضيه باقيمانده هاي چيني مي توان x را به شكل زير بازيابي كرد.

$$x=[\sum X_i s_i s_i^{-1}]_p \qquad (١-٦)$$

كه $s_i=\frac{P}{p_i}$ و $s_i^{-1}$ نمايانگر معكوس ضربي $s_i \bmod p_i$ است. براكت ها نشان دهنده عمليات كاهش پيمانه اي است. با كمك عمليات تبديل سيستم عددي (Base extension) مي توان نمايش عددي كه در سيستم $(p_1,p_2,\dots,p_n)$ نمايش داده شده را در سيستم بزرگتري مثل $(p_1,p_2,\dots,p_n,p_{n+1},\dots,p_m)$ يافت. اين كار را مي توان با عملياتي تقريبي كه در اكثر اوقات جواب صحيح مي دهد (موارد خطا را به سادگي مي توان تشخيص داده و تصحيح كرد) انجام داد. ابتدا $\omega$ را به صورت زير تعريف مي‌كنيم.

$$\omega = \frac{x}{p} = \frac{\sum_{i=1}^{n} x_i s_i s_i^{-1}}{p} = \frac{\sum_{i=1}^{n} x_i \frac{p}{p_i} s_i^{-1}}{p} = \sum x_i \frac{s_i^{-1}}{p_i}$$

حال رابطه (۶-۱) را بصورت زیر بازنویسي مي كنيم.

$$x = \sum_{i=1}^{n} X_i s_i s_i^{-1} - \omega_{int} P$$

تمام $x_j$ها كه در آنها $j=n+1,n+2,...,m$ مي باشد را مي توان بصورت زير محاسبه كرد.

$$x_j = \left[ \left[ \sum_{i=1}^{n} X_i s_i s_i^{-1} \right]_{p_j} - \left[ \omega_{int} P \right]_{p_j} \right]_{p_j}$$

در پياده سازي واقعي دقت محدود نمايش ضريب غير صحيح $\frac{s_i^{-1}}{p_i} > 1$ در هنگامي كه مقدار دقيق $\omega$ به مقادير صحيح نزديك مي شود ايجاد اشكال مي كند اين مسأله با بررسي تعدادي از بيت هاي نمايش باينري بخش اعشاري $\omega$ يعني $\omega_{frac}$ قابل پيش بيني است. ولي اين مسأله در موضوع اصلي مقاله ما تأثيري ندارد و همانطور كه خواهيم ديد محاسبه صحيح $\omega_{int}$ و يا خطادار آن (يكي كمتر از مقدار واقعي) اشكالي ايجاد نمي‌كند.

## ۶-۲- بيان مقاله از ضرب پيمانه اي بدون تقسيم (روش مونتگمري):

روش كاهش پيشنهادي مونتگمري بر اساس دو هم ارزي زير است:

$$\begin{cases} x - (x \bmod R)(D^{-1} \bmod R)D \equiv 0 (\bmod R) \\ x - (x \bmod R)(D^{-1} \bmod R)D \equiv x (\bmod D) \end{cases}$$

از این دو هم ارزي براي حل مسأله $x \bmod D$ مي توان استفاده کرد. مسأله اصلي انتخاب R به گونه اي است که $x \bmod R$ به آساني قابل محاسبه باشد. در سیستم هاي عددي وزن دار R نوعاً توانی صحیح از radix (پایه) r مي باشد. در زیر نشان مي دهیم که دو هم ارزي بیان شده صحیح مي باشند.

رابطه صحیح زیر را در نظر بگیرید:

$$x - x \bmod R = tR = 0 \pmod R$$

D را به گونه اي فرض کنید که $\gcd(D,R)=1$ باشد. در نتیجه معکوس ضربي D به پیمانه R موجود است. از مطالب بیان شده مي توان نتیجه گرفت که روابط هم ارزي صحیح است.

معمولاً $D^{-1} \bmod R$ به عنوان ثابت از قبل محاسبه شده اي موجود است. البته حاصل بکارگیري هم ارزیهاي بالا $x \bmod D$ نبوده و $xR^{-1} \bmod D$ مي باشد که مي توان این مسأله را نیز با همین عملیات ضربي (ضرب در $R^2 \bmod D$) برطرف کرد.

حال روش پیشنهادي مقاله که در آن خطاي ناشي از تغییر RNS بي اثر مي شود را بیان مي کنیم.

| RNS 1 | | RNS 2 |
|---|---|---|
| 1- $P_1 = \prod_{\varepsilon=1}^{n} P_i$ | | $P_2 = \prod_{\varepsilon=1}^{n} P_i$ |
| 2- $x = ab$ | | $x = ab$ |
| | | 3- $x = x \bmod P_1$ |
| | | 4- $t = xD^{-1}$ |
| 5- $t$ | $\Rightarrow$ | $t^*$ |
| 6- | | $u = t^* D$ |
| 7- | | $v = x - u$ |
| 8- | | $y = vP_1^{-1} + 2D$ |
| 9- $y^*$ | $\Leftarrow$ | $y$ |

10- $T = \lfloor 4W_{frac} \rfloor$

11- $z^* = y^* - TD$ $\qquad z = y - TD$

**شکل ٦-١: الگوریتم طرح چهارم**

الگوریتم شکل (٦-١) با دو عدد a,b شروع شده و حاصل

$$z = \frac{ab}{P_1} \bmod D$$

را محاسبه مي كند. براي صحت كار بايد محدوديتهاي زير اعمال شود:

$$\begin{cases} D + \Delta < P_1 < D + \frac{D}{3} \text{ with } \Delta < \frac{D}{6} \\ 4D \le P_2 \le (4 + \varepsilon_{\frac{P_2}{D}})D \text{ with } \varepsilon_{\frac{P_2}{D}} < \frac{1}{12} \\ a,b < D + \Delta < D + \frac{D}{6} \end{cases}$$

اين مفروضات محدوديت بنيادي در صورت بزرگ بودن D روي الگوريتم اعمال نمي كند.

## ٦-٣- بررسي صحت الگوريتم:

1) $x = (ab)_{RNS1}$

حاصلضرب ab را مي توان با كمك $P_2$ و $P_1$ بيان كرد زيرا:

$$x = ab < D^2 + \frac{2D}{6} + \frac{D^2}{36} < P_1 P_2$$

2) $t = (x \bmod P_1) D^{-1} \bmod P_1$

براساس روش مونتگمري اولين قدم محاسبه x mod $P_1$ است. در سيستم RNS 1 اين مسئله واضح است كه $x \bmod P_1 = x$ زيرا $x \bmod P_1 < P_1$ مي توان آنرا در RNS 1 نمايش داد.

ارقام متناظر در RNS2 نیز بسادگي محاسبه مي شود. مباحث بالا بطور متناظر در مورد $t = (x \bmod P_1)D^{-1} \bmod P_1 < P_1$ صدق مي كند.

3) $RNS1 \rightarrow RNS\ 2$

قدم بعدي يعني ضرب در D منجر به نتيجه اي بزرگتر از $P_1$ مي شود. در نتيجه در ابتدا عمليات تبديل RNS $(RNS1 \rightarrow RNS2)$ انجام مي شود. همانطور كه قبلاً گفتيم تقريبي از مقدار واقعي با مقدار $\omega_{int}$ بصورت زير محاسبه مي شود.

$$W_{int} = \left\lfloor \sum_{i=1}^{n} x_i \omega_i \right\rfloor$$

كه $\omega_i = \omega_i - \delta_i$ و $\delta_i << 1$ نمايش خطاي بين مقدار تقريبي و واقعي $\omega = \frac{S_i^{-1}}{P_i}$ است.

تبديل $RNS1 \rightarrow RNS2$ مقدار $t^*$ نتيجه مي دهد كه :

$$t^* = \begin{cases} t & \text{if } \omega^*_{int} = \omega_{int} \\ t + P_1 & \text{if } \omega^*_{int} = \omega_{int} - 1 \end{cases}$$

(تقريب صحيح)

(تقريب با خطا)

محدوده مقادير $t^*$

$$t^* < \begin{cases} P_1 & \omega^*_{int} = \omega_{int} \\ P_1 + \Delta & \omega^*_{int} = \omega_{int} - 1 \end{cases}$$

حالت اول كه مشكل ندارد. تقريب $\omega_{int}$ اگر x خيلي به صفر نزديك باشد يا كمي از $P_1$ بزرگتر باشد دچار اشكال مي شود.

4) $u = ((x \bmod P_1)D^{-1} \bmod P_1)D$

بر اساس روش مونتگمري قدم بعدي محاسبه $u = t^*D$ است.

5) $y = (x - ((x \bmod P_1)D^{-1} \bmod P_1)D)P_1^{-1} + 2D$

میدانیم $x-u=0 \bmod P_1$. پس بدون باقي مانده بر $P_1$ تقسیم مي شود. تصحیح به اندازه 2D مورد نیاز است تا جلوي مقدار منفي براي y را بگیرید.

$$y=(x-u)P_1^{-1}+2D$$

در نتیجه :

$$\frac{D}{2}<y<3D+\frac{D}{3}$$

صحت حدود بالا بصورت زیر مشخص مي شود.

$$0\le x<(D+\Delta)^2<P_1^2$$

$$0\le t^*<P_1+\Delta$$

$$-(\underbrace{D+\frac{D}{3}}_{>P_1}+\underbrace{\frac{D}{6}}_{>\Delta})D<x-t^*D<P_1^2$$

$$-\frac{3}{2}D<(x-t^*D)P_1^{-1}<D+\frac{D}{3}$$

$$\frac{D}{2}<y<3D+\frac{D}{3}$$

$$6)\text{RNS2}\rightarrow\text{RNS1}$$

با اینکه این حقیقت وجود دارد که ممکن است y در محدوده RNS1 نگنجد آنرا به RNS1 برده و تصحیح لازم که کاهش ضریب خاصي از D مي باشد. در مرحله بعد انجام مي شود.

$$y^*=\begin{cases}y & \text{صحیح}\\ y+TD & \text{نیازمند تصحیح}\end{cases}$$

براي تبدیل RNS، $\omega^*$ محاسبه مي شود که در مرحله بعد مورد نیاز است.

$$7)\ z=y-TD, z^*=y^*-TD$$

حال باید بررسي شود که آیا y در RNS1 قابل نمایش است یا خیر. بسیار محتمل است که $y\ge D+\Delta$

باشد ولي فقط مقادير كمتر از $D+\Delta$ را مي توان براي ضرب پيمانه اي بعدي بكار برد در نتيجه بايد ضريب مناسبي، T، از D را از y كم كرد.

$$z = y - TD < D + \Delta \quad \text{in} \quad RNS1$$

$$z = y^{*} - TD < D + \Delta \quad \text{in} \quad RNS2$$

$$8) T = \lfloor 4\omega_{frac} \rfloor$$

محتمل است كه

$$\omega^{*}_{frac} > \omega_{frac} - \varepsilon_{w}$$

$\varepsilon_{w}$ ماكزيمم مقدار خط است.

مقدار $\left(\frac{P_2}{D}\right)^{*}$ محاسبه شده نيز خطايي به اندازه $\varepsilon_{\frac{P_2}{P}}$ دارد

$$\left(\frac{P_2}{D}\right)^{*} > \frac{P_2}{D} - \varepsilon_{\frac{P_2}{D}}$$

خطاي T كه مقدار آن با محاسبه $\left\lfloor W_{frac} \frac{P_2}{D} \right\rfloor$ بدست مي آيد.

$$\varepsilon_{T} = \left(\omega_{frac} \frac{P_2}{D}\right) - \left(\omega^{*}_{frac}\left(\frac{P_2}{D}\right)^{*}\right)$$

$$< \varepsilon_{W} \frac{P_2}{D} + \omega_{frac}\varepsilon_{\frac{P_2}{D}} - \varepsilon_{w}\varepsilon_{\frac{P_2}{D}}$$

$$< 5\varepsilon_{W} + \varepsilon_{\frac{P_2}{D}}$$

مقدار $\varepsilon_{T}$ براي مقاديري از $\omega_{frac} \frac{P_2}{D}$ كه كمي بزرگتر از يك عدد صحيح هستند مهم مي شود و تقريب ممكن است منجر به مقدار غلط T شود. اين

مسئله فقط براي y هايي كه كمي از ضريب صحيحي از D بزرگتر هستند اتفاق مي افتد.

اگر

$$\varepsilon_T < \frac{\Delta}{D} < \frac{1}{6}$$

پس عدم اطمينان از مقدار T فقط براي y در محدوده هاي زير بدست مي آيد.

$$[kD, kD+\Delta] \text{with} \quad k \in \{1,2,3\}$$

مقدار غلط T در نتيجه منجر به z در محدوده $[D,D+\Delta]$ مي شود ولي مقادير در اين محدوده مي توانند بكار روند.

يك توزيع مساوي از خطاي كل $\varepsilon_T$ درميان اجزا نتيجه مي دهد:

$$\varepsilon_w < \frac{1}{60} \quad , \quad \varepsilon_{\frac{P_2}{D}} < \frac{1}{12}$$

در نتيجه

$$\varepsilon_T < 5\varepsilon_w + \varepsilon_{\frac{P_2}{D}} < \frac{1}{6}$$

در نهايت $\frac{P_2}{D}$ كمي بزرگتر از 4 انتخاب مي شود:

$$\frac{P_2}{D} = 4 + \varepsilon_{\frac{P_2}{D}}$$

و از $\left(\frac{P_2}{D}\right)^* = 4$ استفاده مي شود.

### ٦-٤- روش تبديل RNS

در اين مقاله هم از همان روشي كه در طرح سوم استفاده مي شود.

## ٦-٥- پياده سازي سخت افزاري

اصول اصلي پياده سازي طرح چهارم كاملاً مشابه با طرح سوم مي باشد. در واقع در اينجا هم عمليات‌ها به دو دسته اصلي تبديل RNS و عمليات مونتگمري قابل تقسيم هستند. و همچنين از تمام عناصر بكار رفته در طرح سوم (بجز RESCONVS كه بصورت تنها اصلاح شده و در شكل ديگر مورد استفاده قرار مي گيرد) مورد استفاده قرار گرفته اند و يك عنصر مبتني بر ROM به اين مجموعه اضافه مي شود. تفاوت اصلي اين الگوريتم با الگوريتم سوم در انتهاي آن است. در واقع جهت محاسبه ضريب خطاي T از مقادير بدست آمده از تبديل RNS استفاده مي شود كه نتيجه را از RNS كمكي به RNS اوليه منتقل مي كند. در واقع فرض مي شود كه نتيجه نهايي را در همان RNS كه شروع كرديم بيان كنيم. ولي اگر بخاطر داشته باشيد در طرح سوم نتيجه نهايي در همان RNS كمكي باقي ماند و توضيح داريم كه اين مسئله براي ما مشكل ايجاد نمي كند. پس براي مقايسه اين دو طرح فرض را بر همان روش سوم قرار مي دهيم. پس در واقع ديگر به تبديل RNS دوم براي بازگشتن به RNS اوليه بطور كامل احتياج نيست و فقط بخشي از آن كه ما را دريافتن T ياري مي كند مورد استفاده قرار گرفته.

يك نكته ديگر عدم استفاده اين طرح از روش ابتكاري طرح سوم (تبديل RNS اصلاح شده) است كه

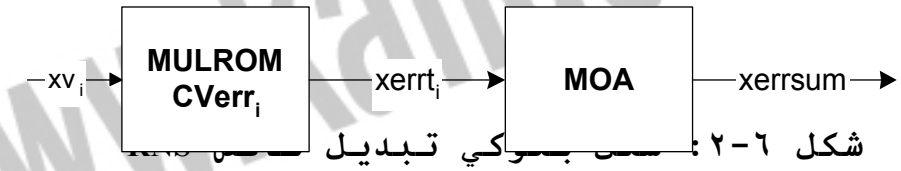
مي توانست بهبود بخش سرعت عمليات باشد. همانگونه كه گفتيم در اينجا از همان روش تبديل RNS طرح سوم استفاده شده پس پياده سازي هم دقيقاً مثل تبديل RNS طرح سوم است. دو بخش ديگر يعني تبديل RNS ناقص (براي محاسبه T) و اصل طرح چهارم در زير مورد بررسي قرار مي گيرند.

### ۶-۵-۱- تبديل ناقصRNS

در اين بخش در واقع فقط يك عمليات ضرب داخلي بايد صورت بگيرد يعني عملياتي بشكل

$$xerr = \sum_{i=1}^{n} x_{v_i}.Cverr_i$$

پس در واقع به دو عنصر يكي ROM جهت ضرب در عدد ثابت Cverr و يك جمع كننده چند عملوندي احتياج داريم. تعريف Cverr مشابه تعريف Cωerr در بخش ۵-۲-۱ مي باشد.

