محاسبه توابع مثلثاتی توسط آرایه های سیستولیک و الگوریتم CORDIC

شادرخ سماوی* و روشنک کلیشادی**

دانشکدهٔ مهندسی برق و کامپیوتر، دانشگاه صنعتی اصفهان

(دریافت مقاله: ۷۸/۱۰/۱۳ – دریافت نسخه نهایی: ۷۹/۱۲/۱٤)

چکیده : *توابع مثلثاتی از کاربردیترین توابع در پردازش سیگنالهای دیجیتال‌اند. طرح ارائه شده در این مقاله توابع مثلثاتی را با استفاده از آرایه های سیستولیک[1]، محاسبه می کند . روش به دست آوردن این توابع برای زاویه θ بر اساس الگوریتم CORDIC[2] است. سلول ساده و استانداردی که برای شبکهٔ سیستولیک مطرح شده، نهایتاً با توجه به ورودیهای خاص سلولها ، بهینه شده است. واحد کنترل و حافظهٔ ROM از اجزای اصلی هر مدار CORDIC هستند و فضای بزرگی را اشغال می‌کنند . سخت افزار طراحی شده از این واحدها استفاده نمی‌کند به همین دلیل طرحی ساده و قابل گسترش است.*

واژگان کلیدی : *آرایه های سیستولیک - الگوریتم CORDIC - توابع مثلثاتی ، FPGA*

Computation of Trigonometric Functions by the Systolic Implementation of the CORDIC Algorithm

Sh. Samavi and R. Kelishadi

Department of Electrical Engineering, Isfahan University of Technology, Isfahan, Iran

Abstract: *Trigonometric functions are among the most useful functions in the digital signal processing applications. The design introduced in this paper computes the trigonometric functions by means of the systolic arrays. The method for computing these functions for an arbitrary angle, θ, is the CORDIC algorithm. A simple standard cell is used for the systolic array. Due to the fixed inputs, in some cases, a number of the cells are optimized. The control unit and a read only memory are the essential parts of any CORDIC implementation. The introduced hardware does not use any of these two structures, which makes it a simple and expandable design.*

Keywords: *Systolic arrays, CORDIC algorithm, Trigonometric functions, FPGA*

۱– مقدمه

انتخاب معماری مناسب برای هر سیستم الکترونیکی وابستگی شدیدی به تکنولوژی پیاده سازی آن دارد. در تکنولوژی VLSI طرحهای خوب باید مدولار، یعنی دارای تنوع سلولی کم باشند. منظم بودن اتصال سلولها به یکدیگر و تکرار پذیری آن نیز مورد قابل توجهی است. دیگر نکته ای که

---

* – استادیار ** – کارشناسی ارشد

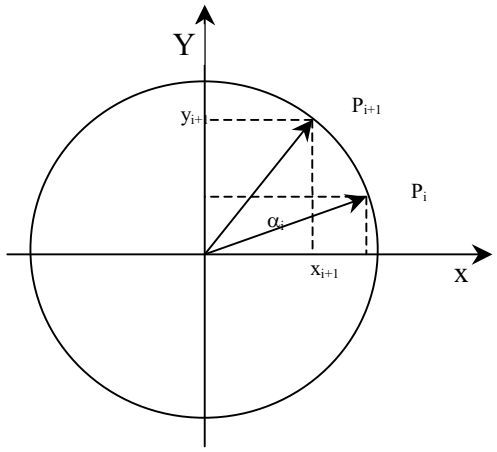
باید رعایت شود کوتاهتر شدن خطوط ارتباطی بین سلولی است . اگر این ارتباط بین سلولهای مجاور باشد، خطوط ارتباطی کوتاهتر شده و تأخیرهای ناشی از آنها کاهش می یابد. معماری آرایه‌های سیستولیک به عنوان راه حلی برای مسائل بالاست . این معماری اولین بار توسط کونگ مطرح و به مرور زمان تکمیل شده است [۱و۲].

یک سیستم سیستولیک، ساختاری توری مانند از عناصر پردازشیِ یکسان[۳] متصل به هم است که هر یک قادر به انجام چند عمل ساده‌اند. داده ها که شامل عناصر ورودی و نتایج میانی هستند ، بین سلولها جریان می یابند و ارتباط با محیط خارج تنها در سلولهای حاشیه‌ای صورت می گیرد . طرزکار آرایه های سیستولیک کاملا شبیه ضربان قلب است که داده ها به جای خون به عناصر پردازشی وارد می شوند . هر سلول می‌تپد و نتیجهٔ خروجی را به سلول بعدی می‌دهد . این معماری برای دستهٔ گسترده‌ای از محاسبات که در آن عملیات روی هر فقره داده به صورت تکراری انجام می شود امکانپذیر است .

توابع مثلثاتی کاربردهای مهندسی متعددی دارند که می‌توان از سیستمهای رادار ، سیستمهای کنترل اتوماتیک و سیستمهای پردازش صوت و تصویر نام برد . در بسیاری از کاربردها مانند INS [۴] نیاز به محاسبهٔ سینوس و کسینوس زوایا به صورت مکرر و مستمر است . این کار باید سریع و دقیق انجام شود و اهمیت آن به قدری است که تراشه‌های آنالوگی[۵] به این منظور طراحی شده است . روشهای مختلفی برای محاسبه سخت افزاری توابع مثلثاتی وجود دارد .روش جستجو در جدول، روش تقریب چند جمله ای با استفاده از بسط تیلور، ترکیبی از دو روش قبل و روش رقم به رقم[۶] با استفاده از الگوریتم CORDIC ، چند نمونه از آنهاست . الگوریتم CORDIC روشی کارامد برای محاسبهٔ توابع مثلثاتی است . این الگوریتم ابتدا توسط ولدر [۶] معرفی شد. این الگوریتم قادر به محاسبه توابع مثلثاتی و متعالی[۷] با استفاده از تکرار شیفت و جمع یا تفریق روی عناصر است. عملیات تکرارپذیر و یکنواخت برای پیاده سازی توسط آرایه های سیستولیک مناسب می باشند [۳و۴]. به همین دلیل ، الگوریتم CORDIC را کاندید مناسبی برای آرایه‌های سیستولیک یافتیم.

۲- توصیف الگوریتم CORDIC

نقطهٔ P_i با مختصات (x_i, y_i) را به اندازهٔ زاویه α_i دوران می‌دهیم و به نقطهٔ P_{i+1} می‌رسیم [۵]. این مطلب در شکل (۱) نمایش داده شده است. هدف، محاسبهٔ مختصات P_{i+1} برحسب مختصات P_i است.



