

# محاسبه توابع مثلثاتی توسط آرایه های سیستمولیک و الگوریتم CORDIC

شادرخ سماوی\* و روشنگر کلشادی\*\*

دانشکده مهندسی برق و کامپیوتر، دانشگاه صنعتی اصفهان

(دریافت مقاله: ۱۳/۷/۷۸ - دریافت نسخه نهایی: ۱۴/۱۲/۷۹)

**چکیده:** توابع مثلثاتی از کاربردیترین توابع در پردازش سیگنالهای دیجیتال اند. طرح ارائه شده در این مقاله توابع مثلثاتی را با استفاده از آرایه های سیستمولیک، محاسبه می کند. روش به دست آوردن این توابع برای زاویه  $\theta$  بر اساس الگوریتم CORDIC است. سلول ساده و استاندارد که برای شبکه سیستمولیک مطرح شده، نهایتاً با توجه به ورودیهای خاص سلولها، بهینه شده است. واحد کنترل و حافظه ROM از اجزای اصلی هر مدار CORDIC هستند و فضای بزرگی را اشغال می کنند. سخت افزار طراحی شده از این واحدها استفاده نمی کند به همین دلیل طراحی ساده و قابل گسترش است.

واژگان کلیدی: آرایه های سیستمولیک - 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,  $\theta$ , 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

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

انتخاب معماری مناسب برای هر سیستم الکترونیکی وابستگی شدیدی به تکنولوژی پیاده سازی آن دارد. در

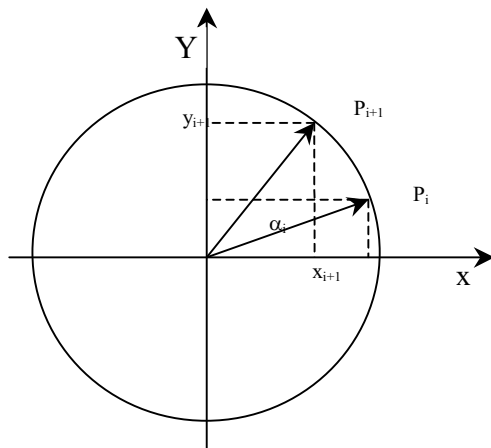
\* - کارشناسی ارشد

\* - استادیار

می باشند [۴و۳]. به همین دلیل، الگوریتم CORDIC را کاندید مناسبی برای آرایه‌های سیستمولیک یافتیم.

## ۲- توصیف الگوریتم CORDIC

نقطه  $P_i$  با مختصات  $(x_i, y_i)$  را به اندازه زاویه  $\alpha_i$  دوران می‌دهیم و به نقطه  $P_{i+1}$  می‌رسیم [۵]. این مطلب در شکل (۱) نمایش داده شده است. هدف، محاسبه مختصات  $P_{i+1}$  برحسب مختصات  $P_i$  است.



شکل ۱- دوران  $P_i$  به اندازه زاویه  $\alpha_i$

مختصات نقطه  $P_{i+1}$  یعنی  $(x_{i+1}, y_{i+1})$ ، به صورت زیر به دست می‌آید

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

در صورتی که بخواهیم تنها از شیفت و جمع یا تفریق استفاده کنیم معادله (۱) را بر  $\cos \alpha_i$  تقسیم می‌کنیم، در نتیجه معادله‌های زیر به دست می‌آیند

$$\frac{x_{i+1}}{\cos \alpha_i} = x_i - y_i \cdot \tan \alpha_i = x'_{i+1} \quad (2)$$

$$\frac{y_{i+1}}{\cos \alpha_i} = y_i + x_i \cdot \tan \alpha_i = y'_{i+1} \quad (3)$$

فرض کنید زاویه ای که می‌خواهیم تابع مثلثاتی را برای آن محاسبه کنیم  $\theta$  باشد. این زاویه را می‌توان از جمع تعدادی

$$\theta \approx \sum_{i=0}^{h-1} d_i \cdot \alpha_i \quad \text{یعنی تشکیل داد،}$$

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 می‌یابد. معماری آرایه‌های سیستمولیک به عنوان راه حلی برای مسائل بالاست. این معماری اولین بار توسط کونگ مطرح و به مرور زمان تکمیل شده است [۱و۲].

یک سیستم سیستمولیک، ساختاری توری مانند از عناصر پردازشی یکسان<sup>۳</sup> متصل به هم است که هر یک قادر به انجام چند عمل ساده‌اند. داده‌ها که شامل عناصر ورودی و نتایج میانی هستند، بین سلولها جریان می‌یابند و ارتباط با محیط خارج تنها در سلولهای حاشیه‌ای صورت می‌گیرد. طرزکار آرایه‌های سیستمولیک کاملاً شبیه ضربان قلب است که داده‌ها به جای خون به عناصر پردازشی وارد می‌شوند. هر سلول می‌تپد و نتیجه خروجی را به سلول بعدی می‌دهد. این معماری برای دسته گسترده‌ای از محاسبات که در آن عملیات روی هر فقره داده به صورت تکراری انجام می‌شود 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<sup>۴</sup> نیاز به محاسبه 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<sup>۵</sup> به این منظور طراحی شده است. روشهای مختلفی برای محاسبه سخت افزاری توابع مثلثاتی وجود دارد. روش جستجو در جدول، روش تقریب چند جمله‌ای با استفاده از بسط تیلور، ترکیبی از دو روش قبل و روش رقم به رقم<sup>۶</sup> با استفاده از الگوریتم CORDIC، چند نمونه از آنهاست. الگوریتم CORDIC روشی کارآمد برای محاسبه توابع مثلثاتی است. این الگوریتم ابتدا توسط ولدر [۶] معرفی شد. این الگوریتم قادر به محاسبه توابع مثلثاتی و متعالی<sup>۷</sup> با استفاده از تکرار شیفت و جمع یا تفریق روی عناصر است. عملیات تکرارپذیر و یکنواخت برای پیاده سازی توسط آرایه‌های سیستمولیک مناسب