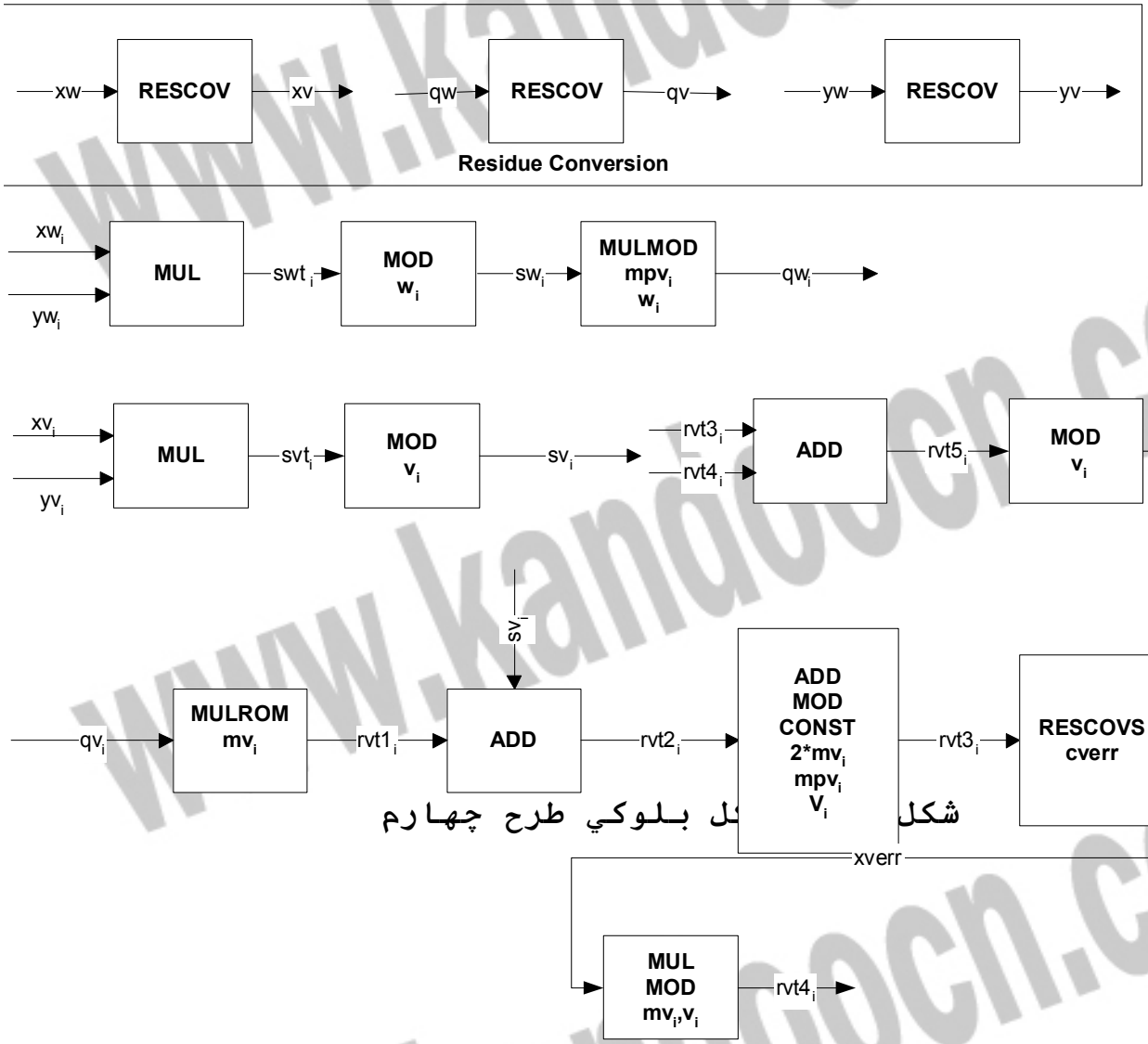



**شكل ۶-۲: [illegible] بلوكي تبديل**

### ۶-۵-۲- پياده سازي بخش اصلي طرح چهارم (الگوريتم مونتگمري):

حال پياده سازي خود طرح چهارم را بررسي مي كنيم. اصول كلي كار با نگاه به بلوك ديگرام طرح كه در شكل (۶-۲) آمده مشخص مي شود. كل عمليات بسيار شبيه طرح سوم است با مقايسه آنها چند تفاوت مشاهده مي شود كه در زير بيان مي شود يكي وجود يك تبديل استاندارد RNS براي تبديل $q_w$ به $q_v$ مي باشد. درواقع در اينجا از

شكل اصلي RNS استفاده شده و از RNS اصلاح شده استفاده نكرديم. ثانياً در اين طرح نياز به يك ضرب جمع كننده پيمانه‌اي با اعداد ثابت داريم كه آنرا نيز پياده كرده ايم و در نهايت استفاده از تبديل RNS ناقص براي محاسبه خطا و كم كردن آن از نتيجه نيز در بلوك دياگرام ديده مي شود.




شكل بلوكي طرح چهارم

## ٦-٦- محاسبه پيچيدگي تأخير و مساحت طرح چهارم

همانگونه كه گفته شد در طرح چهارم از همان عناصر بكار رفته در طرح سوم استفاده شده و تنها دو عنصر تبديل RNS ناقص و يك ضرب/ جمع كننده پيمانه اي با اعداد ثابت اضافه شده. ضرب/جمع كننده پيمانه اي با اعداد ثابت با ROM پياده شده و از همان فرمولها تبعيت مي كند. پس در ادامه تأخير و مساحت بخش تبديل RNS ناقص را محاسبه كرده در ادامه خود طرح را مورد بررسي قرار مي دهيم.

لازم به ذكر است كه در تمام فرمولهاي لگاريتم، پايه ۲ مي باشد و همچنين پارامتر Size تعداد مانده ها مي‌باشد و تعداد بيت $\omega_i$ يعني $\lceil \log_2 \omega_i \rceil$ و $v_i$ يعني $\lceil \log_2 v_i \rceil$ برابر n فرض شده است. در واقع فرض كرديم كه تعداد بيت $\omega_i$ و $v_i$ برابر باشد. البته احتمال تفاوت تعداد بيت وجود دارد ولي ما مجبور به اين فرض براي ساده سازي هستيم.

### ٦-٦-١- محاسبه تأخير و مساحت تبديل ناقص RNS

۱- تاخير MULROM

عمليات MULROM ضرب در عدد ثابت توسط ROM است. تاخير ROM در بخش ٥-٣-١ بيان شده و با توجه به اينكه تعداد بيت هاي ورودي به اين ROM حداكثر $\lceil \log \omega_i \rceil$ مي باشد. پس خواهيم داشت:

$$MULROM_{delay} = 2\lceil \log \omega_i \rceil = 2n$$

۲- تاخیر MOA

با توجه به رابطه تاخیر MOA در بخش ۵-۳-۲ نیازمند تعداد اعداد ورودي و حداکثر بیت هر عدد هستیم. تعداد اعداد ورودي size مي باشد و حداکثر تعداد بیت با توجه به مرحله قبل $\lceil \log(v_i.Cverr_i) \rceil$ است پس مي توان نوشت:

$$MOA_{delay} = \log(size) + \log(\lceil \log(v_i.Cverr_i) \rceil)$$
$$= \log(size) + \log(16+n)$$

پس در کل تاخیر تبدیل ناقص RNS بصورت زیر است:

$$RESCOVS_{delay}=2n+\log(size) + \log(16+n)$$

**- مساحت تبدیل ناقص RNS**

۱) مساحت MULROM

ورودي این واحد حداکثر مقدار $v_i$ وخروجي آن $v_i.Cverr_i$ خواهد بود و به تعداد پیمانه ها از این واحد موجود است. پس مساحت بصورت زیر است:

$$size.\left(2^{\lceil \log(v_i) \rceil}.\lceil \log(Cverr_i.v_i) \rceil\right)$$
$$= size.2^n.(16+n)$$

۲- مساحت MOA

تعداد ورودي هاي این واحد، تعداد پیمانه ها و حداکثر مقدار آنها $Cverr_i.v_i$ مي‌باشد. پس مساحت بصورت زیر است:

$$size.\lceil \log(Cverr_i.v_i) \rceil + \lceil \log(Cverr_i.v_i) \rceil + \log\lceil \log(Cverr_i.v_i) \rceil$$
$$=(16+n).(size+\log(16+n))$$

### ۶-۶-۲- محاسبه تأخیر و مساحت در طرح چهارم

حال با توجه به روابط بالا و روابط بدست آمده در طرح سوم تأخیر و مساحت طرح چهارم را محاسبه مي كنيم. لازم به ذكر است كه در روابط پايين size تعداد پيمانه، RESCOVS تبديل RNS استاندارد (با روابط طرح سوم) وRESCOVS تبديل ناقص RNS كه اخيراً بررسي شده مي باشد.

**- محاسبه تأخيردرطرح چهارم**

اگر بلوك دياگرام طرح چهارم را مورد بررسي قرار دهيم متوجه مي شويم كه در خروجي يك جمع كننده با دو ورودي داريم كه ورودي پايين (rvt4) وابسته به ورودي بالا (rvt3) مي باشد پس در نتيجه تأخير rvt4 در اين قسمت تعيين كننده است. از طرف ديگر در مسير ايجاد rvt4 جمع كننده اي وجود دارد كه دو ورودي دارد. يكي مسير بالايي ($rvt1_i$) و ديگري مسير بالاي ($sv_i$).

ابتدا تاخير هر يك از اين دو مسير را محاسبه مي كنيم:

**۱- تأخير مسير منتهي به $rvt1_i$ :**

الف- تاخير $MUL\ (x_{\omega_i}, y_{\omega_i})$

حداكثر مقدار هر ورودي برابر $\omega_i$ است و با توجه به فرمول ضرب كننده در بخش ۵-۳-۲ داريم:

$$MUL_{delay}=3\lceil \log\lceil \log \omega_i \rceil \rceil = 3\lceil \log n \rceil$$

ب- تاخير $MOD(\omega_i)$

يك ROM با حداكثر ورودي $\omega_i^2$ مي باشد پس داريم:

$$MOD_{delay}=2\lceil \log \omega_i^2 \rceil = 4n$$

ج- تاخير $MULMOD(mp\omega_i, \omega_i)$

يك ROM با حداكثر ورودي $\omega_i$ داريم. پس:

$$MULMOD_{delay}=2\lceil \log \omega_i \rceil = 2n$$

د- تاخير تبديل RNS استاندارد $q_\omega$ به $q_v$

همانطور كه گفتيم در اين طرح از همان روش تبديل RNS استاندارد بكار رفته در طرح سوم استفاده شده و در نتيجه تاخير اين بخش همان $RESCOV_{delay}$ مي باشد.

هـ- تاخير $MULROM(mv_i)$

در اينجا هم ROM با حداكثر ورودي $v_i$ داريم. پس:

$$MULROM_{delay}=2n$$

پس براي $rvt1_i$ داريم:

$$rvt1_{delay} = 3\lceil \log n \rceil + 8n + RESCOV_{delay}$$

## ۲- تاخير مسير منتهاي به $sv_i$

الف- تاخير دو تبديل RNS استاندارد $x_\omega$ به $x_v$ و $y_\omega$ به $y_v$

اين دو تبديل به طور موازي با هم اعمال مي شود و با توجه به اينكه تعداد عملوندهاي هر كدام size و حداكثر مقدار ورودي $x_{\omega_i}$ به $y_{\omega_i}$، $\omega_i$ مي باشد تاخير آن دو برابر است پس كافيست تاخير يكي از آنها را در نظر بگيريم و

همانطور که قبلاً دیده ایم تاخیر یك تبدیل RNS استاندارد $RESCOV_{delay}$ می باشد.

ب- تاخیر $MUL\ (x_{v_i}, y_{v_i})$

حداکثر مقدار هر ورودي این ضرب کننده $v_i$ می باشد و در نتیجه تاخیر آن:

$$MUL_{delay}=3\lceil \log\lceil \log v_i \rceil \rceil = 3\lceil \log n \rceil$$

ج- تاخیر $MOD\ (v_i)$

در اینجا یك ROM با حداکثر مقدار ورودي $v_i^2$ داریم. پس:

$$MOD_{delay}=2\lceil \log v_i^2 \rceil = 4n$$

پس براي $sv_i$ داریم:

$$sv_{i_{delay}} = 4n + 3\lceil \log n \rceil + RESCOV_{delay}$$

با توجه به مقادیر بدست آمده کاملاً مشخص است که تاخیر $rvt1_i$ به اندازه $4n$ از $sv_i$ بیشتر است.

حال به محاسبه تاخیر $rvt4_i$ می پردازیم.

**تاخیر $rvt4_i$**

۱- تأخیر $ADD(rvt1_i, sv_i)$

براي محاسبه تاخیر این تفریق باید ببینیم هر کدام از ورودي هاي آن در بدترین حالت چه مقداري دارند. $sv_i$ خروجي یك $MOD(v_i)$ می باشد و حداکثر مقدار آن $v_i$ است ولي $rvt1_i$ خروجي یك $MULROM(mv_i)$ است که ورودي آن یعني $qv_i$ حداکثر می تواند مقدار $v_i$ داشته باشد پس $rvt1_i$ حداکثر $v_i^2$ می باشد. پس $rvt1_i$ در تاخیر موثرتر است و با توجه به رابطه بخش ۵-۳-۲ براي جمع کننده می توان نوشت:

$$ADD_{delay}=4\log(\lceil \log v_i^2 \rceil) = 4\log(2n)$$

۲- تأخیر ADD MOD CONST $(2mv_i, mpv_i, v_i)$

این واحد که یک عملیات با ROM انجام می دهد عدد ورودي را در $2mv_i$ ضرب کرده با $mpv_i$ جمع کرده و حاصل را در پیمانه $v_i$ محاسبه می کند حداکثر مقدار ورودي این واحد (با توجه به توضیحات ADD بالا) $v_i^2$ می باشد پس در مورد تاخیر این واحد داریم:

$$ADD\ MOD\ CONST_{delay} = 2\lceil \log v_i^2 \rceil = 4n$$

۳- تأخیر تبدیل ناقص RNS بر روي $rvt3_i$

همانطور که قبلاً بررسي شده تاخیر این واحد $RESCOVS_{delay}$ محاسبه شده در طرح چهارم می باشد.

٤- تاخیر $MULMOD(mv_i, v_i)$

یک عملیات با ROM با حداکثر ورودي $size.v_i.Cverr_i$ در واقع ورودي این واحد خروجي واحد تبدیل ناقص RNS می باشد که با توجه به شکل بلوکي تبدیل ناقص RNS خروجي این واحد، خروجي یک MOA است که این MOA به تعداد size ورودي که هر کدام حداکثر مقدار $v_i.Cverr_i$ را دارد را با هم جمع می کند پس مقدار ورودي این واحد حداکثر $size.v_i.Cverr_i$ خواهد بود و تاخیر این واحد:

$$MULMOD_{delay} = 2\lceil \log(size.v_i.Cverr_i) \rceil$$

و با توجه به مباحث گذشته در مورد $Cverr_i$ می توان نوشت:

$$MULMOD_{delay} = 2\lceil \log(size.2^{(n+16)}) \rceil$$

و در مورد تاخیر $rvt4_i$ داریم:

$$rvt4_{i_{delay}} = 4\log(2n) + 4n + 2\lceil \log(size.2^{(n+16)}) \rceil + RESCOVS_{delay}$$

حال باید تاخیر مسیر مشترك نهایي كه شامل یك ADD و یك MOD مي‌باشد حساب كنیم.