شکل ۱- دوران pi به اندازهٔ زاویه α_i

مختصات نقطهٔ P_{i+1} یعنی (y_{i+1} و x_{i+1})، به صورت زیر به دست می‌آید

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \begin{bmatrix} \cos\alpha_i & -\sin\alpha_i \\ \sin\alpha_i & \cos\alpha_i \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} \qquad (۱)$$

در صورتی که بخواهیم تنها از شیفت و جمع یا تفریق استفاده کنیم معادلهٔ (۱) را بر $\text{Cos}\ \alpha_i$ تقسیم می‌کنیم ، در نتیجه معادله‌های زیر به دست می آیند

$$\frac{x_{i+1}}{\text{Cos}\ \alpha_i} = x_i - y_i.\tan\alpha_i = x'_{i+1} \qquad (۲)$$

$$\frac{y_{i+1}}{\text{Cos}\ \alpha_i} = y_i + x_i.\tan\alpha_i = y'_{i+1} \qquad (۳)$$

فرض کنید زاویه ای که می خواهیم تابع مثلثاتی را برای آن محاسبه کنیم θ باشد .این زاویه را می توان از جمع تعدادی زوایای کوچکتر α_i تشکیل داد ، یعنی $\theta \approx \sum_{i=0}^{h-1} d_i.\alpha_i$.

برای سهولت در انجام محاسبات مقدار α_i را بـه صـورت $\alpha_i = \tan^{-1}2^{-i}$ انتخاب می کنیم . بنـابراین چنانچـه محاسـبات در مبنای دو انجام شود، عباراتی که باید در $\tan\alpha_i = 2^{-i}$ ضرب شوند فقط شیفت داده می شوند .

مقدار d_i می تواند ۱ یا ۱- باشد . همـواره بـا $d_0=1$ و $\alpha_0 = 45$ شروع می کنیم و بر حسب اینکه θ از ٤٥ درجه کـوچکـتر یـا بزرگتر باشد ، d_i مرحله بعدی ۱ یا ۱- انتخاب می شود . بعد از h بار تکرار، محاسبات به سمت θ مورد نظر همـگرا می‌شـود. می توان عبارات زیر را از معادله‌های (۲) و (۳) به دست آورد

$$x'_h = \frac{x_0}{\cos\alpha_0.\cos\alpha_1.....\cos\alpha_{h-1}}.\cos(\alpha_0+\alpha_1+...+\alpha_{h-1}) \qquad (٤)$$

$$y'_h = \frac{x_0}{\cos\alpha_0.\cos\alpha_1.....\cos\alpha_{h-1}}.\sin(\alpha_0+\alpha_1....+\alpha_{h-1}) \qquad (٥)$$

اگر $x_0 = \cos\alpha_0.\cos\alpha_1.....\cos\alpha_{h-1}$ انتخـاب شود. آن گاه

$$x'_h = \cos\theta \quad و \quad y'_h = \sin\theta \qquad (٦)$$

به دنبال نشان می‌دهیم که x_0 مقداری ثابت است و تابعی از θ نیست

$$K = \frac{1}{\cos\alpha_0.\cos\alpha_1...\cos\alpha_{h-1}} = \prod_{i=0}^{h-1}\left(\frac{1}{\cos\alpha_i}\right) = \prod_{i=0}^{h-1}\left(1+\frac{\sin^2\alpha_i}{\cos^2\alpha_i}\right)^{\frac{1}{2}} \qquad (٧)$$

$$K = \prod_{i=0}^{h-1}(1+\tan^2\alpha_i)^{\frac{1}{2}} = \prod_{i=0}^{h-1}(1+(2^{-i})^2)^{\frac{1}{2}} = \prod_{i=0}^{h-1}\sqrt{1+2^{-2i}} \qquad (٨)$$

در محاسبهٔ K ، با در نظر گرفتن ۹ رقـم اعشـار، بـرای مقـادیر $i \geq 16$ عبـارت $\sqrt{1+2^{-2i}}$ برابـر ۱ می‌شـود، بنـابراین مقـدار حداکثر را برای i همان شانزده در نظر می‌گیریم

$$K = \prod_{i=0}^{16}\sqrt{1+2^{-2i}} = 1.646760255 \qquad (٩)$$

با توجه به معادلهٔ (۹) مقدار x_0 وقتی تعـداد مراحـل تکـرار ١٦ است به صورت زیر مشخص می شود

$$x_0 = \frac{1}{K} = \frac{1}{1.646760255} = 0.6072252936 \qquad (١٠)$$

متغیر دیگری که به کار برده می شود z_i است که مقادیر زاویه را با آن تغییر می دهیم . مقدار اولیـه z_i یعنـی z_0 همـان زاویـه θ است. با توجه به مطالب ذکر شده، معادله‌های زیـر ، بعـد از n مرحله تکرار، $\sin\theta, \cos\theta$ را تولید می‌کنند

$$x_{i+1} = x_i - d_i \cdot y_i \cdot 2^{-i} \qquad (١١)$$

$$y_{i+1} = y_i + d_i \cdot x_i \cdot 2^{-i} \qquad (١٢)$$

$$z_{i+1} = z_i - d_i \cdot \tan^{-1} 2^{-i} \qquad (١٣)$$

در این معادله‌ها d_i وابسته به علامت z_i است. یعنی اگر $z_i<0$ آن گاه ، $d_i = -1$ و اگر $z_i \geq 0$ آن گاه ، $d_i = ١$.

معادله‌های (۱۱) تا (۱۳) ، معادله‌های اصلـی CORDIC هسـتند که بامقادیر اولیه ذکـر شـده در معادلـهٔ (١٤) محاسـبات شـروع می‌شود و بعد از h مرحله تکرار x_h و y_h بـه ترتیـب $\cos\theta$ و $\sin\theta$ را نشان می‌دهند.

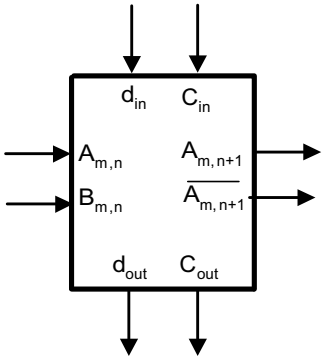
$$x_0 = \frac{1}{K} \quad , \quad y_0 = 0 \quad , \quad z_0 = \theta \qquad (١٤)$$

در الگوریتم CORDIC زاویه θ به بـاینری تبدیـل شـده و بـه عنوان z_0 وارد مـی شـود ، $y_0=0$ و $x_0=0.1001101101110011$ که معادل عدد به دست آمده از معادلـهٔ (۱۰) اسـت . دُگِت [۷] نشان داده است که اعداد n بیتی در سیسـتم بـاینری، بعـد از n مرحلهٔ تکرار جواب دقیقی را خواهند داد.

۳- پیاده سازی

در این بخش راه حلی سـخت افـزاری را بـرای اجـرای سیستولیک الگوریتم CORDIC ارائه می دهیم . داده ها در مدار پیشنهادی ١٦ بیتـی و بـا نمـایش ممـیز ثـابت[۸] در نظـر گرفتـه شده‌اند. برای z ، محل قرار گرفتن ممیز بین بیتهای هشتم و نهم است، یعنی ۸ بیت ارقام صحیح و ۸ بیت ارقام اعشاری داریـم . x و y دو عدد کاملا اعشاری هستند .