$$x_0 = \frac{1}{K} = \frac{1}{1.646760255} = 0.6072252936 \quad (10)$$

متغیر دیگری که به کار برده می شود  $z_i$  است که مقادیر زاویه را با آن تغییر می دهیم. مقدار اولیه  $z_i$  یعنی  $z_0$  همان زاویه  $\theta$  است. با توجه به مطالب ذکر شده، معادله های زیر، بعد از  $n$  مرحله تکرار،  $\sin \theta$ ,  $\cos \theta$  را تولید می کنند

$$x_{i+1} = x_i - d_i \cdot y_i \cdot 2^{-i} \quad (11)$$

$$y_{i+1} = y_i + d_i \cdot x_i \cdot 2^{-i} \quad (12)$$

$$z_{i+1} = z_i - d_i \cdot \tan^{-1} 2^{-i} \quad (13)$$

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

معادله های (11) تا (13)، معادله های اصلی CORDIC هستند که با مقادیر اولیه ذکر شده در معادله (14) محاسبات شروع می شود و بعد از  $h$  مرحله تکرار  $x_h$  و  $y_h$  به ترتیب  $\cos \theta$  و  $\sin \theta$  را نشان می دهند.

$$x_0 = \frac{1}{K}, y_0 = 0, z_0 = \theta \quad (14)$$

در الگوریتم CORDIC زاویه  $\theta$  به باینری تبدیل شده و به عنوان  $z_0 = 0.1001101101110011$  و  $y_0 = 0$  وارد می شود، که معادل عدد به دست آمده از معادله (10) است. دقت [V] نشان داده است که اعداد  $n$  بیتی در سیستم باینری، بعد از  $n$  مرحله تکرار جواب دقیقی را خواهند داد.

### ۳- پیاده سازی

در این بخش راه حلی سخت افزاری را برای اجرای سیستمولیک الگوریتم CORDIC ارائه می دهیم. داده ها در مدار پیشنهادی ۱۶ بیتی و با نمایش ممیز ثابت<sup>۸</sup> در نظر گرفته شده اند. برای  $z$ ، محل قرار گرفتن ممیز بین بیت های هشتم و نهم است، یعنی ۸ بیت ارقام صحیح و ۸ بیت ارقام اعشاری داریم.  $x$  و  $y$  دو عدد کاملاً اعشاری هستند.

عنصر پردازشی در شکل (۲)، اساس ساختمان آرایه سیستمولیک طراحی شده را نشان می دهد.

برای سهولت در انجام محاسبات مقدار  $\alpha_i$  را به صورت  $\alpha_i = \tan^{-1} 2^{-i}$  انتخاب می کنیم. بنابراین چنانچه محاسبات در مبنای دو انجام شود، عبارتی که باید در  $\tan \alpha_i = 2^{-i}$  ضرب شوند فقط شیفیت داده می شوند.

مقدار  $d_i$  می تواند ۱ یا -۱ باشد. همواره با  $d_0 = 1$  و  $\alpha_0 = 45$  شروع می کنیم و بر حسب اینکه  $\theta$  از ۴۵ درجه کوچکتر یا بزرگتر باشد،  $d_i$  مرحله بعدی ۱ یا -۱ انتخاب می شود. بعد از  $h$  بار تکرار، محاسبات به سمت  $\theta$  مورد نظر همگرا می شود. می توان عبارات زیر را از معادله های (۲) و (۳) به دست آورد

$$x'_h = \frac{x_0}{\cos \alpha_0 \cdot \cos \alpha_1 \dots \cos \alpha_{h-1}} \cdot \cos(\alpha_0 + \alpha_1 + \dots + \alpha_{h-1}) \quad (4)$$

$$y'_h = \frac{x_0}{\cos \alpha_0 \cdot \cos \alpha_1 \dots \cos \alpha_{h-1}} \cdot \sin(\alpha_0 + \alpha_1 + \dots + \alpha_{h-1}) \quad (5)$$

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

$$x'_h = \cos \theta \quad \text{و} \quad y'_h = \sin \theta \quad (6)$$

به دنبال نشان می دهیم که  $x_0$  مقداری ثابت است و تابعی از  $\theta$  نیست

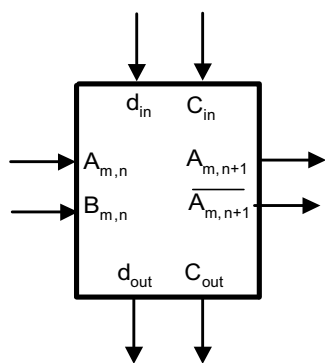
$$K = \frac{1}{\cos \alpha_0 \cdot \cos \alpha_1 \dots \cos \alpha_{h-1}} = \prod_{i=0}^{h-1} \left( \frac{1}{\cos \alpha_i} \right) = \prod_{i=0}^{h-1} \left( 1 + \frac{\sin^2 \alpha_i}{\cos^2 \alpha_i} \right)^{\frac{1}{2}} \quad (7)$$

$$K = \prod_{i=0}^{h-1} (1 + \tan^2 \alpha_i)^{\frac{1}{2}} = \prod_{i=0}^{h-1} (1 + (2^{-i})^2)^{\frac{1}{2}} = \prod_{i=0}^{h-1} \sqrt{1 + 2^{-2i}} \quad (8)$$

