به نام خدا

دانشگاه تهران

دانشکدهٔ مهندسی برق و کامپیوتر

پروژهٔ نهایی درس

طراحی کامپیوتری سیستم های دیجیتال

Pipeline CORDIC Algorithm in Vectoring Mode

مدرس: دکتر بهاروند

اعضای گروه:

غزل کلهر، شقایق لادنی، الهام ابوالحسنی

بهار ۹۹

فهرست مطالب

فهرست تصاویر

فهرست جداول

جدول فعالیت اعضای گروه

	ارزیابی اولیه	پیاده سازی	تست	سنتز	گزارش
غزل کلهر	نوشتن مدل طلایی و یافتن الگوریتم	ماژول‌های control_unit و interface و cordic و rotator و Adder و ALU	اسکریپت پایتون و TCL و تست کلی و تست واحد برای ماژول ها	انجام پیاده‌سازی و اجرای تست فانکشنال پس از پیاده‌سازی	تدوین صفحات و نگارش بخش تست و اسکریپت و پیاده‌سازی و نحوه عملکرد کلی مدار و رسم شماتیک کامپیوتری ماژول‌ها
شقایق لادنی	یافتن مدل طلایی	ماژول‌های mux_4_to_1 و shift_right_var	تست واحد برای ماژول ها	انجام سنتز و اجرای تست فانکشنال پس از پیاده‌سازی	نگارش بخش سنتز و چکیده و پایه ریاضی و توصیف ماژول‌ها
الهام ابوالحسنی	یافتن مدل طلایی	ماژول‌های sign و register و abs	تست واحد برای ماژول ها	انجام سنتز	نگارش بخش سنتز و تاریخچه و کاربردها و پایه ریاضی و توصیف ماژول‌ها

۱ مقدمه

۱-۱ چکیده

در این پروژه، هدف پیاده سازی الگوریتم CORDIC[1] به صورت pipeline و در حالت برداری[2] برای محاسبه ی زاویه یک بردار با محور x است. منظور از pipeline بودن الگوریتم، تکرارپذیری[3] مراحل آن برای محاسبه ی زاویه ی بردار است. در این الگوریتم دیتاهای ورودی و خروجی با اندازه‌ی پارامتری و از نوع علامت دار تعریف شده است و ورودی ها نشان دهنده ی مختصات بردار مورد نظر و خروجی، زاویه ی بردار با محور x است. همچنین الگوریتم پیاده سازی شده تنها بر مبنای عملیات جمع و شیفت می‌باشد و از هیچ عملوند دیگری نظیر ضرب و تقسیم و ... در آن استفاده نشده است. این امر باعث می‌شود که استفاده از این الگوریتم باعث کاهش گیت‌های منطقی، کاهش پیچیدگی‌های سخت افزاری و بهبود سرعت عملکرد در سخت افزار شود. در ادامه هر یک از بخش‌ها و مباحث این الگوریتم به تفصیل شرح داده شده است.

۱-۲ تاریخچه و کاربردها

الگوریتم CORDIC که با نام های digit-by-digit method و Volder's algorithm نیز شناخته می‌شود نخستین بار توسط J. E Volder در سال 1956 در دپارتمان aero electronics در Convair[4] ارائه شد تا analog resolver های موجود در کامپیوتر ناوبری بمب افکن B-58 با یک راه حل دیجیتالی دقیق تر و کارآمدتر جایگزین شود. به همین دلیل است که از CORDIC گاهی به عنوان digital resolver یاد می‌شود.

ولدر در تحقیقات خود از فرمولی در کتابچه CRC[5] شیمی و فیزیک الهام گرفته است. تحقیقات او منجر به یک گزارش فنی شد که در آن الگوریتم CORDIC را برای حل توابع sine, cosine به همراه یک نمونه اولیه پیاده سازی آن، ارائه داد.در این گزارش همچنین امکان محاسبه مختصات چرخشی هیپربولیک و توابع لگاریتمی و نمایی با استفاده از الگوریتم های اصلاح شده CORDIC نیز مورد بحث قرار گرفته است. استفاده از این الگوریتم برای

[1] COordinate Rotation DIgital Computer

[2] Vectoring

[3] Iterative

[4] American aircraft manufacturing company

[5] the 1946 edition of the *CRC Handbook of Chemistry and Physics*

محاسبه ضرب و تقسیم نیز در این زمان متصور شده بود. حتی همکار ولدر در Convair نیز بر اساس اصول این الگوریتم، الگوریتم های تبدیل بین اعداد اعشاری و اعداد باینری رمزنگاری شده[6] را ابداع کرد.

پس از این گزارش در سال 1958، Convair شروع به ساختن سیستمی برای رفع مشکلات رادارها به نام CORDIC I کرد و این پروژه در سال 1360 وقتی تکمیل شد که ولدر شرکت را ترک کرده بود. مدل های جهانی دیگری از جمله CORDIC II نیز توسط سایر همکاران ولدر ساخته و آزمایش شدند.

الگوریتم CORDIC برای اولین بار در سال 1959 به طور عمومی منتشر شد که باعث شد خیلی زود در کامپیوتر های ناوبری شرکت های زیادی از جمله Martin-Orlando, Computer Control, Litton, Kearfott, Lear-Siegler, ... مورد استفاده قرار گیرد.

بعد از آن ولدر با Malcolm MacMillan برای ساختن آتنا[7]، ماشین حسابی رومیزی[8] که از الگوریتم باینری CORDIC استفاده میکرد، همکاری کرد. John Stephen Walther نیز در سال 1971 این الگوریتم را به Unified CORDIC تعمیم داد که می‌توانست توابع هیپربولیک، نماهای طبیعی، لگاریتم های طبیعی، ضرب ها، تقسیم ها و ریشه های مربع را محاسبه کند.

CORDIC یک الگوریتم تکراری است که از روش fixed vector rotation، به منظور ارزیابی و محاسبه توابع مثلثاتی استفاده می‌کند. با فناوری امروزه و محدودیت هایی که در حوزه های توان، فرکانس کارکرد و مصرف انرژی دارد این الگوریتم راه حل سخت افزاری کارآمدی را ارائه کرده است چرا که در آن ضرب کننده ها با جمع کننده ها و شیفت دهنده ها جایگزین شدند که این کار باعث کاهش تعداد گیت ها، پیچیدگی محاسباتی و هزینه سخت افزاری شد و منجر به این شد که محاسبه با استفاده از تقریب چند جمله ای کمتر در کاربرد های real time مورد استفاده قرار گیرند.

این الگوریتم دارای دو حالت است که rotation و vectoring است که ما در این پروژه به پیاده سازی حالت vectoring آن پرداخته ایم. در حالت rotation، وکتور ورودی با استفاده از زاویه های از پیش تعیین شده ای چرخانده می‌شود و در نهایت جمع این زاویه ها برابر با زاویه مورد نظر می‌شود. در حالت vectoring اما مختصات مستطیلی به مختصات قطبی[9] تبدیل می‌شود و بعد به محاسبه زاویه می‌پردازد .CORDIC برای چندین کار محاسباتی از قبیل محاسبه توابع مثلثاتی، هیپربولیک و لگاریتمی، ضرب های واقعی و پیچیده، تقسیم، محاسبه ریشه

[6] binary-coded decimal (BCD)

[7] Athena

[8] fixed-point desktop calculator

[9] Rectangular coordinate

مربعی، راه حل سیستم های خطی، برآورد مقادیر ویژه، تجزیه ارزش منحصر به فرد، فاکتور QR و ... استفاده می‌شود. استفاده از این الگوریتم سریعتر از سایر روش ها برای محاسبات است زمانی که ضرب کننده نداشته باشیم یا تعداد گیت هایی که باید استفاده کنیم باید کم باشند اما به طور کلی استفاده را ضرب کننده ها سرعت بیشتری دارد .

در دهه گذشته این الگوریتم مورد توجه گسترده آکادمی ها و صنعت برای کاربرد های مختلفی از جمله پردازش سیگنال دیجیتال[10] ، پردازش تصویر، گرافیک سه بعدی، شبکه های عصبی، سیستم های MIMO[11] قرار گرفته است. در سال های اخیر نیز استفاده از این الگوریتم برای کاربرد های مختلف زیست پزشکی به خصوص با استفاده از FPGA به طور گسترده ای مورد استفاده قرار گرفته است.

این الگوریتم همچنین در ARM-based STM32G4، پردازنده های Intel سری 8087، 80287، 80387 تا 80486 عمدتا به عنوان روشی به منظور کاهش تعداد گیت ها در واحد محاسبه ی ممیز شناور استفاده می‌شد اما از آنجایی که بیشتر پردازنده های امروزی دارای رجیستر های ممیز شناور با قابلیت تفریق، ضرب، تقسیم ، sine cosine، ریشه مربع و لگاریتم طبیعی هستند دیگر نیاز به این الگوریتم ها در نرم افزار احساس نمی‌شود و فقط در میکروکنترلر ها و یا برنامه های نرم افزاری که به قابلیت اطمینان خاص نیاز دارند یا با محدودیت مواجه هستند مورد استفاده قرار می‌گیرد.

۱-۳ نحوه عملکرد کلی

۱-۳-۱ ماژول های اصلی

ماژول های اصلی از دید کلی به شرح زیر است:

۱-۳-۱-۱ ماژول ورودی

این ماژول مولفه های X و Y را که ورودی های سیستم هستند را دریافت می‌کند. سپس با بررسی علامت مولفه های بردار، ربع مثلثاتی که در آن قرار گرفته است را تعیین می‌کند و سپی قدرمطلق این مولفه ها را برای انجام محاسبات در اختیار سایر ماژول ها قرار می‌دهد.

[10] DSP

[11] Multiple-Input Multiple-Output

۱-۳-۱-۲ ماژول‌های عملیات

این ماژول ها مولفه های بردار را دریافت کرده و با استفاده از دوران و میل دادن مولفه Y بردار به 0، زاویه این بردار با محور X را محاسبه می‌کند.

۱-۳-۱-۳ ماژول خروجی

این ماژول زاویه محاسبه شده را دریافت می‌کند و با توجه به ربع مثلثاتی بردار که در ماژول ورودی تعیین شده بود، این زاویه را با آفست موردنظر جمع می‌کند

۱-۳-۲ مدل طلایی

مدل طلایی سیستم به زبان پایتون نوشته شده است. این مدل به صورت iterative است و با همان الگوریتمی که در توصیف سخت افزاری به کار رفته است کار می‌کند. برای نوشتن کد آن از یک تابع و دیکشنری استفاده شده است که توضیحات آن‌ها به شرح زیر است:

۱-۳-۲-۱ تابع vector_mode

این تابع به ترتیب مولفه‌های X و Y بردار و تعداد تکرار[12] را به عنوان ورودی دریافت می‌کند و پس اجرای عملیات، زاویه محاسبه شده را به عنوان خروجی برمی‌گرداند.

۱-۳-۲-۲ دیکشنری ROM_lookup

این دیکشنری به منظور ذخیره کردن زاویه‌های موردنیاز در الگوریتم نوشته است و تابعی مشابه با آن در توصیف سخت افزاری نوشته شده است. نحوه عملکرد آن به این صورت است که مرحله الگوریتم را به عنوان کلید دریافت می‌کند و زاویه متناظر با این مرحله را برمی‌گرداند

۱-۴ پایه ریاضی

در این قسمت پایه ی ریاضی الگوریتم CORDIC را شرح می‌دهیم. زاویه ی چرخش بردار از 0 تا 360 درجه و بر اساس درجه گزارش می‌شود. در بخش اول به توصیف کلی الگوریتم شرح داده شده است و در بخش دوم به بحث درباره ی زاویه بردار می‌پردازیم.