عنصر پردازشی در شکل (۲) ، اساس ساختمان آرایه سیستولیک طراحی شده را نشان می‌دهد.



$n=1,2,....,16$

$m=1,2,....,16$

$d_{in}=d_{out}$

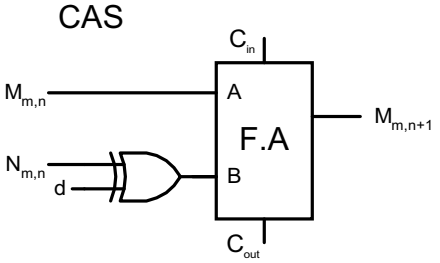
if $d_{in}=0$ then $A_{m,n+1}=A_{m,n}+B_{m,n}$

if $d_{in}=1$ then $A_{m,n+1}=A_{m,n}-B_{m,n}$

شکل ۲ ـ عنصر پردازشی اصلی ساختمان آرایه سیستولیک

در شکل بالا m و n بـه ترتیـب ردیـف پـردازش و مرحلـه پردازش هستند. اعداد ١٦ بیتی بـوده لـذا ١٦ مرحلـه پـردازش داریم یعنی $n=1,....,16$. هر سلول قادر به پردازش یک بیـت است. به این ترتیب بـرای پـردازش ٣ عـدد ١٦ بیتـی x، y و z احتیـاج بـه ٤٨ سـلول در هـر مرحلـه داریـم . بنـابراین شـبکهٔ سیستولیک مورد نظر دارای ٤٨ ردیف و ١٦ ستون است.

هر سلول پردازشی در واقع یک واحد اسـتاندارد CAS[9] اسـت که مدار داخلی آن در شـکل(٣) آمـده اسـت . ١٦ ردیـف اول (گروه z) برای پردازش z ، از ردیـف ١٧ تـا ردیـف ٣٢ (گروه x) برای پردازش x و از ردیف ٣٣ تا ردیف ٤٨ (گروه y) بـرای پردازش y به کار رفته اند .در سلولهای اولیـن ردیـف هـر گروه d_{in} و C_{in} بـه یکدیـگر اتصـال دارنـد. بـرای سـتون اول زاویـه $z_0=\theta$ وارد می شود. این زاویه مثبت است، لذا در ستون اول برای به دست آوردن x و z تفریـق و بـرای بـه دسـت آوردن y جمع داریم .



شکل ۳ ـ شمای کلی یک سلول CAS

یعنی درستون اول برای x و z، $d_{in}=1$ و برای y ، $d_{in}=0$ منظـور شده است، بنابراین مقادیر اولیه بـه شـکل معادلـه‌های (١٥) بـه ستون اول وارد می‌شوند.

z: $d_{in}= C_{in} = 1$

$z_0 = \theta = A_{16,1} \ldots. A_{9,1} . A_{8,1}\ldots.. A_{1,1}$

$\tan^{-1}2^{-0} = 45 = 00101101.00000000 =$

$B_{16,1} \ldots. B_{9,1} . B_{8,1}\ldots.. B_{1,1}$

x: $d_{in}= C_{in} = 1$

$0.A_{32,1} \ldots.. A_{17,1} = x_0 = 0.1001101101110011$

$0.B_{32,1} \ldots.. B_{17,1} = y_0 = 0.0000000000000000$

y: $d_{in}= C_{in} = 0$

$0.A_{48,1} \ldots.. A_{33,1} = y_0 = 0.0000000000000000$

$0.B_{48,1} \ldots.. B_{33,1} = x_0 = 0.1001101101110011$

(١٥)

همان‌طور که در شکل(٢) مشخص است در هر سـلول، مکمـل سـلول پردازشـی نـیز در دسـترس اسـت . در ردیـف شانزدهم گروه اول از این خروجی مکمل اسـتفاده شـده اسـت. این ردیف جایی است که بیت علامت z محاسبه می شـود . بـا توجه به روابط d_i ، مکمـل بیـت علامـت z ، در مرحلـهٔ بعـد کنترل کنندهٔ جمع یا تفریق برای z و x اسـت و خـود آن کنـترل کننـدهٔ جمـع یـا تفریـق y اسـت. بنـابراین اگر در هـر مرحلـه، خروجی $\overline{A_{16,n+1}}$ را به d_{in} سلول اول و سلول هفدهم (بـرای z و x) و خروجی $A_{16,n+1}$ ر ا به d_{in} سلول سی و سـوم (بـرای y) در مرحلهٔ بعد وصل کنیم، به این ترتیب ستون بعد توسط بیـت علامت ستون قبلی کنترل می‌شود.

با اتصال d_{in} به ۱۶ بیت عدد، پردازش یکسان روی هر ۱۶ بیت انجام می شود. این انتشار توسط اتصال d_{out} سلول mام به d_{in} سلول m+1 ام در یک ستون انجام می گیرد . ورودیهای $A_{m,n}$ از خروجیهای $A_{m,n+1}$ مرحلهٔ قبل مستقیماً وارد می‌شوند . هر بیت که پردازش می شود ، بیت نقلی خروجی آن $(C_{out})_m$ به بیت نقلی ورودی سلول ردیف بعد $(C_{in})_{m+1}$ منتقل می شود. ورودیهای B برای گروه z ، در هر مرحله ، $\tan^{-1} 2^{-i}$ برای $i = 0,\ldots,15$ هستند که به صورت باینری محاسبه شده و به $B_{m,n}$ (m=1,2,..,16) وارد می‌شوند.

خروجیهای گروههای x و y هر مرحله باید شیفت داده شوند و به B های مرحلهٔ بعد وارد شوند . شیفت داده شدهٔ x به‌Bهای مرحلهٔ بعد گروه y یعنی سلولهای ردیف ۳۳ تا ۴۸ وصل می شود و شیفت داده شدهٔ y به ورودیهایB مرحلهٔ بعد گروه x یعنی سلولهای ردیف ۱۷ تا ۳۲ اتصال می‌یابد .

