

محاسبه تابع تانژانت معکوس ۴ ربعی جهت برآورد سریع و تقریبی زاویه گرادیان تصاویر خاکستری ۸ بیتی مبتنی بر جداول جستجوی ساده بر روی تراشه های FPGA ارزان قیمت

علی ملکی^۱، پیام سنائی^{۲*}، پناه قربانی^۴، مائده انصاری^۵

^۱ دانشجوی کارشناسی مهندسی کامپیوتر دانشگاه آزاد اسلامی واحد نجف آباد؛ ali.maleki6596@iau.ir

^۲ عضو هیات علمی - استادیار، گروه مهندسی برق، واحد نجف آباد، دانشگاه آزاد اسلامی، نجف آباد، ایران؛ p.sanaee@iau.ac.ir

^۳ عضو هیات علمی، مرکز تحقیقات پردازش دیجیتال و بینایی ماشین، واحد نجف آباد، دانشگاه آزاد اسلامی، نجف آباد، ایران؛ p.sanaee@iau.ac.ir

^۴ دانشجوی کارشناسی مهندسی کامپیوتر دانشگاه آزاد اسلامی واحد نجف آباد؛ panah.ghorbani@iau.ir

^۵ دانشجوی کارشناسی مهندسی کامپیوتر دانشگاه آزاد اسلامی واحد نجف آباد؛ maede.ansari7224@iau.ir

* نویسنده مسئول: دکتر پیام سنائی

چکیده

تابع مثلثاتی معکوس $\text{atan2}(y,x)$ به صورت گسترده ای در پردازش تصویر استفاده می شود. در این مقاله برای محاسبه راستای گرادیان تصاویر، روش نوینی را جهت محاسبه تقریبی تابع $\text{atan2}(y,x)$ با سرعت بالا و مبتنی بر جداول جستجو در تراشه های FPGA ارائه نمودیم. در این روش هیچ نیازی به انجام محاسبات ریاضی ضرب و تقسیم نیست. برای تبدیل محاسبه تابع $\text{atan2}(y,x)$ چهار ربعی به یک ربعی، از روند Angle Range Reduction و برای کاهش حجم حافظه مورد نیاز برای ذخیره سازی جدول جستجو، از روند Scaling Range Reduction استفاده نموده ایم. از این رو حافظه به کار برده شده محدود بوده و در معماری سخت افزاری روش پیشنهادی ۲۵۶ بیت است. برای افزایش دقت محاسبه در این معماری از سه راهکار افزایش تعداد بیت های هر خانه حافظه، افزایش تعداد خانه های حافظه و اصلاح روند محاسبه محتویات جدول استفاده کرده ایم. ارزیابی ها نشان می دهند که بدلیل ترکیبی بودن محاسبات، روش های پیشنهادی از سرعت بسیار بالایی برخوردار می باشند. آزمایش ها نشان می دهند که در روش پیشنهادی سوم به ازای ۲۵۶ خانه حافظه ۸ بیتی نیاز به ۲۵۶ واحد محاسباتی عمومی LB بوده، مقدار متوسط تاخیر ۳۳/۴ نانوثانیه و میزان معیار MSE ۱/۵۷ درجه می باشد. روش پیشنهادی در تراشه XA3S200-4vqg100 از خانواده Spartan3 مبتنی بر زبان توصیف سخت افزار VHDL در بستر ISE14.7 پیاده سازی شده است.

کلمات کلیدی

محاسبات تقریبی، تراشه های آرایه ای منطقی برنامه پذیر، تابع تانژانت معکوس چهار ربعی، محاسبات بی درنگ، ممیز ثابت و روش های مبتنی بر جداول جستجو.

۱- مقدمه

تابع مثلثاتی $\text{atan2}(y,x)$ کاربردهای بسیار زیادی در محاسبات ریاضی داشته، از این رو به صورت فراگیری در علوم مهندسی مانند رباتیک، هندسه کاربردی، پردازش سیگنال های دیجیتال، پردازش تصویر، بینایی ماشین و تشخیص اشیا استفاده می شود [۱ و ۲]. مهمترین و متداولترین کاربرد این تابع در تبدیل بردارها و اعداد مختلط از مختصات دکارتی به قطبی و همچنین محاسبه زاویه گرادیان توابع دو متغیره است. در بسیاری از الگوریتم های علم پردازش دیجیتال تصاویر مانند Image Gradient Direction/Orientation, Histograms of Oriented Gradient (HOG) محاسبات مبتنی بر زاویه لبه ها در تصویر است [۲]. از این رو محاسبه سریع و دقیق این تابع در فرآیند لبه یابی تصاویر از اهمیت به سزایی برخوردار است [۳-۱۰]. عموماً در الگوریتم های پردازش تصویر برای بدست آوردن اندازه و زاویه بردار گرادیان در تصاویر خاکستری از عملگر سوبل با پنجره 3×3 استفاده می شود. این امر باعث افزایش دقت و کاهش زمان و حجم محاسبات لازمه می گردد. عملگر لبه یاب سوبل از دو ماتریس 3×3 مکانی برای کانالو کردن در عناصر پنجره لغزان محاسباتی استفاده می کند. یکی برای محاسبه گرادیان در راستای محور x ها و دیگری برای محاسبه گرادیان در راستای محور y ها. در اینجا ماتریس f تصویر اصلی خاکستری و ماتریس Gx مشتق نقاط در راستای محور x ها (ماتریس گرادیان افقی) و ماتریس

Gy مشتق نقاط در راستای محور yها (ماتریس گرادیان عمودی) است. توسط مجموعه روابط (۱) تا (۲) اندازه و زاویه بردار گرادیان را در تصویر بدست می آوریم [۵،۶،۱۰].

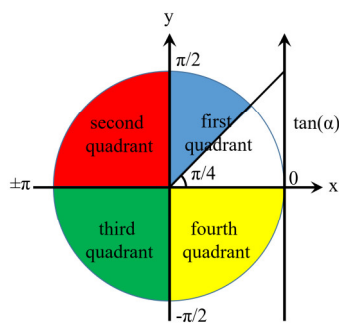
$$\begin{cases} Gy_{(i,j)} = (f_{(i-1,j-1)} + 2 \times f_{(i-1,j)} + f_{(i-1,j+1)}) - (f_{(i+1,j-1)} + 2 \times f_{(i+1,j)} + f_{(i+1,j+1)}) \\ Gx_{(i,j)} = (f_{(i-1,j+1)} + 2 \times f_{(i,j+1)} + f_{(i+1,j+1)}) - (f_{(i-1,j-1)} + 2 \times f_{(i,j-1)} + f_{(i+1,j-1)}) \end{cases} \quad (1)$$

$$\vec{G}_{(i,j)} = \begin{bmatrix} Gx_{(i,j)} \\ Gy_{(i,j)} \end{bmatrix}, \quad \|\vec{G}_{(i,j)}\| = \sqrt{Gx_{(i,j)}^2 + Gy_{(i,j)}^2}, \quad \angle \vec{G}_{(i,j)} = \text{atan2}(Gy_{(i,j)}, Gx_{(i,j)}) \quad (2)$$

برای محاسبه تابع $\text{atan2}(y,x)$ می توان از راهکارهای نرم افزاری و یا سخت افزاری استفاده نمود. راهکارهای نرم افزاری براساس پردازشگرهایی مانند ریزپردازنده ها/ میکروکنترلرها/ پردازشگرهای دیجیتال/ پردازشگرهای گرافیکی هستند [۴،۱۱،۲۹]. حال یا در این پردازشگرها دستورعملی برای محاسبه مستقیم این تابع در واحدهای محاسبات ریاضی NDP/FPU وجود دارد (مانند دستور FPATAN در پردازشگرهایی با معماری اینتل x86). یا توسط برنامه های نرم افزاری مقدار این تابع محاسبه می شود. در هر حال استفاده از پردازشگرها باعث افزایش توان مصرفی و زمان محاسبه خواهد شد. اما راهکار سخت افزاری استفاده از تراشه های ASIC/FPGA بوده که انتخاب اول مهندسين برای پردازش و محاسبات بیدرنگ است [۱۱-۲]. به این ترتیب می توان با حداکثر سرعت این محاسبه را انجام داد. مهمترین معیارها در پیاده سازی سخت افزاری این تابع توسط سامانه های دیجیتال عبارتند از زمان تاخیر، هزینه، توان مصرفی و مساحت در قبال دقت محاسبات از این رو در سامانه های طراحی شده همواره مصالحه ای بین سرعت، هزینه، توان مصرفی و دقت وجود دارد. محاسبه این تابع در حداقل زمان و با حداقل سخت افزار همواره یکی از چالش های بسیار مهم محققین می باشد.