[12] iteration

۱-۴-۱ توصیف کلی الگوریتم CORDIC

این الگوریتم با مراحل تکرارشونده بردار مورد نظر را چرخش می‌دهد.اگر فرض کنیم ϕ زاویه ی چرخش بردار و همچنین X و Y مختصات بردار موردنظر باشد، مختصات جدید بردار چرخش یافته با زاویه ی ϕ در صفحه ی دکارتی به صورت زیر خواهد شد.

$$X' = x \cos\phi - y \sin\phi$$
$$Y' = y \cos\phi + x \sin\phi$$

حال برای اینکه بخواهیم معادلات را فقط با عملیات جمع و شیفت انجام دهیم، باید دو طرف معادلات بالا را بر cosine تقسیم کنیم. در نتیجه با ساده سازی این معادلات به فرم دیگری از این دو معادله می‌رسیم که می‌توان X' و Y' را برحسب $\tan\phi$ بیان کرد که این نمایش به شکل زیر است:

$$X' = \cos\phi \,.\, [x - y \tan\phi]$$
$$Y' = \cos\phi \,.\, [y + x \tan\phi]$$

در اینجا مقدار $\tan\phi$ برابر با $\pm 2^{-i}$ خواهد بود. پس مقدار های x و y باید در توان هایی از دو ضرب شود. این ضرب را می‌توان فقط با شیفت دادن مقادیر x و y انجام داد.در واقع زاویه ی اصلی چرخش بردار، بعد از چندین مرحله تکرار چرخش با زاویه های کوچکتر به دست می‌آید. مقدار i تعداد مراحل تکرار و جهت چرخش بردار را نشان می‌دهد. پس با توجه به این توضیحات فرم معادلات به صورت زیر در می‌آید:

$$X_{i+1} = K_i [x_i - y_i \,.\, d_i \,.\, 2^{-i}]$$
$$Y_{i+1} = K_i [y_i + x_i \,.\, d_i \,.\, 2^{-i}]$$

که در آن:

$$K_i = \cos(\tan^{-1}.2^{-i}) = 1/\sqrt{1 + 2^{-2i}}$$
$$d_i = \pm 1$$

با حذف مقدار K_i از معادله ی بالا، تمام معادلات فقط با عملیات جمع و شیفت امکان پذیر می‌شود. اگر مقدار i به سمت بی‌نهایت برود، مقدار K_i تقریبا برابر با 0.6073 خواهد شد.

$$K = \prod_{n} K_i$$

پس با توجه به مقدار i در نظر گرفته شده و حذف K_i از معادله، مختصات بردار جدید با کمی اختلاف به دست می‌آید.همچنین زاویه ی خروجی در هر مرحله، با استفاده از معادله‌ی زیر به دست می‌آید.

$$Z_{i+1} = Z_i - d_i . tan^{-1}(2^{-i})$$

به طور کلی الگوریتم CORDIC در دو حالت تعریف می‌شود:حالت برداری و چرخشی [13].
در این پروژه فقط به حات برداری آن می‌پردازیم که در آن بردار ورودی به سمت محور xها چرخش می‌یابد تا در نهایت زاویه آن با محور xها به دست آید. پس باید مقدار y بردار به صفر میل کند. ورودی ها و زاویه ی خروجی در هر مرحله ی تکرار به صورت زیر خواهد بود:

$$x_{i+1} = x_i - y_i . d_i . 2^{-i}$$
$$y_{i+1} = y_i + x_i . d_i . 2^{-i}$$
$$z_{i+1} = z_i - d_i . tan^{-1} . (2^{-i})$$

که در آن:

$$d_i = \begin{cases} +1 & yi > 0 \\ -1 & otherwise \end{cases}$$

در آخرین مرحله ی تکرار و بعد از n مرحله، بردار بر روی محور x قرار می‌گیرد. پس مقدار x بردار برابر با طول بردار خواهد شد و مقدار y صفر است. به دلیل وجود مقدار K_i که برای محاسبه ی مختصات جدید صرف نظر شد، طول بردار نیز با اختلاف A_i به دست می‌آید. مقدار خروجی نیز به صورت زیر است.

$$X_n = A_n \sqrt{x0^2 + y0^2}$$
$$Y_n = 0$$
$$Z_n = z_0 + tan^{-1}(y_0/x_0)$$
$$A_n = \prod_n \sqrt{1 + 2^{-2i}}$$

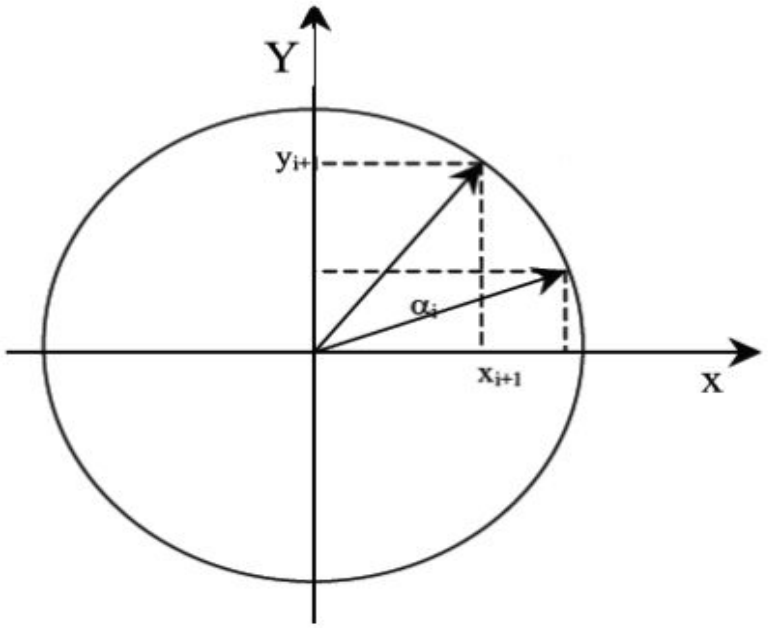
۱-۴-۲ محاسبات زاویه ای الگوریتم CORDIC

اگر فرض کنیم زاویه ای که می‌خواهیم تابع مثلثاتی را برای آن محاسبه کنیم θ باشد، این زاویه را می‌توان از جمع تعدادی زوایای کوچکتر α_i تشکیل داد و در هر مرحله بردار به اندازه ی α_i دوران می‌یابد . یعنی:

$$\theta = \sum_{i}^{n-1} di.\alpha i$$

دوران بردار بر حسب α_i در شکل ۱ نشان داده شده است.

[13] Rotation



شکل ۱ : دوران بردار به اندازه‌ی α_i

برای سهولت در انجام محاسبات مقدار α_i را برابر با

$$\alpha_i = \tan^{-1}(2^{-i})$$

انتخاب می‌کنیم. بنابراین چنانچه محاسبات در مبنای 2 انجام شود، عباراتی که باید در

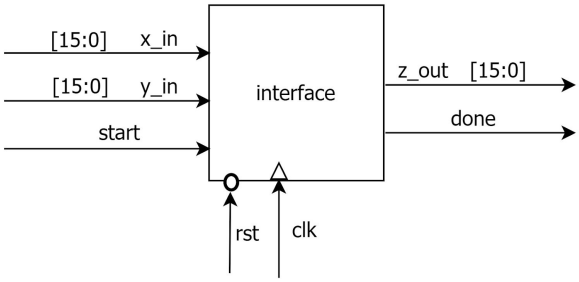
$$2^{-i} = \tan \alpha_i$$

ضرب شوند، فقط شیفت داده می‌شوند. همچنین مقدار d_i می‌تواند -1 یا +1 باشد. در انتها پس از n مرحله، محاسبات به سمت θ مورد نظر همگرا می‌شود.

۲ توصیف معماری سیستم

۲-۱ ماژول interface

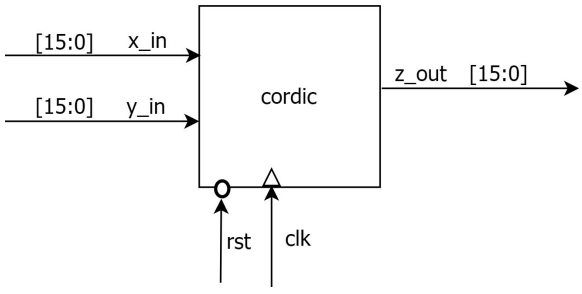
این ماژول به نوعی ماژول اصلی سیستم محسوب می‌شود و یک اینترفیس برای استفاده از سیستم است. نحوه عملکرد آن به این ترتیب است که به عنوان یک اینترفیس ورودی، ورودی های موردنظر کاربر را که مولفه های X و Y بردار هستند را در فرمت باینری ۱۶ بیتی با دقت واحد دریافت می‌کند و در اختیار سیستم قرار می‌دهد. همچنین به عنوان یک اینترفیس خروجی، خروجی سیستم را در فرمت باینری ۱۶ بیتی با دقت ۷ بیت اعشار به کاربر می‌دهد. این ماژول اینترفیس بین مسیر داده و واحد کنترل را برقرار می‌کند.



شکل ۲ : بلوک دیاگرام ماژول interface

۲-۲ ماژول cordic

این ماژول وظیفه پیاده‌سازی الگوریتم شرح داده شده در بخش های قبلی را دارد. برای پیاده‌سازی الگوریتم از روش پایپ‌لاین استفاده شده است. این ماژول از یک تابع به نام برای نگاشت بین مرحله و زاویه استفاده می‌کند. وظیفه تعیین ربع مثلثاتی بردار و جمع کردن آفست موردنظر با نتیجه نهایی نیز بر عهده این ماژول است. ماژول اصلی که در هر stage این الگوریتم تولید می‌شود rotator است که توضیح آن در ادامه آورده شده است.

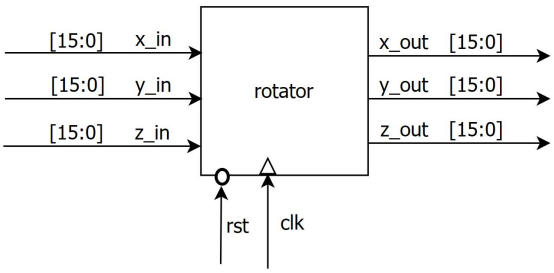


شکل ۳ : بلوک دیاگرام ماژول cordic

۲-۲-۱ ماژول rotator

این ماژول دارای دو پارامتر ITERATION و PHASE است که به ترتیب نشان دهنده تعداد مراحل تکرار و $tan^{-1} . (2^{-i})$ مربوط به آن مرحله است.

در این جا سه ورودی با نام های x_in, y_in, z_in داریم و سه خروجی x_out, y_out, z_out در نظر گرفته شده است.



شکل ۴ : بلوک دیاگرام ماژول rotator

در ابتدای این ماژول یک assign داریم که منفی علامت y_in در هر مرحله را تعیین می‌کند.بعد از آن یک نمونه از ماژول sign داریم که علامت y_in را تعیین می‌کند و از خروجی آن در قسمت قبل استفاده می‌کنیم.

$$d_i = \begin{cases} +1 & yi > 0 \\ -1 & otherwise \end{cases}$$

در واقع دو قسمت بالا برای ما di, -di را ایجاد می‌کنند تا در مراحل بعدی از آنها استفاده کنیم.پس از آن سه نمونه از ALU داریم که برای انجام عملیات های زیر از آنها استفاده شده است .

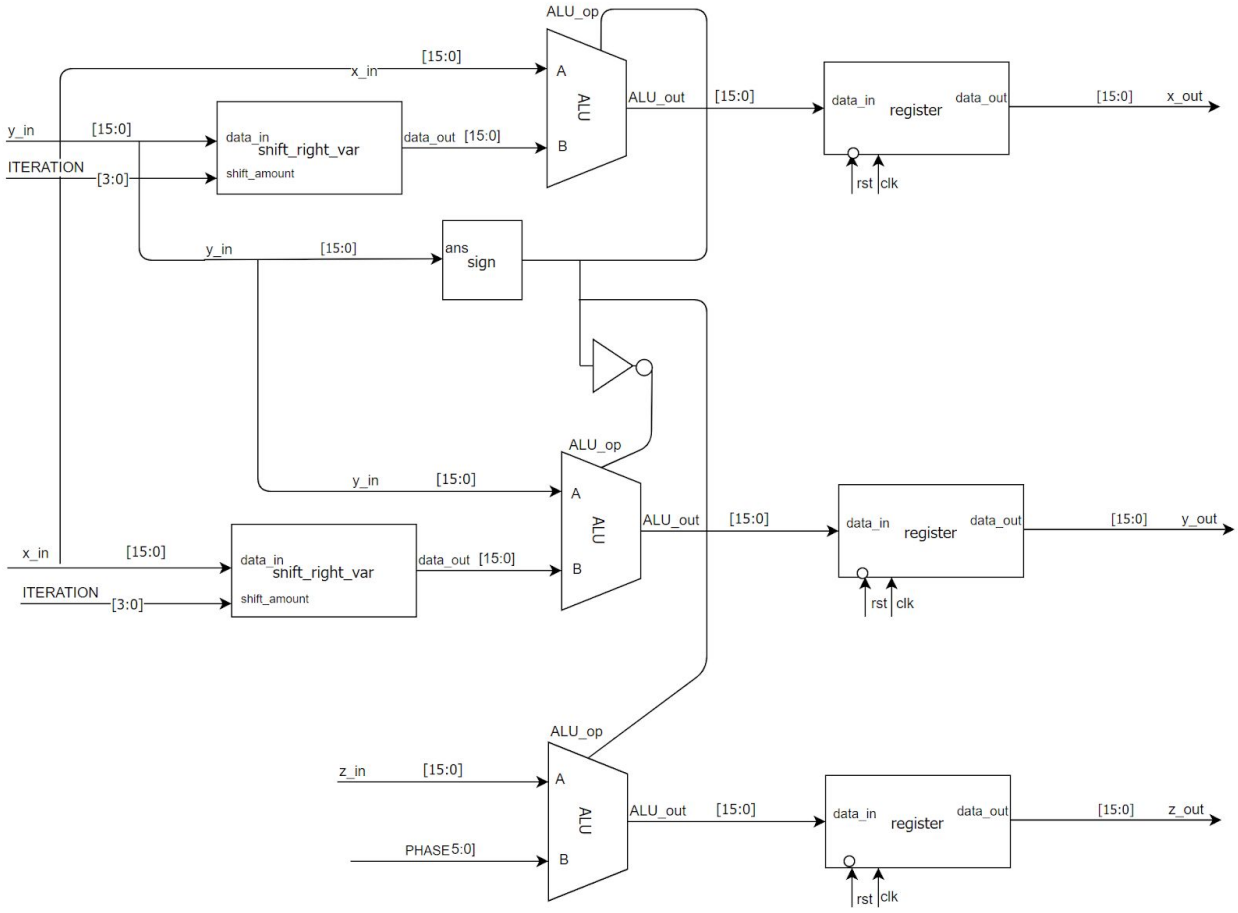
$$x_{i+1} = x_i - y_i \,.\, d_i \,.\, 2^{-i}$$
$$y_{i+1} = y_i + x_i \,.\, d_i \,.\, 2^{-i}$$
$$z_{i+1} = z_i - d_i \,.\, tan^{-1} \,.\, (2^{-i})$$

به این صورت که ALU اول، مقدار شیفت یافته y_in و نوع عملیاتی که باید انجام دهد با توجه به di محاسبه شده در قسمت قبل و x_in را گرفته و محاسبه فرمول اول را انجام می‌دهد. ALU دوم نیز فرمول دوم را شبیه ALU

اول محاسبه می‌کند. ALU سوم نیز مشابه ALU اول است ولی به جای مقدار شیفت یافته از PHASE که به صورت پارامتر به این ماژول داده شده بود استفاده می‌کند تا فرمول سوم را محاسبه کند.