**تاخیر مسیر مشترك:**

۱- تاخیر $ADD(rvt3_i, rvt4_i)$

براي محاسبه تاخیر این ADD نیز باید ببینیم هر كدام از دو ورودي حداكثر چه مقداري دارند. $rvt3_i$ خروجي ADD MOD CONST مي باشد و در نتیجه حداكثر مقدار $v_i$ را دارد. $rvt4_i$ نیز خروجي یك MULMOD است و در نتیجه آنهم حداكثر مقدار $v_i$ دارد پس حداكثر مقدار هر كدام از ورودي ها $v_i$ است. پس براي تاخیر این واحد داریم:

$$ADD_{delay}=4\log(\lceil \log v_i \rceil)=4\log n$$

۲- تاخیر $MOD(v_i)$

این واحد در ورودي خود خروجي ADD مرحله قبل را مي بیند كه حداكثر مي‌تواند مقدار $v_i$ داشته باشد. پس:

$$MOD(v_i)_{delay}=2\lceil \log v_i \rceil=2n$$

پس تاخیر كل طرح چهارم برابر است با:

$$P4_{delay}=rvtl_{i_{delay}}+rvt4_{i_{delay}}+4\log+2n$$

$$=RESCOV_{elay}+RESCOVS_{delay}+3\lceil \log n \rceil+14n+4+8\log n$$

$$+2\lceil \log(size.2^{(n+16)}) \rceil$$

**- محاسبه مساحت طرح چهارم**

۱- مساحت MULها

با توجه به رابطه بخش ۵-۳-۲ براي مساحت MUL و اینكه ورودي هر كدام از واحدهاي MUL به ترتیب حداكثر $\omega_i$ و $v_i$ خواهد بود و از هر كدام به

تعداد پیمانه‌ها داریم براي مساحت هر کدام به ترتیب مقادیر زیر بدست مي آید:

a) $\lceil \log \omega_i \rceil^2 .size = n^2 .size$

b) $\lceil \log v_i \rceil^2 .size = n^2 .size$

۲- مساحت MOD بعد از هر MUL

این واحد با ROM پیاده شده. حداکثر ورودي و خروجي هر کدام به ترتیب $v_i, v_i^2$ و $\omega_i, \omega_i^2$ مي باشد و از هر کدام به تعداد پیمانه ها داریم پس براي مساحت هر کدام به ترتیب مقادیر زیر بدست مي آید:

a) $2^{\lceil \log \omega_i^2 \rceil} . \lceil \log \omega_i \rceil .size = 2^{2n} .n.size$

b) $2^{\lceil \log v_i^2 \rceil} . \lceil \log v_i \rceil .size = 2^{2n} .n.size$

۳- مساحت $MULMOD(mp\omega_i, \omega_i)$

این واحد نیز با ROM پیاده مي شود و به تعداد پیمانه ها تکرار شده است. حداکثر مقدار ورودي و خروجي آن حداکثر $\omega_i$ مي باشد. پس مساحت آن

$2^{\lceil \log \omega_i \rceil} . \lceil \log \omega_i \rceil .size = 2^n .n.size$

٤- مساحت مربوط به RESCOV

این واحد در ۳ جا تکرار شده و در نتیجه مساحت اشغال شده توسط آن

$3.RESCOV_{Area}$

لازم بذکر است که $RESCOV_{Area}$ برابر با مقدار بدست آمده براي تبدیل RNS استاندارد طرح سوم مي باشد.

۵- مساحت $MULROM(mv_i)$

این واحد به تعداد پیمانه ها موجود است. پیاده سازي آن با ROM صورت گرفته و در آن حداکثر مقدار ورودي و خروجي $v_i$ و $v_i^2$ مي باشد پس براي مساحت این قسمت داریم:

$$size.2^{\lceil \log v_i \rceil}.(\lceil \log v_i^2 \rceil) = size.2^n.2n = size.2^{n+1}.n$$

۶- مساحت ADD

همانطور که در هنگام محاسبه تاخیر این واحد دیدیم حداکثر مقدار ورودي آن $v_i^2$ است پس در مورد مساحت آن با توجه به تکرار شدن به تعداد پیمانه ها داریم:

$$size.10\lceil \log v_i^2 \rceil = size.10.2n = size.20n$$

۷- مساحت ADD MOD CONST

با توجه به مباحث تاخیر این واحد مي دانیم که ورودي و خروجي آن حداکثر مقادیر $v_i, v_i^2$ را دارد و به تعداد پیمانه ها از این واحد داریم در نتیجه براي مساحت این بخش داریم:

$$size.2^{\lceil \log v_i^2 \rceil}.(\lceil \log v_i \rceil) = size.2^{2n}.n$$

۸- مساحت RESCOVS

مساحت این واحد یعني $RESCOVS_{Area}$ در بخش محاسبه مساحت تبدیل ناقص RNS صورت گرفته است.

۹- مساحت $MULMOD(mv_i, v_i)$

این واحد هم به تعداد size موجود است، با ROM پیاده شده و ورودي و خروجي آن به ترتیب حداکثر مقادیر $size.v_i.Cverr_i$ و $v_i$ را دارد و مساحت آن بشکل زیر خواهد بود:

$$size.2^{\lceil \log(size.v_i.Cverr_i) \rceil}.\lceil \log v_i \rceil = size.2^{\lceil \log(size.2^{n+16}) \rceil}.n$$

۱۰- مساحت $ADD(rvt3_i, rvt4_i)$

همانطور که در قسمت محاسبه تاخیر دیدیم هر کدام از ورودي ها حداکثر مقدار $v_i$ دارند پس با توجه به وجود size عدد از این بلوك براي مساحت داریم:

$$size.10.\lceil \log v_i \rceil = 10.n.size$$

۱۱- مساحت $MOD(v_i)$ در خروجي نهایي

این واحد هم به تعداد size در طرح تکرار شده و با توجه به اینکه توسط یك ROM پیاده شده و ورودي و خروجي آن حداکثر مقدار $v_i$ را دارد. مساحت به شکل زیر خواهد بود:

$$size.2^{\lceil \log v_i \rceil}.\lceil \log v_i \rceil = size.2^n.n$$

و مساحت طرح چهارم برابر است با:

$$P4_{Area} = 2n^2.size + 2^{2n+1}.n.size + 2^n.size.n + 3RESCOV_{Area}$$
$$+ size.2^{n+1}.n + size.20n + size.2^{2n}.n + RESCOVS_{Area}$$
$$+ size.2^{\lceil \log(size.2^{n+16}) \rceil}.n + 10.n.size + size.2^n.n$$

# فصل چهارم

# طرح دوم

## ٦-٥- پياده سازي سخت افزاري

اصول اصلي پياده سازي طرح چهارم كاملاً مشابه با طرح سوم مي باشد. در واقع در اينجا هم عمليات‌ها به دو دسته اصلي تبديل RNS و عمليات مونتگمري قابل تقسيم هستند. و همچنين از تمام عناصر بكار رفته در طرح سوم (بجز RESCONVS كه بصورت تنها اصلاح شده و در شكل ديگر مورد استفاده قرار مي گيرد) مورد استفاده قرار گرفته اند و يك عنصر مبتني بر ROM به اين مجموعه اضافه مي شود. تفاوت اصلي اين الگوريتم با الگوريتم سوم در انتهاي آن است. در واقع جهت محاسبه ضريب خطاي T از مقادير بدست آمده از تبديل RNS استفاده مي شود كه نتيجه را از RNS كمكي به RNS اوليه منتقل مي كند. در واقع فرض مي شود كه نتيجه نهايي را در همان RNS كه شروع كرديم بيان كنيم. ولي اگر بخاطر داشته باشيد در طرح سوم نتيجه نهايي در همان RNS كمكي باقي ماند و توضيح داريم كه اين مسئله براي ما مشكل ايجاد نمي كند. پس براي مقايسه اين دو طرح فرض را بر همان روش سوم قرار مي دهيم. پس در واقع ديگر به تبديل RNS دوم براي بازگشتن به RNS اوليه بطور كامل احتياج نيست و فقط بخشي از آن كه ما را دريافتن T ياري مي كند مورد استفاده قرار گرفته.

يك نكته ديگر عدم استفاده اين طرح از روش ابتكاري طرح سوم (تبديل RNS اصلاح شده) است كه

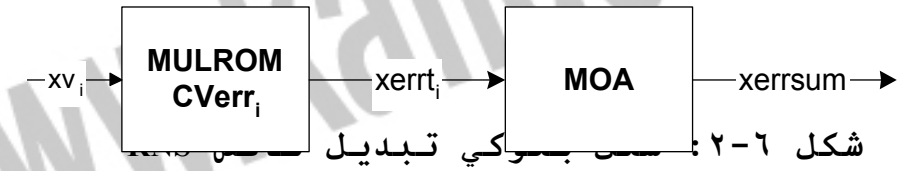
مي توانست بهبود بخش سرعت عمليات باشد. همانگونه كه گفتيم در اينجا از همان روش تبديل RNS طرح سوم استفاده شده پس پياده سازي هم دقيقاً مثل تبديل RNS طرح سوم است. دو بخش ديگر يعني تبديل RNS ناقص (براي محاسبه T) و اصل طرح چهارم در زير مورد بررسي قرار مي گيرند.

### ۶-۵-۱- تبديل ناقصRNS

در اين بخش در واقع فقط يك عمليات ضرب داخلي بايد صورت بگيرد يعني عملياتي بشكل

$$xerr = \sum_{i=1}^{n} x_{v_i}.Cverr_i$$

پس در واقع به دو عنصر يكي ROM جهت ضرب در عدد ثابت Cverr و يك جمع كننده چند عملوندي احتياج داريم. تعريف Cverr مشابه تعريف Cωerr در بخش ۵-۲-۱ مي باشد.

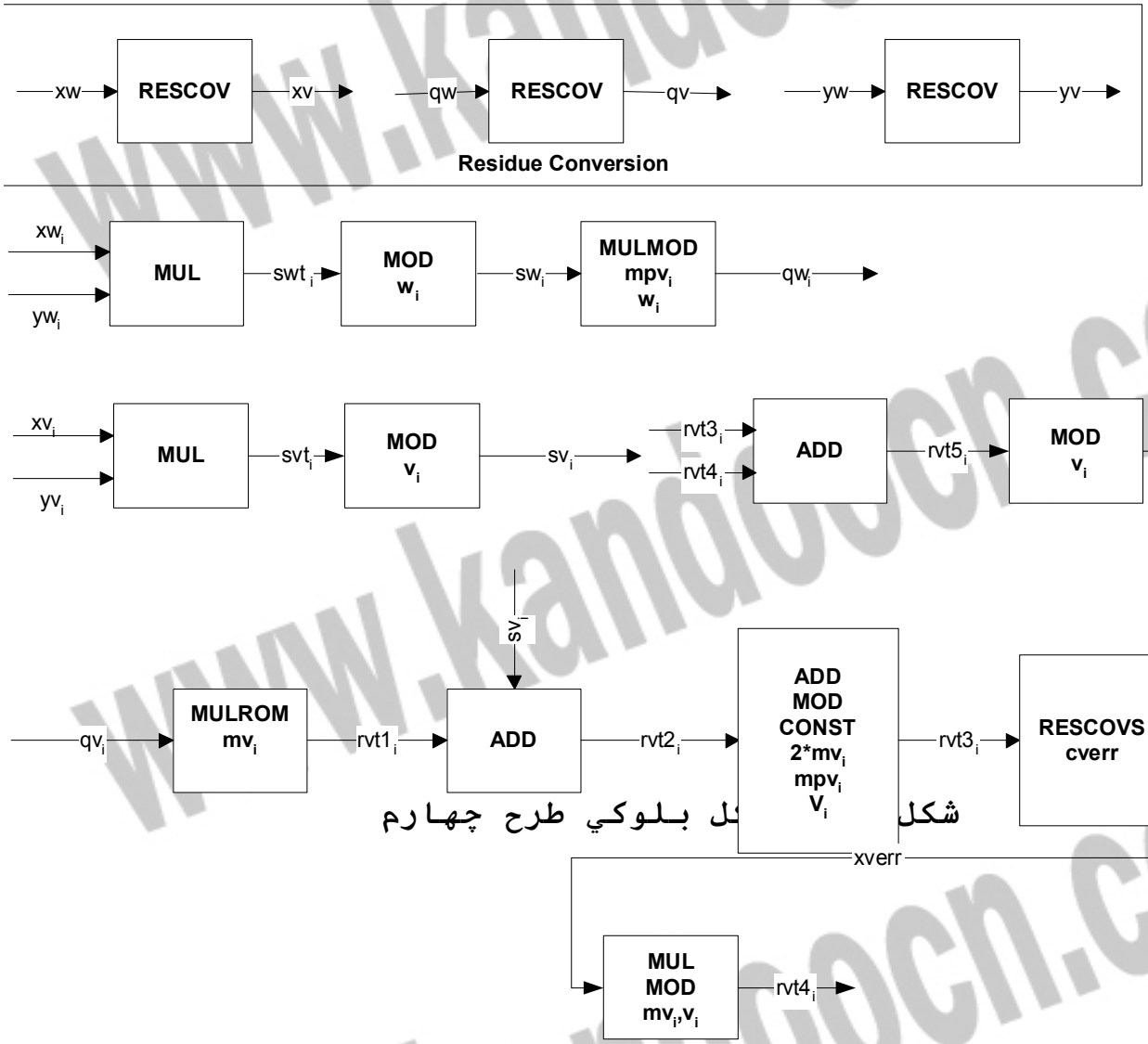



**شكل ۶-۲: [illegible] بلوكي تبديل**

### ۶-۵-۲- پياده سازي بخش اصلي طرح چهارم (الگوريتم مونتگمري):

حال پياده سازي خود طرح چهارم را بررسي مي كنيم. اصول كلي كار با نگاه به بلوك ديگرام طرح كه در شكل (۶-۲) آمده مشخص مي شود. كل عمليات بسيار شبيه طرح سوم است با مقايسه آنها چند تفاوت مشاهده مي شود كه در زير بيان مي شود يكي وجود يك تبديل استاندارد RNS براي تبديل $q_w$ به $q_v$ مي باشد. درواقع در اينجا از

شكل اصلي RNS استفاده شده و از RNS اصلاح شده استفاده نكرديم. ثانياً در اين طرح نياز به يك ضرب جمع كننده پيمانه‌اي با اعداد ثابت داريم كه آنرا نيز پياده كرده ايم و در نهايت استفاده از تبديل RNS ناقص براي محاسبه خطا و كم كردن آن از نتيجه نيز در بلوك دياگرام ديده مي شود.




شكل بلوكي طرح چهارم

## ٦-٦- محاسبه پيچيدگي تأخير و مساحت طرح چهارم

همانگونه كه گفته شد در طرح چهارم از همان عناصر بكار رفته در طرح سوم استفاده شده و تنها دو عنصر تبديل RNS ناقص و يك ضرب/ جمع كننده پيمانه اي با اعداد ثابت اضافه شده. ضرب/جمع كننده پيمانه اي با اعداد ثابت با ROM پياده شده و از همان فرمولها تبعيت مي كند. پس در ادامه تأخير و مساحت بخش تبديل RNS ناقص را محاسبه كرده در ادامه خود طرح را مورد بررسي قرار مي دهيم.

لازم به ذكر است كه در تمام فرمولهاي لگاريتم، پايه ۲ مي باشد و همچنين پارامتر Size تعداد مانده ها مي‌باشد و تعداد بيت $\omega_i$ يعني $\lceil \log_2 \omega_i \rceil$ و $v_i$ يعني $\lceil \log_2 v_i \rceil$ برابر n فرض شده است. در واقع فرض كرديم كه تعداد بيت $\omega_i$ و $v_i$ برابر باشد. البته احتمال تفاوت تعداد بيت وجود دارد ولي ما مجبور به اين فرض براي ساده سازي هستيم.

### ٦-٦-١- محاسبه تأخير و مساحت تبديل ناقص RNS

١- تاخير MULROM

عمليات MULROM ضرب در عدد ثابت توسط ROM است. تاخير ROM در بخش ٥-٣-١ بيان شده و با توجه به اينكه تعداد بيت هاي ورودي به اين ROM حداكثر $\lceil \log \omega_i \rceil$ مي باشد. پس خواهيم داشت:

$$MULROM_{delay} = 2\lceil \log \omega_i \rceil = 2n$$

۲- تاخیر MOA

با توجه به رابطه تاخیر MOA در بخش ۵-۳-۲ نیازمند تعداد اعداد ورودي و حداکثر بیت هر عدد هستیم. تعداد اعداد ورودي size مي باشد و حداکثر تعداد بیت با توجه به مرحله قبل $\lceil \log(v_i.Cverr_i) \rceil$ است پس مي توان نوشت:

$$MOA_{delay} = \log(size) + \log(\lceil \log(v_i.Cverr_i) \rceil)$$
$$= \log(size) + \log(16+n)$$

پس در کل تاخیر تبدیل ناقص RNS بصورت زیر است:

$$RESCOVS_{delay}=2n+\log(size) + \log(16+n)$$

**- مساحت تبدیل ناقص RNS**

۱) مساحت MULROM

ورودي این واحد حداکثر مقدار $v_i$ وخروجي آن $v_i.Cverr_i$ خواهد بود و به تعداد پیمانه ها از این واحد موجود است. پس مساحت بصورت زیر است:

$$size.\left(2^{\lceil \log(v_i) \rceil}.\lceil \log(Cverr_i.v_i) \rceil\right)$$
$$= size.2^n.(16+n)$$

۲- مساحت MOA

تعداد ورودي هاي این واحد، تعداد پیمانه ها و حداکثر مقدار آنها $Cverr_i.v_i$ مي‌باشد. پس مساحت بصورت زیر است:

$$size.\lceil \log(Cverr_i.v_i) \rceil + \lceil \log(Cverr_i.v_i) \rceil + \log\lceil \log(Cverr_i.v_i) \rceil$$
$$=(16+n).(size+\log(16+n))$$

### ۶-۶-۲- محاسبه تأخیر و مساحت در طرح چهارم

حال با توجه به روابط بالا و روابط بدست آمده در طرح سوم تأخیر و مساحت طرح چهارم را محاسبه مي كنيم. لازم به ذكر است كه در روابط پايين size تعداد پيمانه، RESCOVS تبديل RNS استاندارد (با روابط طرح سوم) وRESCOVS تبديل ناقص RNS كه اخيراً بررسي شده مي باشد.