۲- پیش زمینه نظری و روابط ریاضی بنیادین

هر دو تابع حقیقی $\text{atan2}(y,x)$ و $\tan^{-1}(u)$ جز توابع معکوس مثلثاتی هستند. دامنه تعریف تابع یک متغیره $\tan^{-1}(u)$ اعداد حقیقی بوده و بُرد آن از زاویه -90° تا $+90^\circ$ می باشد. اما تابع $\text{atan2}(y,x)$ دو متغیره بوده و دامنه تعریف آن زوج مرتبی از اعداد حقیقی است. این تابع نگاشتی از فضای دوبعدی به یک بعدی $\mathbb{R}^2 \rightarrow \mathbb{R}$ است. البته می توان تابع مزبور را در حوزه اعداد مختلط نیز تعریف نمود و به صورت $\tan^{-1}(x+iy)$ استفاده کرد. در این صورت این تابع نیز یک متغیره می شود، اما پارامتر ورودی عددی مختلط بوده و تابع نگاشتی از اعداد مختلط به اعداد حقیقی $\mathbb{Q} \rightarrow \mathbb{R}$ خواهد شد. بُرد تابع $\text{atan2}(y,x)$ کل صفحه مثلثاتی (0° تا $+360^\circ$) را پوشش می دهد. روند محاسبه تابع $\text{atan2}(y,x)$ به صورت چهار ربعی از روی ربع اول دایره مثلثاتی، کاهش دامنه تعریف زاویه Angle Range Reduction نامیده می شود (شکل ۱). با استفاده از روند کاهش دامنه تعریف زاویه تابع $\text{atan2}(y,x)$ توسط مجموعه روابط (۳) بیان می شود.



$$\text{atan2}(y,x) = \begin{cases} \tan^{-1}\left(\frac{|y|}{|x|}\right), & \text{if } x > 0 \text{ and } y \geq 0 \text{ 1'st quadrant} \\ +180^\circ - \tan^{-1}\left(\frac{|y|}{|x|}\right), & \text{if } x < 0 \text{ and } y \geq 0 \text{ 2'ed quadrant} \\ +180^\circ + \tan^{-1}\left(\frac{|y|}{|x|}\right), & \text{if } x < 0 \text{ and } y < 0 \text{ 3'rd quadrant} \\ +360^\circ - \tan^{-1}\left(\frac{|y|}{|x|}\right), & \text{if } x > 0 \text{ and } y < 0 \text{ 4'th quadrant} \\ +90^\circ, & \text{if } x = 0 \text{ and } y > 0 \\ +270^\circ, & \text{if } x = 0 \text{ and } y < 0 \\ \text{undefined}, & \text{if } x = 0 \text{ and } y = 0 \end{cases} \quad (3)$$

شکل ۱: ربع های مثلثاتی

حال برای بدست آوردن زاویه متناظر با تابع $\text{atan2}(y,x)$ که آن را θ می نامیم از جدول ۱ استفاده می کنیم [۱۲]. جدول ۱ ارتباط بین زاویه اصلی θ ، زاویه کمکی φ و علامت های پارامترهای x, y را بیان می کند. برای محاسبه تابع $\text{atan2}(y,x)$ نیازمند انجام محاسبات اندازه گیری قدرمطلق، تقسیم y/x و تابع مثلثاتی $\tan^{-1}(|y|/|x|)$ می باشیم. در انتها مقدار تابع $\text{atan2}(y,x)$ براساس محاسبات پس پردازشی جدول ۱ مشخص می شود.

جدول ۱: مقدار زاویه اصلی $\theta = \text{atan2}(y,x)$ بر اساس شرایط مرزی، زاویه کمکی $\varphi = \tan^{-1}(|y|/|x|)$ و علامت های پارامترهای x, y

	$y < 0$	$y = 0$	$y > 0$
$x < 0$	$\theta = +180^\circ + \varphi$	$\theta = +180^\circ$	$\theta = +180^\circ - \varphi$
$x = 0$	$\theta = +270^\circ$	$\theta = \text{undefined}$	$\theta = +90^\circ$
$x > 0$	$\theta = +360^\circ - \varphi$	$\theta = 0^\circ$	$\theta = \varphi$

برای محاسبه تابع مثلثاتی $\tan^{-1}(u)$ توسط مدارات دیجیتال ۳ راهکار متداول توسط محققین و مهندسين ارائه شده است [۱۵-۱۴،۴،۱۱،۱۳].

۱- استفاده از سری ها، تقریبات چند جمله ای و توابع ریاضی معادل

۲- استفاده از روش جابه جایی و جمع، الگوریتم Cordic (COordinate Rotation DIgital Computer)

۳- استفاده از جداول جستجو Look Up Table, Table based Method, ROM-based approach

اما نحوه پیاده سازی تابع $\tan^{-1}(u)$ توسط روابط ریاضی به دو دسته عمده تقسیم می شوند [۱].

۱- استفاده از سری های نامتناهی مانند بسط سری تیلور، بسط سری مکلورن، سری اویلر و چند جمله ای چبی شف [۱۵]

۲- تقریب با استفاده از توابع چند جمله ای کسری Rational Approximation [۱]

رابطه (۴) سری مکلورن و رابطه (۵) سری اویلر را نشان می دهند [۱۶، ۱۷]. بدلیل نامتناهی بودن سری ها و غیرخطی بودن اکید تابع $\tan^{-1}(u)$ برای محاسبه این تابع، نیاز به تعداد زیادی از جملات سری ها خواهیم داشت. از این رو نیازمند حجم بالایی از پردازش هستیم. همچنین از دیگر مشکلات، نیاز به واحدهای ضرب کننده و تقسیم کننده است. دقت این روش ها وابسته به تعداد جملات چند جمله ای به کار رفته می باشد. پیاده سازی تابع $\tan^{-1}(u)$ بر اساس سری ها از دقت بالایی برخوردار است اما نیازمند هزینه سخت افزاری بالا و زمان محاسبه نسبتاً طولانی خواهد بود. بدین ترتیب در محاسبات بی درنگ پیاده سازی تابع $\tan^{-1}(u)$ بر اساس سری های ریاضی انتخاب مناسبی نخواهد بود. سری تیلور همگرایی کندی داشته و حداقل نیاز به ۷ جمله دارد تا دقت خوبی داشته باشد [۱۶، ۱۷]. عموماً در سری ها زاویه برحسب رادیان محاسبه می شود.

$$\tan^{-1}(u) = \sum_{n=0}^{\infty} (-1)^n \frac{u^{2n+1}}{2n+1} \quad : -1 \leq x \leq +1 = u - \frac{u^3}{3} + \frac{u^5}{5} - \frac{u^7}{7} + \frac{u^9}{9} - \frac{u^{11}}{11} \dots \quad (4)$$

$$\tan^{-1}(u) = \frac{u}{1+u^2} \sum_{n=0}^{\infty} \prod_{k=1}^n \frac{2ku^2}{(2k+1)(1+u^2)} = \sum_{n=0}^{\infty} \frac{2^{2n}(n!)^2}{(2n+1)!} \times \frac{u^{2n+1}}{(1+u^2)^{n+1}} \quad (5)$$

اما این امکان وجود دارد که تابع $\tan^{-1}(u)$ به صورت تقریبی توسط چند جمله ای های کسری معادل سازی شود [۱، ۱۳]. شکل عمومی چند جمله ای های کسری توسط رابطه (۶) بیان می شود. در جدول ۲ تعدادی از چند جمله ای های کسری معرفی شده که در مراجع مختلف آورده شده اند.

$$\tan^{-1}(u) \approx \varphi_{n,d}(u) = \frac{P_n(u)}{D_d(u)} = \frac{\sum_{i=0}^n a_i u^i}{\sum_{j=0}^d b_j u^j} \quad (6)$$

جدول ۲: برخی از چند جمله ای های کسری معادل تابع $\tan^{-1}(u)$ زاویه خروجی برحسب رادیان

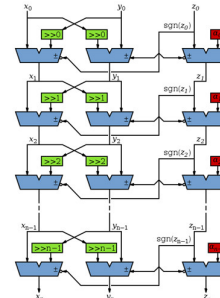
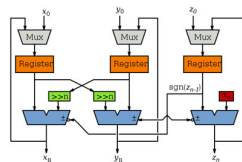
مرجع	تابع
[1]	$\frac{u}{1 + 0.0443 u + 0.2310u^2}$
[14]	$u \left(\frac{\pi}{4} + (1 - u)(0.2447 + 0.0663 u) \right)$
[14]	$u \left(\frac{\pi}{4} + 0.273(1 - u) \right)$
[14]	$u \left(\frac{\pi}{4} + 0.186982 - 0.191942u^2 \right)$

از دیگر روش ها، تقریب پاره خطی است که در مرجع [۱۸] استفاده شده است و در رابطه (۷) آورده شده است. در این روش بازه اطلاعاتی به چهار ناحیه تقسیم شده و تابع توسط چهار پاره خط مدل می شود.

$$\tan^{-1}(u) \cong \begin{cases} 56u & , 0.00 < u \leq 0.25 \\ 50u + 1.5 & , 0.25 < u \leq 0.50 \\ 40u + 6.5 & , 0.50 < u \leq 0.75 \\ 32u + 13 & , 0.75 < u \leq 1.00 \end{cases} \quad \text{Return value in Degree} \quad (7)$$

برای کاهش خطا می توان تعداد بازه ها را افزایش داد. در مرجع [۱۹] نیز از روش تقریب پاره خطی برای یافتن تابع $\tan^{-1}(u)$ استفاده شده و بازه اطلاعاتی به صورت غیرخطی به ۳۲ ناحیه تقسیم شده است.

الگوریتم CORDIC راهکار کارآمد و مقرون به صرفه ای جهت محاسبه توابع غیرخطی است که از روش جابه جایی و جمع استفاده می کند. این الگوریتم در سخت افزارهایی که دارای واحد ضرب کننده نیستند استفاده شده و جز اولین انتخاب ها و محبوب ترین شیوه ها برای پیاده سازی در تراشه های FPGA می باشد [۲۴-۲۰ و ۱۵ و ۱۳]. با این الگوریتم بسیاری از توابع متعالی جبری مانند جذر، توابع مثلثاتی، لگاریتمی و هذلولوی را می توان محاسبه کرد. در این الگوریتم دو روند Rotation و Vectoring وجود داشته و همچنین تعداد دفعات تکرار یا واحدهای محاسباتی متناسب با دقت و تعداد بیت های خروجی است. اگر چه این الگوریتم نیاز به تکرار بالا دارد اما مزیت آن عدم نیاز به واحد ضرب کننده است از این رو مساحت کمی را از تراشه FPGA اشغال می کند. از مشکلات این الگوریتم همگرایی کند آن است.