بعد از آن سه register داریم که خروجی ALU ها را در هنگام لبه بالا رونده clk روی خروجی های rotator قرار می‌دهند.

در آخر نیز دو نمونه شیفت دهنده داریم که مقادیر شیفت یافته x_in, y_in مورد نیاز برای ALU ها را محاسبه می‌کنند.

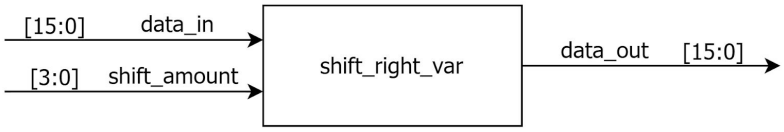


شکل ۵ : قسمت های ترکیبی و ترتیبی ماژول rotator

۲-۲-۲ ماژول shift_right_var

این ماژول یک shifter به سمت راست است که برای تقسیم بر توان های دو و برای عدد های علامت دار طراحی شده است. متغیر shift_amount، نشان دهنده ی تعداد شیفت مورد نظر و data_in یک عدد علامت‌دار با سایز WORD_WIDTH است که به عنوان ورودی به ماژول داده می‌شود. خروجی و همچنین رجیستر data_out نیز

تعبیه شده است که سایز آن با مقدار ثابت WORD_WIDTH مشخص شده است و این خروجی به عنوان عددی علامت‌دار در نظر گرفته شده است.

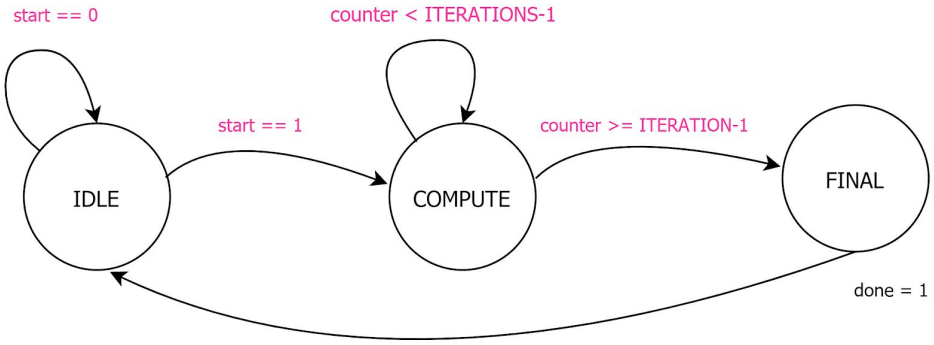


شکل ۶ : بلوک دیاگرام ماژول shift_right_var

در این ماژول با هر تغییر روی ورودی، هر بار data_out برابر با ورودی data_in قرار می‌گیرد و بعد از آن، data_out برابر با shift_amount بار(تعداد شیفت مورد نظر)، به راست شیفت داده می‌شود و در نهایت برای تعیین علامت، با عددی برابر با رقم پر ارزش data_out به همراه تعداد WORD_WIDTH صفر در سمت راست آن، or خواهد شد.

۴-۲ واحد کنترل

ماژول control_unit با دریافت سیگنال start کار خود را آغاز می‌کند و به حالت COMPUTE منتقل می‌شود و تا زمانی که ۱۶ کلاک سپری شود در این حالت باقی می‌ماند. در نهایت با توجه به این که محاسبات خاتمه یافته است به حالت FINAL منتقل می‌شود و سیگنال done را فعال می‌کند که خاتمه کار مدار را اعلام کند.



شکل ۷ : واحد کنترل

۲-۴ واسط کاربری

یک واسط کاربری به زبان پایتون برای تولید نمونه‌های تصادفی و اجرای مدل طلایی نوشته شده است. واسط کاربری دیگری به زبان TCL برای شبیه‌سازی مدار نوشته شده است.

۲-۴-۱ اسکریپت پایتون

در کد واسط کاربری به ازای هر تست‌کیس ابتدا مولفه های بردار به صورت تصادفی در بازه مجاز مشخص شده تولید می‌شوند و با فرمت رشته باینری در فایل های موردنظر نوشته می‌شوند. سپس این ورودی ها به مدل طلایی داده می‌شود و خروجی آن در فایل python_phase.txt نوشته می‌شود. علاوه بر نوشته شدن خروجی در فایل، در ترمینال نیز اطلاعات خروجی با جزئیات کافی (شامل مولفه‌های بردار و مقدار زاویه) چاپ می‌شود.

۲-۴-۲ اسکریپت TCL

این اسکریپت ابتدا تمامی فایل ها را کامپایل می‌کند. پس از کامپایل فایل ها شبیه‌سازی را آغاز می‌کند و سیگنال های موردنظر را به Wave اضافه می‌کند. در نهایت شبیه‌سازی تا به انتها انجام می‌شود.

۲-۴-۳ فایل‌های ورودی

دو فایل ورودی x_values.txt و y_values.txt در این پروژه وجود دارد که به ترتیب برای مقادیر مولفه X بردارها و مقادیر مولفه Y بردارها هستند.

هر خط از این فایل های ورودی به صورت زیر است:

<x/y_value>

هر یک از این مقادیر به صورت عدد باینری ۱۶ بیتی هستند و تمام بیت های آن به بخش صحیح عدد تعلق دارد. در این پروژه زاویه بین این بردار (که مولفه های X و Y آن مشخص شده) و محور X بر حسب درجه محاسبه می‌شود.

۲-۴-۴ فایل خروجی

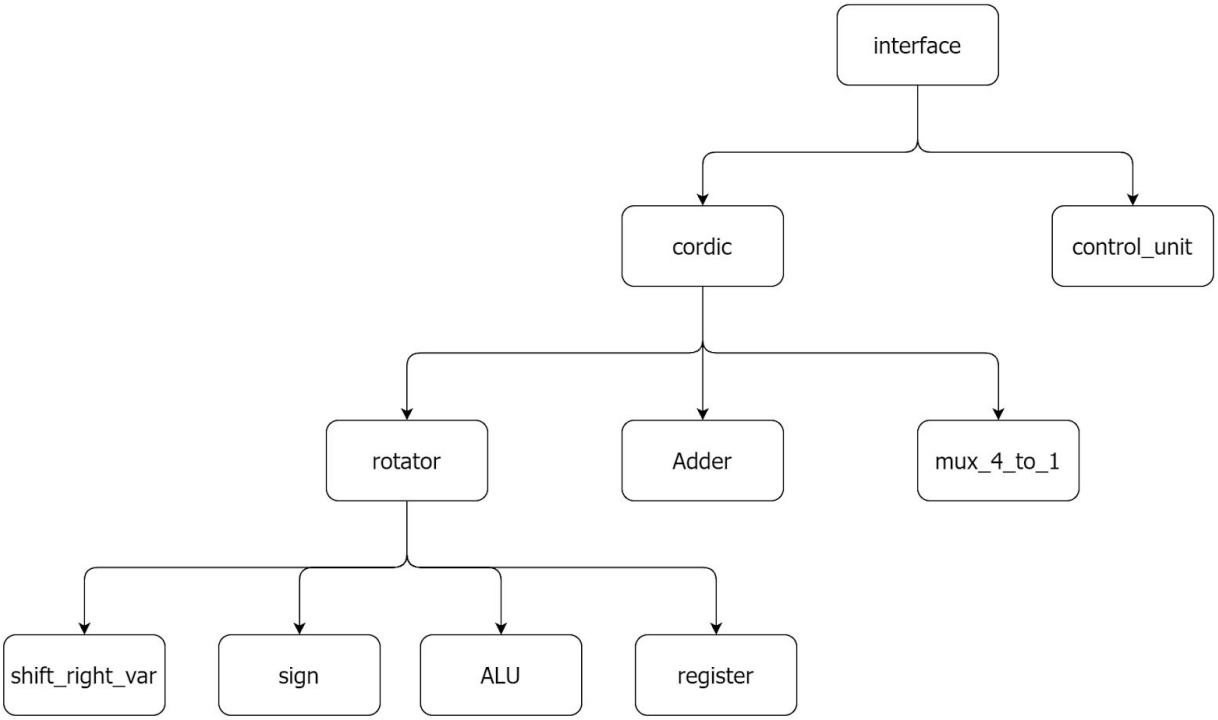
پس از اجرای کد نتایج در فایل verilog_phase.txt نوشته می‌شوند. هر خط از این فایل با همان خطوط در فایل‌های ورودی متناظر است.

هر خط از این فایل خروجی به صورت زیر است:

<phase_value>

هر یک از این مقادیر به صورت عدد عدد اعشاری با حداکثر ۷ رقم اعشاری هستند (که متناظر است با ۷ بیت اعشاری در حالت باینری).

۵-۲ ساختار درختی سیستم و Design Hierarchy



شکل ۸ : *Design Hierarchy*

۳ شبیه سازی و تست

روی 65536 نمونه تصادفی تست انجام شده است و تمامی آن‌ها با دقت ۹۹.۷۱٪ درست آزمایی شده است.
نحوه تست کلی مدار در بخش های زیر شرح داده شده است.

۳-۱ cordic_test

علاوه بر Test Bench های ماژول های درونی یک Test Bench کلی هم برای مدار نوشته شده که در آن یک نمونه از ماژول interface را تست می‌کنیم. بخش های مختلف آن به شرح زیر است.

۳-۱-۱ ورودی و خروجی

برای این که بتوان این ماژول را در تعداد بالا تست کرد و به گونه‌ای که نیاز به تغییر زیادی در test bench نباشد و همچنین برای بررسی خودکار تست‌ها ورودی‌ها را از فایل می‌خوانیم و خروجی‌ها را نیز در فایل می‌نویسیم (برای تغییر تعداد تست‌ها کافیست عدد TESTCASES را تغییر دهیم). برای خواندن و نوشتن نیز از فایل های txt استفاده می‌کنیم که هم فرمت ساده‌ای دارد و هم عملیات تست و بررسی درستی خروجی‌ها ساده‌تر انجام می‌شود.

۳-۱-۲ کلاک عملیات

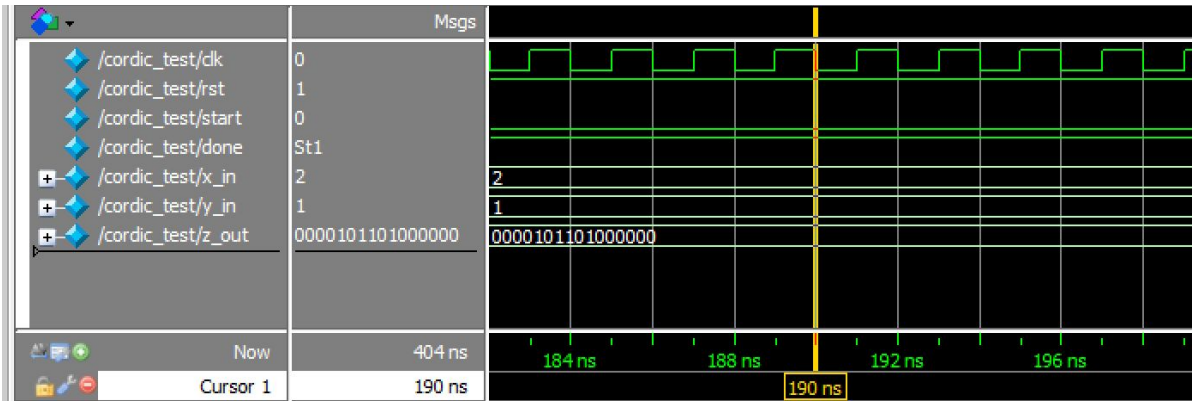
این بخش در یک بلوک initial جداگانه پیاده‌سازی شده است و سیگنال کلاک مورد نیاز برای مدار را تولید می‌کند.

۳-۱-۳ دادن ورودی‌ها به ماژول اصلی

در بلاک اصلی initial به ازای هر تست‌کیس، مولفه‌های بردار از فایل‌های نظیر آن خوانده شده و پس از آن تعداد مناسبی کلاک صبر می‌کنیم تا خروجی تولید شده و پایدار شود. در نهایت خروجی با فرمت مناسب (دسیمال) در فایل موردنظر نوشته می‌شود. علاوه بر نوشته شدن خروجی در فایل، در ترمینال نرم افزار نیز اطلاعات خروجی با جزئیات کافی (شامل زمان شبیه‌سازی، مولفه‌های بردار و مقدار زاویه) چاپ می‌شود. همچنین این بلاک، سیگنال‌های ری ست و start مدار را مقداردهی می‌کند.