در محاسبه  $K$ ، با در نظر گرفتن ۹ رقم اعشار، برای مقادیر  $i \geq 16$  عبارت  $\sqrt{1 + 2^{-2i}}$  برابر ۱ می شود، بنابراین مقدار حداکثر را برای  $i$  همان شانزده در نظر می گیریم

$$K = \prod_{i=0}^{16} \sqrt{1 + 2^{-2i}} = 1.646760255 \quad (9)$$

با توجه به معادله (۹) مقدار  $x_0$  وقتی تعداد مراحل تکرار ۱۶ است به صورت زیر مشخص می شود



$$n=1,2,\dots,16$$

$$m=1,2,\dots,16$$

$$d_{in}=d_{out}$$

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

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

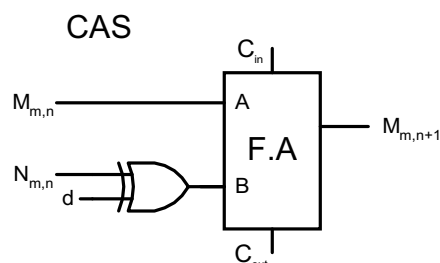
شکل ۲ - عنصر پردازشی اصلی ساختمان آرایه سیستمولیک

یعنی درستون اول برای  $x$  و  $z$ ،  $d_{in}=1$ ، و برای  $y$ ،  $d_{in}=0$  منظور شده است، بنابراین مقادیر اولیه به شکل معادله‌های (۱۵) به ستون اول وارد می‌شوند.

$$\begin{aligned} z: & d_{in}=C_{in}=1 \\ & z_0 = \theta = A_{16,1} \dots A_{9,1} \cdot A_{8,1} \dots A_{1,1} \\ & \tan^{-1}2^{-0} = 45 = 00101101.00000000 = \\ & B_{16,1} \dots B_{9,1} \cdot B_{8,1} \dots B_{1,1} \\ x: & d_{in}=C_{in}=1 \\ & 0.A_{32,1} \dots A_{17,1} = x_0 = 0.1001101101110011 \\ & 0.B_{32,1} \dots B_{17,1} = y_0 = 0.0000000000000000 \\ y: & d_{in}=C_{in}=0 \\ & 0.A_{48,1} \dots A_{33,1} = y_0 = 0.0000000000000000 \\ & 0.B_{48,1} \dots B_{33,1} = x_0 = 0.1001101101110011 \end{aligned} \quad (15)$$

همان‌طور که در شکل (۲) مشخص است در هر سلول، مکمل سلول پردازشی نیز در دسترس است. در ردیف شانزدهم گروه اول از این خروجی مکمل استفاده شده است. این ردیف جایی است که بیت علامت  $z$  محاسبه می‌شود. با توجه به روابط  $d_i$ ، مکمل بیت علامت  $z$ ، در مرحله بعد کنترل‌کننده جمع یا تفریق برای  $x$  است و خود آن کنترل‌کننده جمع یا تفریق  $y$  است. بنابراین اگر در هر مرحله، خروجی  $A_{16,n+1}$  را به  $d_{in}$  سلول اول و سلول هفدهم (برای  $z$  و  $x$ ) و خروجی  $A_{16,n+1}$  را به  $d_{in}$  سلول سی و سوم (برای  $y$ ) در مرحله بعد وصل کنیم، به این ترتیب ستون بعد توسط بیت علامت ستون قبلی کنترل می‌شود.

در شکل بالا  $m$  و  $n$  به ترتیب ردیف پردازش و مرحله پردازش هستند. اعداد ۱۶ بیتی بوده لذا ۱۶ مرحله پردازش داریم یعنی  $n=1, \dots, 16$ . هر سلول قادر به پردازش یک بیت است. به این ترتیب برای پردازش ۳ عدد ۱۶ بیتی  $x$ ،  $y$  و  $z$  احتیاج به ۴۸ سلول در هر مرحله داریم. بنابراین شبکه سیستمولیک مورد نظر دارای ۴۸ ردیف و ۱۶ ستون است. هر سلول پردازشی در واقع یک واحد استاندارد CAS<sup>۱</sup> است که مدار داخلی آن در شکل (۳) آمده است. ۱۶ ردیف اول (گروه  $z$ ) برای پردازش  $z$ ، از ردیف ۱۷ تا ردیف ۳۲ (گروه  $x$ ) برای پردازش  $x$  و از ردیف ۳۳ تا ردیف ۴۸ (گروه  $y$ ) برای پردازش  $y$  به کار رفته‌اند. در سلولهای اولین ردیف هر گروه  $d_{in}$  و  $C_{in}$  به یکدیگر اتصال دارند. برای ستون اول زاویه  $z_0 = \theta$  وارد می‌شود. این زاویه مثبت است، لذا در ستون اول برای به دست آوردن  $x$  و  $z$  تفریق و برای به دست آوردن  $y$  جمع داریم.