**- محاسبه تأخيردرطرح چهارم**

اگر بلوك دياگرام طرح چهارم را مورد بررسي قرار دهيم متوجه مي شويم كه در خروجي يك جمع كننده با دو ورودي داريم كه ورودي پايين (rvt4) وابسته به ورودي بالا (rvt3) مي باشد پس در نتيجه تأخير rvt4 در اين قسمت تعيين كننده است. از طرف ديگر در مسير ايجاد rvt4 جمع كننده اي وجود دارد كه دو ورودي دارد. يكي مسير بالايي ($rvt1_i$) و ديگري مسير بالاي ($sv_i$).

ابتدا تاخير هريك از اين دو مسير را محاسبه مي كنيم:

**۱- تأخير مسير منتهي به $rvt1_i$ :**

الف- تاخير $MUL\ (x_{\omega_i}, y_{\omega_i})$

حداكثر مقدار هر ورودي برابر $\omega_i$ است و با توجه به فرمول ضرب كننده در بخش ۵-۳-۲ داريم:

$$MUL_{delay}=3\lceil \log\lceil \log \omega_i \rceil \rceil = 3\lceil \log n \rceil$$

ب- تاخير $MOD(\omega_i)$

يك ROM با حداكثر ورودي $\omega_i^2$ مي باشد پس داريم:

$$MOD_{delay}=2\lceil \log \omega_i^2 \rceil = 4n$$

ج- تاخير $MULMOD(mp\omega_i, \omega_i)$

يك ROM با حداكثر ورودي $\omega_i$ داريم. پس:

$$MULMOD_{delay}=2\lceil \log \omega_i \rceil = 2n$$

د- تاخير تبديل RNS استاندارد $q_\omega$ به $q_v$

همانطور كه گفتيم در اين طرح از همان روش تبديل RNS استاندارد بكار رفته در طرح سوم استفاده شده و در نتيجه تاخير اين بخش همان $RESCOV_{delay}$ مي باشد.

هـ- تاخير $MULROM(mv_i)$

در اينجا هم ROM با حداكثر ورودي $v_i$ داريم. پس:

$$MULROM_{delay}=2n$$

پس براي $rvt1_i$ داريم:

$$rvt1_{delay} = 3\lceil \log n \rceil + 8n + RESCOV_{delay}$$

## ۲- تاخير مسير منتهاي به $sv_i$

الف- تاخير دو تبديل RNS استاندارد $x_\omega$ به $x_v$ و $y_\omega$ به $y_v$

اين دو تبديل به طور موازي با هم اعمال مي شود و با توجه به اينكه تعداد عملوندهاي هر كدام size و حداكثر مقدار ورودي $x_{\omega_i}$ به $y_{\omega_i}$، $\omega_i$ مي باشد تاخير آن دو برابر است پس كافيست تاخير يكي از آنها را در نظر بگيريم و

همانطور که قبلاً دیده ایم تاخیر یك تبدیل RNS استاندارد $RESCOV_{delay}$ می باشد.

ب- تاخیر $MUL\ (x_{v_i}, y_{v_i})$

حداکثر مقدار هر ورودي این ضرب کننده $v_i$ می باشد و در نتیجه تاخیر آن:

$$MUL_{delay}=3\lceil \log\lceil \log v_i \rceil \rceil = 3\lceil \log n \rceil$$

ج- تاخیر $MOD\ (v_i)$

در اینجا یك ROM با حداکثر مقدار ورودي $v_i^2$ داریم. پس:

$$MOD_{delay}=2\lceil \log v_i^2 \rceil = 4n$$

پس براي $sv_i$ داریم:

$$sv_{i_{delay}} = 4n + 3\lceil \log n \rceil + RESCOV_{delay}$$

با توجه به مقادیر بدست آمده کاملاً مشخص است که تاخیر $rvt1_i$ به اندازه $4n$ از $sv_i$ بیشتر است.

حال به محاسبه تاخیر $rvt4_i$ می پردازیم.

**تاخیر $rvt4_i$**

۱- تأخیر $ADD(rvt1_i, sv_i)$

براي محاسبه تاخیر این تفریق باید ببینیم هر کدام از ورودي هاي آن در بدترین حالت چه مقداري دارند. $sv_i$ خروجي یك $MOD(v_i)$ می باشد و حداکثر مقدار آن $v_i$ است ولي $rvt1_i$ خروجي یك $MULROM(mv_i)$ است که ورودي آن یعني $qv_i$ حداکثر می تواند مقدار $v_i$ داشته باشد پس $rvt1_i$ حداکثر $v_i^2$ می باشد. پس $rvt1_i$ در تاخیر موثرتر است و با توجه به رابطه بخش ۵-۳-۲ براي جمع کننده می توان نوشت:

$$ADD_{delay}=4\log(\lceil \log v_i^2 \rceil) = 4\log(2n)$$

۲- تأخیر ADD MOD CONST $(2mv_i, mpv_i, v_i)$

این واحد که یک عملیات با ROM انجام می دهد عدد ورودي را در $2mv_i$ ضرب کرده با $mpv_i$ جمع کرده و حاصل را در پیمانه $v_i$ محاسبه می کند حداکثر مقدار ورودي این واحد (با توجه به توضیحات ADD بالا) $v_i^2$ می باشد پس در مورد تاخیر این واحد داریم:

$$ADD\ MOD\ CONST_{delay} = 2\lceil \log v_i^2 \rceil = 4n$$

۳- تأخیر تبدیل ناقص RNS بر روي $rvt3_i$

همانطور که قبلاً بررسي شده تاخیر این واحد $RESCOVS_{delay}$ محاسبه شده در طرح چهارم می باشد.

٤- تاخیر $MULMOD(mv_i, v_i)$

یک عملیات با ROM با حداکثر ورودي $size.v_i.Cverr_i$ در واقع ورودي این واحد خروجي واحد تبدیل ناقص RNS می باشد که با توجه به شکل بلوکي تبدیل ناقص RNS خروجي این واحد، خروجي یک MOA است که این MOA به تعداد size ورودي که هر کدام حداکثر مقدار $v_i.Cverr_i$ را دارد را با هم جمع می کند پس مقدار ورودي این واحد حداکثر $size.v_i.Cverr_i$ خواهد بود و تاخیر این واحد:

$$MULMOD_{delay} = 2\lceil \log(size.v_i.Cverr_i) \rceil$$

و با توجه به مباحث گذشته در مورد $Cverr_i$ می توان نوشت:

$$MULMOD_{delay} = 2\lceil \log(size.2^{(n+16)}) \rceil$$

و در مورد تاخیر $rvt4_i$ داریم:

$$rvt4_{i_{delay}} = 4\log(2n) + 4n + 2\lceil \log(size.2^{(n+16)}) \rceil + RESCOVS_{delay}$$

حال باید تاخیر مسیر مشترك نهایي كه شامل یك ADD و یك MOD مي‌باشد حساب كنیم.

**تاخیر مسیر مشترك:**

۱- تاخیر $ADD(rvt3_i, rvt4_i)$

براي محاسبه تاخیر این ADD نیز باید ببینیم هر كدام از دو ورودي حداكثر چه مقداري دارند. $rvt3_i$ خروجي ADD MOD CONST مي باشد و در نتیجه حداكثر مقدار $v_i$ را دارد. $rvt4_i$ نیز خروجي یك MULMOD است و در نتیجه آنهم حداكثر مقدار $v_i$ دارد پس حداكثر مقدار هر كدام از ورودي ها $v_i$ است. پس براي تاخیر این واحد داریم:

$$ADD_{delay}=4\log(\lceil \log v_i \rceil)=4\log n$$

۲- تاخیر $MOD(v_i)$

این واحد در ورودي خود خروجي ADD مرحله قبل را مي بیند كه حداكثر مي‌تواند مقدار $v_i$ داشته باشد. پس:

$$MOD(v_i)_{delay}=2\lceil \log v_i \rceil=2n$$

پس تاخیر كل طرح چهارم برابر است با:

$$P4_{delay}=rvtl_{i_{delay}}+rvt4_{i_{delay}}+4\log+2n$$

$$=RESCOV_{elay}+RESCOVS_{delay}+3\lceil \log n \rceil+14n+4+8\log n$$

$$+2\lceil \log(size.2^{(n+16)}) \rceil$$

**- محاسبه مساحت طرح چهارم**

۱- مساحت MULها

با توجه به رابطه بخش ۵-۳-۲ براي مساحت MUL و اینكه ورودي هر كدام از واحدهاي MUL به ترتیب حداكثر $\omega_i$ و $v_i$ خواهد بود و از هر كدام به

تعداد پیمانه‌ها داریم براي مساحت هر کدام به ترتیب مقادیر زیر بدست مي آید:

a) $\lceil \log \omega_i \rceil^2 .size = n^2 .size$

b) $\lceil \log v_i \rceil^2 .size = n^2 .size$

۲- مساحت MOD بعد از هر MUL

این واحد با ROM پیاده شده. حداکثر ورودي و خروجي هر کدام به ترتیب $v_i, v_i^2$ و $\omega_i, \omega_i^2$ مي باشد و از هر کدام به تعداد پیمانه ها داریم پس براي مساحت هر کدام به ترتیب مقادیر زیر بدست مي آید:

a) $2^{\lceil \log \omega_i^2 \rceil} . \lceil \log \omega_i \rceil .size = 2^{2n} .n.size$

b) $2^{\lceil \log v_i^2 \rceil} . \lceil \log v_i \rceil .size = 2^{2n} .n.size$

۳- مساحت $MULMOD(mp\omega_i, \omega_i)$

این واحد نیز با ROM پیاده مي شود و به تعداد پیمانه ها تکرار شده است. حداکثر مقدار ورودي و خروجي آن حداکثر $\omega_i$ مي باشد. پس مساحت آن

$2^{\lceil \log \omega_i \rceil} . \lceil \log \omega_i \rceil .size = 2^n .n.size$

٤- مساحت مربوط به RESCOV

این واحد در ۳ جا تکرار شده و در نتیجه مساحت اشغال شده توسط آن

$3.RESCOV_{Area}$

لازم بذکر است که $RESCOV_{Area}$ برابر با مقدار بدست آمده براي تبدیل RNS استاندارد طرح سوم مي باشد.

۵- مساحت $MULROM(mv_i)$

این واحد به تعداد پیمانه ها موجود است. پیاده سازي آن با ROM صورت گرفته و در آن حداکثر مقدار ورودي و خروجي $v_i$ و $v_i^2$ مي باشد پس براي مساحت این قسمت داریم:

$$size.2^{\lceil \log v_i \rceil}.(\lceil \log v_i^2 \rceil) = size.2^n.2n = size.2^{n+1}.n$$

۶- مساحت ADD

همانطور که در هنگام محاسبه تاخیر این واحد دیدیم حداکثر مقدار ورودي آن $v_i^2$ است پس در مورد مساحت آن با توجه به تکرار شدن به تعداد پیمانه ها داریم:

$$size.10\lceil \log v_i^2 \rceil = size.10.2n = size.20n$$

۷- مساحت ADD MOD CONST

با توجه به مباحث تاخیر این واحد مي دانیم که ورودي و خروجي آن حداکثر مقادیر $v_i, v_i^2$ را دارد و به تعداد پیمانه ها از این واحد داریم در نتیجه براي مساحت این بخش داریم:

$$size.2^{\lceil \log v_i^2 \rceil}.(\lceil \log v_i \rceil) = size.2^{2n}.n$$

۸- مساحت RESCOVS

مساحت این واحد یعني $RESCOVS_{Area}$ در بخش محاسبه مساحت تبدیل ناقص RNS صورت گرفته است.

۹- مساحت $MULMOD(mv_i, v_i)$

این واحد هم به تعداد size موجود است، با ROM پیاده شده و ورودي و خروجي آن به ترتیب حداکثر مقادیر $size.v_i.Cverr_i$ و $v_i$ را دارد و مساحت آن بشکل زیر خواهد بود:

$$size.2^{\lceil \log(size.v_i.Cverr_i) \rceil}.\lceil \log v_i \rceil = size.2^{\lceil \log(size.2^{n+16}) \rceil}.n$$

۱۰- مساحت $ADD(rvt3_i, rvt4_i)$

همانطور که در قسمت محاسبه تاخیر دیدیم هر کدام از ورودي ها حداکثر مقدار $v_i$ دارند پس با توجه به وجود size عدد از این بلوك براي مساحت داریم:

$$size.10.\lceil \log v_i \rceil = 10.n.size$$

۱۱- مساحت $MOD(v_i)$ در خروجي نهایي

این واحد هم به تعداد size در طرح تکرار شده و با توجه به اینکه توسط یك ROM پیاده شده و ورودي و خروجي آن حداکثر مقدار $v_i$ را دارد. مساحت به شکل زیر خواهد بود:

$$size.2^{\lceil \log v_i \rceil}.\lceil \log v_i \rceil = size.2^n.n$$

و مساحت طرح چهارم برابر است با:

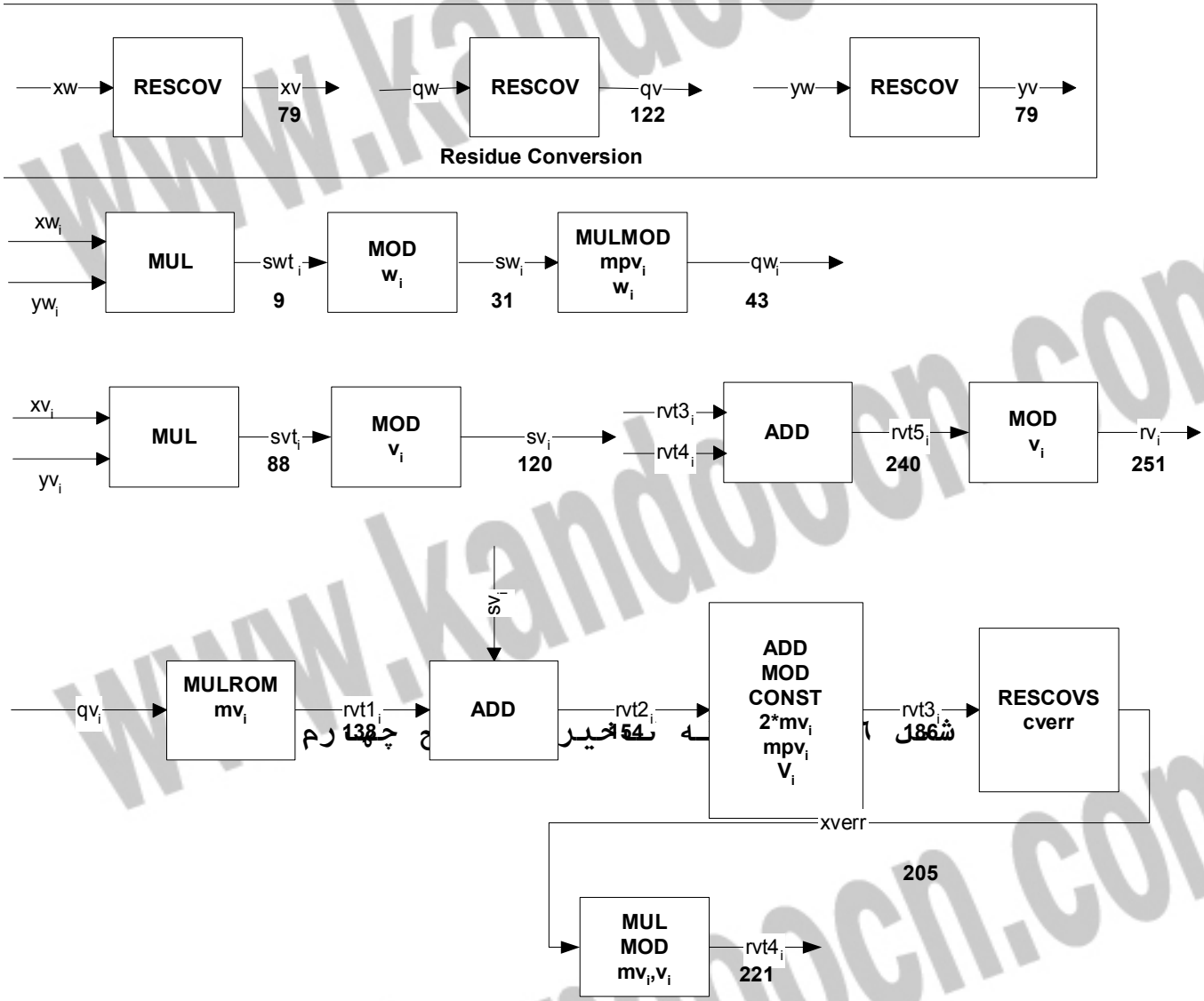
$$P4_{Area} = 2n^2.size + 2^{2n+1}.n.size + 2^n.size.n + 3RESCOV_{Area}$$
$$+ size.2^{n+1}.n + size.20n + size.2^{2n}.n + RESCOVS_{Area}$$
$$+ size.2^{\lceil \log(size.2^{n+16}) \rceil}.n + 10.n.size + size.2^n.n$$

# فصل پنجم

# طرح سوم

## ٦-٧- نتايج شبيه سازي در طرح چهارم

اعداد بكار رفته در آزمايش طرح چهارم همان اعداد طرح سوم هستند و در محدوده‌اي طرح چهارم صدق مي كنند. نتيجه حاصله همان (١٧٢ و ٥و٢٥) است كه صحيح مي باشد. حجم كل برحسب گيت ١١٦٩٥٤٩ و تأخيرها را در شكل (٦-٤) مي بينيد.