شکل (٤) بخش کوچکی از آرایه که متعلق به گروه x ، ردیفهای ۲۰ تا ۲۳ و مراحل هفتم تا یازدهم است را نشان می‌دهد. سیگنال ساعت به تمام Latch های مدار متصل است. این سیگنال را با خطوط نقطه چین و Latch ها را با دایره‌های کوچک در شکل (٤) نمایش داده‌ایم. همه ورودیهای سلولها قبل از ورود از یک Latch عبور می کنند . از مدار Latch برای جلوگیری از تداخل داده های ورودی سلولها استفاده می شود. بعد از اولین پالس ساعت،مقادیر اولیه و زاویه وارد ستون اول می شود. درسلولهای گروه zو x عمل تفریق و در سلولهای گروه y عمل جمع صورت می‌گیرد. خروجی هرمرحله وارد Latch شده ، $\overline{A_{16,2}}$ توسط یک Latch وارد d_{in} سلول اول ستون دوم می شود. همین عدد توسط یک Latch دیگر به d_{in} سلول هفدهم ستون دوم وارد می شود . بعد از اعمال پالس دوم همه خروجیها به ورودیهای مرحلهٔ دوم می روند ومرحلهٔ اول، زاویه دیگری ومقادیر اولیه را می گیرد. پردازش روی عناصر هر دوستون انجام می شود و به همین ترتیب ادامه می یابد تا محاسبات مربوط به زاویه θ، در خروجی مرحله شانزدهم ظاهر شود . خروجیهای $A_{m,17}$ جواب نهایی هستند، معادله‌های (١٦) و (١٧) .

$$x_{16,16}x_{15,16}\ldots. \; x_{1,16} = A_{32,17} \;\; A_{31,17} \ldots.. \; A_{17,17} = \text{Cos } z_0 \qquad (١٦)$$

$$y_{16,16}y_{15,16}\ldots. \; y_{1,16} = A_{48,18} \;\; A_{47,17} \ldots. \; A_{33,17} = \text{Sin } z_0 \qquad (١٧)$$

بعد از گذشت شانزده پالس اولیه، مانند هر ساختار سیستولیک، با هر پالس ساعت نتیجهٔ محاسبات برای یک زاویه تولید می‌شود. عباتهای زیر نتیجهٔ محاسبات را برای زاویه $\theta = 42$ نشان می‌دهند

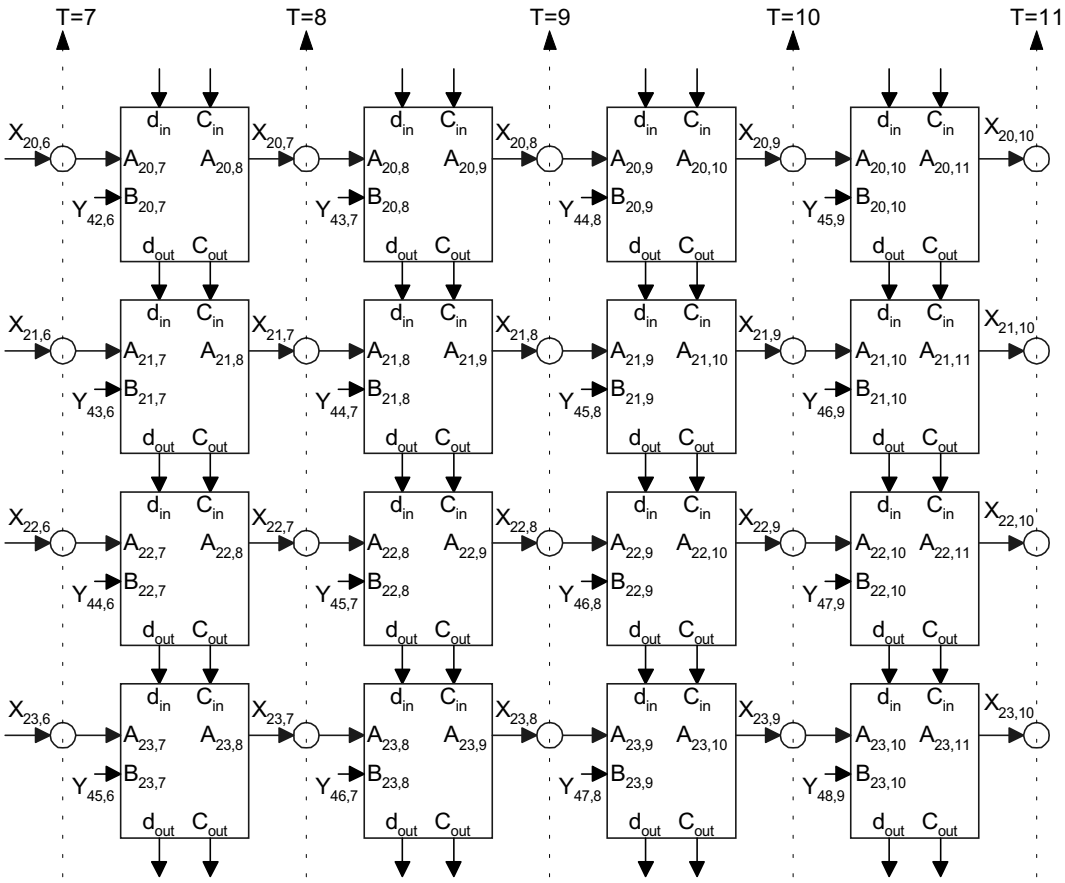
$$x = \cos 42 = 1011111000111101 = 0.743118284$$
$$y = \sin 42 = 1010101101001011 = 0.669113294 \qquad (١٨)$$

در مقایسه با نتایجی که نرم افزار matlab تولید می‌کند، برای Cos 42 ، 0.00357% و برای Sin 42 ، 0.00259% خطای نسبی داریم . واضح است که اگر محاسبات توسط اعداد با تعداد بیت بیشتری انجام گیرد، کاهش خطا محسوستر است.

٤- بهینه سازی

در این بخش قصد بهینه سازی طرح را داریم . منظور از بهینه سازی در اینجا ، ساده کردن بعضی از سلولهای مدار است که دارای ورودیهای ثابت‌اند . واضح است که این کار، یکنواختی ساختار را به هم می زند و تاثیر چندانی در کم کردن تأخیر مدار نخواهد داشت زیرا در ساختارهای سیستولیک پریود ساعت تابعی از تاخیر کندترین مرحله است .

مدار شکل (٣) را در سطح ترانزیستور به طرق مختلف می توان پیاده سازی کرد . تعداد ترانزیستورهای مصرف شده تابعی از تکنولوژی به کار رفته ، خواهد بود . به عنوان مثال، شکل (٥) یک نوع پیاده سازی را در تکنولوژی CMOS نشان می‌دهد . حتی در تکنولوژی CMOS هم همین مدار را به طرق متفاوتی طراحی کرده اند که بعضا تعداد ترانزیستور بیشتر یا کمتری نسبت به شکل (٥) دارند . هدف از آنچه در ادامه می‌آید این است که مدار پایه شکل (٣) را با هر ساختار