شکل ۳ - شمای کلی یک سلول CAS

ظاهر شود. خروجیهای  $A_{m,17}$  جواب نهایی هستند، معادله‌های (۱۶) و (۱۷).

$$x_{16,16}x_{15,16}\dots x_{1,16} = A_{32,17} A_{31,17} \dots A_{17,17} = \cos z_0 \quad (16)$$

$$y_{16,16}y_{15,16}\dots y_{1,16} = A_{48,18} A_{47,17} \dots A_{33,17} = \sin z_0 \quad (17)$$

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

$$\begin{aligned} x &= \cos 42 = 1011111000111101 = 0.743118284 \\ y &= \sin 42 = 1010101101001011 = 0.669113294 \end{aligned} \quad (18)$$

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.

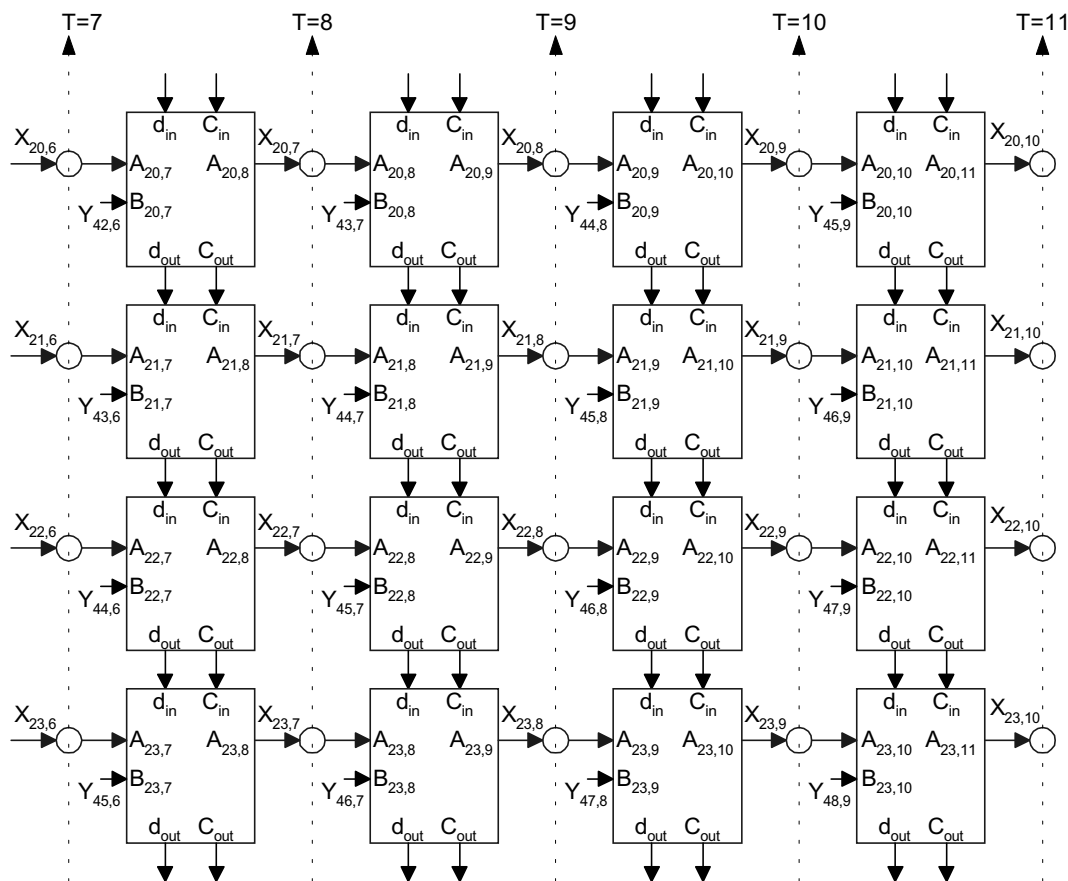
#### ۴- بهینه سازی

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 می‌زند و تأثیر چندانی در کم کردن تأخیر مدار نخواهد داشت زیرا در ساختارهای سیستمیک پریود ساعت تابعی از تأخیر کندترین مرحله است.

مدار شکل (۳) را در سطح ترانزیستور به طرق مختلف می‌توان پیاده سازی کرد. تعداد ترانزیستورهای مصرف شده تابعی از تکنولوژی به کار رفته، خواهد بود. به عنوان مثال، شکل (۵) یک نوع پیاده سازی را در تکنولوژی CMOS نشان می‌دهد. حتی در تکنولوژی CMOS هم همین مدار را به طرق متفاوتی طراحی کرده‌اند که بعضاً تعداد ترانزیستور بیشتر یا کمتری نسبت به شکل (۵) دارند. هدف از آنچه در ادامه می‌آید این است که مدار پایه شکل (۳) را با هر ساختار

با اتصال  $d_{in}$  به ۱۶ بیت عدد، پردازش یکسان روی هر ۱۶ بیت انجام می‌شود. این انتشار توسط اتصال  $d_{out}$  سلول  $m$ ام به  $d_{in}$  سلول  $m+1$ ام در یک ستون انجام می‌گیرد. ورودیهای  $A_{m,n}$  از خروجیهای  $A_{m,n+1}$  مرحله قبل مستقیماً وارد می‌شوند. هر بیت که پردازش می‌شود، بیت نقلی خروجی آن  $(C_{out})_m$  به بیت نقلی ورودی سلول ردیف بعد  $(C_{in})_{m+1}$  منتقل می‌شود. ورودیهای  $B$  برای گروه  $z$ ، در هر مرحله،  $\tan^{-1} 2^{-i}$  برای  $i=0, \dots, 15$  هستند که به صورت باینری محاسبه شده و به  $B_{m,n}$  ( $m=1,2,\dots,16$ ) وارد می‌شوند. خروجیهای گروههای  $x$  و  $y$  هر مرحله باید شیفته داده شوند و به  $B$  های مرحله بعد وارد شوند. شیفته داده شده  $x$  به  $B$  های مرحله بعد گروه  $y$  یعنی سلولهای ردیف ۳۳ تا ۴۸ وصل می‌شود و شیفته داده شده  $y$  به ورودیهای  $B$  مرحله بعد گروه  $x$  یعنی سلولهای ردیف ۱۷ تا ۳۲ اتصال می‌یابد.