شکل ۲: الف- پیاده سازی مدار ترکیبی الگوریتم کوردیک ب- پیاده سازی مدار ترتیبی الگوریتم کوردیک

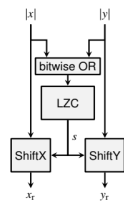
برای پیاده سازی سخت افزاری این الگوریتم دو روند به کار گرفته می شود. روش تمام موازی و ترکیبی که از به دنبال هم بستن سلول های محاسباتی پیاده سازی می شود (شکل ۲-الف) [۲۰]. این روش نه تنها هزینه بالایی را می طلبد بلکه تاخیر واحدهای جمع کننده، پارامتر بسیار حیاتی و تاثیرگذاری خواهد بود. اما در روش ترتیبی Iterative که مبتنی بر یک ماشین حالت محدود (FSM) است از یک واحد محاسباتی همه منظوره CORDIC استفاده کرده که به صورت بازخوردی نتایج خروجی را به ورودی با پارامترهای جدید اعمال می کند (شکل ۲-ب). در این شیوه واحد محاسباتی ساده بوده اما واحد کنترل پیچیده خواهد بود و در ضمن زمان محاسبات بسیار طولانی خواهد شد. برای محاسبه تابع $\text{atan2}(y,x)$ باید از روش Vectoring استفاده کرد [۲۴].

اما روش جداول جستجو جز سریعترین روش ها بوده که به حجم حافظه زیادی نیاز دارد. این روش هنگامی مناسب است که سیستم عددی اتخاذ شده ممیز ثابت بوده و دقت بیتی مورد نیاز اطلاعات ورودی بین ۸ الی ۱۶ بیت باشد. در روش متعارف، پارامتر ورودی u آدرس خانه حافظه ای را تعیین کرده که پاسخ $f(u)$ در آن واقع شده است [۲۵]. در نتیجه عدد u روی خطوط آدرس حافظه قرار گرفته و مقدار تابع $f(u)$ از روی خطوط داده حافظه خوانده می شود. از این رو هنگامی که ابعاد بیتی پارامتر u زیاد شود تعداد خانه های حافظه افزایش چشمگیری پیدا کرده و در نتیجه حجم حافظه به صورت نمایی افزایش می یابد. در این شیوه با افزایش تعداد خطوط آدرس حافظه، دقت پارامتر ورودی u افزایش یافته و با افزایش عرض بیتی خانه های حافظه دقت $f(u)$ افزایش می یابد. اکنون برای تابع $\text{atan2}(y,x)$ اگر ابعاد بیتی هر کدام از پارامترهای ورودی x, y برابر با w بیت باشد آنگاه حجم حافظه مورد نیاز 2^{2w} کلمه خواهد شد که حتی برای مقادیر کوچک w نیز غیر عملی و غیر کاربردی خواهد بود. همان گونه که مشاهده می شود با افزایش یک بیت ابعاد اطلاعات ورودی (افزایش دقت اطلاعات ورودی) حجم حافظه مورد نیاز چهار برابر و به صورت نمایی افزایش می یابد [۲۸-۲۶ و ۱۳ و ۱۲]. روش های متعددی مانند جداول دو بخشی BiPartite method (BP)، جداول چند بخشی MultiPartite method (MP) و جداول سلسله مراتبی چند بخشی Hierarchical MultiPartite method (HMP) جهت کاهش ابعاد حافظه جستجو پیشنهاد شده اند [۲۵ و ۱۴-۱۲]. در این مجموعه از روش ها بیت های با ارزش بیشتر تابع توسط یک جدول بیان شده و بیت های با ارزش کمتر توسط جدول دیگری بیان می شوند سپس با ترکیب خطی خروجی این جداول مقدار اصلی تابع بدست می آید. البته روش های ترکیبی توسط بسیاری از محققین برای رسیدن به دقت بیشتر و زمان محاسبه کمتر نیز ارائه شده اند مانند تلفیق روش جداول جستجو با محاسبات ریاضی به صورت درونیابی [۱].

۳- کارهای مرتبط

Malik و همکاران در سال ۲۰۲۵ برای محاسبه لبه تصاویر به روش Canny از تراشه FPGA به شماره Zync 7000 Ap SoC استفاده کردند [۳]. آنان برای محاسبه زاویه گرادیان از یک الگوریتم تقریبی استفاده کردند و فقط یکی از چهار زاویه 0° ، 90° ، 135° و 180° را بدست آورده اند. هاشمی و همکاران نیز در سال ۲۰۲۴ برای محاسبه لبه تصاویر به روش Canny از تراشه های FPGA به شماره Arria 10GX-1150 و Startix 10SX-2800 استفاده کردند. آنها نیز زاویه گرادیان را در راستای ۴ زاویه محاسبه کردند [۴]. Guo و همکاران در سال ۲۰۲۳ به روش Cordic زوایای لبه را در تصاویر محاسبه نمودند. این زوایا توسط ۸ مقدار 0° ، 22.5° ، 45° ، 67.5° ، 90° ، 112.5° ، 135° ، 157.5° و 180° بیان می شدند. آنان از تراشه FPGA به شماره Cyclon VI استفاده نمودند [۵]. Khalil و همکاران در سال ۲۰۲۳ برای محاسبه لبه تصاویر به روش Sobel از بُرد FPGA به شماره Zynq ZC7010 Ap SoC استفاده کردند [۶]. Kromichev در سال ۲۰۲۲ با استفاده از روش مقایسه زاویه لبه را به صورت تقریبی محاسبه کرد [۷]. ایشان نیز زوایا را به صورت تقریبی و توسط ۴ زاویه 0° ، 90° ، 135° و 180° بدست آورده و نتایج را روی تراشه های FPGA خانواده Cyclon، Statix آزمایش نمود. Abdullah و همکاران در سال ۲۰۲۱ مبتنی بر تراشه FPGA به شماره Spartan 3EXC3S1600 زاویه لبه ها را محاسبه کردند [۸]. آنان یک بار محاسبات را مبتنی بر الگوریتم Cordic و IP Core مربوطه انجام داده و بار دیگر از جداول جستجو استفاده کردند. جدول جستجو در حافظه BRAM به ابعاد 16384 خانه ۸ بیتی پیاده سازی شده بود. Milić و همکاران در سال ۲۰۲۱ زوایای لبه ها را در تصاویر به روش Cordic در تراشه های FPGA محاسبه کردند [۹]. آنها با استفاده از راهکار خط لوله سرعت محاسبات را به صورت چشمگیری افزایش دادند. Abed و همکاران در سال ۲۰۲۰ مبتنی بر تراشه FPGA به شماره Virtex 4 LX200 سرعت محاسبات را به صورت چشمگیری افزایش دادند [۱۰]. آنان نیز برای محاسبه زاویه لبه از الگوریتم Cordic استفاده کردند. Kim و همکاران در سال ۲۰۱۹ با استفاده از سری تیلور مقدار زاویه لبه در تصاویر را محاسبه نمودند [۲].

Benammar و همکاران در سال ۲۰۱۹ با تلفیق روش جدول جستجو و یک چند جمله ای کسری درجه دو ریاضی توانستند تابع تانژانت معکوس را با دقت بالایی روی بُردهای دیجیتال eZDSP-F28335 بدست آورند [۱]. Torres و همکاران در سال ۲۰۱۷ برای بدست آوردن دقیق مقدار زاویه یک عدد مختلط یک روش دو مرحله ای را ارائه کردند. در مرحله اول برای تخمین اولیه تابع تانژانت معکوس از تقریب پاره خطی استفاده کرده و سپس در مرحله دوم برای رسیدن به دقت بالاتر از جدول جستجو استفاده کردند [۲۸]. De Dinechin و همکارش در سال ۲۰۱۵ محاسبه تابع $\text{atan2}(y,x)$ با استفاده از خواص این تابع به صورت چهار ربعی انجام دادند [۱۲]. آنان برای افزایش دقت محاسبات خود از روش کاهش محدوده توسط مقیاس بندی Scaling Range Reduction استفاده کردند (رابطه ۸). در این روند تا هنگامی که بیت های با ارزش بیشتر هر دو پارامتر x, y صفر است، این دو پارامتر را به سمت چپ شیفت می دهیم تا بیت با ارزش بیشتر یکی از این دو پارامتر یک شود (شکل ۳). در این طرح واحد تشخیص صفر بودن بیت های با ارزش بیشتر، شمارش بیت های صفر آغازین LZC (Leading Zero Count) است.