شکل ٤- بخشی از شبکهٔ سیستولیک

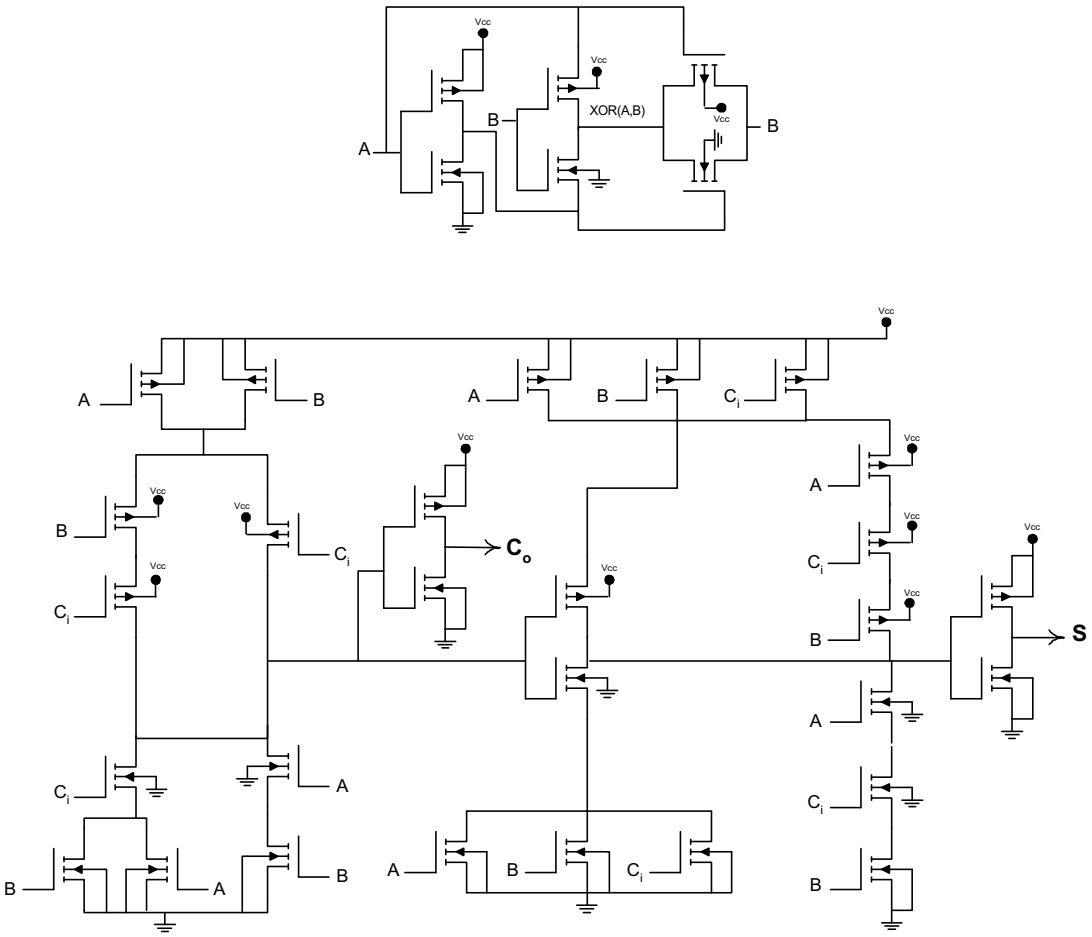
ترانزیستوری که داشته باشد می توان ساده کرد . به همین منظور مدار شکل (٥) را به عنوان مبنا فرض می کنیم و بهینه سازی را بر اساس تعداد ترانزیستوری که از این شکل کم می‌شود به پیش می بریم . برای پیاده سازیهای دیگر نیز می‌توان همین بحث را تعمیم داد .

آرایه سیستولیک این طرح دارای ٧٦٨=١٦×٤٨ سلول پردازشی و تعداد Latch به کار رفته برای هر ستون ،٨٣ عدد است. پس برای کل ١٦ ستون ١٣٢٨=١٦×٨٣ عدد Latch در مدار داریم . با توجه به اینکه سلول پردازشی از ٣٤ ترانزیستور و مدارLatch از ١٨ ترانزیستور ساخته شده است ، بنابراین مدار از تعداد ٥٠٠١٦ ترانزیستور ساخته می‌شود. بهینه سازی را در چهار گام توضیح می‌دهیم.

گام اول بهینه سازی

همان‌طور که در شکل (٣) مشخص است ، $B_{m,n}$ و d_{in} ورودیهای یک گیت XOR هستند. اگر $B_{m,n}=0$ باشد، حاصل XOR آن با d_{in}، خود d_{in} خواهد بود و اگر $B_{m,n}=1$ باشد، حاصل XOR آن با d_{in} مکمل d_{in} ($\overline{d_{in}}$) خواهد بود. بنابراین، در حالت اول ($B_{m,n}=0$) ،می توان XOR را حذف کرد و مستقیما d_{in} را به ورودی F.A وصل کرد، این سلول را OCAS1[١٠] می نامیم. این سلول ٢٨ ترانزیستور دارد و به علت حذف XOR، ٦ عدد از ترانزیستورها کم می شود . یعنی با جایگزینی OCAS1 ، ٦ عدد ترانزیستور کمتر استفاده می شود.

در حالت دوم ($B_{m,n}=1$) می توان XOR را حذف کرد و d_{in} را بعد از عبور دادن از یک NOT به ورودی B جمع کننده



شکل ۵- یک ساختار ترانزیستوری برای مدار CAS

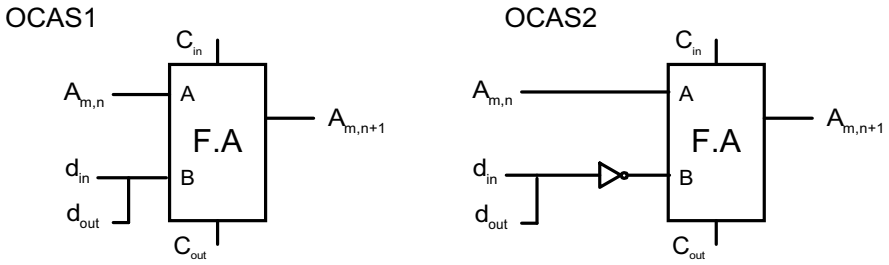
وصل کرد. این سلول را OCAS2 می نـامیم. ایـن سـلول از ۳۰ ترانزیستور ساخته شده است، بنابراین جـایگزینی OCAS2 ، ٤ ترانزیستور از مدار کم می کند.

ورودیـهای $B_{n,m}$ بـــرای گروه z همـــگی $\tan^{-1} 2^{-i}$ و مقادیری ثـابت‌اند. سـتون اول z در گام دوم بهینـه مـی شـود. جمعا تعداد صفرها از مرحلهٔ دوم تا مرحلهٔ شانزدهم ۱۹۱ عـــدد است. بنابراین ۱۹۱ سلول به OCAS1 تبدیل می شـود و ۱۱٤٦ ترانزیستور کاهش مـی یـابد . تعـداد یک‌هـا، از مرحلـهٔ دوم تـا مرحلهٔ شانزدهم، ٤۹ عدد است . بنابراین ٤۹ سلول بـه OCAS2 تبدیل شده و ۱۹٦ ترانزیستور از مدار کاسته می شود .