شکل (۴) 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 (۴) نمایش داده‌ایم. همه ورودیهای سلولها قبل از ورود از یک  $Latch$  عبور می‌کنند. از مدار  $Latch$  برای جلوگیری از تداخل داده‌های ورودی سلولها استفاده می‌شود. بعد از اولین پالس ساعت، مقادیر اولیه و زاویه وارد ستون اول می‌شود. در سلولهای گروه  $z$  عمل تفریق و در سلولهای گروه  $y$  عمل جمع صورت می‌گیرد. خروجی هر مرحله وارد  $Latch$  شده،  $A_{16,2}$  توسط یک  $Latch$  وارد  $d_{in}$  سلول اول ستون دوم می‌شود. همین عدد توسط یک  $Latch$  دیگر به  $d_{in}$  سلول هفدهم ستون دوم وارد می‌شود. بعد از اعمال پالس دوم همه خروجیها به ورودیهای مرحله دوم می‌روند و مرحله اول، زاویه دیگری و مقادیر اولیه را می‌گیرد. پردازش روی عناصر هر دو ستون انجام می‌شود و به همین ترتیب ادامه می‌یابد تا محاسبات مربوط به زاویه  $\theta$ ، در خروجی مرحله شانزدهم



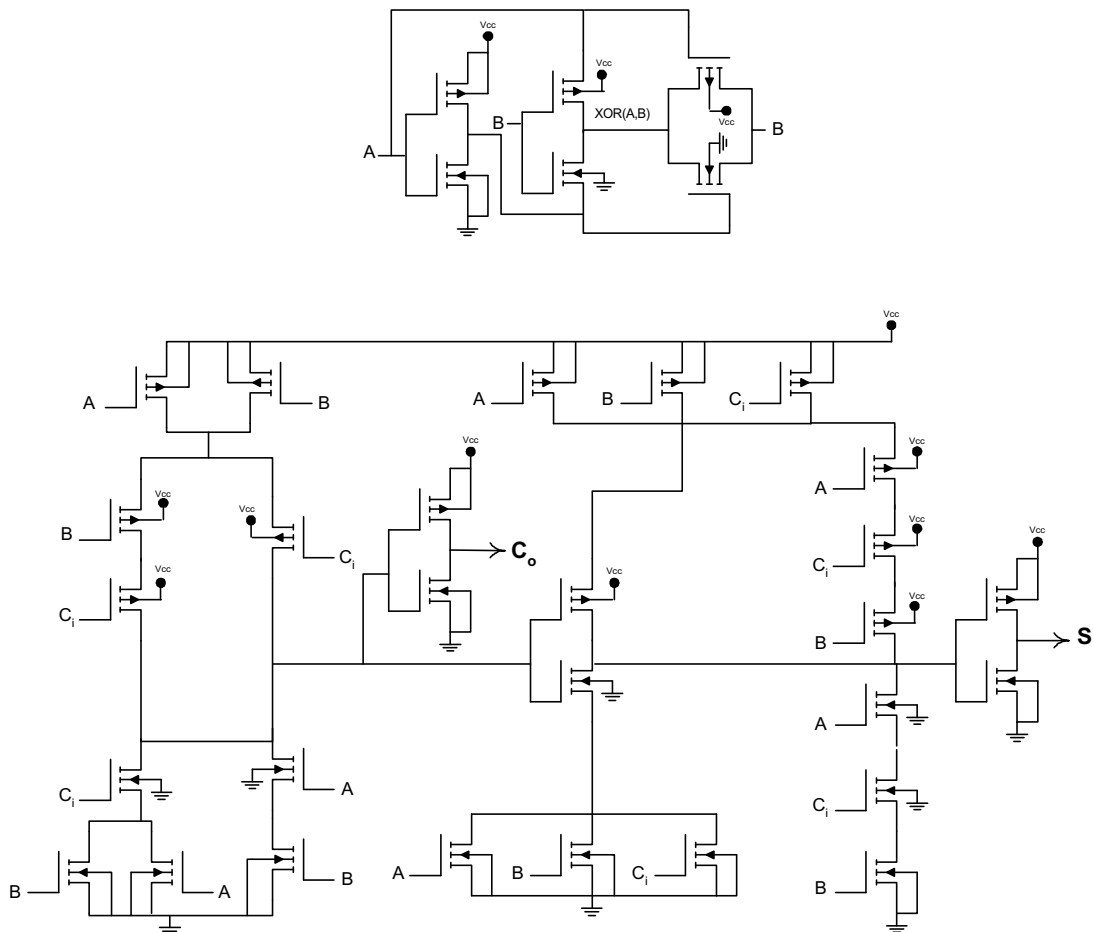
شکل ۴- بخشی از شبکه سیستمولیک

#### گام اول بهینه سازی

همان‌طور که در شکل (۳) مشخص است،  $d_{in}$  و  $B_{m,n}$  ورودیهای یک گیت XOR هستند. اگر  $B_{m,n}=0$  باشد، حاصل XOR آن با  $d_{in}$  خود خواهد بود و اگر  $B_{m,n}=1$  باشد، حاصل XOR آن با مکمل  $d_{in}$  ( $\overline{d_{in}}$ ) خواهد بود. بنابراین، در حالت اول ( $B_{m,n}=0$ )، می‌توان XOR را حذف کرد و مستقیماً  $d_{in}$  را به ورودی F.A وصل کرد، این سلول را OCAS1 می‌نامیم. این سلول ۲۸ ترانزیستور دارد و به علت حذف XOR، ۶ عدد از ترانزیستورها کم می‌شود. یعنی با جایگزینی OCAS1، ۶ عدد ترانزیستور کمتر استفاده می‌شود. در حالت دوم ( $B_{m,n}=1$ ) می‌توان XOR را حذف کرد و  $d_{in}$  را بعد از عبور دادن از یک NOT به ورودی B جمع‌کننده

ترانزیستوری که داشته باشد می‌توان ساده کرد. به همین منظور مدار شکل (۵) را به عنوان مبنا فرض می‌کنیم و بهینه سازی را بر اساس تعداد ترانزیستوری که از این شکل کم می‌شود به پیش می‌بریم. برای پیاده‌سازیهای دیگر نیز می‌توان همین بحث را تعمیم داد.

آرایه سیستمولیک این طرح دارای  $48 \times 16 = 768$  سلول پردازشی و تعداد Latch به کار رفته برای هر ستون، ۸۳ عدد است. پس برای کل ۱۶ ستون  $16 \times 83 = 1328$  عدد Latch در مدار داریم. با توجه به اینکه سلول پردازشی از ۳۴ ترانزیستور و مدار Latch از ۱۸ ترانزیستور ساخته شده است، بنابراین مدار از تعداد  $50016$  ترانزیستور ساخته می‌شود. بهینه‌سازی را در چهار گام توضیح می‌دهیم.