شکل ۳: روند حذف بیت های صفر مشترک جلوی پارامترهای x, y [۱۲]

$$\tan^{-1}\left(\frac{y}{x}\right) = \tan^{-1}\left(\frac{y \times 2^k}{x \times 2^k}\right), k \in \mathbb{N} \quad (8)$$

آنها به دو روش تابع $\text{atan2}(y, x)$ را با ورودی و خروجی های ممیز ثابت بر روی تراشه Virtex6 (6vhx380tff1923) پیاده سازی نمودند. در روش اول تابع $\text{atan2}(y, x)$ دو متغیره به عنوان ترکیب دو تابع تک متغیره مبتنی بر سری تیلور محاسبه می شود. مقدار متقابل و تانژانت که هر کدام با استفاده از روش های تقریب دو بخشی یا چند جمله ای ارزیابی می شوند. در راهکار دوم مستقیماً از روی تقریب های چند جمله ای دو متغیره تکه ای درجه ۱ یا ۲، مقدار تابع محاسبه شد. در هر یک از این رویکردها در ابتدا به کاهش آرگومان مربوطه پرداخته می شود. Gutierrez و Valls در سال ۲۰۱۰ یک معماری را برای محاسبه تابع $\text{atan2}(y/x)$ پیشنهاد کردند. بدین ترتیب که توسط دو جدول جستجو این محاسبه را انجام می دادند. در مرحله اول توسط یک جدول مقدار $1/x$ را یافته سپس حاصل $1/x$ را توسط یک ضرب کننده در y ضرب می کردند. سپس توسط جدول جستجوی دوم مقدار تانژانت معکوس پارامتر y/x را محاسبه می کردند [۲۶]. در کل این عملیات ظرف مدت دو پالس ساعت محاسبه می شد. آنان نیز برای مقیاس بندی از مدار آشکارساز صفرهای جلوی عدد و شیفت دهنده های بشکته ای به سمت چپ استفاده نمودند. Gutierrez, Torres و Valls در سال ۲۰۰۹ یک معماری را برای محاسبه تابع $\text{atan2}(y/x)$ پیشنهاد کردند. بدین ترتیب که عملیات تقسیم y/x با استفاده از تبدیل لگاریتمی به تفریق تبدیل شد. سپس با استفاده از ترکیب تقسیم بندی غیریکنواخت و تکنیک LUT چند بخشی مقدار تابع بدست می آید. این معماری در یک تراشه FPGA Xilinx پیاده سازی شد. این روش نسبت به الگوریتم CORDIC و روندهای مبتنی بر جداول جستجو از توان عملیاتی بالاتر و مساحت کمتری برخوردار بود [۲۷]. Ukil و همکاران در سال ۲۰۱۱ مقدار تابع $\text{atan2}(y, x)$ را با استفاده از تراشه ARM7 توسط درونیایی مبتنی بر تقریب پاره خطی و جدول جستجو ۱۰۱ نقطه ای، با دقت بسیار بالایی محاسبه نمودند [۱۷]. آنها در طرح پیشنهادیشان پس از استفاده از جدول جستجو، توسط درونیایی خطی دقت را افزایش دادند.

۴- روشهای پیشنهادی

۴.۱- تئوری های اولیه

در روش های پیشنهادی، روند محاسبه تابع $\text{atan2}(Gy, Gx)$ مبتنی بر جداول جستجو Look Up Table/Table Base است. از مزایای این شیوه این است که محاسبات تقسیم Gy/Gx و تابع مثلثاتی معکوس $\tan^{-1}(u)$ توانمان انجام شده و نیازی به تفکیک محاسبات نخواهد بود. از این رو مقادیر گرادیان های عمودی و افقی Gy, Gx آدرس خانه های حافظه را تعیین کرده و محتوای خانه های حافظه، مقدار زاویه متناظر خواهند بود. همچنین از آنجایی که مقدار گرادیان های عمودی و افقی Gy, Gx اعداد شمارشی می باشند؛ در نتیجه تعداد حالات ممکنه محدود بوده و استفاده از روش جدول جستجو بسیار مناسب خواهد بود. از این رو در ابتدا ابعاد بیتی اطلاعات Gy, Gx را بدست می آوریم. در تصاویر خاکستری با عمق شدت روشنایی ۸ بیت، حداکثر مقدار پارامتر گرادیان عمودی Gy هنگامی است که تمامی سه پیکسل واقع در سطر سوم پنجره ۳ در ۳ مشکی بوده یعنی مقادیر شدت روشنایی متناظر با این پیکسل ها صفر باشند و نیز تمامی سه پیکسل واقع در سطر اول پنجره ۳ در ۳ سفید بوده یعنی مقادیر شدت روشنایی متناظر با این پیکسل ها معادل ۲۵۵ باشند. حداقل مقدار پارامتر گرادیان عمودی Gy هنگامی است که اطلاعات تصویری سطر اول را با سطر سوم عوض نماییم (روابط ۹). به همین روش می توان حدود گرادیان افقی Gx را نیز بدست آورد. از این رو مقادیر گرادیان های عمودی و افقی را می توان توسط اعداد شمارشی علامتدار ۱۱ بیتی در سیستم عددی مکمل دو $[-1024, +1024]$ بیان کرد. زوایای نظیر جهت گرادیان تصویر را بر حسب درجه و بر اساس سیستم عددی ممیز ثابت بیان می کنیم (جدول ۳). بنابراین تابع $\text{atan2}(Gy, Gx)$ نگاهی از حوزه اعداد شمارشی علامتدار به اعداد اعشاری $\mathbb{R} \rightarrow \mathbb{Z}^2$ خواهد بود.

$$\begin{aligned} \{Gy \max &= (1 \times 255 + 2 \times 255 + 1 \times 255) - (1 \times 0 + 2 \times 0 + 1 \times 0) = +1020 \\ \{Gy \min &= (1 \times 0 + 2 \times 0 + 1 \times 0) - (1 \times 255 + 2 \times 255 + 1 \times 255) = -1020 \\ \{(Gx, Gy) &|-1020 \leq Gx \leq 1020, -1020 \leq Gy \leq 1020, (Gx, Gy) \in \mathbb{Z}^2\} \rightarrow [0.0^\circ, 360.0^\circ] \end{aligned} \quad (9)$$

جدول ۳: سیستم های عددی اتخاذ شده و دامنه و بُرد تابع $\text{atan2}(Gy, Gx)$

atan2(Gy, Gx) تابع	بُرد تابع atan2(Gy, Gx)
دامنه تابع atan2(Gy, Gx)	بُرد تابع atan2(Gy, Gx)
اعداد شمارشی علامتدار سیستم عددی مکمل دو ۱۱ بیتی [-1020, +1020]	اعداد اعشاری علامتدار ممیز ثابت بدون علامت بر اساس درجه و w بیتی Q(w, f) [0.0°, +360.0°]

۲.۴- روش پیشنهادی

۴.۲.۱- تئوری اصلی روش پیشنهادی

برای یافتن جهت بردار گردایان در تصاویر خاکستری ۸ بیتی مطابق با روش کاهش دامنه تعریف زاویه در ابتدا پارامترهای Gx , Gy را تعیین علامت کرده تا ربع مثلثاتی را بیابیم که زاویه اصلی در آن واقع شده است. سپس قدرمطلق پارامترهای Gx , Gy را محاسبه کرده و در پارامترهای x , y ذخیره می کنیم. تابع $\text{atan2}(y, x)$ دو متغیره است. از این رو در روش جداول جستجو دو پارامتر ورودی x , y وجود دارند که آدرس خانه های حافظه را تعیین می کنند. در این روش پارامترهای x , y به یکدیگر چسبانیده شده تا آدرس خانه حافظه ای از ROM را تعیین کند که پاسخ در آن واقع شده است. این نکته شایان ذکر است که بدین ترتیب نیازی به انجام عمل تقسیم نیست. از روابط (۱۰) در می یابیم که پارامترهای x , y شمارشی بدون علامت و ۱۰ بیتی هستند. از این رو ظرفیت حافظه 2^{20} Word معادل یک مگا کلمه بوده که بسیار بزرگ و غیر کاربردی می باشد.

$$\begin{cases} x = |Gx| \Rightarrow 0 \leq x \leq 1020 \\ y = |Gy| \Rightarrow 0 \leq y \leq 1020 \end{cases} \Rightarrow x \& y \text{ are unsigned and 10 bit.} \quad (10)$$

جهت کاهش تعداد خطوط آدرس از این راهکار استفاده می کنیم. اگر پارامترهای x , y با ابعاد w بیت باشند فضای اطلاعاتی شامل 2^{2w} نقطه خواهد بود. این مجموعه را M می نامیم (رابطه ۱۱). برای کاهش فضای حالت، فقط زوایای نظیر نقاط واقع در منطقه U را در حافظه قرار می دهیم. مجموعه U شامل نقاطی است که به صورت k بیتی بیان می شوند رابطه (۱۲). حال نقاطی که خارج منطقه U هستند، یا در امتداد نقاط واقع در منطقه U می باشند و یا با تقریب توسط نقاط واقع در منطقه U مقدار زاویه متناظر آنها قابل بیان است. اکنون می توانیم زوج مرتب های خارج از منطقه U را به دو گروه زیر دسته بندی کنیم.

۱- گروهی که مختصات آنها ضریب توان دویی از مختصات نقاط داخل ناحیه U هستند (رابطه ۱۳).