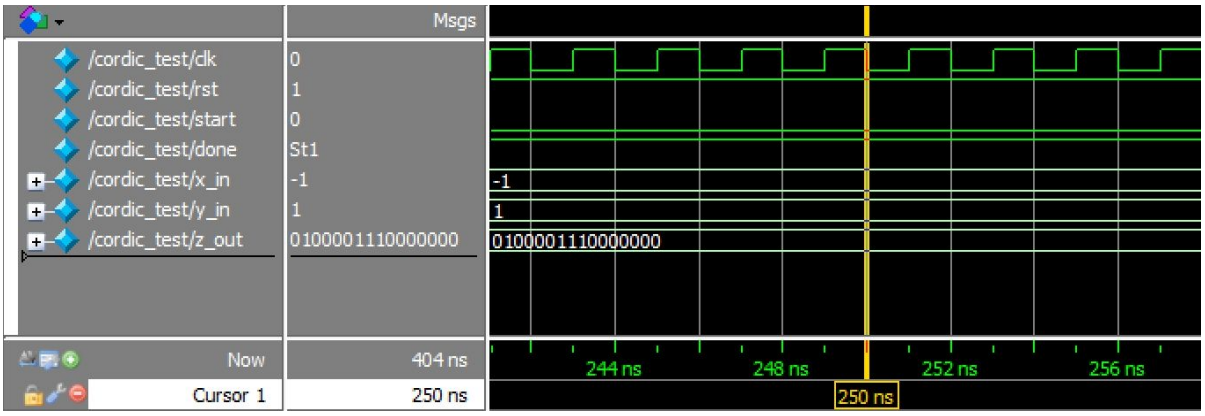
۲-۳ نمونه‌هایی از تست مدار روی Wave

پس از کامپایل کد ها، شبیه‌سازی انجام می‌شود. جهت سهولت مشاهده نتایج، ما شبیه‌سازی را توسط vsim در محیط ModelSim انجام دادیم.



شکل ۹ : نتیجه شبیه‌سازی ۱-

همانگونه که در تصویر مشاهده می‌کنید مولفه‌های X و Y بردار به ترتیب اعداد 2 و 1 هستند. زاویه‌ای که این بردار با محور X می‌سازد عدد 22.5 درجه است که دقت آن در حالت باینری 7 بیت اعشار است.



شکل ۱۰ : نتیجه شبیه‌سازی ۲-

همانگونه که در تصویر مشاهده می‌کنید مولفه‌های X و Y بردار به ترتیب اعداد 1- و 1 هستند. زاویه‌ای که این بردار با محور X می‌سازد عدد 135 درجه است که دقت آن در حالت باینری 7 بیت اعشار است.

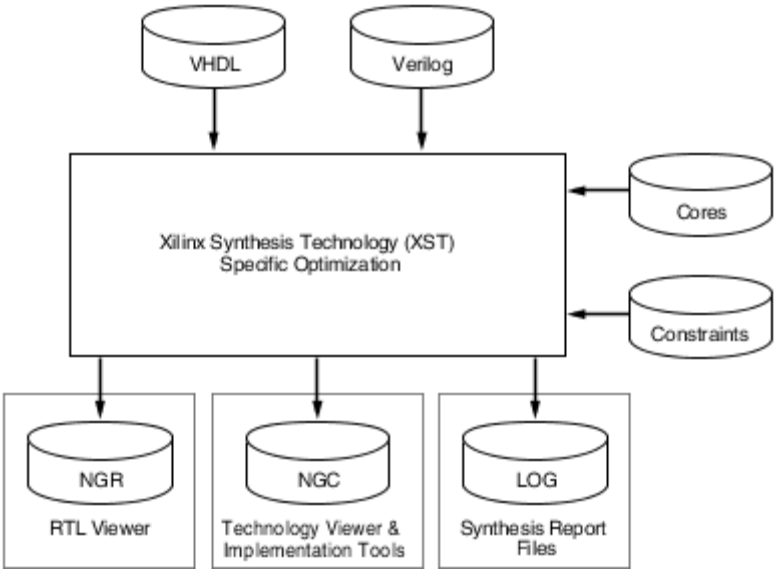
۳-۳ درستی آزمایی

همانطور که در بخش‌های قبل بیان گردید خروجی حاصل از اجرای مدل طلایی در پوشه GoldenModel در فایل python_phase.txt نوشته می‌شود. همچنین خروجی حاصل از شبیه‌سازی نیز در پوشه ModelSim در فایل verilog_phase.txt نوشته می‌شود. به منظور درست آزمایی نتایج، یک اسکریپت پایتون در پوشه اصلی پروژه قرار داده شده است که محتویات این دو فایل را خط به خط به طور موازی پیمایش می‌کند و خطای جذر میانگین مربعات نتایج[14] را محاسبه کرده و از روی آن دقت نتایج را بدست آورده و این دو را در خروجی چاپ می‌کند.
برای اجرای این اسکریپت دستور زیر را در ترمینال وارد می‌کنیم:

```
python3 diff_detector.py
```

[14] Root Mean Square Error

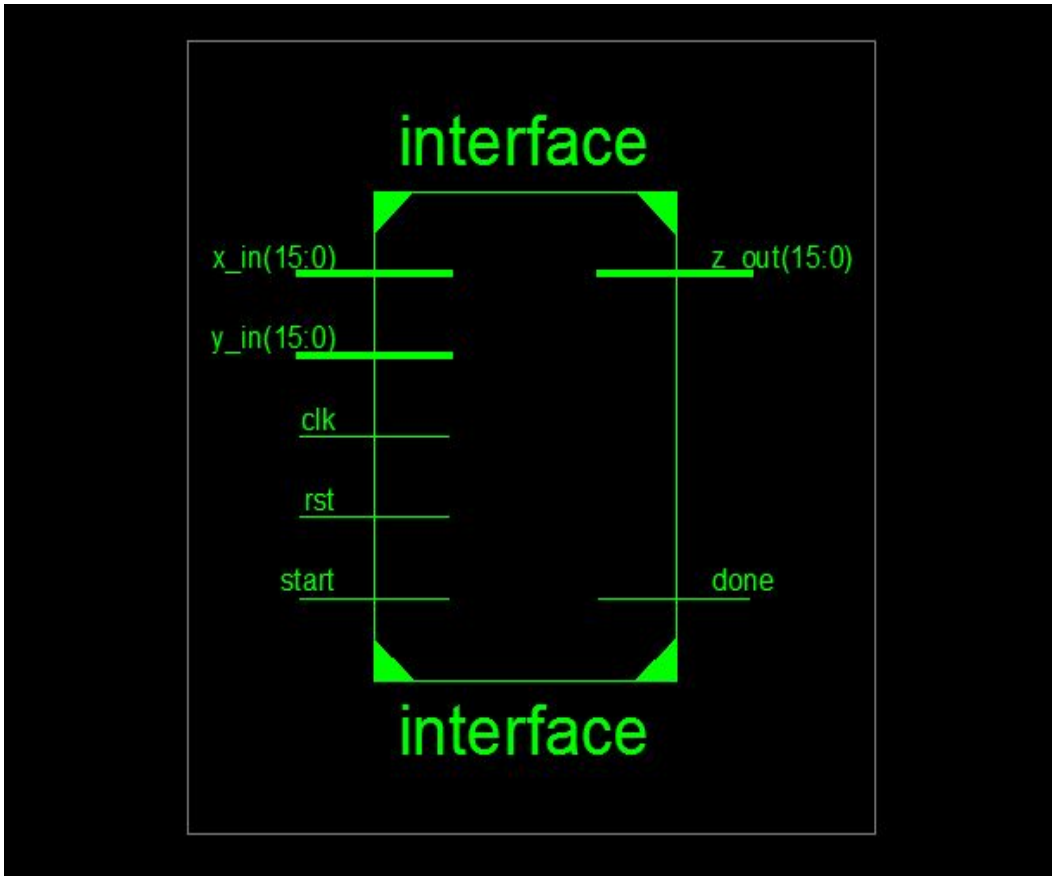
۴ سنتز



برای این قسمت از نرم افزار Xilinx استفاده کرده ایم. نام دیگر این نرم افزار Navigator Project است. برای شروع کار ابتدا تمام فایل های وریلاگ به علاوه ی فایل settings.h به پروژه ی ساخته شده اضافه می‌شود و فایل interface_inst را به عنوان top module قرار می‌دهیم. گزارش تمام مراحل سنتز در بخش های بعدی آمده است.

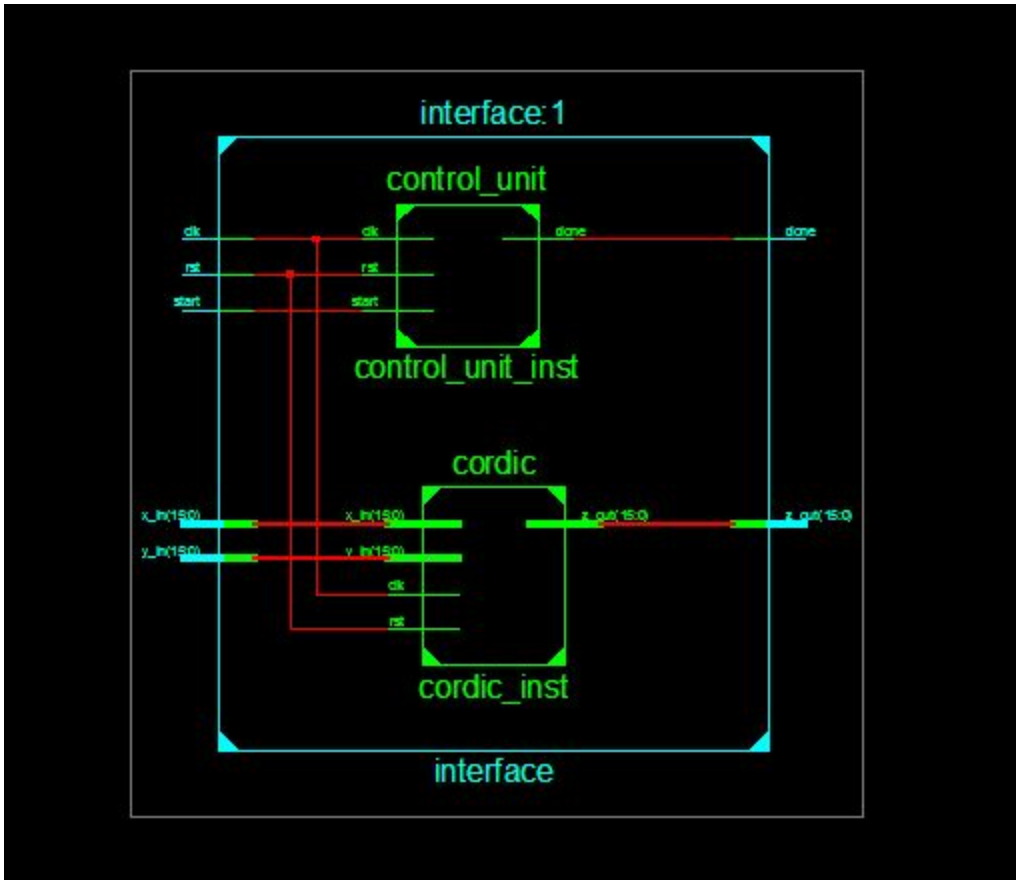
۴-۱ طرح شماتیک

در این قسمت طرح شماتیکی از نرم افزار در شکل های ۱۱ تا ۱۴ نشان داده شده است.شکل ۱۱ دارای ورودی ها و خروجی های اصلی است.



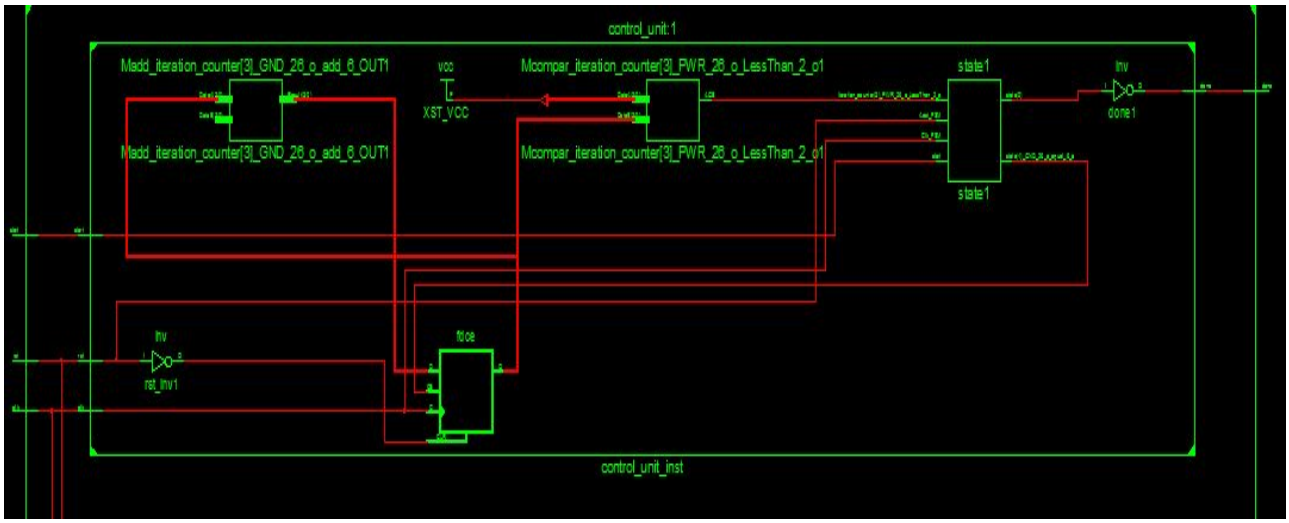
شکل ۱۱ : شماتیک ماژول *interface*

در شکل ۱۲ اجزای داخلی interface نشان داده شده است.این ماژول از دو ماژول داخلی control unit و cordic ساخته شده است. control unit بخش کنترلر است و cordic نقش data path را دارد.در شکل ۱۳ و ۱۴ اجزای داخلی این دو ماژول را بررسی می‌کنیم.