شکل ۵- یک ساختار ترانزیستوری برای مدار CAS

باید شیفتم شوند که از مرحله دوم این کار شروع می شود که این مرحله را در گام سوم بهینه می کنیم. تعداد صفرها از مرحله سوم تا مرحله شانزدهم ۲۳۸ است که به همین تعداد سلول به OCAS1 تبدیل می شود. بنابراین ۱۴۲۸ ترانزیستور 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 بهینه سازی شد که در آن ۲۷۷۰ ترانزیستور و ۲۳۸ Latch حذف شد.

#### گام دوم بهینه سازی

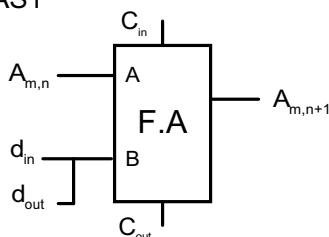
گام دوم برای ساده سازی ستون اول گروه z است. در این ستون،  $d_{in}=1$  است. می دانیم که اگر  $B_{m,n}=0$  باشد، سلول OCAS1 و اگر  $B_{m,n}=1$  باشد، سلول OCAS2 را می توان به کار برد. با توجه به ساده سازی گیتها، توابع (۱۹) و (۲۰) را

وصل کرد. این سلول را OCAS2 می نامیم. این سلول از ۳۰ ترانزیستور ساخته شده است، بنابراین جایگزینی OCAS2، ۴ ترانزیستور از مدار کم می کند.

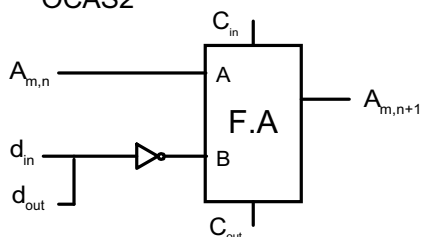
ورودیهای  $B_{n,m}$  برای گروه z همگی  $2^{-i} \tan^{-1}$  و مقادیری ثابت اند. ستون اول z در گام دوم بهینه می شود. جمعاً تعداد صفرها از مرحله دوم تا مرحله شانزدهم ۱۹۱ عدد است. بنابراین ۱۹۱ سلول به OCAS1 تبدیل می شود و ۱۱۴۶ ترانزیستور کاهش می یابد. تعداد یکها، از مرحله دوم تا مرحله شانزدهم، ۴۹ عدد است. بنابراین ۴۹ سلول به OCAS2 تبدیل شده و ۱۹۶ ترانزیستور از مدار کاسته می شود.

می دانیم که به ازای هر شیفتم به راست، یک صفر در موضع پرارزش وارد می شود. گروههای y و x در هر مرحله

OCAS1

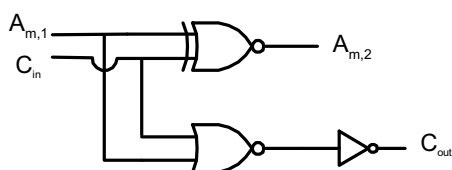


OCAS2

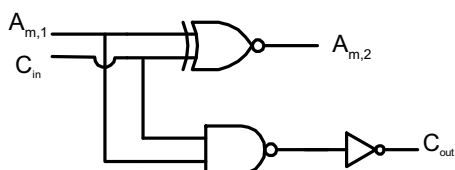


شکل ۶- سلولهای بهینه شده در گام اول

OCAS3



OCAS4



شکل ۷- سلولهای بهینه شده در گام دوم

سلول نهم ستون اول دارای  $C_{in}=1$  (حاصل از سلول هشتم) و  $B_{9,1}=1$  است. بنابراین سلول OCAS4 را جایگزین آن می‌کنیم. 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 وارد می‌کنیم، خروجی آن را به  $A_{9,2}$  وصل می‌کنیم. در این حالت با حذف سلول اصلی و جایگزینی یک NOT، ۳۲ ترانزیستور از مدار کاسته می‌شود.