۲- گروهی که مختصات آنها ضریب توان دویی از مختصات نقاط داخل ناحیه U نیستند (رابطه ۱۴).

$$M = \{(x, y) | 0 \leq x < 2^w \text{ and } 0 \leq y < 2^w \text{ and } x \in \mathbb{Z} \text{ and } y \in \mathbb{Z}\} \quad (11)$$

$$U = \{(x, y) \in M | 0 \leq x < 2^k \text{ and } 0 \leq y < 2^k\} \quad (12)$$

$$H1 = \left\{ (x, y) \in M | (x, y) \notin U \text{ and } \exists (\alpha, \beta) \in U \mid \frac{y}{\beta} = \frac{x}{\alpha} = 2^e \text{ and } e \in \mathbb{N} \right\} \quad (13)$$

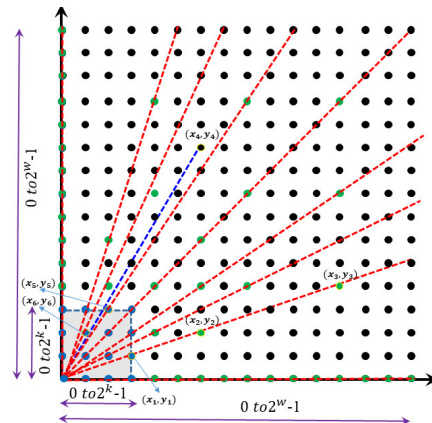
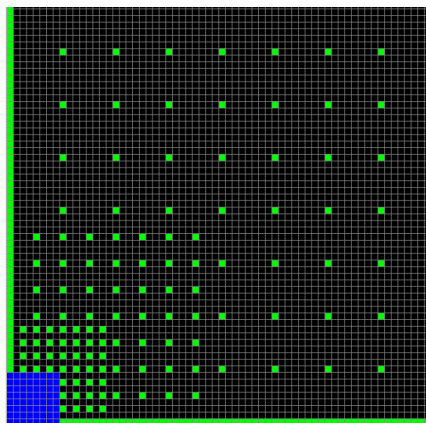
$$H2 = \{(x, y) \in M | (x, y) \notin U \text{ and } (x, y) \notin H1\} \quad (14)$$

با استفاده از رابطه (۱۳) می توان مقدار تابع $\text{atan2}(y, x)$ را برای نقاط واقع در ناحیه $H1$ بدون نیاز به عمل تقسیم از روی نقاط متناظر واقع در ناحیه U و با استفاده از عمل شیفت به سمت چپ به صورت دقیق محاسبه کرد (رابطه ۸). در شکل (۴-الف) این دسته بندی به ازای $k=3$ و $w=6$ نمایش داده شده است. در این شکل نقاط به رنگ آبی متعلق به مجموعه U ، نقاط به رنگ سبز متعلق به مجموعه $H1$ و نقاط سیاه رنگ متعلق به مجموعه $H2$ هستند. کلیه زوایا تبدیل به یکی از ۹ راستایی می شوند که توسط خطوط خط چین قرمز رنگ در شکل (۴-ب) نمایش داده شده اند. این خطوط از حداقل یکی از نقاط اعضای مجموعه U عبور می کنند. همانگونه که در شکل (۴-ب) مشاهده می شود. ارتباط بین نقاط $(x1, y1)$ تا $(x6, y6)$ توسط روابط (۱۵) تا (۱۷) بیان می شود.

$$\text{atan2}(y_1, x_1) = \tan^{-1}\left(\frac{1}{3}\right) = \text{atan2}(y_2, x_2) = \tan^{-1}\left(\frac{2}{6}\right) = \text{atan2}(y_3, x_3) = \tan^{-1}\left(\frac{4}{12}\right) \quad (15)$$

$$\text{atan2}(y_4, x_4) = \tan^{-1}\left(\frac{10}{6}\right), \text{atan2}(y_5, x_5) = \tan^{-1}\left(\frac{3}{2}\right), \text{atan2}(y_6, x_6) = \tan^{-1}\left(\frac{2}{1}\right) \quad (16)$$

$$\text{atan2}(y_5, x_5) \leq \text{atan2}(y_4, x_4) \leq \text{atan2}(y_6, x_6) \quad (17)$$



شکل ۴: الف) گروه های $U, H1, H2$ به ازای $k=3$ و $w=6$ ب) گروه U به ازای $k=2$ و $w=4$

حال در ابتدا برای یافتن تابع $\text{atan2}(y,x)$ توسط روند کاهش محدوده محاسبات به روش مقیاس بندی، صفرهای مشترک با ارزش بیشتر دو پارامتر x, y را با شیفت دادن اطلاعات به سمت چپ حذف می کنیم و این روند را تا هنگامی که بیت های با ارزش بیشتر دو پارامتر x, y صفر است تا حداکثر $w-k$ بار تکرار می کنیم. حال از k بیت با ارزش بیشتر پارامترهای اصلاح شده x, y برای آدرس دهی جدول جستجو استفاده می کنیم و به این ترتیب حجم اطلاعات جدول کاهش چشمگیری پیدا می کند. با این ترفند مقدار تابع $\text{atan2}(y,x)$ برای نقاط متعلق به مجموعه های $U, H1$ به صورت دقیق محاسبه می شود. اما در محاسبه تابع برای نقاط متعلق به مجموعه $H2$ دچار خطا می شویم. از آنجایی که تابع $\tan^{-1}(y/x)$ در ربع اول مثلثاتی اکیداً صعودی است. به ازای $k=4$ و $w=10$ و پارامترهای اصلاح شده x, y مجموعه روابط (۱۸) تا (۲۰) را خواهیم داشت. در جدول ۴ تعداد و درصد هر یک از دسته ها را نسبت به کل نقاط به ازای $w=10$ و k های مختلف نشان داده ایم.

$$\frac{y_9 y_8 y_7 y_6 000000}{x_9 x_8 x_7 x_6 111111} \leq \frac{y_9 y_8 y_7 y_6 y_5 y_4 y_3 y_2 y_1 y_0}{x_9 x_8 x_7 x_6 x_5 x_4 x_3 x_2 x_1 x_0} \leq \frac{y_9 y_8 y_7 y_6 111111}{x_9 x_8 x_7 x_6 000000} \quad (18)$$

$$\text{atan2}(y_9 y_8 y_7 y_6 000000, x_9 x_8 x_7 x_6 111111) \leq \text{atan2}(y_9 y_8 y_7 y_6 y_5 y_4 y_3 y_2 y_1 y_0, x_9 x_8 x_7 x_6 x_5 x_4 x_3 x_2 x_1 x_0) \quad (19)$$

$$\text{atan2}(y_9 y_8 y_7 y_6 y_5 y_4 y_3 y_2 y_1 y_0, x_9 x_8 x_7 x_6 x_5 x_4 x_3 x_2 x_1 x_0) \leq \text{atan2}(y_9 y_8 y_7 y_6 111111, x_9 x_8 x_7 x_6 000000) \quad (20)$$

جدول ۴: تعداد درصد هر یک از دسته ها را نسبت به کل نقاط به ازای $w=10$ و k های مختلف

k	U		H1		H2	
	Total point	percent	Total point	percent	Total point	percent
4	256	0.024%	3072	0.293%	1045248	99.682%
5	1024	0.098%	5664	0.540%	1041888	99.362%
6	4096	0.391%	13952	1.331%	1030528	98.279%
7	16384	1.563%	38272	3.650%	993920	94.788%
8	65536	6.250%	99328	9.472%	883712	84.277%

۴. ۲. معماری روش پیشنهادی اول

در این روش پیشنهادی جهت کاهش حجم حافظه مورد نیاز برای جدول جستجو از ویژگی های منحصر به فرد توابع تقسیم و atan2 استفاده می کنیم. این ویژگی ها در مجموعه روابط (۳ و ۸) مشاهده می شوند. همچنین k را ۴ اختیار کرده و از یک حافظه ۲۵۶ کلمه ای ۸ بیتی استفاده می کنیم. این طرح پیشنهادی شامل سه بخش پیش پردازش، جدول جستجو و پس پردازش است (شکل ۷). روند کار بدین صورت است که در ابتدا در بخش پیش پردازش برای تعیین ربع مثلثاتی که گرادیان در آن واقع است؛ پارامترهای Gx, Gy را تعیین علامت می کنیم. سپس برای استفاده از روش کاهش دامنه تعریف زاویه، قدرمطلق پارامترهای Gx, Gy محاسبه کرده و آنها را در پارامترهای x, y ذخیره می نماییم. از آنجایی که پارامترهای اولیه x, y مثبت بوده بنابراین زوایا در ربع اول مثلثاتی بوده و بین 0° تا 90° خواهند بود. در مرحله کاهش محدوده توسط مقیاس بندی، برای کاهش خطای محاسبات و افزایش دقت، صفرهای مشترک با ارزش بیشتر دو پارامتر x, y را حذف می کنیم. از آنجایی که پارامتر k چهار اختیار شده در نتیجه حافظه مورد نیاز ۲۵۶ کلمه بوده و به ۸ خط جهت آدرس دهی حافظه نیازمندیم. از این رو تا هنگامی که بیت های با ارزش بیشتر اطلاعات ۱۰ بیتی x, y صفر هستند اعداد را تا حداکثر ۶ بیت به سمت چپ شیفت می دهیم. این امر به معنای ضرب در دو کردن اعداد x, y است. فرآیند شیفت دادن اطلاعات به سمت چپ توسط مالتی پلکسرها و مبتنی بر شیفت دهنده های بشکه ای انجام می گیرد. اکنون ۴ بیت با ارزش بیشتر پارامتر x اصلاح شده و ۴ بیت با ارزش بیشتر پارامتر y اصلاح شده را به یکدیگر ملحق نموده تا آدرس ۸ بیتی جدول جستجو بدست آید. این امر به معنای تقسیم بر 2^4 کردن پارامترهای اصلاح شده x, y است که توسط فرآیند برش Truncation بخش اعشاری (۶ بیت با ارزش کمتر) آنها را حذف نموده ایم. از آنجایی که در محاسبات اطلاعات x, y در توان های ۲ ضرب کرده و بر توان های ۲ تقسیم نموده ایم. زاویه متناظر بدست آمده تقریباً برابر با زاویه گرادیان می باشد. بدین ترتیب تمامی نقاط مجموعه های $H1, H2$ به نقاط مجموعه U نگاشت می یابند. شایان ذکر است که راستای نقاط متعلق به مجموعه های $U, H1$ بدون هیچ خطایی محاسبه شده، اما در محاسبه راستای نقاط مجموعه $H2$ دچار خطا خواهیم شد (شکل ۵).

```
x = 00,0101,0010B  2 bit shift left => new x = 01,0100,1000B  a = 0101B
y = 00,1110,1010B  2 bit shift left => new y = 11,1010,1000B  b = 1110B
Mem Addr = y9y8y7y6x9x8x7x6 = 111001010B
tan-1(0011101010B/0001010010B) ≈ tan-1(1110B/0101B) tan-1(234/82) ≈ tan-1(14/5), tan-1(2.8536) ≈ tan-1(2.8)
```

```
x = 00,0000,0010B  6 bit shift left => new x = 00,1000,0000B  a = 0010B
y = 00,0000,0001B  6 bit shift left => new y = 00,0100,0000B  b = 0001B
Mem Addr = y9y8y7y6x9x8x7x6 = 00010010B
tan-1(0000000001B/0000000010B) = tan-1(0001B/0010B) tan-1(1/2) = tan-1(1/2), tan-1(0.5) = tan-1(0.5)
```