# ۷- مقایسه طرح ها و جمع بندي

## ۷-۱- مقایسه طرح ها

در این بخش سعي بر مقایسه چهار طرح خواهیم داشت. ابتدا نتایج بدست آمده براي هر طرح را بررسي مي کنیم و سپس به مقایسه طرحها با هم از لحاظ کیفي و کمي خواهیم پرداخت.

با توجه به روابط تاخیر و پیچیدگي هر طرح متوجه مي شویم که دو متغیر اصلي در هر کدام از این دو پارامتر تاثیر گذار هستند. یکي تعداد بیت هر پیمانه RNS و دیگري تعداد پیمانه ها. در این بخش سعي مي کنیم که اثر این متغیر ها را در روابط پیچیدگي تاخیر و مساحت بررسي کنیم. همانطور که در فصول محاسبه تاخیر و مساحت هر طرح دیدیم براي امکان پذیر ساختن روابط تاخیر و پیچیدگي تعداد بیت هاي همه پیمانه هاي RNS را با هم یکسان گرفتیم. نکته دیگر رابطه تعداد بیت عملیات و اعداد ورودي با تعداد بیت پیمانه هاي RNS و تعداد پیمانه هاي RNS مي باشد. اگر اعدادي که مي خواهیم با هم ضرب پیمانه اي کنیم (و همینطور پیمانه عملیات) n بیتي باشد و تعداد پیمانه هاي RNS، size و تعداد بیت متوسط هر کدام t بیت باشد باید رابطه زیر برقرار باشد.

$$t = \left\lceil \frac{n}{size} \right\rceil$$

مثلاً اگر پیمانه عملیات ٦٤ بیتي باشد و بخواهیم ۸ پیمانه داشته باشیم. هر پیمانه

باید بطور متوسط ۸ بیتی باشد. البته این رابطه در واقع یک مقدار متوسطی را نتیجه میدهد و باز هم نوعی ساده سازی برای امکان پذیر ساختن مقایسه است. مطالبی که رابطه بالا نشان می دهد، رابطه معکوس بین تعداد بیت هر پیمانه و تعداد پیمانه با بودن یک برای تعداد پیمانه های RNS و تعداد بیت هر کدام در نظر گرفت.

با ذکر یک مثال از پیمانه ٦٤ بیتی (که در تعریف پروژه به عنوان پیمانه مورد نظر ما بوده) مطالب بالا را روشن می سازیم. فرض کنید پیمانه عملیات بصورت عدد "13434002501101518 85" باشد یک عدد ٦٤ بیتی است. این پیمانه را می توان در هر کدام از سیستم های RNS زیر بیان کرد.

RNS1 → (3,13,17,19,23,29,31,37,41,43,47,53,59)

RNS2 → (177,689,799,817,943,851,899)

RNS3 → (109635747,679949,77043)

تعداد اعداد قابل نمایش در هر کدام از این سه سیستم عدد

27468005002203037701

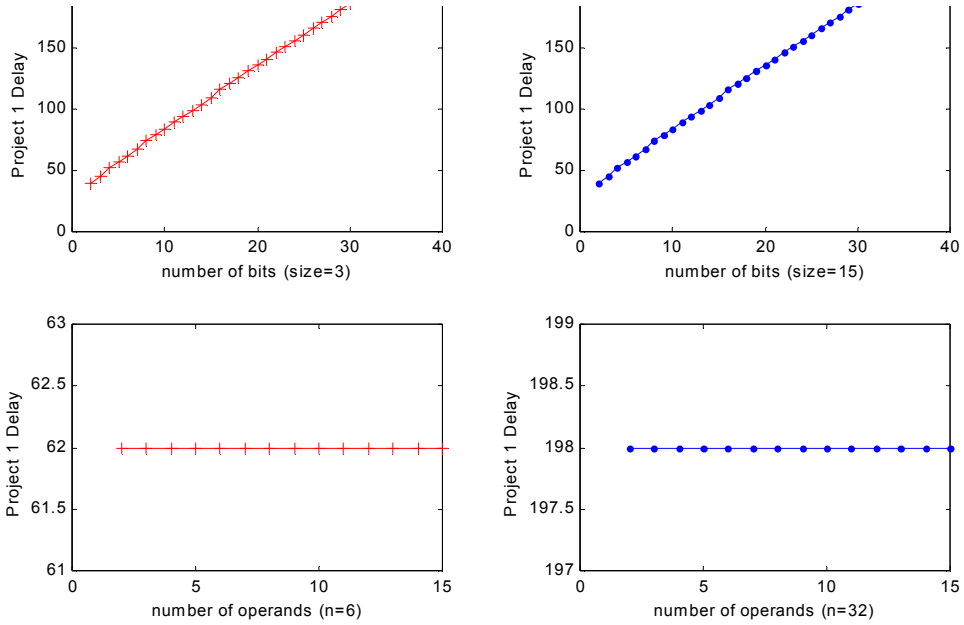
است که با توجه به بزرگتر بودن پیمانه ها عملیات در پیمانه گفته شده را امکان پذیر می کند تعداد پیمانه هر کدام به ترتیب 4,7,13 و تعداد متوسط هر کدام 22,10,5 بیت می باشد که با رابطه (۱-۷) تطبیق دارد. همچنین رابطه معکوس بین تعداد پیمانه های RNS و تعداد بیت هر پیمانه RNS مشخص است. یعنی می توان با بالا

بردن تعداد پیمانه هاي RNS و کوچکتر کردن اندازه هر کدام سیستم مختلفي بدست آورد که انجام محاسبه در عملیات پیمانه اي nبیتي را ممکن ساخت.

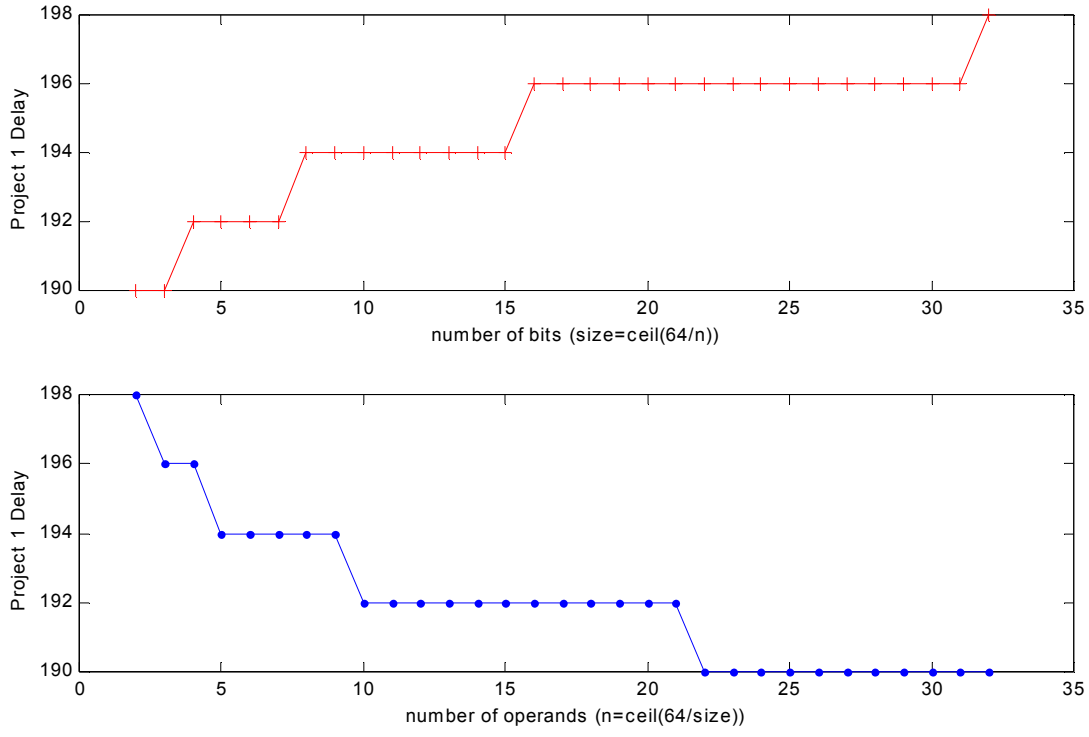
حالا باید دید اثر هر کدام از این دو متغیر در طرحهاي مختلف چگونه است. در ادامه براي هر طرح سه دسته نمودار ترسیم شده است. در دسته اول با فرض تعداد پیمانه ثابت (مقادیر ۳و۱۵ پیمانه به طور اختیاري در نظر گرفته شده)، تغییرات تاخیر و مساحت هر طرح براساس تغییر تعداد بیت هر پیمانه بررسي شده است. دست بعدي براي پیمانهRNS با تعداد بیت ثابت (که تعداد بیت 16,6 به طور اختیاري انتخاب شده) همان تغییرات را براساس تغییر تعداد پیمانه ها بررسي مي کنیم. در دسته آخر استفاده از رابطه (۷-۱) براي یك پیمانه عملیاتي ٦٤ بیتي وضعیت تاخیر و حجم هر طرح بررسي شده است. در واقع هر بار تعداد پیمانه ها (و یا تعداد بیت هر پیمانه) متغیر مي باشد و به طور متناظر تعداد بیت هر پیمانه (و یا تعداد پیمانه ها) با رابطه (۷-۱) محاسبه مي شود. در ادامه نمودار هاي بیان شده در بالا را براي هر طرح و توضیحات مربوط به هر کدام را خواهید دید.

شکل (۷-۱) دسته اول و دوم نمودارها را براي تاخیر طرح اول نشان مي دهد. از نمودارها چنین بر مي آید که تعداد پیمانه ها در تاخیر

تاثيري ندارد و آنچه در تاخير طرح اول تاثير گذار است فقط تعداد بيت هر پيمانه است كه تاثيرگذاري آن تقريباً خطي است. اين مساله با آنچه در مورد بيان RNS طرح اول در فصل سوم بيان شد مطابقت دارد. رابطه تاخير طرح اول يعني رابطه ۳-۵ مطلب بالا را تصديق مي كند.

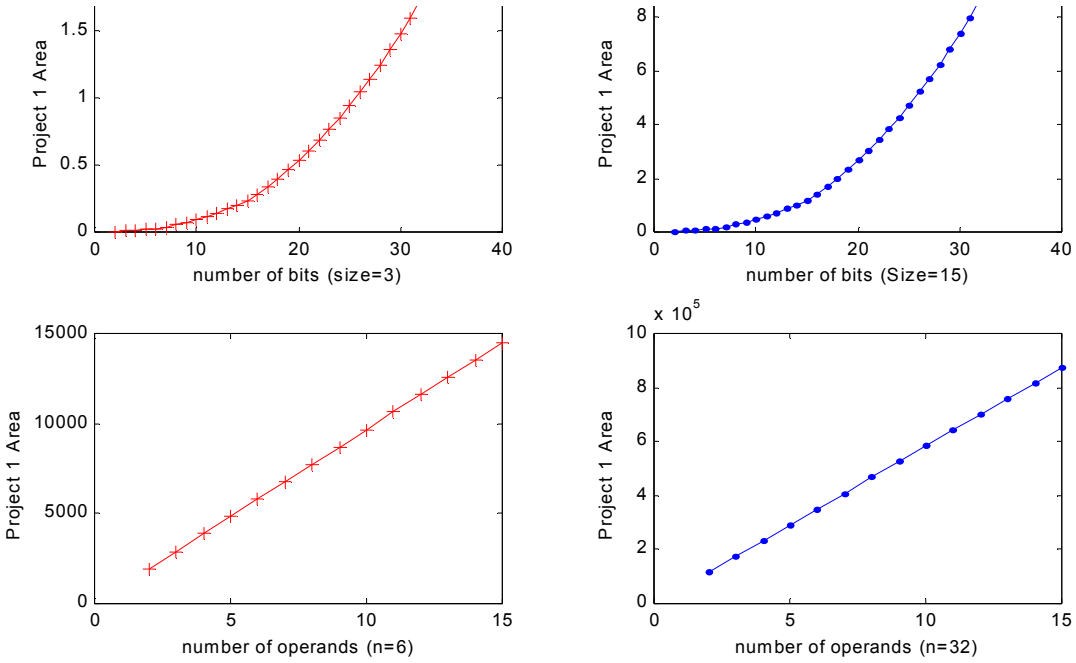



طرح اول به نمایش در آمده است.



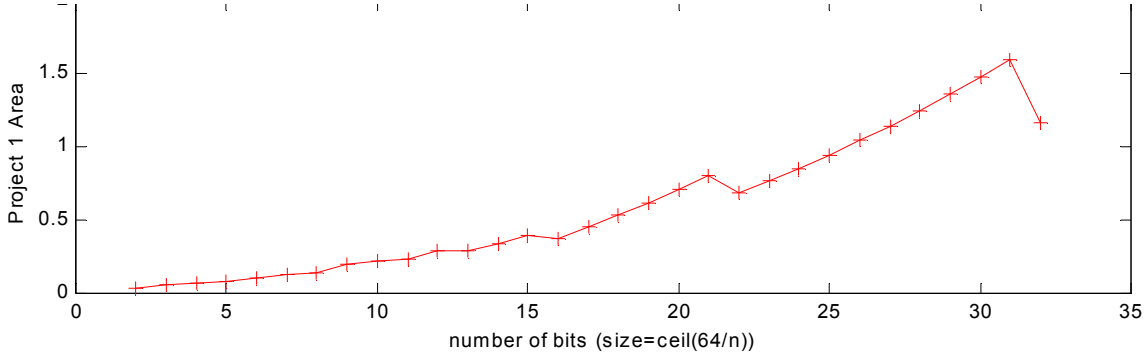

کاهش تاخیر طرح اول موثر است.

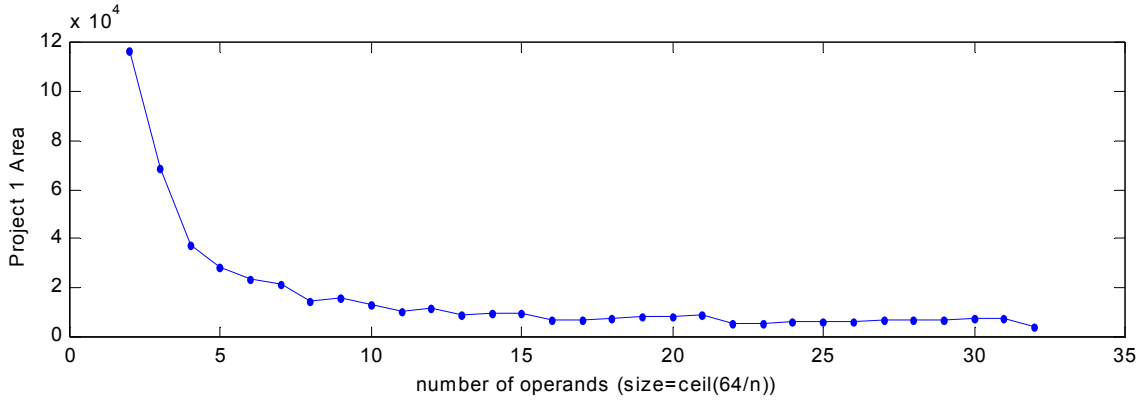
در شکل (۷-۳) دسته اول و دوم نمودارها برای مساحت طرح اول نشان داده شده است.




رفتن تعداد بيت هر پيمانه به شكل غيرخطي در مساحت طرح دوم تاثير دارد ولي بالا رفتن تعداد پيمانه ها به صورت خطي در اين مساحت تاثيرگذار است. اين مساله نيز توسط رابطه ۳-٦ كه بيان كننده مساحت طرح اول در حالت RNS است تصديق مي شود، در اين رابط m فقط به صورت يك ضريب در مساحت پايه طرح اول ضرب شده و در نتيجه تاثيري خطي خواهد داشت.