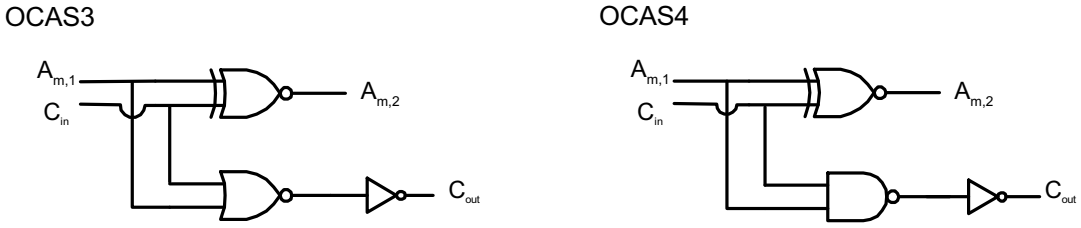
می دانیم که به ازای هر شیفت به راست، یـک صفـر در موضع پرارزش وارد می شـود. گروهـهای yو x در هـر مرحلـه باید شیفت شوند که از مرحله دوم این کار شروع می شود کـه این مرحله را در گام سوم بهینـه مـی کنیـم. تعـداد صفرهـا از مرحلهٔ سوم تا مرحلهٔ شانزدهم ۲۳۸ است که بـه همیـن تعـداد سلول به OCAS1 تبدیل می شود. بنابراین۱٤۲۸ ترانزیسـتور از مدار کاسته می شود و ۲۳۸ عدد Latch برای ورود $B_{m,n}$ از مدار حذف می شوند. در مجموع در گام اول، ۳ مورد بهینـه سـازی شد که در آن ۲۷۷۰ ترانزیستور وLatch۲۳۸ حذف شد.

گام دوم بهینه سازی

گام دوم بـرای ســاده ســازی ستــون اول گروه z اسـت. در این ستون ، $d_{in}=1$ است. می دانیم که اگر $B_{m,n}=0$ باشد ، سـلول OCAS1 و اگر $B_{m,n}=1$ باشد، سلول OCAS2 را می‌توان به کار برد. با توجه بـه سـاده سـازی گیتها ، توابــع (۱۹) و (۲۰) را



شکل ٦- سلولهای بهینه شده در گام اول



شکل ۷- سلولهای بهینه شده در گام دوم

و (۲۰) را برای $B_{m,n}=0$ و $B_{m,n}=1$ با $d_{in}=1$ داریم

$$Cout = A_{m,1} + C_{in}\ ,\ SUM = A_{m,1}.C_{in} + \overline{A_{m,1}}.\overline{C_{in}} \qquad (۱۹)$$

$$Cout = A_{m,1}.C_{in}\ ,$$

$$SUM = \overline{A_{m,1}}.C_{in} + A_{m,1}.\overline{C_{in}} \qquad (۲۰)$$

سلول OCAS3 و OCAS4 توابع بالا را پیاده سازی می کنند. این سلولها ۱۲ ترانزیستور دارند، یعنی با قرار دادن هر کدام ۲۲ ترانزیستور از مدار پایه کاسته می شود .

برای اولین سلول اولین ستون همواره $B_{1,1}=0$ است. بنابراین سلول OCAS3 را جایگزین آن می کنیم. پس از ساده سازی به علت یک بودن C_{in}، این سلول را حذف می کنیم و به ورودی $A_{1,2}$ مستقیما $A_{1,1}$ را وصل می‌کنیم. بیت نقلی ورودی (C_{in}) برای سلول دوم، ستون اول، نیز ۱ می شود. این سلول (سلول دوم، ستون اول) و شش سلول بعد از آن (سلول سوم تا سلول هشتم در ستون اول) همگی دارای $B_{m,1}=0, d_{in}=1, C_{in}=1$ هستند، پس شرایط بالا برای آنها نیز صادق است. یعنی مجموعاً ۸ سلول اول ستون اول را می‌توانیم حذف کنیم و $A_{m,2}$ را به $A_{m,1}$ وصل می کنیم و با این کار ۸ سلول اصلی و ۸ عدد Latch ورودی آنها را حذف کنیم. این کار ۲۷۲ ترانزیستور را از مدار کم می کند.

سلول نهم ستون اول دارای $C_{in}=1$ (حاصل از سلول هشتم) و $B_{9,1}=1$ است. بنابراین سلول OCAS4 را جایگزین آن می کنیم . با توجه به $C_{in}=1$ ، آن را حذف کرده و بیت نقلی خروجی سلول که همان $A_{9,1}$ است ، به C_{in} بعدی متصل می‌شود . $A_{9,1}$ را به یک NOT واردمی‌کنیم، خروجی آن را به $A_{9,2}$ وصل می‌کنم. در این حالت با حذف سلول اصلی و جایگزینی یک NOT ، ۳۲ ترانزیستور از مدار کاسته می شود.

در سلول شانزدهم ستون اول ورودیهای $B_{16,1}, A_{16,1}$ همواره صفرند. $B_{16,1}$که عدد ثابتی است ($\tan^{-1} 2^{-1}=45$)و نیز به دلیل قرارگیری زاویه ورودی بین صفر و ۹۰°، $A_{16,1}$ همیشه مثبت است، همواره بیت شانزدهم صفر دارند (بیت هشتم صحیح). از آنجا که $B_{16,1}=1$ است، پس یک سلول OCAS3 داریم که ساده سازی روی آن انجام می شود و خروجی $\overline{C_{in}}$ را داریم. در سلول شانزدهم محاسبه بیت نقلی خروجی ضرورتی ندارد. بنابراین سلو ل اصلی را حذف می‌کنیم و به جای آن یک NOT قرار می دهیم که ورودی آن C_{out} سلول پانزدهم است. خروجی این NOT را به $A_{16,2}$ اتصال می‌دهیم. این خروجی به عنوان d_{in} مرحله بعدی برای گروههای z و x است. ورودی این NOT را نیز به d_{in} برای گروهy مرحلهٔ بعد وصل می کنیم . با این کار ۳۲

ترانزیستور از مدار کاسته می‌شود. Latchهای ورودی این سلول نیز حذف می شود. پس ۱ عدد Latch از کل مدار کم می شود.

ده سلول این ستون با مطالب بالا بهینه شد، (m=1,2,3,4,5,6,7,8,9,16). از ٦ سلول باقی مانده ۳ عدد دارای $B_{m,1}=0$ هستند (m=10,13,15)، که با OCAS3 جایگزین می شوند و ٦٦ ترانزیستور از مدار کم می کنند و ۳ عدد دارای $B_{m,1}=1$ هستند(m=11,12,14) که هر سه با OCAS4 جایگزین می شوند و ٦٦ ترانزیستور را حذف می‌کنند. در مجموع در گام دوم ۹ عدد Latch و ٤٦٨ عدد ترانزیستور حذف می‌شود.