در سلول شانزدهم ستون اول ورودیهای  $A_{16,1}$ ،  $B_{16,1}$  همواره صفرند.  $B_{16,1}$  که عدد ثابتی است ( $\tan^{-1} 2^{-1}=45^\circ$ ) و نیز به دلیل قرارگیری زاویه ورودی بین صفر و  $90^\circ$ ،  $A_{16,1}$  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 می‌شود و خروجی  $\overline{C_{in}}$  را داریم. در سلول شانزدهم محاسبه بیت نقلی خروجی ضرورتی ندارد. بنابراین سلول اصلی را حذف می‌کنیم و به جای آن یک NOT قرار می‌دهیم 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 گروه  $y$  مرحله بعد وصل می‌کنیم. با این کار ۳۲

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

$$C_{out} = A_{m,1} + C_{in}, \text{ SUM} = A_{m,1} \cdot C_{in} + \overline{A_{m,1}} \cdot \overline{C_{in}} \quad (19)$$

$$C_{out} = A_{m,1} \cdot C_{in},$$

$$\text{SUM} = \overline{A_{m,1}} \cdot C_{in} + A_{m,1} \cdot \overline{C_{in}} \quad (20)$$

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

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



$$\text{Cout} = d_{in} + C_{in}, \text{SUM} = d_{in}C_{in} + \bar{d}_{in}\bar{C}_{in} \quad (25)$$

$$\text{Cout} = \bar{d}_{in} \cdot C_{in}, \text{SUM} = d_{in} \cdot C_{in} + \bar{d}_{in} \cdot \bar{C}_{in} \quad (26)$$

$$\text{Cout} = C_{in} \cdot \bar{d}_{in}, \text{SUM} = C_{in} \cdot \bar{d}_{in} + \bar{C}_{in} \cdot d_{in} \quad (27)$$

برای معادله (24) سلول OCAS5 را داریم که از سلول اصلی، 22 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 8 ترانزیستور و 8 عدد Latch از سلولها کاسته می شود.

معادله (25) 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 12 ترانزیستور داشته و با این 10 مورد جانشینی، 220 ترانزیستور و 20 Latch از مدار کم می شود.

معادله (26) سلول OCAS7 را با 12 ترانزیستور نتیجه می دهد. 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 8 سلول را می توان با OCAS7 عوض کرده و 176 ترانزیستور و 16 Latch از سلولها را کم کنیم.

سلول OCAS8 معادله (27) را می سازد که 12 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 1 هستند، پس با جایگزینی این سلولها 220 ترانزیستور و 20 عدد Latch از سلولها کم می شود. در کل گام سوم تعداد 1792 ترانزیستور و 128 Latch از مدار کاهش می یابد.

### گام چهارم بهینه سازی

در محاسبه 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 Cout را 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  $B_{32,2}=0, B_{48,2}=0, A_{32,2}=1, A_{48,2}=1$  بودند که سلول OCAS6 جایگزین آنها شد. در این سلولها باید

ترانزیستور از مدار کاسته می شود. Latch های ورودی این سلول نیز حذف می شود. پس 1 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 6 سلول باقی مانده 3 عدد دارای  $B_{m,1}=0$  هستند  $(m=10,13,15)$ ، که با OCAS3 جایگزین می شوند و 66 ترانزیستور از مدار کم می کنند و 3 عدد دارای  $B_{m,1}=1$  هستند  $(m=11,12,14)$  که هر سه با OCAS4 جایگزین می شوند و 66 ترانزیستور را حذف می کنند. در مجموع در گام دوم 9 عدد Latch و 468 عدد ترانزیستور حذف می شود.

### گام سوم بهینه سازی

در مرحله اول برای گروه  $x$ ،  $(m=17 \dots 32)$ ، عمل تفریق و برای گروه  $y$  عمل جمع  $(m=33 \dots 48)$  داریم. برای  $x$  ورودیهای  $A_{m,1}$  و  $B_{m,1}$  به ترتیب  $x_0, y_0$  هستند بنابراین

$$A_1 - B_1 = X_0 - Y_0 = X_0 - 0 = X_0 \quad (21)$$

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

$$A_1 + B_1 = Y_0 + X_0 = 0 + X_0 = X_0 \quad (22)$$

با توجه به معادله های (21) و (22)، خروجی  $x$  و  $y$  همان  $x_0$  می شود، بنابراین کل 32 سلول مربوط به  $x$  و  $y$ ،  $(m=17, \dots, 48)$ ، را در ستون اول حذف می کنیم. این حذف 64 عدد Latch و 1088 ترانزیستور را از مدار کم می کند. ورودیهای  $A_{m,2}$  برای  $x$   $(m=17, \dots, 32)$  و ورودیهای  $A_{m,2}$  برای  $y$   $(m=33, \dots, 48)$ ، هر دو  $x_0$  هستند و ورودیهای  $B_{m,2}$  برای هر دو سری  $(y$  و  $x)$  یکی شیفته یافته  $x_0$  هستند، بنابراین