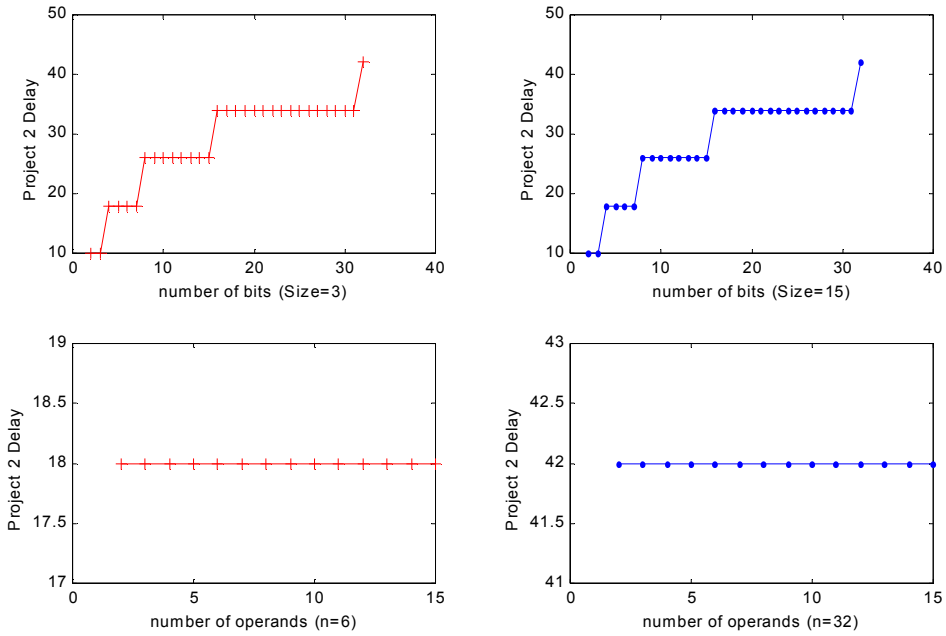
شكل (۷-٤) دسته سوم نمودارها را براي مساحت طرح اول نشان داده و نتيجه حاصله با نتيجه بدست آمده از دسته سوم نمودارهاي مساحت طرح اول يكسان است.



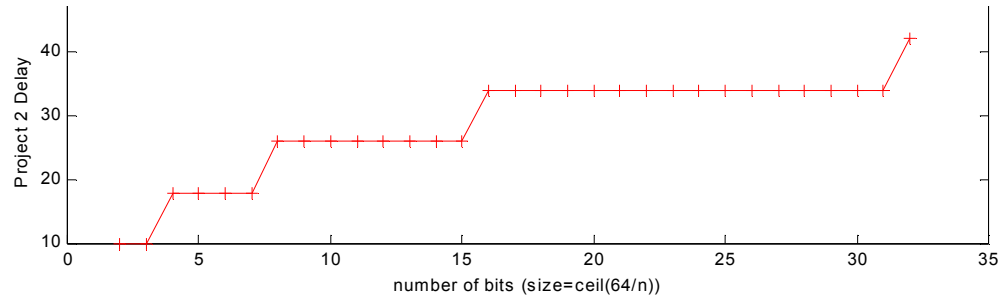

که اثر کاهش دهندگي بالا بردن تعداد پيمانه ها و کم کردن تعداد بيت هر پيمانه به تدريج محدود مي شود در واقع براي پيمانه عمليات ٦٤ بيت نمودارها نشان مي دهند که بالا بردن تعداد پيمانه ها از عدد ١٠ تاثير چنداني در مساحت و تاخير نخواهد گذاشت. اين مساله بخصوص در مورد مساحت مشهودتر است.

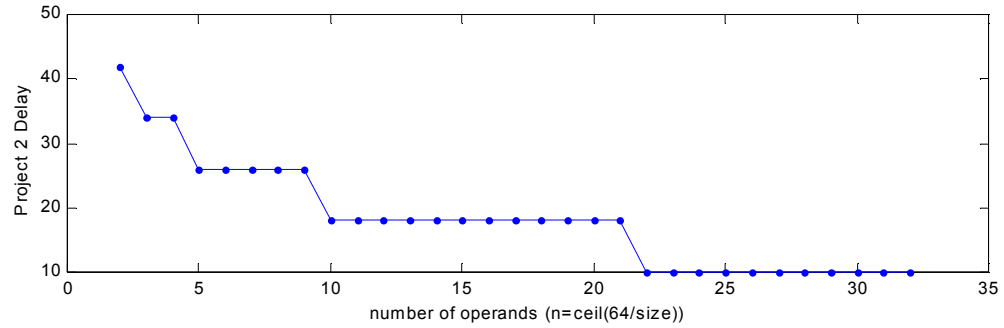
حالا به سراغ طرح دوم مي رويم. در شکل (٧-٥) دسته اول و دوم نمودارها براي تاخير طرح دوم به نمايش درآمد است.

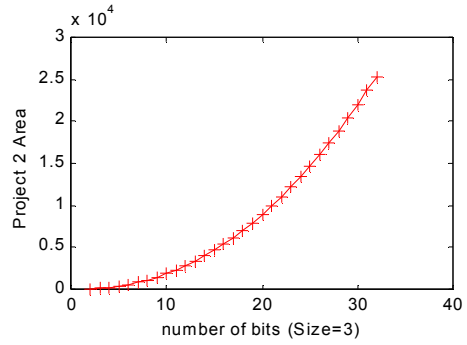


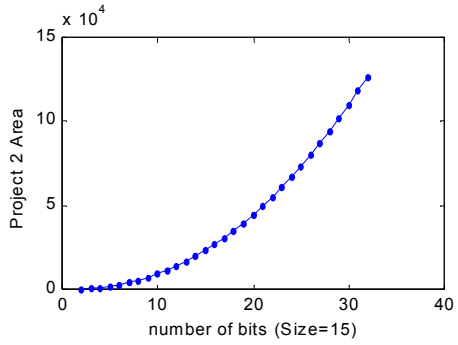

تاخیر وابستگي خاصي را به تعداد پیمانه ها نشان نمي دهد. درشکل (۷-٦) دسته سوم نمودارها براي تاخیر طرح دوم مشاهد مي شود.

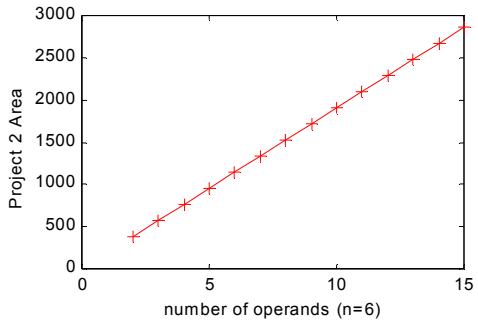
در اینجا نیز مي بینید که بالا بردن تعداد پیمانه ها و کاهش متناظر تعداد بیت هر کدام در کاهش تاخیر موثر است.

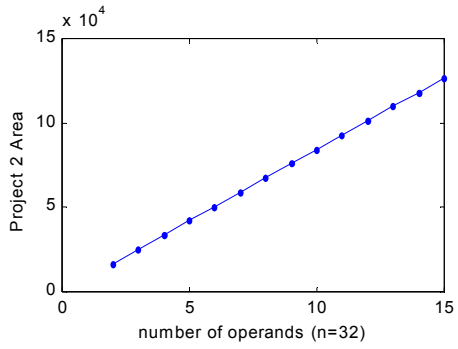


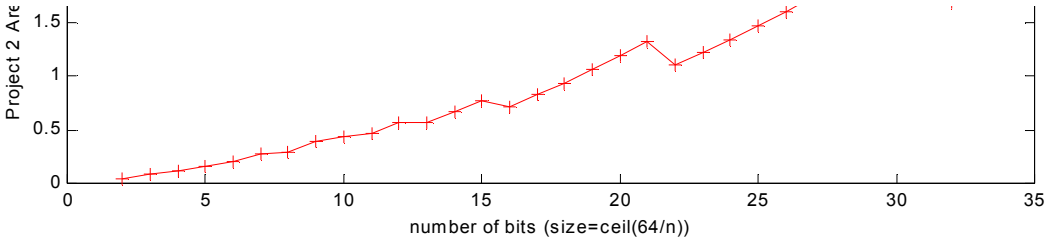



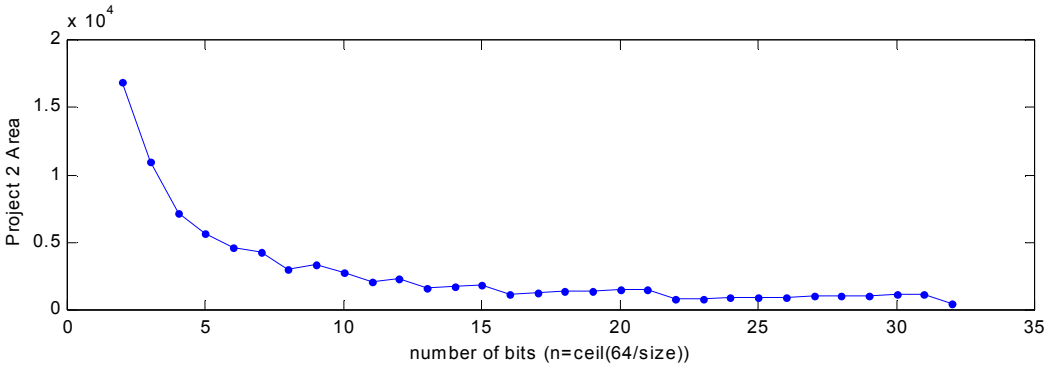



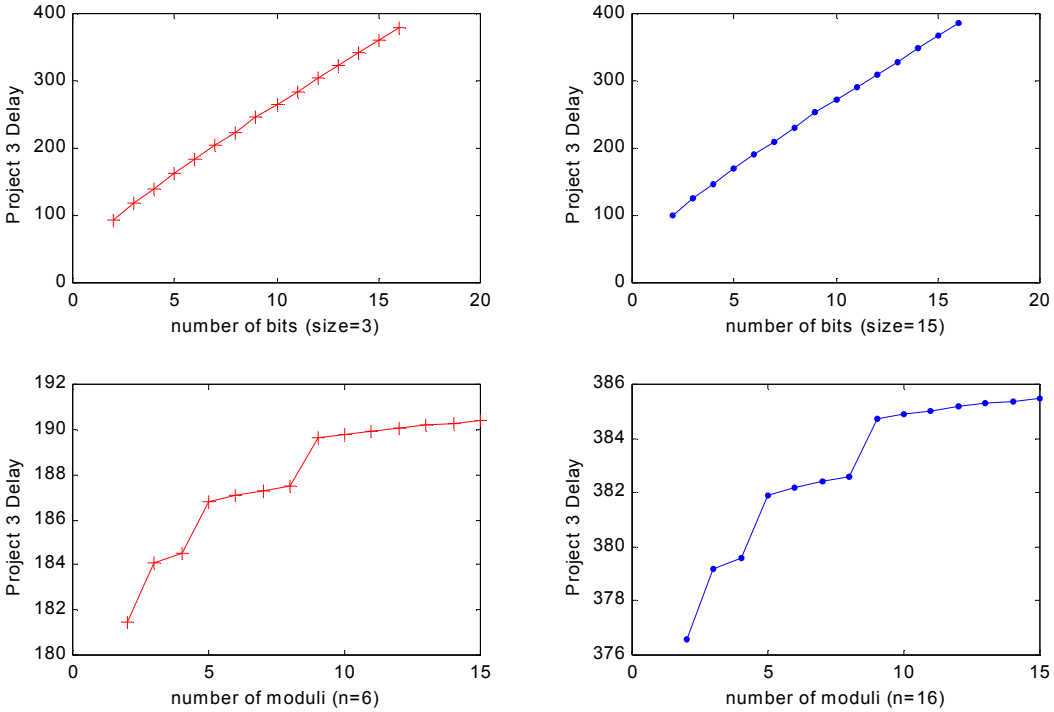




نمودارها براي مساحت طرح دوم را مي بينيد. در اينجا نيز توان نموداري شبيه طرح اول مشاهده كرد و در نتيجه تحليل هاي انجام شده براي طرح اول در مورد اين دسته نمودارها و همچنين نتايج حاصله، مشابه طرح اول است.

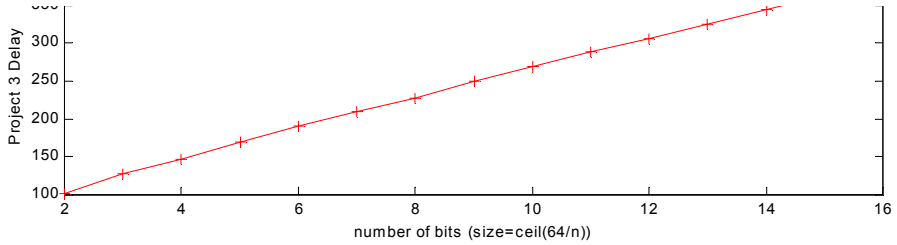


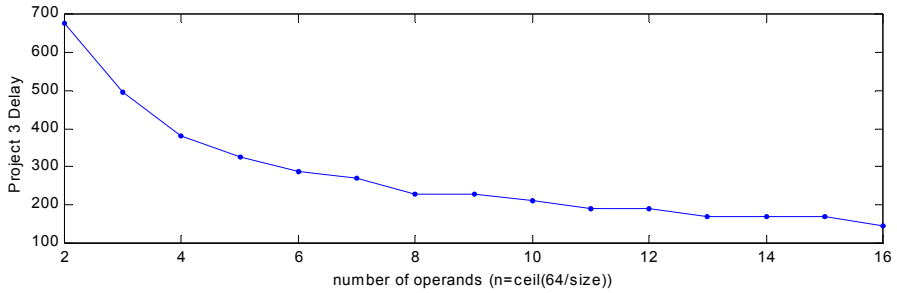



طرح سوم هدف بعدي بررسي ما خواهد بود. در شكل (۷-۹) دسته اول و دوم نمودارها را براي



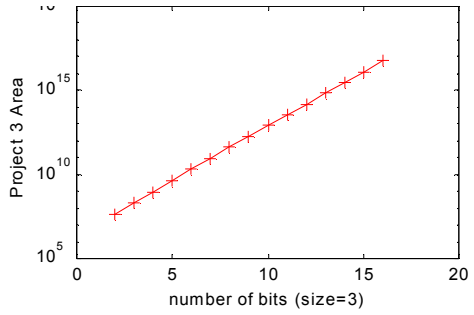

مشاهده اين نمودارهاست. نكته جالب ديگر، اين مساله است كه با پنج برابر شدن تعداد پيمانه ها، اندازه تاخير، تغيير زيادي نمي كند؛ ولي سه برابر شدن (تقريبي) تعداد بيت هر پيمانه تقريباً به همين نسبت يعني سه برابر تاخير را افزايش مي دهد. دسته سوم نمودار طرح سوم در مورد تاخير در شكل (۷-۱۰) مشاهده مي شود. در اينجا تاثير كاهش تاخير با كاهش تعداد بيت هر پيمانه و افزايش متناظر تعداد پيمانه ها براي يك پيمانه عمليات ثابت مشهود است.

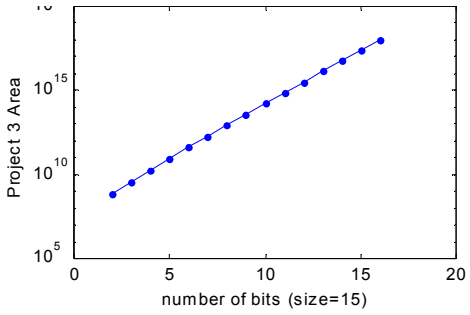


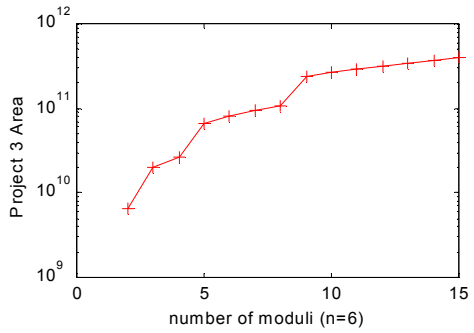



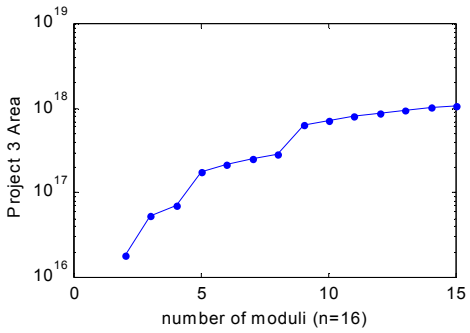

بیشتري دارد و بعد از آن کم کم بسیار کم اثر مي شود.