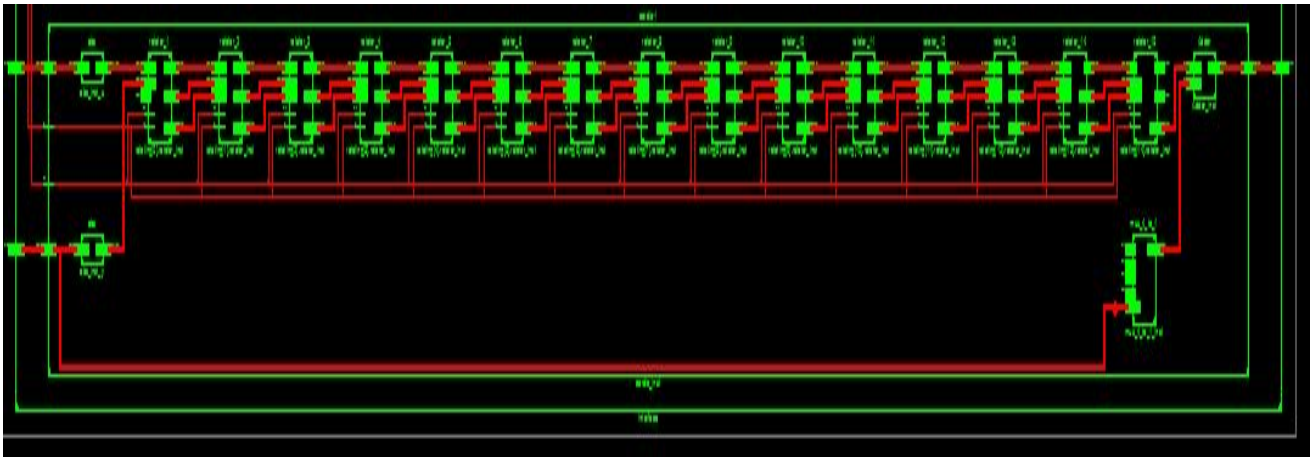
شکل ۱۲ : اجزای داخلی ماژول *interface*

در شکل زیر بخش های داخلی کنترلر نشان داده شده است .



شکل ۱۳ : اجزای داخلی ماژول *control unit*

همچنین در شکل زیر اجزای داخلی ماژول cordic نشان داده شده است.



شکل ۱۴ : اجزای داخلی ماژول cordic

۲-۴ خلاصه آپشن های سنتز[15]

در این بخش اطلاعات کلی پروژه از جمله نام فایل سنتز شده و ماژول های حافظه ای و ویژگی های اضافه شده به پروژه شرح داده شده است.

بخش اول مربوط به پارامتر هاست.

```
---- Source Parameters
Input File Name                    : "interface.prj"
Ignore Synthesis Constraint File   : NO

---- Target Parameters
Output File Name                   : "interface"
Output Format                      : NGC
Target Device                      : xa7a100t-2I-csg324
```

شکل ۱۵ : پارامتر ها

در شکل بالا می‌بینیم که فایل interface.prj ورودی فرایند سنتز است . این فایل همان XST project file است که هر فایل دیگری به پروژه اضافه می‌کنیم در واقع به این فایل اضافه می‌شود.

[15] Synthesis Options Summary

نرم افزار ISE® از XST[16] برای انجام فرایند سنتز فایل هایی که به زبان verilog, hdl نوشته شده است استفاده می‌کند که خروجی آن Xilinx®-specific netlist یا همان فایل های NCG است. در شکل بالا نیز می‌توان دید که خروجی سنتز فایلی با فرمت NGC است. این فایل در مسیر پروژه قرار می‌گیرد و به عنوان ورودی در مرحله Translate فرایند Implement Design استفاده می‌شود.

بخش دوم این قسمت در واقع مربوط به آپشن ها و گزینه هایی است که برای سنتز مورد استفاده قرار گرفته است.

```
---- Source Options
Top Module Name                    : interface
Automatic FSM Extraction           : YES
FSM Encoding Algorithm             : Auto
Safe Implementation                : No
FSM Style                          : LUT
RAM Extraction                     : Yes
RAM Style                          : Auto
ROM Extraction                     : Yes
Shift Register Extraction          : YES
ROM Style                          : Auto
Resource Sharing                   : YES
Asynchronous To Synchronous        : NO
Shift Register Minimum Size        : 2
Use DSP Block                      : Auto
Automatic Register Balancing       : No

---- Target Options
LUT Combining                      : Auto
Reduce Control Sets                : Auto
Add IO Buffers                     : YES
Global Maximum Fanout              : 100000
Add Generic Clock Buffer(BUFG)     : 32
Register Duplication               : YES
Optimize Instantiated Primitives   : NO
Use Clock Enable                   : Auto
Use Synchronous Set                : Auto
Use Synchronous Reset              : Auto
Pack IO Registers into IOBs        : Auto
Equivalent register Removal        : YES
```

[16] Xilinx Synthesis Technology

```
---- General Options
Optimization Goal                  : Speed
Optimization Effort                : 1
Power Reduction                    : NO
Keep Hierarchy                     : No
Netlist Hierarchy                  : As_Optimized
RTL Output                         : Yes
Global Optimization                : AllClockNets
Read Cores                         : YES
Write Timing Constraints           : NO
Cross Clock Analysis               : NO
Hierarchy Separator                : /
Bus Delimiter                      : <>
Case Specifier                     : Maintain
Slice Utilization Ratio            : 100
BRAM Utilization Ratio             : 100
DSP48 Utilization Ratio            : 100
Auto BRAM Packing                  : NO
Slice Utilization Ratio Delta      : 5
```

شکل ۱۶ : آپشن ها و گزینه های سنتز

این آپشن ها را از مراحل زیر می‌توان به مقدار دلخواه تنظیم کرد.

1. Select a source file from the **Source File** window.
2. Right-click **Synthesize - XST** in the **Process** window.
3. Select **Properties**.
4. Select **Synthesis Options**.

با انجام مراحل بالا بر اساس نوع device انتخاب شده یک بخشی نمایش داده می‌شود که در آن می‌توان هر کدام از پارامتر های زیر را تنظیم کرد.

- OptimizationGoal (OPT_MODE)
- OptimizationEffort (OPT_LEVEL)
- UseSynthesisConstraintsFile (-iuc)
- SynthesisConstraintFile (-uc)
- LibrarySearchOrder (-lso)

- GlobalOptimizationGoal (-glob_opt)
- GenerateRTLSchematic (-rtlview)
- WriteTimingConstraints (-write_timing_constraints)
- Verilog2001 (-verilog2001)

۳-۴ گزارش HDL Parsing

در این بخش XST چک می‌کند که آیا کدها نوشته شده درست بوده اند یا نه و هر گونه خطای syntax در این بخش گزارش می‌شود.

بخشی از این گزارش را می‌توان در شکل زیر مشاهده کرد:

```
=========================================================================

                        HDL Parsing

=========================================================================

Analyzing Verilog file "E:\term 6\CAD\Project\report\pipeline-
cordic-algorithm-in-vectoring-mode-master\pipeline-cordic-
algorithm-in-vectoring-mode-master\ModelSim\sign.v" into library
work
Parsing verilog file "E:\term 6\CAD\Project\report\pipeline-
cordic-algorithm-in-vectoring-mode-master\pipeline-cordic-
algorithm-in-vectoring-mode-master\ModelSim\/settings.h"
included at line 1.
Parsing module <sign>.
Analyzing Verilog file "E:\term 6\CAD\Project\report\pipeline-
cordic-algorithm-in-vectoring-mode-master\pipeline-cordic-
algorithm-in-vectoring-mode-master\ModelSim\shift_right_var.v"
into library work
Parsing verilog file "E:\term 6\CAD\Project\report\pipeline-
cordic-algorithm-in-vectoring-mode-master\pipeline-cordic-
algorithm-in-vectoring-mode-master\ModelSim\/settings.h"
included at line 1.
Parsing module <shift_right_var>.
Analyzing Verilog file "E:\term 6\CAD\Project\report\pipeline-
cordic-algorithm-in-vectoring-mode-master\pipeline-cordic-
algorithm-in-vectoring-mode-master\ModelSim\register.v" into
library work
```

شکل ۱۷ : HDL Parsing

۴-۴ گزارش HDL Elaboration

این قسمت سلسله مراتب طراحی و نحوه اتصال آنها را گزارش می‌کند. پارامترهایی که در طراحی استفاده کردیم در اینجا نیز دیده می‌شود. این گزارش در شکل زیر دیده می‌شود

```
*                          HDL Elaboration
*
=========================================================================
=========

Elaborating module <interface>.

Elaborating module <cordic>.

Elaborating module <abs(WORD_WIDTH=16)>.

Elaborating module <mux_4_to_1(WORD_WIDTH=16)>.

Elaborating module <Adder(WORD_WIDTH=16)>.

Elaborating module <rotator(ITERATION=0,PHASE=
16'b01011010000000)>.

Elaborating module <sign(WORD_WIDTH=16)>.

Elaborating module <ALU(WORD_WIDTH=16)>.

Elaborating module <register(WORD_WIDTH=16)>.

Elaborating module <shift_right_var(WORD_WIDTH=16,SHIFT_WIDTH=
4)>.

Elaborating module <rotator(ITERATION=1,PHASE=
16'b0101101000000)>.

Elaborating module <rotator(ITERATION=2,PHASE=
16'b010110100000)>.

Elaborating module <rotator(ITERATION=3,PHASE=16'b01011010000)>.

Elaborating module <rotator(ITERATION=4,PHASE=16'b0101101000)>.

Elaborating module <rotator(ITERATION=5,PHASE=16'b010110100)>.

Elaborating module <rotator(ITERATION=6,PHASE=16'b01011010)>.

Elaborating module <rotator(ITERATION=7,PHASE=16'b0101101)>.

Elaborating module <rotator(ITERATION=8,PHASE=16'b010110)>.

Elaborating module <rotator(ITERATION=9,PHASE=16'b01011)>.

Elaborating module <rotator(ITERATION=10,PHASE=16'b0101)>.

Elaborating module <rotator(ITERATION=11,PHASE=16'b010)>.

Elaborating module <rotator(ITERATION=12,PHASE=16'b01)>.

Elaborating module <rotator(ITERATION=13,PHASE=16'b0)>.

Elaborating module <rotator(ITERATION=14,PHASE=16'b0)>.

Elaborating module <control_unit(ITERATION_WIDTH=4,ITERATIONS=
16)>.
```

شکل ۱۸ : *HDL Elaboration*

۵-۴ گزارش HDL Synthesis

در این بخش XSL به آنالیز کد می‌پردازد تا بتواند آن را با specific design building blocks یا مارکو ها بسازد. ابزار XSL برای کاهش تعداد اجزای استفاده شده برای هر قسمت یک چک resource sharing انجام می‌دهد تا بتواند به طرز بهینه از اجزا استفاده کند. این کار باعث کاهش فضای اشغال شده و افزایش فرکانس کلاک منجر می‌شود.