گام سوم بهینه سازی

در مرحلهٔ اول برای گروهx ، (۳۲و...وm=۱۷)، عمل تفریق و برای گروه y عمل جمع (٤٨و...وm=۳۳) داریم. برای x ورودیهای $A_{m,1}$ و $B_{m,1}$ به ترتیب x_0, y_0 هستند بنابراین

$$A_1 - B_1 = X_0 - Y_0 = X_0 - 0 = X_0 \qquad (۲۱)$$

و برای y ورودیهای $A_{m,1}$ و $B_{m,1}$ به ترتیب x_0, y_0 هستند یعنی

$$A_1 + B_1 = Y_0 + X_0 = 0 + X_0 = X_0 \qquad (۲۲)$$

با توجه به معادله‌های (۲۱) و (۲۲) ، خروجی x و y همان x_0 می شود ، بنابراین کل ۳۲ سلول مربوط به x و y ، (m=17,...,48) ، را در ستون اول حذف می کنیم. این حذف ٦٤ عدد Latch و ۱۰۸۸ترانزیستوررا از مدار کم می‌کند. ورودیهای $A_{m,2}$ برای x (m=17,...,32) و ورودیهای $A_{m,2}$ برای y (m=33,...,48)، هر دو x_0 هستند و ورودیهای $B_{m,2}$ برای هر دو سری (yوx) یکی شیفت یافته x_0 هستند، بنابراین

$$\begin{aligned} &A_{32,2}\ A_{31,2}\ ....\ A_{17,2} = 1001101101110011 \\ &B_{32,2}\ B_{31,2}\ .....\ B_{17,2} = 0100110110111001 \\ &A_{48,2}\ A_{47,2}\ .....\ A_{33,2} = 1001101101110011 \\ &B_{48,2}\ B_{47,2}\ .....\ B_{33,2} = 0100110110111001 \end{aligned} \qquad (۲۳)$$

به ازای حالتهای مختلف $A_{m,2}$ و $B_{m,2}$، سلول اصلی به سلولهای ساده تر تبدیل می شود.

برای $B_{m,2}=0$ و $A_{m,2}=0$ ، $B_{m,2}=0$ و $A_{m,2}=1$ ، $B_{m,2}=1$ و $A_{m,2}=0$ و نهایتا $B_{m,2}=1$ و $A_{m,2}=1$ به ترتیب معادله‌های (٢٤) تا (۲۷) را داریم .

$$Cout = d_{in}.C_{in}\ ,\ SUM = d_{in}\overline{C}_{in} + \overline{d}_{in}C_{in} \qquad (٢٤)$$

$$Cout = d_{in} + C_{in}\ ,SUM = d_{in}C_{in} + \overline{d}_{in}.\overline{C}_{in} \qquad (٢٥)$$

$$Cout = \overline{d}_{in}.C_{in}\ ,\ SUM = d_{in}.C_{in} + \overline{d}_{in}.\overline{C}_{in} \qquad (٢٦)$$

$$Cout = C_{in}.\overline{d}_{in}\ ,\ SUM = C_{in}.\overline{d}_{in} + \overline{C}_{in}.d_{in} \qquad (۲۷)$$

برای معادلهٔ (٢٤) سلول OCAS5 را داریم که از سلول اصلی، ۲۲ ترانزیستور کمتر دارد. در ستون دوم برای m =19,30,35,46 ، ورودیهای $B_{m,2}, A_{m,2}$، هر دو صفرند. با این جایگزینی ۸۸ترانزیستورو ۸ عدد Latch از سلولها کاسته می‌شود.

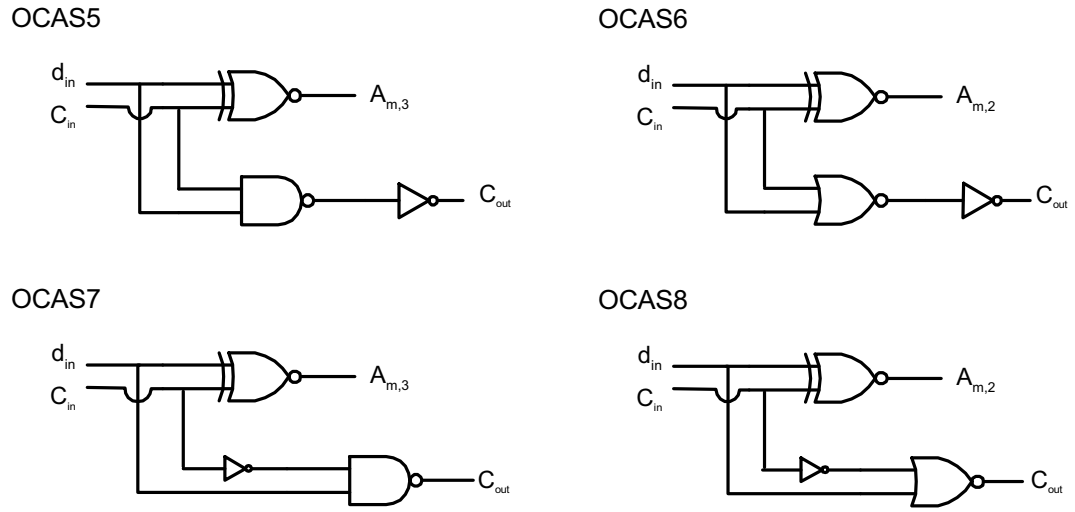
معادلهٔ (۲۵) سلول OCAS6 را نتیجه می دهد . در ستون دوم برای m=18,23,26,29,32,34,39,42,45,48 ، حالت $B_{m,2}=0$ و $A_{m,2}=1$ را داریم، هر سلول ۱۲ ترانزیستور داشته و با این ۱۰ مورد جانشینی، ۲۲۰ ترانزیستورو۲۰ Latch از مدار کم می شود .

معادلهٔ (٢٦) سلول OCAS7 را با۱۲ ترانزیستور نتیجه می‌دهد. در ستون دوم حالت $B_{m,2}=1$ و $A_{m,2}=0$ برای سلولهای m=20,24,27,31,36,40,43,47 وجود دارد ، پس ۸ سلول را می‌توان با OCAS7 عوض کرده و ۱۷٦ترانزیستور و۱٦Latch از سلولها را کم کنیم .

سلول OCAS8 معادلهٔ (۲۷) را می‌سازد که ۱۲ ترانزیستور دارد . ورودیهای $B_{m,2}$ و $A_{m,2}$ در ستون دوم برای m=17,21,22,25,37,38,41,44,28,33، هر دو ۱ هستند، پس با جایگزینی این سلولها۲۲۰ ترانزیستور و ۲۰ عدد Latch از سلولها کم می شود . در کل گام سوم تعداد۱۷۹۲ ترانزیستورو ۱۲۸ Latch از مدارکاهش می یابد .