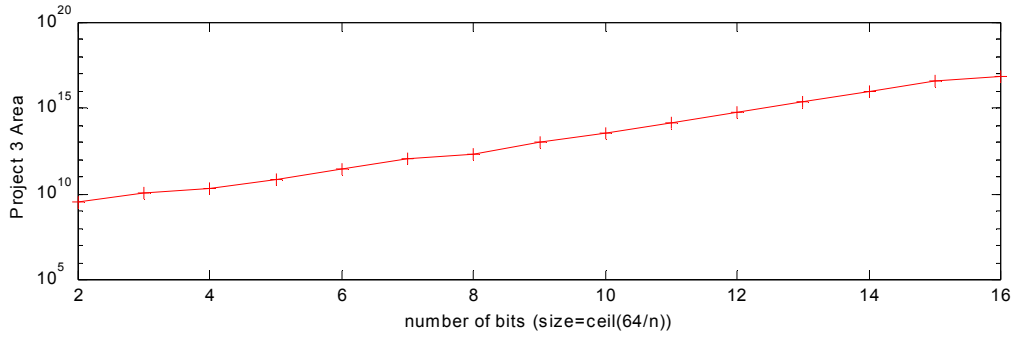
شکل (۷-۱۱) نمایانگر دسته اول و دوم نمودار ها براي مساحت طرح سوم ميباشد. در اینجا نیز نتایجي مشابه نتایج نمودار هاي متناظر براي تاخیر این طرح حاصل ميشود.

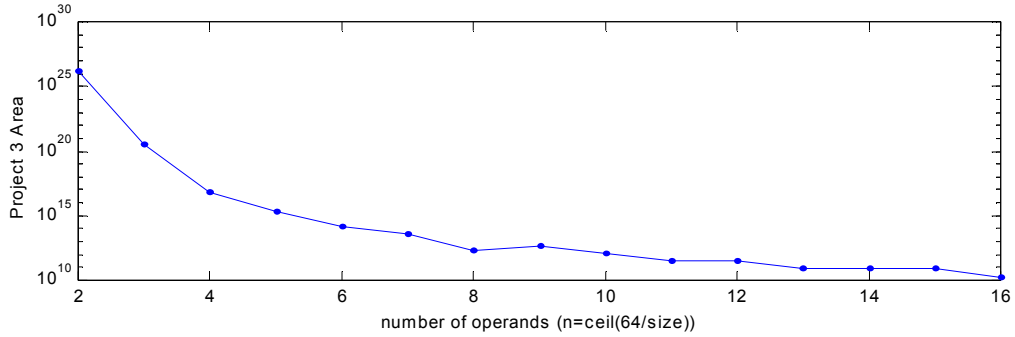


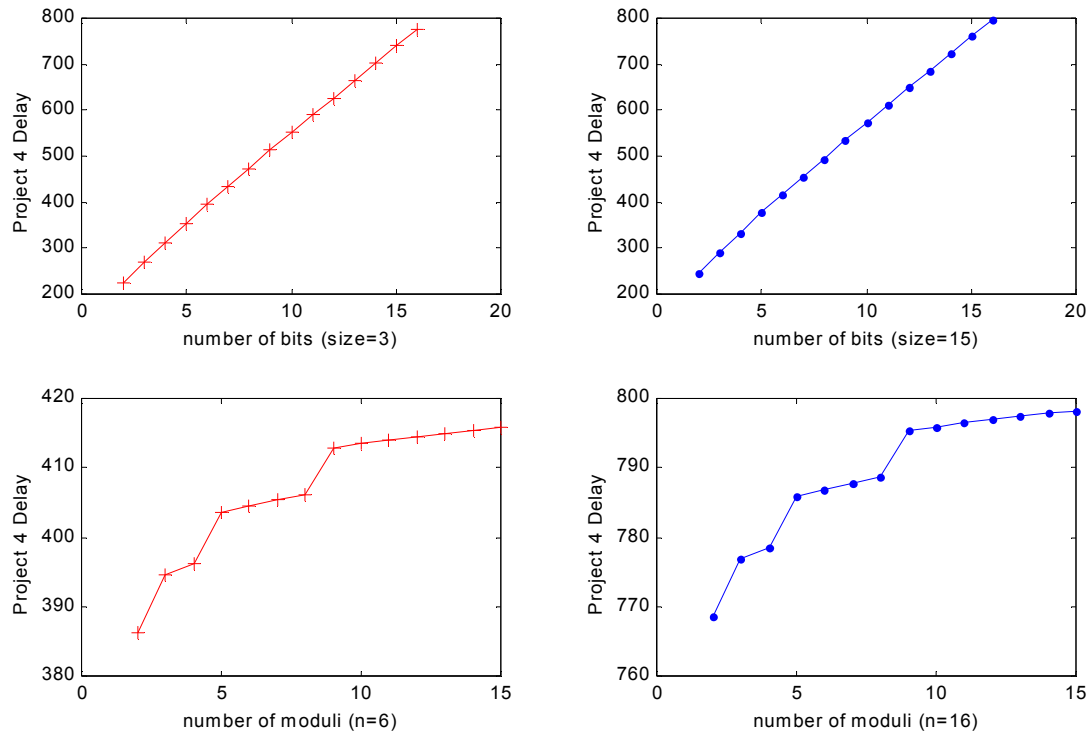



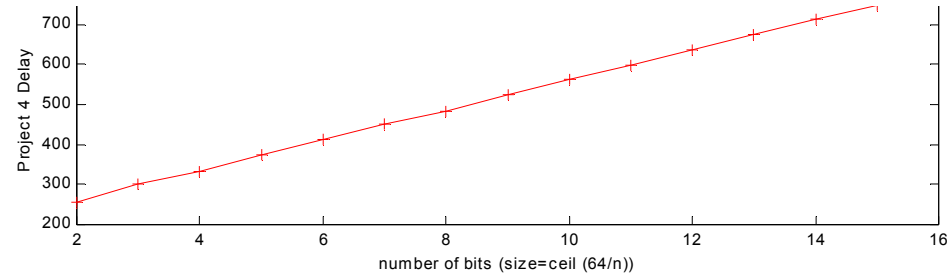



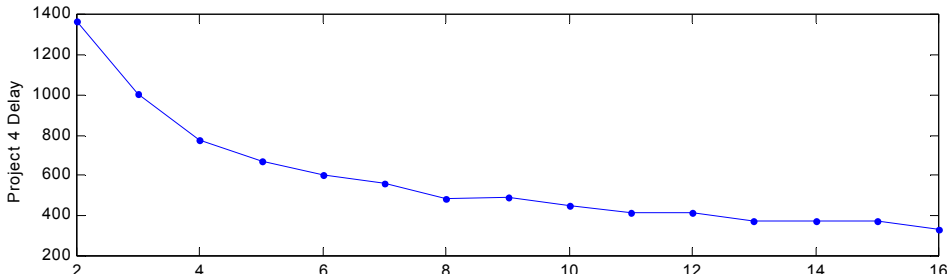



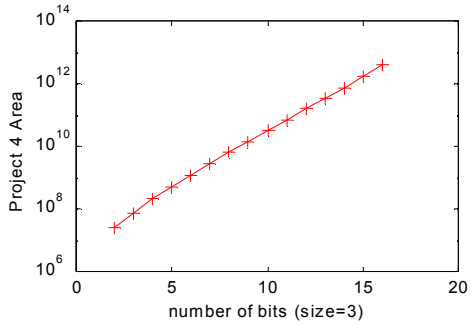
آنچه در این نمودارها دیده می شود، نتایجی مشابه نتایج مربوط به تاخیر همین دسته نمودار طرح سوم را در بر دارد.

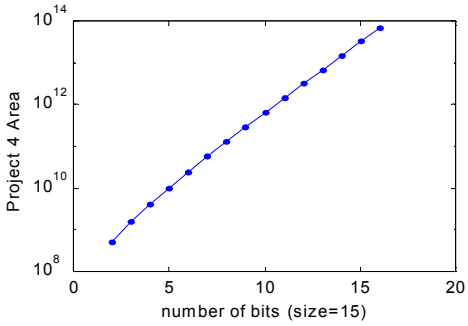
شکل (۷-۱۳)، (۷-۱۴)، (۷-۱۵) و (۷-۱۶) نشان دهنده نمودارهای مربوط به تاخیر و مساحت طرح چهارم می باشد که به علت تشابه نتایج شهودی با آنچه در مورد طرح سوم بیان شد از تکرار مطالب خودداری می شود.

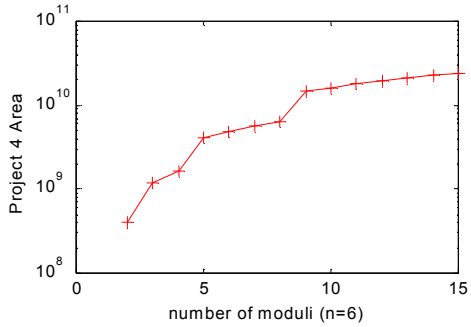


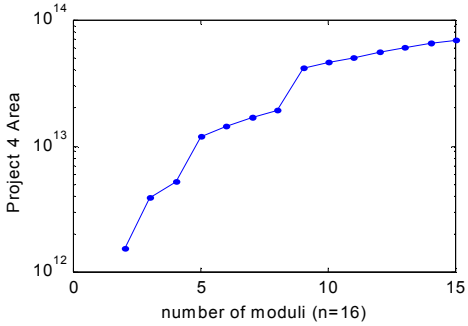


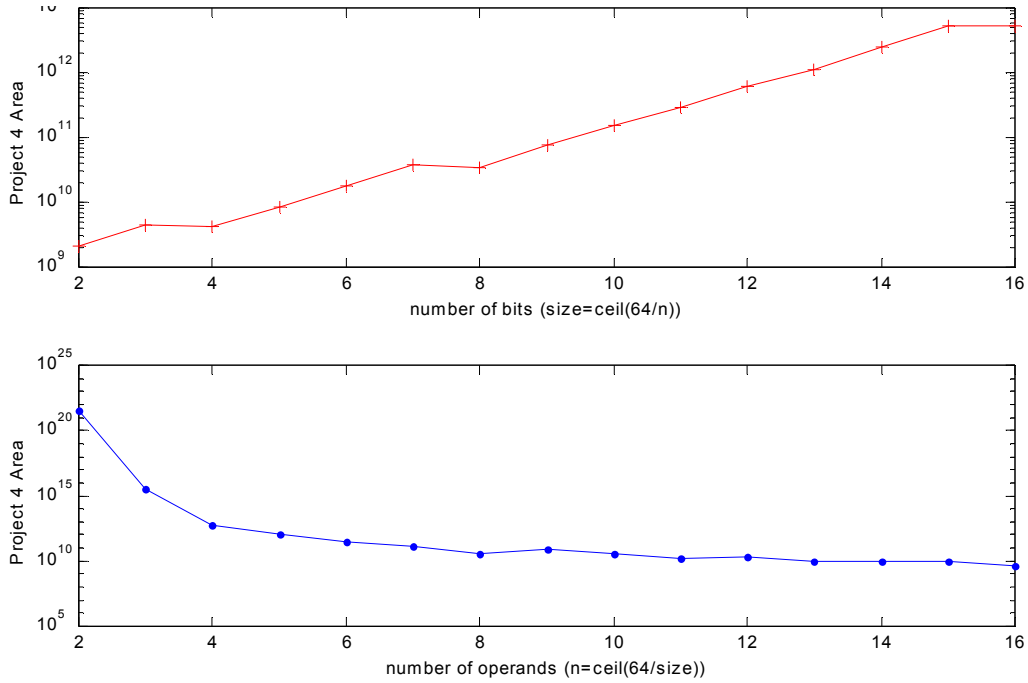



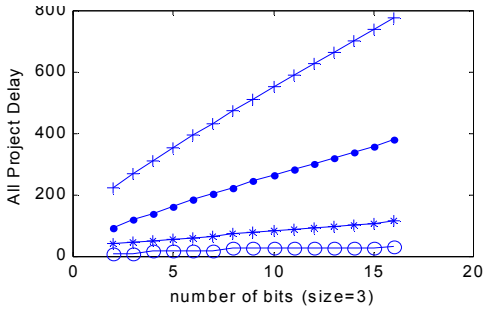



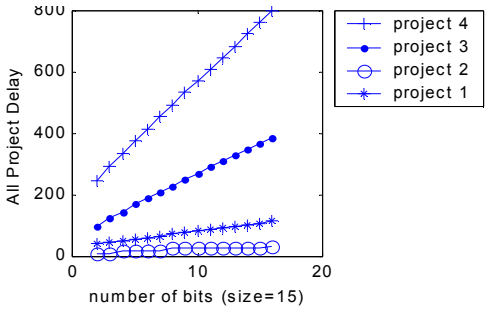



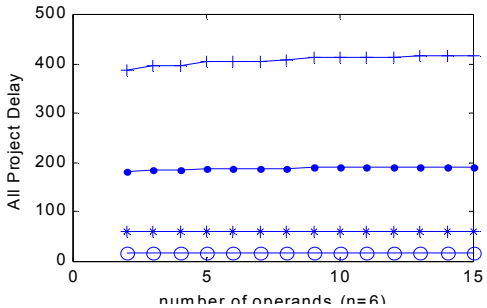


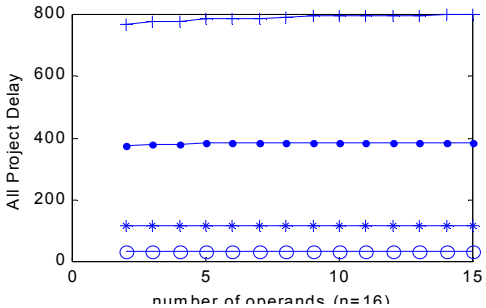

کمي مي باشد و در واقع بدون در نظر گرفتن موارد کاربرد نحوه عملکرد هر طرح انجام شده است. مباحث مربوط به تحليل مسائل غير کمي اين چهار طرح در بخش بعدي يعني جمع بندي مورد بررسي قرار مي گيرد.

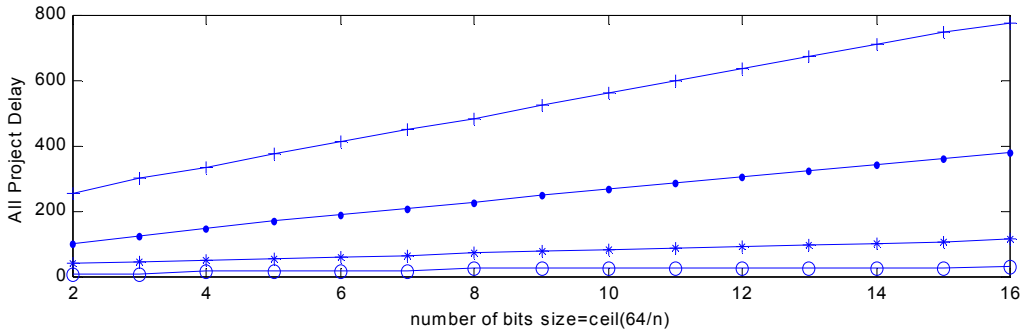
نمودار هاي (۷-۱۷)، (۷-۱۸)، (۷-۱۹) و (۷-۲۰) نشان دهنده دسته اول، دوم و سوم نمودارهاي هر چهار طرح به طور همزمان مي باشد.