شکل ۵: دو مثال از روش مقیاس بندی

مجموعه روابط (۲۱) نحوه پر کردن حافظه ۲۵۶ کلمه ای را بیان می کنند. حال در مرحله جدول جستجو با توجه به آدرس ۸ بیتی بدست آمده مقدار زاویه متناظر از جدول جستجو استخراج می شود.

$$\text{Mem}[y_9 y_8 y_7 y_6 x_9 x_8 x_7 x_6] = \text{atan2}(y_9 y_8 y_7 y_6, x_9 x_8 x_7 x_6) \quad (21)$$

$$\text{Mem}[y_9 y_8 y_7 y_6 x_9 x_8 x_7 x_6] = \begin{cases} \tan^{-1}(y_9 y_8 y_7 y_6 / x_9 x_8 x_7 x_6) & \text{if } x_9 x_8 x_7 x_6 \neq 0 \\ 0^\circ & \text{if } x_9 x_8 x_7 x_6 = 0 \text{ and } y_9 y_8 y_7 y_6 = 0 \\ 90^\circ & \text{if } x_9 x_8 x_7 x_6 = 0 \text{ and } y_9 y_8 y_7 y_6 \neq 0 \end{cases}$$

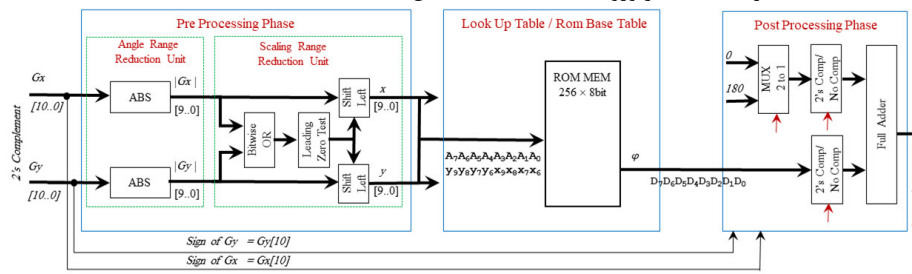
نحوه بیان زاویه بر اساس ابعاد خانه های حافظه است. اگر هر خانه حافظه ۸ بیت باشد آنگاه می توانیم اطلاعات را به صورت $Q(8,1)$ یعنی ۷ بیت شمارشی و ۱ بیت اعشاری ذخیره نماییم. در این صورت بازه اطلاعاتی که در هر خانه حافظه ذخیره می گردد از 0b00000000 تا 0b10110100 است. همان گونه که مشاهده می شود بازه اطلاعاتی ۸ بیتی 0b1010110101 تا 0b11111111 پوشش داده نشده است و دقت زوایای ذخیره شده در حافظه 0.5° خواهد بود. در شکل (۶- الف) محتویات خانه های حافظه و در شکل (۶- ب) زوایایی از ربع اول که بدین صورت در حافظه ذخیره شده اند را نمایش داده ایم. شایان ذکر است که اگر چه فضای حافظه ۲۵۶ خانه است اما به دلیل وجود عناصر تکراری در زوایای ذخیره شده در حافظه، ۱۲۳ عنصر یکتا داریم. خانه های حافظه به آدرس 0b01010001, 0b10100010, 0b11110011 به آدرس 0b01010001, محتویات یکسانی دارند. از سوی دیگر به دلیل غیرخطی بودن تابع atan2 تمامی زوایا با گام های 0.5° مانند $6/0^\circ$ و $13/5^\circ$ در جدول وجود ندارند.

اکنون در مرحله پس پردازش، جهت بازیابی زاویه اصلی به این صورت عمل می کنیم که پس از بدست آوردن زاویه متناظر با قدرمطلق گرادیان های عمودی و افقی، و بر اساس علامت گرادیان ها، مبتنی بر روابط (۴) و جدول ۱ زاویه متناظر اصلی را بدست می آوریم. این واحد صرفاً نیاز به یک جمع کننده/تفریق کننده با امکان انتخاب عملوندهای ورودی دارد. در این پژوهش تمرکز بر روی تراشه XA3S200-4vqg100 از خانواده Spartan3 بوده و پیاده سازی معماری پیشنهادی مبتنی بر زبان توصیف سخت افزار VHDL در بستر ISE14.7 می باشد.

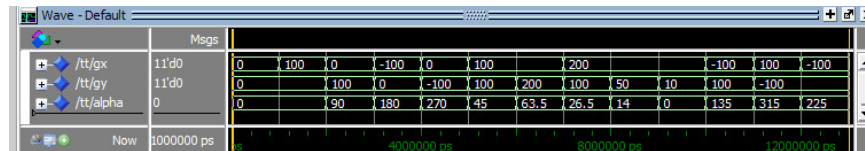


شکل ۶: الف) محتویات خانه های حافظه ۲۵۶ کلمه ای ۸ بیتی ب) زوایای واقع در ربع اول موجود در حافظه جستجو

کدهای VHDL الگوریتم پیشنهادی به ازای $w=10$, $k=4$ و مبتنی بر روش Data Flow و به صورت تمام ترکیبی بوده که در مرجع [۳۰] قرار داده شده اند. در شکل ۷ شمای کلی و معماری سخت افزاری طرح پیشنهادی اول به ازای $w=10$, $k=4$ و در شکل ۸ نتایج شبیه سازی شده توسط بسته نرم افزاری Model Sim برای تعدادی از ورودی های Gx, Gy نمایش داده شده است.



شکل ۷: معماری سخت افزاری طرح پیشنهادی اول



شکل ۸: نتایج شبیه سازی به ازای برخی از زوایا در محیط ModelSim برای طرح پیشنهادی اول

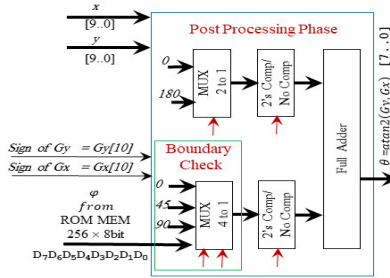
۳. ۲. ۴- معماری روش پیشنهادی دوم

این روش همان روش اول است اما فقط نحوه پر کردن جدول جستجو متفاوت می باشد. به این ترتیب که تمامی زوایای گرادیان هایی را بدست می آوریم که طبق الگوریتم کاهش ابعاد به روش مقیاس بندی به یک آدرس k بیتی یکسان نگاشت می شوند (رابطه ۲۲). اکنون میانگین آن زوایا را در خانه حافظه متناظر ذخیره می نماییم. زوایایی که مقدار i یا j در آنها صفر بوده و یا i یا j برابر باشند به صورت مستقیم بدون نیاز به جدول جستجو قابل محاسبه هستند. از این رو این زوایا در فرآیند متوسط گیری لحاظ نمی شوند.

$$\text{Mem}[y_9 y_8 y_7 y_6 x_9 x_8 x_7 x_6] = \text{Mean}(\text{atan2}(j, i)) \quad \begin{cases} 0000000000B < i \leq 1111111111B \\ 0000000000B < j \leq 1111111111B \\ i \neq j \end{cases} \quad (22)$$

$$\begin{cases} x_9 x_8 x_7 x_6 = \text{Slicing Range Reduction}(i) [9.4] \\ y_9 y_8 y_7 y_6 = \text{Slicing Range Reduction}(j) [9.4] \end{cases}$$

در این شیوه برای کاهش خطا شرایط مرزی را نیز در نظر می گیریم. به این ترتیب که اگر مقدار $|Gx| = |Gy|$ بدون نیاز به مراجعه به حافظه مقدار زاویه ϕ برابر با 45° ، اگر $|Gy| = 0$ باشد زاویه ϕ برابر با 0° و اگر $|Gy| \neq 0$ ، $|Gx| = 0$ باشد زاویه ϕ برابر با 90° خواهد شد. معماری سخت افزاری طرح پیشنهادی دوم به ازای $w=10$ ، $k=4$ مشابه با شکل ۷ بوده و فقط واحد پس پردازش متفاوت است (شکل ۹).