در شکل زیر گزارش این بخش برای ماژول ها دیده می‌شود.

```
Synthesizing Unit <abs>.
    Related source file is "E:\term 6\CAD\Project\report
\pipeline-cordic-algorithm-in-vectoring-mode-master\pipeline-
cordic-algorithm-in-vectoring-mode-master\ModelSim\abs.v".
        WORD_WIDTH = 16
    Found 16-bit subtractor for signal <ans[15]_unary_minus_2
_OUT> created at line 14.
    Summary:
     inferred   1 Adder/Subtractor(s).
     inferred   1 Multiplexer(s).
Unit <abs> synthesized.

Synthesizing Unit <mux_4_to_1>.
    Related source file is "E:\term 6\CAD\Project\report
\pipeline-cordic-algorithm-in-vectoring-mode-master\pipeline-
cordic-algorithm-in-vectoring-mode-master\ModelSim\mux_4_to_
1.v".
        WORD_WIDTH = 16
    Found 4x16-bit Read Only RAM for signal <out>
    Summary:
     inferred   1 RAM(s).
Unit <mux_4_to_1> synthesized.

Synthesizing Unit <Adder>.
    Related source file is "E:\term 6\CAD\Project\report
\pipeline-cordic-algorithm-in-vectoring-mode-master\pipeline-
cordic-algorithm-in-vectoring-mode-master\ModelSim\Adder.v".
        WORD_WIDTH = 16
    Found 16-bit adder for signal <result> created at line 16.
    Summary:
     inferred   1 Adder/Subtractor(s).
Unit <Adder> synthesized.

Synthesizing Unit <shift_right_var>.
    Related source file is "E:\term 6\CAD\Project\report
\pipeline-cordic-algorithm-in-vectoring-mode-master\pipeline-
cordic-algorithm-in-vectoring-mode-master\ModelSim
\shift_right_var.v".
        WORD_WIDTH = 16
        SHIFT WIDTH = 4


    Summary:
     inferred  15 Comparator(s).
     inferred  15 Multiplexer(s).
Unit <shift_right_var> synthesized.
```

```
Synthesizing Unit <rotator_1>.
    Related source file is "E:\term 6\CAD\Project\report
\pipeline-cordic-algorithm-in-vectoring-mode-master\pipeline-
cordic-algorithm-in-vectoring-mode-master\ModelSim\rotator.v".
        ITERATION = 4'b0000
        PHASE = 16'sb0001011010000000
    Found 4x2-bit Read Only RAM for signal <neg_sign_out_y>
    Summary:
     inferred   1 RAM(s).
Unit <rotator_1> synthesized.

Synthesizing Unit <sign>.
    Related source file is "E:\term 6\CAD\Project\report
\pipeline-cordic-algorithm-in-vectoring-mode-master\pipeline-
cordic-algorithm-in-vectoring-mode-master\ModelSim\sign.v".
        WORD_WIDTH = 16
    Found 32-bit comparator greater for signal <ans[15]_GND_8
_o_LessThan_1_o> created at line 12
    Summary:
     inferred   1 Comparator(s).
     inferred   1 Multiplexer(s).
Unit <sign> synthesized.

Synthesizing Unit <ALU>.
    Related source file is "E:\term 6\CAD\Project\report
\pipeline-cordic-algorithm-in-vectoring-mode-master\pipeline-
cordic-algorithm-in-vectoring-mode-master\ModelSim\ALU.v".
        WORD_WIDTH = 16
    Found 16-bit adder for signal <A[15]_B[15]_add_0_OUT>
created at line 25.
    Found 16-bit adder for signal <n0021> created at line 28.
    Found 16-bit adder for signal <A[15]_GND_9_o_add_3_OUT>
created at line 28.
    Found 16-bit 3-to-1 multiplexer for signal <result>
created at line 23.
    Summary:
     inferred   2 Adder/Subtractor(s).
     inferred   3 Multiplexer(s).
Unit <ALU> synthesized.

Synthesizing Unit <register>.
    Related source file is "E:\term 6\CAD\Project\report
\pipeline-cordic-algorithm-in-vectoring-mode-master\pipeline-
cordic-algorithm-in-vectoring-mode-master\ModelSim
\register.v".
        WORD_WIDTH = 16
    Found 16-bit register for signal <data_out>.
    Summary:
     inferred  16 D-type flip-flop(s).
Unit <register> synthesized.
```

شکل ۱۹ : *HDL Synthesis*

در این قسمت نیز control unit سنتز شده است.

```
Synthesizing Unit <control_unit>.
    Related source file is "E:\term 6\CAD\Project\report
\pipeline-cordic-algorithm-in-vectoring-mode-master\pipeline-
cordic-algorithm-in-vectoring-mode-master\ModelSim
\control_unit.v".
        ITERATION_WIDTH = 4
        ITERATIONS = 16
    Found 4-bit register for signal <iteration_counter>.
    Found 2-bit register for signal <state>.
    Found finite state machine <FSM_0> for signal <state>.

-----------------------------------------------------------------
---------
    | States             | 3
|
    | Transitions        | 5
|
    | Inputs             | 2
|
    | Outputs            | 2
|
    | Clock              | clk (rising_edge)
|
    | Reset              | rst (negative)
|
    | Reset type         | asynchronous
|
    | Reset State        | 00
|
    | Encoding           | auto
|
    | Implementation     | LUT
|

-----------------------------------------------------------------
---------
    Found 4-bit adder for signal <iteration_counter[3]_GND_26
_o_add_6_OUT> created at line 73.
    Found 4-bit comparator greater for signal
<iteration_counter[3]_PWR_26_o_LessThan_2_o> created at line
48
    Summary:
     inferred   1 Adder/Subtractor(s).
     inferred   4 D-type flip-flop(s).
     inferred   1 Comparator(s).
     inferred   1 Finite State Machine(s).
Unit <control_unit> synthesized.
```

شکل ۲۰ : سنتز *control unit*

جمع کل و خلاصه ی این بخش

```
HDL Synthesis Report

Macro Statistics
# RAMs                                          : 16
 4x16-bit single-port Read Only RAM             : 1
 4x2-bit single-port Read Only RAM              : 15
# Adders/Subtractors                            : 94
 16-bit adder                                   : 91
 16-bit subtractor                              : 2
 4-bit adder                                    : 1
# Registers                                     : 46
 16-bit register                                : 45
 4-bit register                                 : 1
# Comparators                                   : 466
 32-bit comparator greater                      : 15
 4-bit comparator greater                       : 1
 4-bit comparator lessequal                     : 450
# Multiplexers                                  : 602
 16-bit 2-to-1 multiplexer                      : 587
 2-bit 2-to-1 multiplexer                       : 15
# FSMs                                          : 1
```

شکل ۲۱ : خلاصه بخش

۶-۴ گزارش Advanced HDL Synthesis

در این قسمت کد به صورت پیشرفته سنتز می‌شود .خلاصه حاصل از این بخش را می‌توان در شکل زیر مشاهده کرد:

```
Advanced HDL Synthesis Report

Macro Statistics
# RAMs                                          : 16
 4x16-bit single-port distributed Read Only RAM : 1
 4x2-bit single-port distributed Read Only RAM  : 15
# Adders/Subtractors                            : 93
 16-bit adder                                   : 91
 16-bit subtractor                              : 2
# Counters                                      : 1
 4-bit up counter                               : 1
# Registers                                     : 720
 Flip-Flops                                     : 720
# Comparators                                   : 466
 32-bit comparator greater                      : 15
 4-bit comparator greater                       : 1
 4-bit comparator lessequal                     : 450
# Multiplexers                                  : 587
 16-bit 2-to-1 multiplexer                      : 587
# FSMs                                          : 1
```

شکل ۲۲ : *Advanced HDL Synthesis*

۴-۷ گزارش Low Level Synthesis

در این مرحله ابتدا آنالیزی روی FSM مدار انجام شده و سپس به بهینه سازی مدار در سطح پایین سیستم می‌پردازد. در بهینه سازی سطح پایین، ماکروهای استنباط شده به یک اجرای با فناوری خاص تبدیل می‌شود. این جریان برای FPGA و CPLD ها تفاوت هایی دارند که به شرح زیر است:

۴-۷-۱ FPGA

جریان FPGA کاملا زمان محور است و با اعمال محدودیت هایی از قبیل PERIOD و OFFSET کنترل می‌شود. در طی این مرحله به موارد خاصی اشاره دارد.از جمله:

- Carry logic (MUXCY, XORCY, MULT_AND)
- RAM (block or distributed)
- Shift Register LUTs (SRL16, SRL32)
- Clock Buffers (IBUFG, BUFG, BUFGP, BUFR)
- Multiplexers (MUXF5, MUXF6, MUXF7, MUXF8)
- Arithmetic Functions (DSP48, MULT18X18)

استفاده از ویژگی های فناوری خاص ممکن است از مکانیسم پیاده سازی ماکروها یا Logic Mapping ناشی شود. اگرچه با توجه به این ویژگی‌ها ممکن است نتوان از تمام ویژگی های در دسترس FPGA ها استفاده کرد. جریان سنتز FPGA، از طراحی های پیشرفته و تکنیک های بهینه سازی مانند متعادل کردن[17] رجیسترها پشتیبانی می‌کند.

۴-۷-۲ CPLD

جریان CPLD زمان محور نیست و نمی‌توان فرکانس Clock یا مقدار آفست را مشخص کرد. هدف CPLD ها کاهش تعداد سطوح منطقی است. در طی بهینه سازی سطح پایین، XST یک netlist خالص ایجاد می‌کند که حاوی گیت های AND و OR است. سپس چگونگی چینش این گیت ها برای رسیدن به هدف نهایی سیستم تعیین می‌شود.

[17] Balance

گزارش ارائه شده از این بخش در شکل زیر آمده است.

```
=========================================================================
*                          Low Level Synthesis                          *
=========================================================================
Analyzing FSM <MFsm> for best encoding.
Optimizing FSM <FSM_0> on signal <state[1:2]> with user encoding.
-------------------
 State | Encoding
-------------------
 00    | 00
 01    | 01
 10    | 10
-------------------

Optimizing unit <interface> ...

Optimizing unit <cordic> ...

Optimizing unit <rotator_1> ...

Optimizing unit <ALU> ...

Optimizing unit <shift_right_var> ...

Optimizing unit <register> ...

Optimizing unit <rotator_2> ...

Optimizing unit <rotator_3> ...

Optimizing unit <rotator_4> ...

Optimizing unit <rotator_5> ...

Optimizing unit <rotator_6> ...

Optimizing unit <rotator_7> ...

Optimizing unit <rotator_8> ...

Optimizing unit <rotator_9> ...

Optimizing unit <rotator_10> ...

Optimizing unit <rotator_11> ...

Optimizing unit <rotator_12> ...

Optimizing unit <rotator_13> ...

Optimizing unit <rotator_14> ...

Optimizing unit <rotator_15> ...
```

```
Mapping all equations...
Building and optimizing final netlist ...
Found area constraint ratio of 100 (+ 5) on block interface, actual ratio is 6.

Final Macro Processing ...

=========================================================================
Final Register Report

Macro Statistics
# Registers                                          : 633
 Flip-Flops                                          : 633
```

شکل ۲۳ : *Low Level Synthesis*

۸-۴ خلاصه طراحی[18]

در این قسمت گزارش طراحی این پروژه آورده شده است.این گزارش اطلاعاتی درباره ی تعداد بلاک های استفاده شده در پروژه مانند تعداد flip flop ها در اختیار طراح قرار می‌دهد. همچنین می‌توان به اطلاعات کلی پروژه و دستگاه های مورد استفاده در آن دسترسی پیدا کرد. برای یافتن این گزارش ابتدا از کادر سمت چپ آیکون Detailed Reports را باز کرده و سپس روی Synthesis Report کلیک می‌کنیم. حال در صفحه‌ی بازشده اطلاعات مربوط به خلاصه ی طراحی را جستجو می‌کنیم. این گزارش خودش شامل بخش های کوچک‌تری است که در ادامه توضیح داده خواهد شد.

۱-۸-۴ Primitive and Black Box Usage

این بخش آماری از بلوک های اساسی و Primitive های استفاده شده را نشان می‌دهد. در اصل این بخش یک نمونه ماژول vhdl / verilog خالی است. وقتی چنین قطعه ای از کد را سنتز می‌کنید، هیچ یک از Primitive ها را استنباط نمی‌کنید. در ISE، معمولاً نمونه های Black Box را با "؟" می‌بینید. این نماد در پنجره سلسله مراتبی قرار می‌گیرد.

[18] Design Summary

```
Top Level Output File Name         : interface.ngc

Primitive and Black Box Usage:
------------------------------
# BELS                             : 5224
#      GND                         : 1
#      INV                         : 204
#      LUT1                        : 46
#      LUT2                        : 446
#      LUT3                        : 1445
#      LUT4                        : 315
#      LUT5                        : 62
#      LUT6                        : 163
#      MUXCY                       : 1249
#      VCC                         : 1
#      XORCY                       : 1292
# FlipFlops/Latches                : 633
#      FDC                         : 629
#      FDCE                        : 4
# Clock Buffers                    : 1
#      BUFGP                       : 1
# IO Buffers                       : 51
#      IBUF                        : 34
#      OBUF                        : 17
```

شکل ۲۴ : *Primitive and Black Box Usage*

۲-۸-۴ Device utilization summary

ابزار سنتز Primitive ها را استنباط می‌کند یا از آن ها استفاده می‌کند. تعداد این Primitive های استفاده شده در این بخش گزارش شده است.

```
Device utilization summary:
---------------------------

Selected Device : 7a100tcsg324-3


Slice Logic Utilization:
 Number of Slice Registers:             633  out of  126800     0%
 Number of Slice LUTs:                 2681  out of  63400      4%
    Number used as Logic:              2681  out of  63400      4%

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:   2685
   Number with an unused Flip Flop:    2052  out of   2685     76%
   Number with an unused LUT:             4  out of   2685      0%
   Number of fully used LUT-FF pairs:   629  out of   2685     23%
   Number of unique control sets:         2

IO Utilization:
 Number of IOs:                          52
 Number of bonded IOBs:                  52  out of    210     24%

Specific Feature Utilization:
 Number of BUFG/BUFGCTRLs:               1  out of     32      3%
```

شکل ۲۵ : *Device utilization summary*

۹-۴ گزارش زمان بندی[19]

این بخش شامل چندین گزارش از زمان بندی بخش های مختلف پروژه و ماژول های بکار رفته در آن است:

۱-۹-۴ Clock Information

این بخش شامل اطلاعات مربوط به کلاک سیستم طراحی شده است.

```
Timing Report

NOTE: THESE TIMING NUMBERS ARE ONLY A SYNTHESIS ESTIMATE.
      FOR ACCURATE TIMING INFORMATION PLEASE REFER TO THE TRACE REPORT
      GENERATED AFTER PLACE-and-ROUTE.

Clock Information:
------------------
+-----------------------------------+-------------------------+-------+
| Clock Signal                      | Clock buffer(FF name)   | Load  |
+-----------------------------------+-------------------------+-------+
| clk                               | BUFGP                   | 633   |
+-----------------------------------+-------------------------+-------+
```

شکل ۲۶ : *Clock Information*

۲-۹-۴ Timing summary

خلاصه ی زمان بندی نمای کلی ای از طراحی را نشان می‌دهد و شامل بخش هایی نظیر آمار طراحی، خطاهای زمان بندی، ماکزیمم دوره ی زمانی و فرکانس مدار و تاخیر های مدار است.