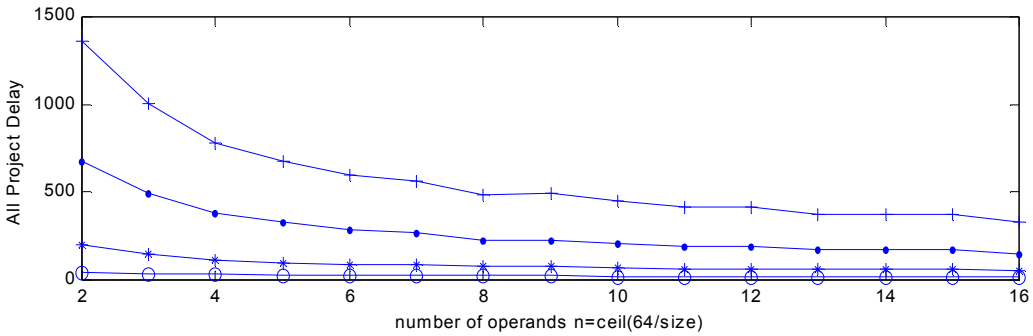


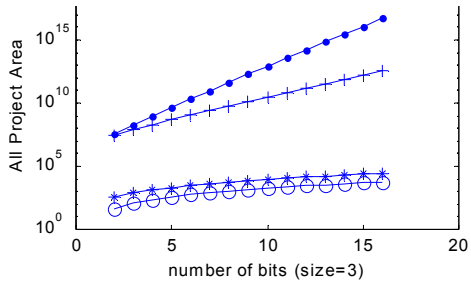



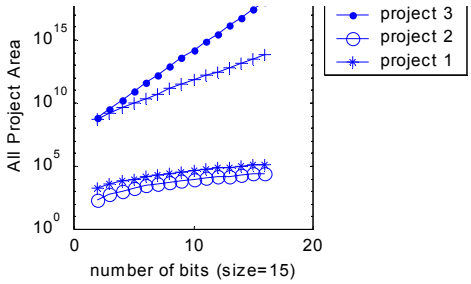



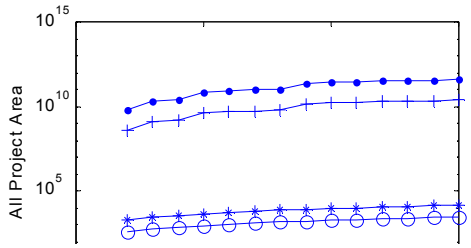



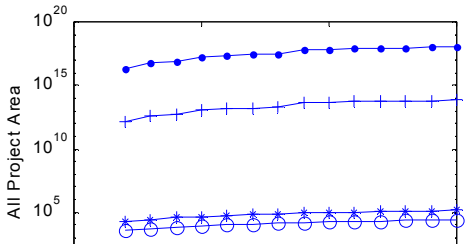



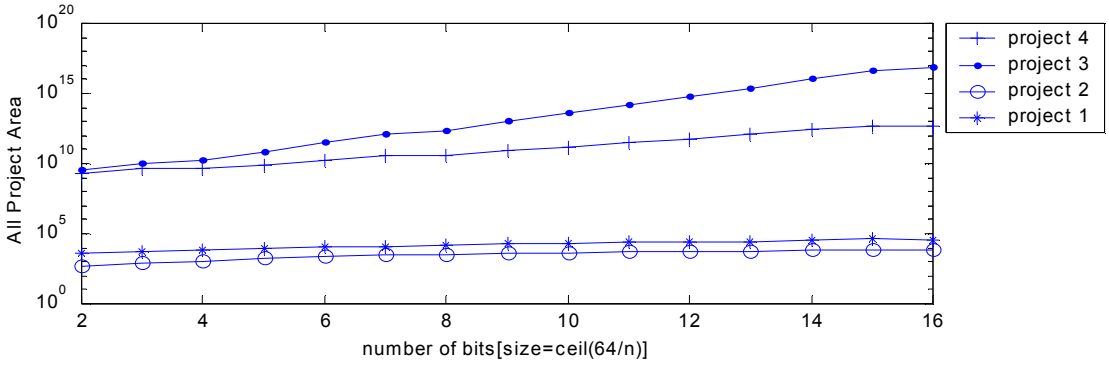



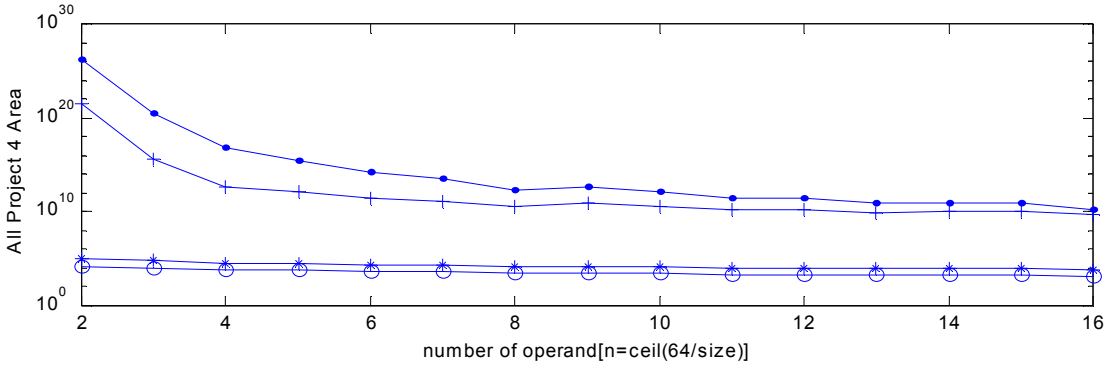


همانطور که مشاهده می شود طرح چهارم، کندترین طرح، طرح سوم حجیم ترین طرح و طرح دوم سریعترین و کم حجم ترین طرح می باشند. همچنین دیده می شود که هر چهار طرح وابستگی زیادی به تعداد پیمانه ها ندارند و تغییرات اساسی در تاخیر و مساحت با بالا رفتن تعداد بیت هر پیمانه رخ می دهد. مطلب دیگری که از این نمودارها بر می آید این است که وابستگی طرح سوم و چهارم (شیب نمودار) به تغییرات بیت ها بیشتر از طرح اول و دوم می باشد. نکته دیگری که از شکل (۷-۱۸) و (۷-۲۰) بر می آید امکان کاهش تاخیر و حجم با بالا بردن تعداد پیمانه و کاهش متناظر تعداد بیت هر پیمانه می باشد و این مساله در طرح سوم و چهارم مشهودتر است. البته این اثرگذاری با افزایش تعداد پیمانه ها کم کم کاهش می یابد و مثلاً در مثال ما که پیمانه ٦٤ بیتی است، برای مقادیر بزرگ تر از هشت پیمانه، تاثیر افزایش پیمانه ها بر کاهش تاخیر و مساحت بسیار کم شده است.

## ۷-۲- جمع بندي

با بررسي نتايج پياده سازي ها در مي يابيم كه طرحهاي يك و دو براي مواردي مناسب هستند كه هدف خود ضرب RNS باشد ولي طرحهاي سه و چهار وقتي مناسب هستند كه از ضرب RNS بعنوان هسته توان رساني استفاده مي شود. دليل اين امر وجود تبديلات بين سيستم هاي RNS و نيز خطاهاي قابل رفع در روش هاي RDM است.

مورد ديگر اين است كه در طرحهاي سه وچهار پيمانه ضرب ربطي به سيستمهاي عددي RNS ندارد و در واقع از RNS بعنوان كمك عمليات بهره مي برد، ولي در طرحهاي يك و دو اعداد در خود RNS ضرب مي شوند. در اصل طرحهاي يك و دو گونه اي از ضرب پيمانه اي هستند.

با توجه به كاربرد هاي بالا در مواردي كه عمليات مورد نظر ما توسط طرح اول و دوم قابل پياده سازي باشد بهتر است كه از اين دو طرح استفاده كرد. كه از بين اين دو طرح اول از لحاظ تاخير و مساحت ارجحيت دارد؛ ولي در مواردي كه كاربرد ما محدود به بكارگيري طرح سوم يا چهارم مي شود بايد ديد كه كداميك از دو پارامتر تاخير و مساحت براي ما مهم است و پس از آن به ترتيب طرح سوم يا چهارم برگزيد.

# ۸- مراجع

[1] D.E. knuth, the Art of Computer Programming Vol.2/Seminumerical Algorithms, Third Edition, 1998.

[2] P.L. Montgomery, "Modular Multiplication without trival division", Mathematics of Computation, 44(170); 519-521, April 1985

[3] A.A. Hiasat, "New Efficient Structure for a Modular Multiplier for RNS", IEEE Transaction on Computers, Vol 49,pp. 170-174 Feb 2000

[4] A.A. Hiasat, "RNS Arithmatic Multiplicr for Medium and Large Moduli", IEEE transaction on Circuit and systems-11: Analog and digital Signal processing," vol. 47, No.q, pp 931-940, sep 2000

[5] D.Pearson, "A parallel implementation of RSA", Proceedings in the symposium on Computer, Arithmatic, pp 110, July 1996.

[6] K.C posch, R Posch, "Residue Number system: a key to parallelism in public key cryptography." IEEE Transaction, Vol.32, no.2, pp:332-335, July 1992.

[7] G.A. Jullien, "Implementation of Multiplication, Modulo a Prime Number, with Application to Theoritic Transforms," IEEE Transaction on Computers Vol.29,no.10,pp 899-905, oct 1980.

[8] M.soderstand and Cvernia, "A High speed low-cost modulo P, Multiplier with RNS Arithmatic Application", Proc. IEEE, Vol 68, pp. 529-532, Apr. 1980

[9] D.Radhakrishan and Y.Yuan, "Novel Approches to the design of VLSI RNS Multipllier", IEEE Transaction on Circuits and systems-II; Analog and digital signal processing. Vol 39 pp 52-57, Jan 1992.

[10] M Dugdale, "Residue Multipliers Using Factor Decomposition," IEEE Transaction on Circuits and systems, II: Analog and Digital signal Processing, Vol 41, pp 623-627. Sept 1994.

[11] A.S. Ramnarayan, “Practicul Realization of mod p,p Prime Multipliers,” Electronic Letters. Vol. 16, pp 466-467, June 1980

[12] F.J. Taylor, “A VLSI Residue Arithmatic Multiplier.” IEEE Trans. Computers, Vol 341, no.6, pp.540-546, June 1982.

[13] A.A. Hiasat, “A memory-less mod (2n+1) Residue Multiplier,” Electronic Letters, Vol. 28, pp. 414-415, Jan 1991

[14] C.D, walter, “Systolic Modular Multiplier,” IEEE Trans computers, Vol. 42 no.3. pp. 375-378. Mar 1993

[15] E.Di Claudio, F Piazza and G.Orlandi, “Fast Combinational RNS multiplier for DSP Applications, “IEEE Iran. Computers, Vol.44 no.5, pp. 624-633. May 1985.

[16] 6.Alia, E.Martinelli, “A VLSI Modulo m multiplier,” IEEE Irans on Circuits and system II Analog and Digital Signal Processing, vol 42, pp 725-729, Nov. 1995.

[18] A wrzyszcz, D Milford and E.Duglas, “A new Approach to Fixed-Coeffient inner product over finite Rings,” IEEE Trans computer, Vol 47, no.7, pp 760-776, July 1998.

[19] Si Piestrak “Design of residue generators and Multioperand modular adders using carry-save Adders, “IEEE Trans. Comput. Vol. 43 pp. 6877. Jan 1994.

[20] M. Soderstrand, M. A. W. Jenkins, G. Jullien, and F. Taylor, Eds., Residue Number System Arithmetic: Modern Applications in Digital Signal Processing. Piscataway, NJ: IEEE Press, 1986.

[21] N. Szabo and R. Tanaka, Residue Arithmetic and Its Applications to Computer Technology. New York: McGraw Hill, 1967.

[22] A. S. Ramnarayan, “Practical realization of mod p,p prime multiplier” Electron. Lett., Vol. 16, pp. 466-467, June. 1980.

[23] M. Soderstrand and C. Vernia, "A high-speed low-cost modulo $P_i$ multiplier with RNS arithmetic application," Proc. IEEE, vol. 68, pp. 529-532, Apr. 1980.

[24] G. A. Jullien, "Implementation of multiplication, modulo a prime number, with applications to theoretic transforms," IEEE Trans comput., Vol. C-29, pp. 899-905, Oct. 1980.

[25] F. J. Taylor, " Large moduli multipliers for signal processing," IEEE Trans. Circuits Syst., Vol. CAS-28, pp. 731-436, July 1981.

[26] F. J. Taylor, "A VLSI residue arithmetic multiplier," IEEE Trans Comput., Vol. C-31, pp. 540-546, June 1982.

[27] F. J. Taylor and Huandg, "An autoscale residue multiplier," IEEE Trans. Comput., Vol. C-31, pp. 321-325, Apr. 1982.

[28] A. Hiasat, "A memoryless mod $(2^n \pm 1)$ residue multiplier," Elecron. Lett., Vol. 28, pp. 414-415, Jan. 1991.

[29] M. Dugdale, "Residue multipliers using factored decomposition," IEEE Trans. Circuits Syst. II, Vol, 41, pp. 623-627, Sept. 1994.

[30] A. Haisat, "Semi- Custom VLSI design for RNS multipliers using combinational logic approach," in Proc. 3rd IEEE Int. Conf. Electronics. Circuits and Systems (ICECS'96), Vol. 2, Oct. 1996, pp. 935-938.

[31] D. Radhakrishnan and Y. Yuan, "Novel approaches to the design of VLSI RNS multiplier," IEEE Trans. Circuits Syst. II, Vol. 39, pp. 52-57, Jan. 1992.

[32] Markus A. Hitz and Erich Kaltofen. Integer division in residue number systems. IEEE Transactions on Computers, 44(8): 983-989, August 1995.

[33]http://www.iis.ee.ethz.ch/zimmi/publication/comp-arith-otes.ps.gz

[34] P. SINHA, "Fast Parallel Algorithms for Binary Multiplication and Their Implementation on Systolic Architectures," IEEE Transactions on Computers, Vol. 38, No.3, March 1989.

[35] K.C.Posch, R.Posch, “Modulo Reduction In Residue Number Systems,” IEEE Transactions on Parallel and distributed systems, Vol.6, No. 5, May 1995.

[36] R. Rivest, A. Shamir, and L. Adleman, “A method for obtaining digital signatures and public-key cryptosystems,” Commun. ACM, pp. 120-126, 1978.

[37] K. C. Posch and R. Posch, “Approaching encryption at ISDN speed using partial parallel modulus multiplication,” Microprocessing and Microprogramming. Amsterdam, The Netherlands: North-Holand, 1990, Vol. 29, pp. 177-184.

# ضمیمه

# MOMA

## جمع کننده پیمانه اي چند عملوندي (MOMA)

فرض کنید $X_K(x_{ka-1},...,x_{(k-1)a}),X_1(x_{a-1},...,x_1,x_0)$ ، k عدد به پیمانه A باشند. جمع کننده k عملوندي به پیمانه A مقدار زیر را محاسبه مي کند:

$$[X]_A=\left[\sum_{i=0}^{k} x_i\right]_A$$

که هم ارز است با

$$[X]_A=\left[\sum_{j=0}^{a}(\sum_{i=0}^{k} x_{ia+j}).[2^j]_A\right]_A$$

روش پیشنهادي در ابتدا جمع زیر را انجام مي دهد:

$$\left[\sum_{j=0}^{a}(\sum_{i=0}^{k} x_{ia+j}).[2^j]_A\right]_{2^{P(A)}-1}$$

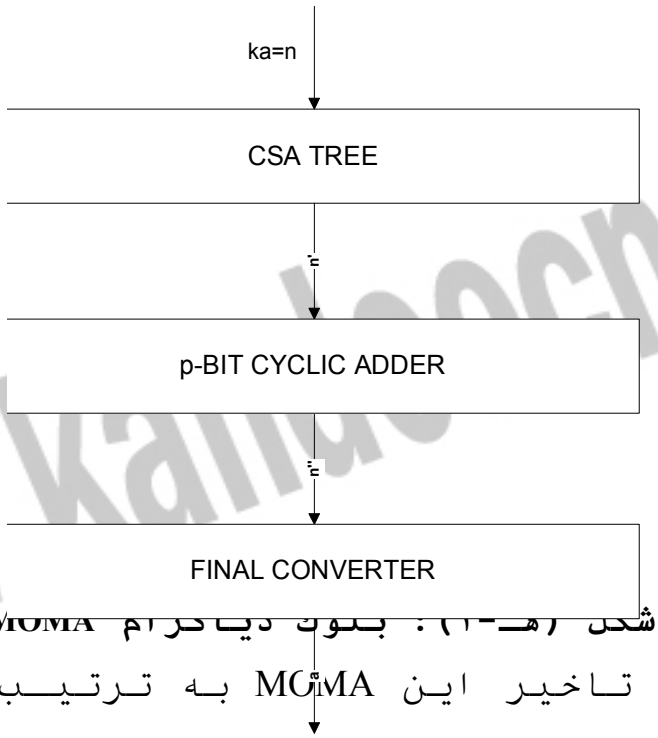
که در آن P(A) بصورت زیر تعریف مي شود:

$$P(A)=\min\{j\mid j>0 \text{ and } [2^j]_A=1\}$$

عملیات جمع بالا به کمك یك شبکه CSA با (End Around Carry) EAC انجام مي گیرد و سپس محاسبه نهایي زیر با یك جدول انجام مي گیرد.

$$\left[\left[\sum_{j=0}^{a}(\sum_{i=0}^{k} x_{ia+j}).[2^j]_A\right]_{2^{P(A)}-1}\right]_A$$

بلوك دياگرام طرح فوق به شكل زير مي باشد.




شكل (هـ-۱): بلوك دياگرام MOMA

مساحت و تاخير اين MOMA به ترتيب به شرح زير است:

$$A = (k-1)a - m - 1 + 2^{m+1}$$

$$T = 2(k + a - 2) + 2m$$

در روابط بالا k تعداد عملوندها، a تعداد بيت هر عملوند و m بصورت زير است.

$$m = \lceil \log k(A-1) \rceil$$