گام چهارم بهینه سازی

در محاسبهٔ بیت شانزدهم x، y و z احتیاجی به محاسبه بیت نقلی نداریم . در گام دوم، سلول شانزدهم واقع در ستون اول، بهینه شد و در آن Coutرا دیگر محاسبه نکردیم. در گام سوم، سلول سی و دوم و چهل و هشتم ستون اول کلا حذف شدند. در ستون دوم سلولهای سی و دوم و چهل و هشتم دارای ورودیهای $B_{322}=0, B_{482}=0, A_{322}=1, A_{482}=1$ بودند که سلول OCAS6 جایگزین آنها شد. در این سلولها باید



شکل ۸- سلولهای بهینه شده در گام سوم

قسمت مربوط به تولید Cout را حذف کنیم ، بنابراین ۱۲ ترانزیستور از مدار کم می شود..

بقیه سلولهای مدار در ردیفهای شانزدهم، سی و سوم و چهل و هشتم OCAS1 یا OCAS2 هستند . در هر دو مورد از خروجی $\overline{Cout}$ استفاده نشده است و به این ترتیب یک NOT را می توانیم از آنها حذف کنیم. یعنی برای ستون دوم، سلول شانزدهم و در دیگر ستونها، سلولهای شانزدهم، سی و دوم و چهل هشتم، یک NOT حذف می شود. مجموعا ٤٣عدد NOT حذف می شود که ٨٦ترانزیستور از مدار می کاهد.پس در این گام ۹۸ ترانزیستور حذف می شود .

در کل چهار گام بهینه سازی ۳۷۵ مدار Latch حذف شد که ٦٧٥٠ ترانزیستور از مدار می کاهد ، تعداد ترانزیستور حذف شده نیز درطی این چهار مرحله ٥١٢٨ عدد است، بنابراین تعداد کل ترانزیستورهای حذف شده ١١٨٧٨ عدد است ،که ۲۳/۷۵ درصد کل ترانزیستورها ست.

۵- نتیجه گیری

در این مقاله یک روش سخت افزاری محاسبهٔ Sinو Cos به صورت سیستولیک ارائه شد. مدار ارائه شده قادر است در هر پالس ساعت یک زاویه را دریافت کرده و با رفتار خط لوله ای[11] در هر پالس همزمان Sin و Cos یک زاویه را تولید کند . بازدهی[12] این ساختار تابعی از بزرگی کلمات و دقت محاسبات نیست. طرح ارائه شده را می‌توان برای افزایش دقت ، گسترش داد . به ازای هر بیت افزایش دقت به سه ردیف و یک ستون اضافی نیاز است.

طرح پیشنهادی، توسط نرم افزار Foundation شرکت Xilinx برای پیاده سازی روی تراشهٔ FPGA شمارهٔ XC4020XL شبیه سازی شد. این شبیه سازی صحت عمل طرح را نشان داد. از ٧٨٤ عدد CLB تراشهٔ مذکور ٧٦٨ عدد مصرف شد و تأخیری برابر با 52.2ns برای هر یک از مراحل شانزده‌گانهٔ طرح، ثبت شد. به این ترتیب فرکانس تپش 19.15MHz تعیین شد.

جزء جدا ناپذیر طرحهایی که تا کنون براساس الگوریتم CORDIC بناشده اند ROM است. کانتابوترا [٨] در طراحی خود سه پالس ساعت را به خواندن ROM اختصاص داده است. در مدار پیشنهادی ما ROM که سطح بزرگی از تراشه را اشغال می‌کند کاملا حذف شده است . پالیوراس [٩] نیز با درونیابی و استفاده از جداول (١٦ کیلو بیت حافظه) و ٨٥٠٠

گیت (هـر گیـت معـادل یـک NAND دو ورودی) بـه تـاخیر $3t_{MAC}$ می رسد .هـر t_{MAC} تـاخیر محاسـبه در عمـل ضـرب و جمع[13] است . حال آنکه مدار پیشنهادی مقالهٔ حاضر تاخیری به مراتب کمتر از طرح پالیوراس دارد. دلیل ایـن امـر هـم وجـود ساختار خط لوله‌ای و استفاده از عمل جمـع در هـر مرحلـه بـه جای عمل ضرب است.

در تمام پیاده سازیهای CORDIC واحد کنترل پیچیده‌تریـن بخش سخت افزار است که با گسـترش مـدار نیـاز بـه طراحـی مجدد و عموما پیچیده تری دارد . در ایـن مقالـه واحـد کنـترل کاملا حذف شده است که نبـود آن در موقـع گسـترش طـرح و افزایش دقت می تواند مزیت بزرگی باشد .

واژه نامه

1. systolic arrays
2. coordinate rotation digital computer
3. processing element
4. inertial navigation system
5. AD 639
6. digit by digit
7. transentental
8. fixed point
9. controlled add-subtract
10. Optimized-Controlled-Add-Subtract
11. pipeline
12. throught-put
13. multiply-accumulate (MAC)

مراجع

1. Kung , H.T., "Why Systolic Architectures?," *IEEE Transactions on Computers*, pp.37-46, Jan. 1982.
2. Parhami, B., *Computer Arithmetic: Algorithms and Hardware Designs*, Oxford University Press, 2000.
3. Johnson, K.T., Hurson, AR., And Shirazi, B., "General – Purpose Systolic Arrays," *IEEE Transactions on Computers*, vol. 26, No. 11, pp.20-31, Nov. 1993.
٤. سماوی، ش.، و ثقفی ، ر. ، "طراحی و پیاده سازی کمـک پردازندهٔ توابع متعالی توسط FPGA"، *مجموعـه مقـالات هشتمین کنفرانس مهندسی برق*، دانشگاه صنعتی اصفهان، ص‌ص. ۲۰۶-۲۱۱، اردیبهشت ۱۳۷۹.
5. Dupart, J., and Muler, J.M., " The CORDIC Algorithm : New Result for Fast VLSI Implementation," *IEEE Transactions on Computers*, Vol. 42 , No. 2 , pp. 920-930 , Feb. 1993.
6. Volder, J.E., " The CORDIC Trigonometric Computing Technique," *IRE Transactions on Electronic Computers*, Vol. 8, pp. 330-334 , Sept. 1959.
7. Daggett, D.H., "Decimal – Binary Conversions in CORDIC," *IRE Transactions on Electronic Computers*, Vol. EC-8, No. 3, pp.335-339, Sept. 1959.
8. Kantabutra, V., "On Hardware for Computing Exponential and Trigonometric Functions," *IEEE Transactions on Computers,* Vol. 45, No. 3 , March 1996.
9. Paliouras, V., et al , "A Floating – point Processor for Fast and Accurate Sine/Cosine Evaluation," *IEEE Transactions on Circuits and Systems*, Vol. 47, No. 5 , May 2000.