[19] Timing Report

```
Timing Summary:
---------------
Speed Grade: -3

   Minimum period: 3.648ns (Maximum Frequency: 274.123MHz)
   Minimum input arrival time before clock: 5.370ns
   Maximum output required time after clock: 2.110ns
   Maximum combinational path delay: 1.742ns
```

شکل ۲۷ : خلاصه ی زمان‌بندی

Timing Details ۳-۹-۴

جزئیات زمان بندی اطلاعاتی را درباره ی محدودیت زمان بندی در دسترس قرار می‌دهد. می‌توان با استفاده از گزارش زمان بندی constraint خروجی، زمان‌بندی خروجی را تحلیل کرد. زمان بندی constraint خروجی مسیر داده را از پین کلاک خارجی به FPGA، از طریق هر logic درون FPGA و از عنصر همزمان در داخل FPGA تا پین داده خارجی پوشش می‌دهد. این اطلاعات در شکل زیر آمده است. تمامی زمان های نشان داده شده در واحد نانو ثانیه هستند.

```
Timing constraint: Default period analysis for Clock 'clk'
  Clock period: 3.648ns (frequency: 274.123MHz)
  Total number of paths / destination ports: 215788 / 597
-------------------------------------------------------------------------
Delay:               3.648ns (Levels of Logic = 20)
  Source:            cordic_inst/rotating[11].rotator_inst/register_inst_x/data_out_0 (FF)
  Destination:       cordic_inst/rotating[12].rotator_inst/register_inst_x/data_out_15 (FF)
  Source Clock:      clk rising
  Destination Clock: clk rising

  Data Path: cordic_inst/rotating[11].rotator_inst/register_inst_x/data_out_0 to cordic_inst
                                     Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     FDC:C->Q              3   0.361   0.389  cordic_inst/rotating[11].rotator_inst/register
     LUT2:I0->O            1   0.097   0.000  cordic_inst/rotating[12].rotator_inst/ALU_inst
     MUXCY:S->O            1   0.353   0.000  cordic_inst/rotating[12].rotator_inst/ALU_inst
     MUXCY:CI->O           1   0.023   0.000  cordic_inst/rotating[12].rotator_inst/ALU_inst
     MUXCY:CI->O           1   0.023   0.000  cordic_inst/rotating[12].rotator_inst/ALU_inst
     MUXCY:CI->O           1   0.023   0.000  cordic_inst/rotating[12].rotator_inst/ALU_inst
     MUXCY:CI->O           1   0.023   0.000  cordic_inst/rotating[12].rotator_inst/ALU_inst
     MUXCY:CI->O           1   0.023   0.000  cordic_inst/rotating[12].rotator_inst/ALU_inst
     MUXCY:CI->O           1   0.023   0.000  cordic_inst/rotating[12].rotator_inst/ALU_inst
     MUXCY:CI->O           1   0.023   0.000  cordic_inst/rotating[12].rotator_inst/ALU_inst
     MUXCY:CI->O           1   0.023   0.000  cordic_inst/rotating[12].rotator_inst/ALU_inst
     MUXCY:CI->O           1   0.023   0.000  cordic_inst/rotating[12].rotator_inst/ALU_inst
     MUXCY:CI->O           1   0.023   0.000  cordic_inst/rotating[12].rotator_inst/ALU_inst
     MUXCY:CI->O           1   0.023   0.000  cordic_inst/rotating[12].rotator_inst/ALU_inst
     XORCY:CI->O           1   0.370   0.295  cordic_inst/rotating[12].rotator_inst/ALU_inst
     LUT3:I2->O            1   0.097   0.279  cordic_inst/rotating[12].rotator_inst/ALU_inst
     MUXCY:DI->O           1   0.337   0.000  cordic_inst/rotating[12].rotator_inst/ALU_inst
     MUXCY:CI->O           1   0.023   0.000  cordic_inst/rotating[12].rotator_inst/ALU_inst
     MUXCY:CI->O           0   0.023   0.000  cordic_inst/rotating[12].rotator_inst/ALU_inst
     XORCY:CI->O           1   0.370   0.295  cordic_inst/rotating[12].rotator_inst/ALU_inst
     LUT3:I2->O            1   0.097   0.000  cordic_inst/rotating[12].rotator_inst/ALU_inst
     FDC:D                     0.008          cordic_inst/rotating[12].rotator_inst/register
    ----------------------------------------
    Total                      3.648ns (2.389ns logic, 1.259ns route)
                                       (65.5% logic, 34.5% route)

Timing constraint: Default OFFSET IN BEFORE for Clock 'clk'
  Total number of paths / destination ports: 119208 / 674
-------------------------------------------------------------------------
Offset:              5.370ns (Levels of Logic = 37)
  Source:            y_in<0> (PAD)
  Destination:       cordic_inst/rotating[0].rotator_inst/register_inst_y/data_out_15 (FF)
  Destination Clock: clk rising

  Data Path: y_in<0> to cordic_inst/rotating[0].rotator_inst/register_inst_y/data_out_15
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     IBUF:I->O             7   0.001   0.307  y_in_0_IBUF (y_in_0_IBUF)
     INV:I->O              1   0.113   0.000  cordic_inst/abs_inst_y/Msub_ans[15]_unary_minus_
     MUXCY:S->O            1   0.353   0.000  cordic_inst/abs_inst_y/Msub_ans[15]_unary_minus_
     MUXCY:CI->O           1   0.023   0.000  cordic_inst/abs_inst_y/Msub_ans[15]_unary_minus_
     MUXCY:CI->O           1   0.023   0.000  cordic_inst/abs_inst_y/Msub_ans[15]_unary_minus_
     MUXCY:CI->O           1   0.023   0.000  cordic_inst/abs_inst_y/Msub_ans[15]_unary_minus_
     MUXCY:CI->O           1   0.023   0.000  cordic_inst/abs_inst_y/Msub_ans[15]_unary_minus_
     MUXCY:CI->O           1   0.023   0.000  cordic_inst/abs_inst_y/Msub_ans[15]_unary_minus_
     MUXCY:CI->O           1   0.023   0.000  cordic_inst/abs_inst_y/Msub_ans[15]_unary_minus_
     MUXCY:CI->O           1   0.023   0.000  cordic_inst/abs_inst_y/Msub_ans[15]_unary_minus_
     MUXCY:CI->O           1   0.023   0.000  cordic_inst/abs_inst_y/Msub_ans[15]_unary_minus_
     MUXCY:CI->O           1   0.023   0.000  cordic_inst/abs_inst_y/Msub_ans[15]_unary_minus_
     MUXCY:CI->O           1   0.023   0.000  cordic_inst/abs_inst_y/Msub_ans[15]_unary_minus_
     MUXCY:CI->O           1   0.023   0.000  cordic_inst/abs_inst_y/Msub_ans[15]_unary_minus_
     MUXCY:CI->O           1   0.023   0.000  cordic_inst/abs_inst_y/Msub_ans[15]_unary_minus_
     XORCY:CI->O           6   0.370   0.318  cordic_inst/abs_inst_y/Msub_ans[15]_unary_minus_
     LUT3:I2->O            3   0.097   0.703  cordic_inst/abs_inst_y/Mmux_abs_ans51 (cordic_in
     LUT6:I0->O            1   0.097   0.295  cordic_inst/rotating[0].rotator_inst/sign_inst/a
     LUT6:I5->O           31   0.097   0.800  cordic_inst/rotating[0].rotator_inst/sign_inst/a
     LUT6:I0->O            1   0.097   0.000  cordic_inst/rotating[0].rotator_inst/ALU_inst_y/
```

```
    MUXCY:S->O               1    0.353   0.000  cordic_inst/rotating[0].rotator_inst/ALU_inst
    MUXCY:CI->O              1    0.023   0.000  cordic_inst/rotating[0].rotator_inst/ALU_inst
    MUXCY:CI->O              1    0.023   0.000  cordic_inst/rotating[0].rotator_inst/ALU_inst
    MUXCY:CI->O              1    0.023   0.000  cordic_inst/rotating[0].rotator_inst/ALU_inst
    MUXCY:CI->O              1    0.023   0.000  cordic_inst/rotating[0].rotator_inst/ALU_inst
    MUXCY:CI->O              1    0.023   0.000  cordic_inst/rotating[0].rotator_inst/ALU_inst
    MUXCY:CI->O              1    0.023   0.000  cordic_inst/rotating[0].rotator_inst/ALU_inst
    MUXCY:CI->O              1    0.023   0.000  cordic_inst/rotating[0].rotator_inst/ALU_inst
    MUXCY:CI->O              1    0.023   0.000  cordic_inst/rotating[0].rotator_inst/ALU_inst
    MUXCY:CI->O              1    0.023   0.000  cordic_inst/rotating[0].rotator_inst/ALU_inst
    MUXCY:CI->O              1    0.023   0.000  cordic_inst/rotating[0].rotator_inst/ALU_inst
    MUXCY:CI->O              1    0.023   0.000  cordic_inst/rotating[0].rotator_inst/ALU_inst
    MUXCY:CI->O              1    0.023   0.000  cordic_inst/rotating[0].rotator_inst/ALU_inst
    MUXCY:CI->O              1    0.023   0.000  cordic_inst/rotating[0].rotator_inst/ALU_inst
    MUXCY:CI->O              0    0.023   0.000  cordic_inst/rotating[0].rotator_inst/ALU_inst
    XORCY:CI->O              1    0.370   0.295  cordic_inst/rotating[0].rotator_inst/ALU_inst
    LUT4:I3->O               1    0.097   0.000  cordic_inst/rotating[0].rotator_inst/ALU_inst
    FDC:D                         0.008          cordic_inst/rotating[0].rotator_inst/register
   ----------------------------------------
   Total                          5.370ns (2.651ns logic, 2.719ns route)
                                          (49.4% logic, 50.6% route)

Timing constraint: Default OFFSET OUT AFTER for Clock 'clk'
  Total number of paths / destination ports: 73 / 17
-------------------------------------------------------------------------
Offset:              2.110ns (Levels of Logic = 10)
  Source:            cordic_inst/rotating[14].rotator_inst/register_inst_z/data_out_8 (FF)
  Destination:       z_out<15> (PAD)
  Source Clock:      clk rising

  Data Path: cordic_inst/rotating[14].rotator_inst/register_inst_z/data_out_8 to z_out<15>
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     FDC:C->Q              1   0.361   0.511  cordic_inst/rotating[14].rotator_inst/regist
     LUT3:I0->O            1   0.097   0.000  cordic_inst/Adder_inst/Madd_result_lut<8> (c
     MUXCY:S->O            1   0.353   0.000  cordic_inst/Adder_inst/Madd_result_cy<8> (cc
     MUXCY:CI->O           1   0.023   0.000  cordic_inst/Adder_inst/Madd_result_cy<9> (cc
     MUXCY:CI->O           1   0.023   0.000  cordic_inst/Adder_inst/Madd_result_cy<10> (c
     MUXCY:CI->O           1   0.023   0.000  cordic_inst/Adder_inst/Madd_result_cy<11> (c
     MUXCY:CI->O           1   0.023   0.000  cordic_inst/Adder_inst/Madd_result_cy<12> (c
     MUXCY:CI->O           1   0.023   0.000  cordic_inst/Adder_inst/Madd_result_cy<13> (c
     MUXCY:CI->O           0   0.023   0.000  cordic_inst/Adder_inst/Madd_result_cy<14> (c
     XORCY:CI->O           1   0.370   0.279  cordic_inst/Adder_inst/Madd_result_xor<15> (
     OBUF:I->O                 0.000          z_out_15_OBUF (z_out<15>)
    ----------------------------------------
    Total                      2.110ns (1.319ns logic, 0.791ns route)
                                       (62.5% logic, 37.5% route)

Timing constraint: Default path analysis
  Total number of paths / destination ports: 60 / 8
-------------------------------------------------------------------------
Delay:               1.742ns (Levels of Logic = 11)
  Source:            y_in<15> (PAD)
  Destination:       z_out<15> (PAD)

  Data Path: y_in<15> to z_out<15>
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     IBUF:I->O           148   0.001   0.504  y_in_15_IBUF (y_in_15_IBUF)
     LUT3:I1->O            1   0.097   0.000  cordic_inst/Adder_inst/Madd_result_lut<8> (co
     MUXCY:S->O            1   0.353   0.000  cordic_inst/Adder_inst/Madd_result_cy<8> (cor
     MUXCY:CI->O           1   0.023   0.000  cordic_inst/Adder_inst/Madd_result_cy<9> (cor
     MUXCY:CI->O           1   0.023   0.000  cordic_inst/Adder_inst/Madd_result_cy<10> (co
     MUXCY:CI->O           1   0.023   0.000  cordic_inst/Adder_inst/Madd_result_cy<11> (co
     MUXCY:CI->O           1   0.023   0.000  cordic_inst/Adder_inst/Madd_result_cy<12> (co
     MUXCY:CI->O           1   0.023   0.000  cordic_inst/Adder_inst/Madd_result_cy<13> (co
     MUXCY:CI->O           0   0.023   0.000  cordic_inst/Adder_inst/Madd_result_cy<14> (co
     XORCY:CI->O           1   0.370   0.279  cordic_inst/Adder_inst/Madd_result_xor<15> (z
     OBUF:I->O                 0.000          z_out_15_OBUF (z_out<15>)
    ----------------------------------------
    Total                      1.742ns (0.959ns logic, 0.783ns route)
                                       (55.0% logic, 45.0% route)
```