شکل ۹: معماری سخت افزاری بخش پس پردازش طرح پیشنهادی دوم

۵- نتایج پیاده سازی و جمع بندی

کدهای VHDL نظیر معماری های پیشنهادی به ازای حجم های متفاوتی از حافظه و همچنین ۸ و یا ۱۲ بیتی بودن هر خانه حافظه در بسته نرم افزاری ISIM14.7 شبیه سازی و بررسی شدند. روند برنامه آزمونه (Testbench) بدین صورت است که توسط دو حلقه تو در تو مقادیر $Gx = -1020$ to $+1020$ و $Gy = -1020$ to $+1020$ را به معماری اعمال کرده تا کل فضای حالت پوشش داده شود. سپس مقادیر Gx ، Gy به همراه نتایج در یک فایل متنی ذخیره می شوند. در نهایت برنامه ای به زبان متلب (Matlab) محاسبات آماری را انجام داده و اختلاف بین زاویه گرادین واقعی و محاسبه شده را بدست می آورد. نتایج بررسی شبیه سازی کلیه روش های پیشنهادی در جدول ۷ آورده شده است. در جدول ۷ در ستون اول نوع روش پیشنهادی به کار برده شده، ستون دوم شماره روش، ستون سوم تعداد خانه حافظه و تعداد بیت هر خانه حافظه که جدول جستجو در آن ذخیره شده، ستون چهارم حداقل و حداکثر فاصله بین دو زاویه متوالی در فضای حافظه جستجو، ستون پنجم حداکثر قدرمطلق خطای بین زاویه واقعی و زاویه محاسبه شده، ستون ششم معیار متوسط مجذور خطا، ستون هفتم تعداد واحدهای محاسباتی عمومی مورد نیاز در تراشه FPGA برای پیاده سازی طرح و در انتها ستون هشتم حداکثر تاخیر بین سیگنال های ورودی و خروجی آورده شده اند.

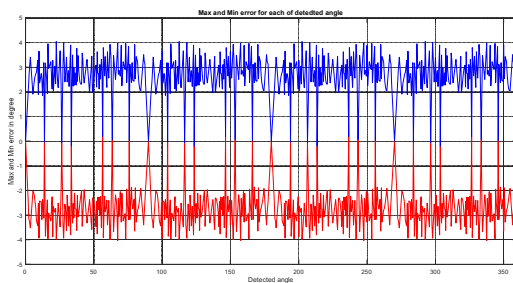
جدول ۷: مقایسه روش های پیشنهادی چهارگانه به ازای $w=10$ و k های مختلف

proposed method	No	Memory	min, max angle	Max Abs Error	MSE	LB	Propagation Delay
First	1	256 * 8 bit v1	0.50, 4.00	7.0148	3.3792	226	34.43 ns
	2	256 * 8 bit + 256 Byte Ext MEM	0.50, 4.00	7.0148	3.3792	101+ Ext 256 Byte Mem	22.64 ns + Ext Mem Access Time
	3a	256 * 8 bit + 256 Byte Int MEM	0.50, 4.00	7.0148	3.3792	454	25.08 ns
	3b	256 * 8 bit + 256 Byte Int RAMB	0.50, 4.00	7.0148	3.3792	101+ 1 RAMB	23.89 ns+2 CP
	4	256 * 8 bit v2	0.35, 3.88	7.0148	3.3936	216	30.68 ns
	5	256 * 12 bit	0.16, 3.81	7.0148	3.3547	311	31.68 ns
	6	1024 * 8bit	0.50, 2.00	3.5148	0.8162	672	40.42 ns
	7	1024 * 12bit	0.03, 1.84	3.4649	0.7969	1079	40.72 ns
	8	4096 * 8bit	0.50, 1.00	1.8601	0.0512	1698	42.67 ns
	9	4096 * 12bit	0.03, 0.91	1.6836	0.0465	3054	43.03 ns
Second	10	256 * 8 bit	0.50, 2.00	4.0352	1.5656	256	33.41 ns
	11	256 * 12 bit	0.03, 2.13	4.1317	1.5450	329	34.31 ns
	12	1024 * 8bit	0.50, 1.00	2.2727	0.4056	696	39.05 ns
	13	1024 * 12bit	0.03, 0.94	2.0829	0.3860	929	41.04 ns
	14	4096 * 8bit	0.50, 0.50	1.2520	0.1165	1759	42.10 ns
	15	4096 * 12bit	0.03, 0.44	1.0265	0.0960	3152	48.18 ns

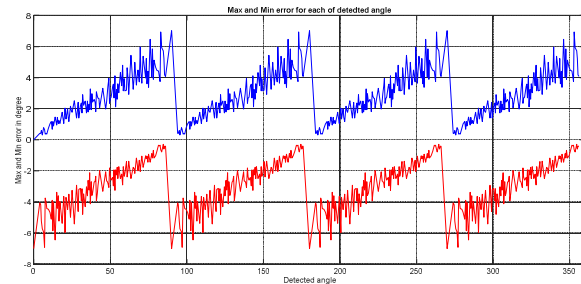
نتایج شبیه سازی طرح پیشنهادی اول به ازای $k=4$ و ذخیره سازی زوایا با قالب Q(8,1) در ردیف 1 آورده شده است. میزان معیار MSE برابر با 3.3792×10^{-3} و حداکثر قدرمطلق خطا برابر با 7.0148×10^{-3} می باشد. در این سخت افزار از ۲۲۶ واحد محاسباتی عمومی استفاده شده است. همچنین جدول جستجو شامل ۱۲۳ زاویه یکتا با گام های $5^\circ/0$ است. در شکل (۱۰-الف) خط قرمز، حداکثر خطای مثبت و خط آبی، حداکثر خطای منفی را نمایش می دهند که به ازای هر زاویه تشخیص داده شده است. اولین نکته این است که ترجیح سنتز کننده ISE 14.7 بر این بود که جدول جستجو تعریف شده توسط بردار ثابت را بوسیله واحدهای محاسباتی عمومی تراشه FPGA پیاده سازی کند. می توان از حافظه خارجی برای ذخیره سازی جداول جستجو استفاده کرد. بدیهی است که حجم سخت افزار کاهش چشمگیری خواهد داشت. لذا در طرح ردیف 2 جدول جستجو خارج از تراشه FPGA لحاظ شده و بدین ترتیب تعداد واحدهای محاسباتی به ۱۰۱ واحد عمومی کاهش پیدا نمود. اما نیاز به یک حافظه موازی ۸ بیتی ۲۵۶ کلمه ای بوده

که زمان دستیابی به محتویات این حافظه به تاخیر افزوده می شود. در طرح ردیف 3a تلاش شد تا از حافظه داخلی استفاده شود که باز هم سنتز کننده توسط واحدهای محاسباتی عمومی این حافظه را پیاده سازی کرد و حجم واحدهای محاسباتی مورد نیاز برای پیاده سازی طرح ردیف 3a به ۴۵۴ واحد افزایش یافت. در طرح پیشنهادی ردیف 3b جهت ذخیره سازی اطلاعات جدول جستجو در حافظه RAMB داخلی تراشه spartan3 از IP Core های از پیش آماده استفاده کردیم. این راهکار باعث شد تا حجم بلوک های محاسباتی عمومی به ۱۰۱ واحد تقلیل یابد اما بدلیل این که حافظه های RAMB ذاتاً ترتیبی هستند به مقدار ۲ پالس ساعت به زمان محاسبه افزوده شد. همان طور که مشاهده می شود معیارهای MSE, Max Abs Error در همه چهار روش پیاده سازی طرح پیشنهادی اول ردیف های 1,2,3a,3b یکسان می باشد. در نسخه دوم از طرح پیشنهادی اول ردیف 4 برای پوشش حداکثری محتوای خانه های حافظه، زوایا را در ضرب اصلاح $255 \div 90$ ضرب نموده و سپس زوایا را پس از فرآیند گرد کردن به صورت ۸ بیتی و شمارشی در جدول جستجو ذخیره می نماییم تا تمامی حالات ممکنه پوشش داده شود. در این روش حداقل فاصله بین زوایای متوالی در حافظه جستجو 0.35° است. مشاهده می شود که تغییر فاحشی در معیارها ایجاد نشده و حتی کمی مقدار معیار MSE نیز بیشتر شده است. اما نسبت به طرح ردیف 1 هم تعداد واحد محاسباتی عمومی ۱۰ واحد کاسته شده و هم زمان تاخیر در حدود ۴ نانو ثانیه کمتر شده است. در طرح ردیف 5 از ۲۵۶ خانه حافظه با عرض ۱۲ بیت استفاده نمودیم تا دقت بیان اطلاعات افزایش یابد. در جدول جستجو اطلاعات با قالب Q(12,5) ذخیره شده است یعنی ۵ رقم اعشار در سیستم عددی ممیز ثابت اتخاذ نمودیم. بدین ترتیب دقت بیان زوایای واقع در حافظه افزایش چشمگیری نسبت به طرح ردیف 1 پیدا می کند. اما از آنجایی که تعداد نقاط واقع در جدول جستجو محدود است با وجود کاهش فاصله بین دو زاویه متوالی که در بهترین شرایط 0.16° شده است، معیار MSE بهبود چشمگیری پیدا نمی کند. در طرح ردیف 6 تعداد عناصر جدول جستجو را بیشتر کردیم $(k = 5)$ معیارهای حداکثر قدرمطلق خطا نسبت به طرح ردیف 1 نصف و MSE تقریباً یک چهارم شده اند. با بررسی ردیف های 6 تا 9 جدول ۷ در می یابیم که افزایش تعداد عناصر جدول جستجو نسبت به افزایش عرض بیتی خانه های حافظه تاثیر بهتری بر معیارهای MSE, MaxAbs Error دارد.