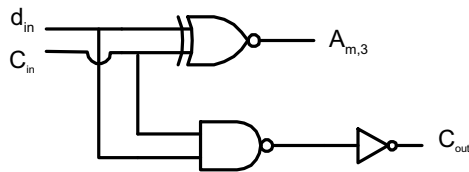
$$\begin{aligned} A_{32,2} A_{31,2} \dots A_{17,2} &= 1001101101110011 \\ B_{32,2} B_{31,2} \dots B_{17,2} &= 0100110110111001 \\ A_{48,2} A_{47,2} \dots A_{33,2} &= 1001101101110011 \\ B_{48,2} B_{47,2} \dots B_{33,2} &= 0100110110111001 \end{aligned} \quad (23)$$

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

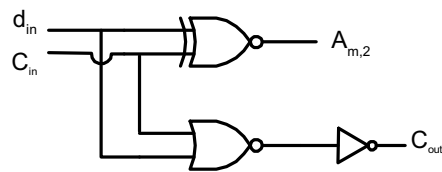
برای  $B_{m,2}=0$  و  $A_{m,2}=0$ ،  $B_{m,2}=0$  و  $A_{m,2}=1$ ،  $B_{m,2}=1$  و  $A_{m,2}=0$  و نهایتاً  $B_{m,2}=1$  و  $A_{m,2}=1$  به ترتیب معادله های (24) تا (27) را داریم.

$$\text{Cout} = d_{in} \cdot C_{in}, \text{SUM} = \bar{d}_{in} \bar{C}_{in} + \bar{d}_{in} C_{in} \quad (24)$$

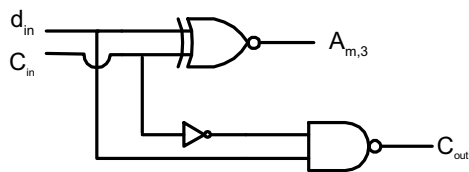
OCAS5



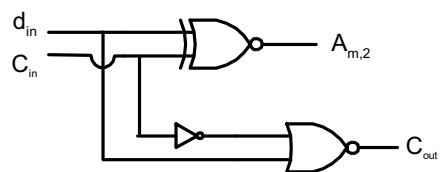
OCAS6



OCAS7



OCAS8



شکل ۸- سلولهای بهینه شده در گام سوم

در هر پالس ساعت یک زاویه را دریافت کرده و با رفتار خط لوله ای<sup>۱۱</sup> در هر پالس همزمان Sin و Cos یک زاویه را تولید کند. بازدهی<sup>۱۲</sup> این ساختار تابعی از بزرگی کلمات و دقت محاسبات نیست. طرح ارائه شده را می توان 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 تراشه FPGA شماره XC4020XL شبیه سازی شد. این شبیه سازی صحت عمل طرح را نشان داد. از ۷۸۴ عدد CLB تراشه مذکور ۷۶۸ عدد مصرف شد و تأخیری برابر با 52.2ns برای هر یک از مراحل شانزده گانه طرح، ثبت شد. به این ترتیب فرکانس تپش 19.15MHz تعیین شد.

جزء جدا ناپذیر طرحهایی که تا کنون براساس الگوریتم CORDIC بنا شده اند ROM است. کانتابوترا [۸] 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 می کند کاملاً حذف شده است. پالیوراس [۹] نیز با درونیابی و استفاده از جداول (۱۶ کیلو بیت حافظه) و ۸۵۰۰

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 چهارم و هشتم OCAS1 یا OCAS2 هستند. در هر دو مورد از خروجی 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 چهارم هشتم، یک NOT حذف می شود. مجموعاً ۴۳ عدد NOT حذف می شود که ۸۶ ترانزیستور 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 ۶۷۵۰ ترانزیستور از مدار می کاهد، تعداد ترانزیستور حذف شده نیز در طی این چهار مرحله ۵۱۲۸ عدد است، بنابراین تعداد کل ترانزیستورهای حذف شده ۱۱۸۷۸ عدد است، که ۲۳/۷۵ درصد کل ترانزیستورها است.

## ۵- نتیجه گیری

در این مقاله یک روش سخت افزاری محاسبه Sin و Cos به صورت سیستمولیک ارائه شد. مدار ارائه شده قادر است

در تمام پیاده سازیهای CORDIC واحد کنترل پیچیده‌ترین بخش سخت افزار است که با گسترش مدار نیاز به طراحی مجدد و عموماً پیچیده تری دارد. در این مقاله واحد کنترل کاملاً حذف شده است که نبود آن در موقع گسترش طرح و افزایش دقت می تواند مزیت بزرگی باشد.

گیت ( هر گیت معادل یک NAND دو ورودی) به تاخیر  $3t_{MAC}$  می رسد. هر  $t_{MAC}$  تاخیر محاسبه در عمل ضرب و جمع<sup>۱۳</sup> است. حال آنکه مدار پیشنهادی مقاله حاضر تاخیری به مراتب کمتر از طرح پالیوراس دارد. دلیل این امر هم وجود ساختار خط لوله‌ای و استفاده از عمل جمع در هر مرحله به جای عمل ضرب است.

## واژه نامه

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

## 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.
4. سماوی، ش.، و ثقفی، ر.، "طراحی و پیاده سازی کمک پردازنده توابع متعالی توسط FPGA"، مجموعه مقالات 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.