در طرح پیشنهادی دوم محتویات جدول اصلاح گردید. نتایج شبیه سازی به ازای $k = 4$ و ذخیره سازی زوایا با قالب Q(8,1) در ردیف 10 آورده شده است. میزان معیار MSE برابر با $1/5656^\circ$ و حداکثر قدرمطلق خطا برابر با $4/0.352^\circ$ می باشد. در این سخت افزار از ۲۵۶ واحد محاسباتی عمومی استفاده شده است. در شکل (۱۰-ب) خط قرمز حداکثر خطا مثبت و خط آبی حداکثر خطای منفی را به ازای هر زاویه تشخیص داده شده نمایش می دهند. با بررسی شکل (۱۰-ب) مشاهده می کنیم که میزان خطاهای مثبت و منفی در تشخیص زوایا در طرح پیشنهادی دوم برخلاف طرح پیشنهادی اول، تقریباً یکنواخت است. همان گونه که مشاهده می شود صرفاً با عوض کردن محتوای جدول جستجو، حداکثر خطا و مقدار MSE کاهش چشمگیری می یابد. در شکل های (۱۱-ب) و (۱۱-الف) تعداد واحد محاسباتی و معیار MSE به ازای حجم متفاوت حافظه و عرض بیتی متفاوت نمایش داده شده است. با مشاهده دقیق این دو نمودار در می یابیم که فقط افزایش تعداد خانه حافظه تاثیر مثبتی بر کاهش معیار MSE دارد. همان گونه که نتایج نشان می دهند افزایش تعداد خانه های حافظه ناشی از افزایش تعداد خطوط آدرس تاثیر بهتری در کاهش معیار MSE دارد و افزایش ابعاد بیتی خانه های حافظه تاثیر آنچنانی در بهبود معیارها ندارد. با بررسی های به عمل آمده میزان توان مصرفی در کلیه روش های ردیف 1 و 10 در حدود ۲۷ میلی وات می باشد.

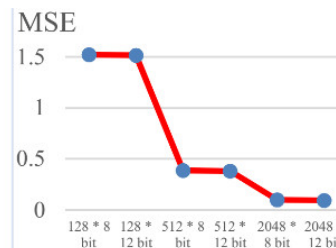
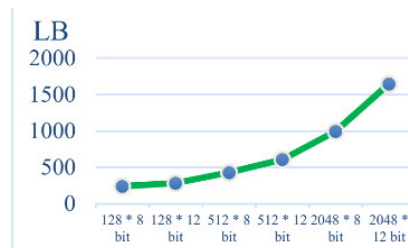


ب) روش پیشنهادی دوم ردیف ۱۰



الف) روش پیشنهادی اول ردیف ۱

شکل ۱۰: میزان حداقل (قرمز) و حداکثر خطا (آبی) در تشخیص زاویه نظیر گرادین های Gx, Gy



شکل ۱۱: الف) میزان تغییرات معیار MSE با تغییر حجم و عرض بیتی حافظه

ب) میزان تغییرات مقدار واحدهای محاسباتی عمومی به کار برده شده در تراشه FPGA با تغییر حجم و عرض بیتی حافظه

۶- مراجع

- [1] Benammar, M., Alassi, A., Gastli, A., Ben-Brahim, L., Touati, F., "New fast arctangent approximation algorithm for generic real-time embedded applications", *Sensors*, No. 19(23), pp. 5148-5163, 2019.
 - [2] Kim, Y., Shahdoost, H., Jadhav, S., Gloster, C.S., "Improving the Accuracy of Arctan for Face Detection", *25th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, IEEE Canada, Apr 2017.
 - [3] Malik, S., H., K., Mirzaei, S., "An Optimized FPGA Based Canny Edge Detection System Using High-Level Synthesis", *IEEE International systems Conference (SysCon)*, IEEE Canada, Apr 2025.
 - [4] Hashemi, M., Mohammed, A., K., Roche, C., "SYCL-based Acceleration of Canny Edge Detector Algorithm Using DPC++", *IEEE 67th International Midwest Symposium on Circuits and Systems (MWSCAS)*, IEEE USA, Aug 2024.
 - [5] Guo, L., Wu, S., "FPGA Implementation of a Real-Time Edge Detection System Based on an Improved Canny Algorithm", *Applied Sciences*, No. 13(2), pp 870-887, 2023.
 - [6] Khalil, A., S., Shalaby, M., Hegazi, E., "An Enhanced System on Chip-Based Sobel Edge Detector", *International Telecommunications Conference (ITC-Egypt)*, Alexandria, Egypt, Jul 2023.
 - [7] Kromichev, D., "FPGA Based Edge Detection: Integer Inverse Tangent Algorithm", *16th International Engineering Conference on Communications, Electromagnetics and Medical Applications (CEMA'22)*, Bulgaria, Oct 2022.
 - [8] Abdullah, H., A., T., Younis, B., M., K., "FPGA implementation of canny edge detection algorithm for X-ray images", *1ST SAMARRA INTERNATIONAL CONFERENCE FOR PURE AND APPLIED SCIENCES (SICPS2021)*, March 2021.
 - [9] Milić, M., Nenad, B., "Hardware Accelerators for Calculating the Magnitude and Angle of Image Gradient", *29th Telecommunications Forum (TELFOR)*, IEEE Serbia, Nov 2021.
 - [10] Abed, S., "Implementation of an Edge Detection Algorithm using FPGA Reconfigurable Hardware", *Journal of Engineering Research*, No. 1, pp 179-197, 2020.
 - [11] Kusaka, T., Tanaka, T., "Fast and Accurate Approximation Methods for Trigonometric and Arctangent Calculations for Low-Performance Computers", *Electronics*, No. 1(15), pp. 2285-2301, 2022.
 - [12] De Dinechin, F., Istioan, M., "Hardware implementations of fixed-point Atan2", *22nd Symposium on Computer Arithmetic*, IEEE France, Jun 2015.
 - [13] Muller, J.M., "Elementary functions and approximate computing", *Proceedings of the IEEE*, No. 108(12), pp. 2136-2149, 2020.
 - [14] Hsiao, S.F., Wen, C.S., Chen, Y.H., Huang, K.C., "Hierarchical multipartite function evaluation", *IEEE Transactions on Computers*, No. 66(1), pp. 89-99, 2016.
 - [15] Meyer-Baese, U.; *Digital Signal Processing with Field Programmable Gate Arrays*, Springer, 2014.
- [۱۶] هارونی، م، سنائی، پ، "طراحی و پیاده‌سازی یک ماشین حالت محدود جهت محاسبه تابع مثلثاتی تانژانت معکوس مبتنی بر سری تیلور عقب‌رونده و با استفاده از دو واحد ضرب‌کننده DSP48-E بر روی تراشه‌های FPGA شرکت AMD-XILINX به صورت زمان متغیر"، سی و سومین کنفرانس بین المللی مهندسی برق، ایران، اردیبهشت ۱۴۰۴.
- [17] Ukil, A., Shah, V.H., Deck, B., "Fast computation of arctangent functions for embedded applications: A comparative analysis". *International Symposium on Industrial Electronics*, IEEE Poland, Jun 2011.
 - [18] Saber, M., Jitsumatsu, Y., Kohda, T., "A low-power implementation of arctangent function for communication applications using FPGA", *Fourth International Workshop on Signal Design and its Applications in Communications*, IEEE Japan, Oct 2009.
 - [19] Zeyad, O., "Design and Implementation Hardware Architecture for Four-Quadratic Arctangent", *International Journal of Computer Applications*, No. 176, pp. 10-13, 2017.
 - [20] Sharma, R., Thakre, P., "A CORDIC Architecture Implementation for Rectangular to Polar Conversion", *International Journal of Scientific Research in Science and Technology (IJSRST)*, No. 4(8), pp. 389-395, 2018.
 - [21] Pilato, L., Fanucci, L., Saponara, S., "Real-time and high-accuracy arctangent computation using CORDIC and fast magnitude estimation", *Electronics*, No. 6(1), pp. 22-32, 2017.
 - [22] Meher, P.K., Valls, J., Juang, T.B., Sridharan, K., Maharatna, K., "50 Years of CORDIC: Algorithms, Architectures, and Applications", *IEEE Trans. Circuit Syst. I*, No. 56, pp. 1893-1907, 2009.
 - [23] Zhu, H., Ge, Y., Jiang, B., "Modified CORDIC algorithm for computation of arctangent with variable iterations", *13th International Conference on Signal Processing (ICSP)*, IEEE China, Nov 2016.
 - [24] Torres, V., Valls, J., Canet, M.J., "Optimised CORDIC-based atan2 computation for FPGA implementations", *Electronics Letters*, No. 53(19), pp. 1296-1308, 2017.
 - [25] Gener, Y.S., Gören, S., Ugurdag, H.F., "Lossless look-up table compression for hardware implementation of transcendental functions", *27th International Conference on Very Large Scale Integration (VLSI-SoC)*, IFIP/IEEE Peru, Oct 2019.
 - [26] Gutierrez, R., Torres, V., Valls, J., "FPGA-implementation of atan (Y/X) based on logarithmic transformation and LUT-based techniques", *Journal of Systems Architecture*, No. 56(11), pp. 588-596, 2010.
 - [27] Gutierrez, R., Valls, J., "Low-power FPGA-implementation of atan (Y/X) using look-up table methods for communication applications", *Journal of Signal Processing Systems*, No. 56, pp. 25-33, 2009.
 - [28] Torres, V., Valls, J., "A fast and low-complexity operator for the computation of the arctangent of a complex number", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, No. 25(9), pp. 2663-2667, 2017.
 - [29] Emrani, Z., Bateni, S. Rabbani, H., "A new parallel approach for accelerating the gpu-based execution of edge detection algorithms", *Journal of Medical Signals & Sensors*, No 7(1), pp.33-42, 2017.
 - [30] https://drive.google.com/drive/folders/1yW8vpPO8ZG7KG8WaZHKfoNLPbO3_8v8L?usp=sharing