شکل ۲۸ : جزئیات زمان‌بندی

۴-۹-۴ Cross Clock Domains Report

در این گزارش تعداد Cross Clock Domains های رخ داده شده در پروژه اطلاع داده شده است که در گزارش این پروژه یک مورد یافت شد.

```
Clock to Setup on destination clock clk
---------------+---------+---------+---------+---------+
               | Src:Rise| Src:Fall| Src:Rise| Src:Fall|
Source Clock   |Dest:Rise|Dest:Rise|Dest:Fall|Dest:Fall|
---------------+---------+---------+---------+---------+
clk            |    3.648|         |         |         |
---------------+---------+---------+---------+---------+
```

شکل ۲۹ : Cross Clock Domains Report

۱۰-۴ نتیجه نهایی سنتز و میزان حافظه ی اشغال شده

در این بخش تعداد اورها و هشدارهای کل پروژه و میزان حافظه مصرفی گزارش شده است.

```
Total REAL time to Xst completion: 30.00 secs
Total CPU time to Xst completion: 30.07 secs

-->

Total memory usage is 456528 kilobytes

Number of errors   :    0 (   0 filtered)
Number of warnings :   98 (   0 filtered)
Number of infos    :   23 (   0 filtered)
```

شکل ۳۰ : نتیجه نهایی سنتز

۴-۱۱ گزارش Data Sheet

این گزارش، پارامترهای زمان‌بندی خارجی را در طراحی خلاصه می‌کند و تعدادی جدول را در بر می‌گیرد. فقط ورودی ها، خروجی ها و ساعت هایی که محدودیت دارند در گزارش برای یافتن خطا ظاهر می‌شوند. جداول نشان داده شده بستگی به نوع مسیر زمان بندی موجود در طرح و همچنین محدودیت های زمان بندی کاربردی دارد. در زیر جدول هایی وجود دارد که در این گزارش آمده است:

۴-۱۱-۱ Cross Clock Domains Report

در این جدول، setup time و hold time برای سیگنال های ورودی را نشان می‌دهد. تمامی زمان های نشان داده شده در واحد نانو ثانیه هستند.

جدول ۱ : Data Sheet Report در سنتز

Setup/Hold to clock clk

Source	Max Setup to clk (edge)	Process Corner	Max Hold to clk (edge)	Process Corner	Internal Clock(s)	Clock Phase
rst	4.281(R)	SLOW	1.185(R)	SLOW	clk_BUFGP	0.000
start	0.428(R)	FAST	1.291(R)	SLOW	clk_BUFGP	0.000
x_in<0>	4.983(R)	SLOW	2.175(R)	SLOW	clk_BUFGP	0.000
x_in<1>	5.346(R)	SLOW	1.983(R)	SLOW	clk_BUFGP	0.000
x_in<2>	5.100(R)	SLOW	2.077(R)	SLOW	clk_BUFGP	0.000
x_in<3>	5.557(R)	SLOW	2.222(R)	SLOW	clk_BUFGP	0.000
x_in<4>	4.764(R)	SLOW	1.979(R)	SLOW	clk_BUFGP	0.000
x_in<5>	4.862(R)	SLOW	1.701(R)	SLOW	clk_BUFGP	0.000
x_in<6>	5.092(R)	SLOW	2.419(R)	SLOW	clk_BUFGP	0.000
x_in<7>	4.738(R)	SLOW	1.838(R)	SLOW	clk_BUFGP	0.000
x_in<8>	4.627(R)	SLOW	2.496(R)	SLOW	clk_BUFGP	0.000
x_in<9>	4.826(R)	SLOW	2.649(R)	SLOW	clk_BUFGP	0.000
x_in<10>	4.874(R)	SLOW	2.339(R)	SLOW	clk_BUFGP	0.000
x_in<11>	3.885(R)	SLOW	2.513(R)	SLOW	clk_BUFGP	0.000
x_in<12>	3.157(R)	SLOW	2.102(R)	SLOW	clk_BUFGP	0.000
x_in<13>	3.191(R)	SLOW	2.389(R)	SLOW	clk_BUFGP	0.000
x_in<14>	2.420(R)	SLOW	2.263(R)	SLOW	clk_BUFGP	0.000
x_in<15>	4.429(R)	SLOW	2.311(R)	SLOW	clk_BUFGP	0.000
y_in<0>	7.723(R)	SLOW	2.638(R)	SLOW	clk_BUFGP	0.000
y_in<1>	8.229(R)	SLOW	2.040(R)	SLOW	clk_BUFGP	0.000
y_in<2>	7.773(R)	SLOW	2.337(R)	SLOW	clk_BUFGP	0.000
y_in<3>	7.788(R)	SLOW	2.333(R)	SLOW	clk_BUFGP	0.000
y_in<4>	7.251(R)	SLOW	2.199(R)	SLOW	clk_BUFGP	0.000
y_in<5>	7.367(R)	SLOW	2.388(R)	SLOW	clk_BUFGP	0.000
y_in<6>	7.444(R)	SLOW	2.442(R)	SLOW	clk_BUFGP	0.000
y_in<7>	7.955(R)	SLOW	2.167(R)	SLOW	clk_BUFGP	0.000
y_in<8>	7.545(R)	SLOW	2.187(R)	SLOW	clk_BUFGP	0.000
y_in<9>	7.744(R)	SLOW	1.951(R)	SLOW	clk_BUFGP	0.000
y_in<10>	7.343(R)	SLOW	2.108(R)	SLOW	clk_BUFGP	0.000
y_in<11>	7.760(R)	SLOW	2.095(R)	SLOW	clk_BUFGP	0.000
y_in<12>	7.998(R)	SLOW	1.750(R)	SLOW	clk_BUFGP	0.000
y_in<13>	7.851(R)	SLOW	1.780(R)	SLOW	clk_BUFGP	0.000
y_in<14>	7.741(R)	SLOW	1.849(R)	SLOW	clk_BUFGP	0.000
y_in<15>	9.510(R)	SLOW	1.408(R)	SLOW	clk_BUFGP	0.000

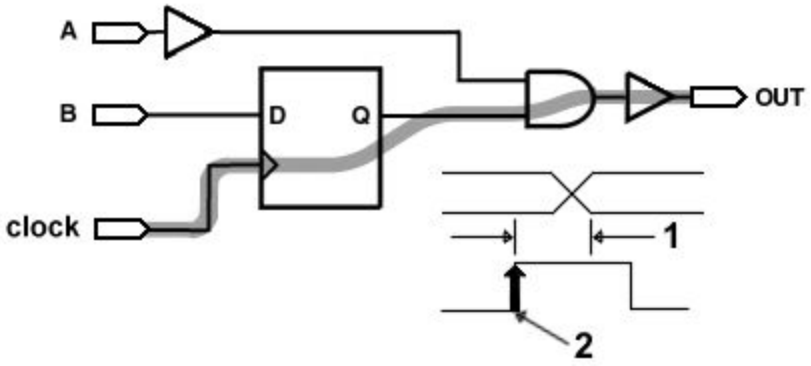
جدول ۲ : *Data Sheet Report* در پیاده سازی

Setup/Hold to clock clk

Source	Max Setup to clk (edge)	Process Corner	Max Hold to clk (edge)	Process Corner	Internal Clock(s)	Clock Phase
rst	5.522(R)	SLOW	1.031(R)	SLOW	clk_BUFGP	0.000
start	1.052(R)	SLOW	-0.306(R)	SLOW	clk_BUFGP	0.000
x_in<0>	9.839(R)	SLOW	-1.087(R)	FAST	clk_BUFGP	0.000
x_in<1>	9.602(R)	SLOW	-0.839(R)	SLOW	clk_BUFGP	0.000
x_in<2>	9.530(R)	SLOW	-1.009(R)	FAST	clk_BUFGP	0.000
x_in<3>	9.725(R)	SLOW	-0.837(R)	FAST	clk_BUFGP	0.000
x_in<4>	9.169(R)	SLOW	-0.945(R)	FAST	clk_BUFGP	0.000
x_in<5>	8.873(R)	SLOW	-0.230(R)	SLOW	clk_BUFGP	0.000
x_in<6>	8.750(R)	SLOW	-0.791(R)	FAST	clk_BUFGP	0.000
x_in<7>	8.854(R)	SLOW	-0.806(R)	SLOW	clk_BUFGP	0.000
x_in<8>	8.822(R)	SLOW	0.437(R)	SLOW	clk_BUFGP	0.000
x_in<9>	8.563(R)	SLOW	0.217(R)	SLOW	clk_BUFGP	0.000
x_in<10>	8.578(R)	SLOW	0.433(R)	SLOW	clk_BUFGP	0.000
x_in<11>	8.035(R)	SLOW	-0.126(R)	SLOW	clk_BUFGP	0.000
x_in<12>	8.365(R)	SLOW	-0.129(R)	SLOW	clk_BUFGP	0.000
x_in<13>	7.287(R)	SLOW	-0.490(R)	SLOW	clk_BUFGP	0.000
x_in<14>	6.604(R)	SLOW	-0.203(R)	SLOW	clk_BUFGP	0.000
x_in<15>	9.913(R)	SLOW	-0.951(R)	FAST	clk_BUFGP	0.000
y_in<0>	10.410(R)	SLOW	0.300(R)	SLOW	clk_BUFGP	0.000
y_in<1>	10.501(R)	SLOW	0.118(R)	SLOW	clk_BUFGP	0.000
y_in<2>	10.597(R)	SLOW	0.203(R)	SLOW	clk_BUFGP	0.000
y_in<3>	10.239(R)	SLOW	0.001(R)	SLOW	clk_BUFGP	0.000
y_in<4>	10.641(R)	SLOW	0.252(R)	SLOW	clk_BUFGP	0.000
y_in<5>	10.346(R)	SLOW	-0.092(R)	SLOW	clk_BUFGP	0.000
y_in<6>	10.746(R)	SLOW	-0.776(R)	SLOW	clk_BUFGP	0.000
y_in<7>	10.460(R)	SLOW	-0.669(R)	SLOW	clk_BUFGP	0.000
y_in<8>	11.071(R)	SLOW	-0.523(R)	SLOW	clk_BUFGP	0.000
y_in<9>	10.702(R)	SLOW	-0.438(R)	SLOW	clk_BUFGP	0.000
y_in<10>	10.802(R)	SLOW	-0.670(R)	SLOW	clk_BUFGP	0.000
y_in<11>	10.636(R)	SLOW	-0.650(R)	SLOW	clk_BUFGP	0.000
y_in<12>	10.913(R)	SLOW	-0.708(R)	SLOW	clk_BUFGP	0.000
y_in<13>	10.388(R)	SLOW	-0.330(R)	SLOW	clk_BUFGP	0.000
y_in<14>	10.464(R)	SLOW	-0.358(R)	SLOW	clk_BUFGP	0.000
y_in<15>	11.537(R)	SLOW	-1.159(R)	FAST	clk_BUFGP	0.000

Clock clk to Pad ۴-۱۱-۲

حداکثر زمان لازم برای عبور داده های ورودی به فلیپ فلاپ یا لچ از طریق logic ها و مسیر مورد نظر را قبل از لبه کلاک بعدی و رسیدن ورودی به خروجی تراشه، مشخص می‌کند. این گزارش شامل تاخیر clock-to-Q و تاخیر در عبور از مسیر آن به سمت خروجی است.



گزارش این تاخیر ها در جدول زیر آمده است:

جدول ۳ : Clock clk to pad در سنتز

Destination	Max (slowest) clk (edge) to PAD	Process Corner	Min (fastest) clk (edge) to PAD	Process Corner	Internal Clock(s)	Clock Phase
done	8.089(R)	SLOW	3.412(R)	FAST	clk_BUFGP	0.000
z_out<0>	9.021(R)	SLOW	3.866(R)	FAST	clk_BUFGP	0.000
z_out<1>	9.183(R)	SLOW	3.962(R)	FAST	clk_BUFGP	0.000
z_out<2>	9.205(R)	SLOW	4.039(R)	FAST	clk_BUFGP	0.000
z_out<3>	9.164(R)	SLOW	4.034(R)	FAST	clk_BUFGP	0.000
z_out<4>	8.886(R)	SLOW	3.810(R)	FAST	clk_BUFGP	0.000
z_out<5>	9.014(R)	SLOW	3.879(R)	FAST	clk_BUFGP	0.000
z_out<6>	9.166(R)	SLOW	3.989(R)	FAST	clk_BUFGP	0.000
z_out<7>	9.695(R)	SLOW	4.293(R)	FAST	clk_BUFGP	0.000
z_out<8>	10.038(R)	SLOW	4.425(R)	FAST	clk_BUFGP	0.000
z_out<9>	10.559(R)	SLOW	4.478(R)	FAST	clk_BUFGP	0.000
z_out<10>	10.525(R)	SLOW	4.345(R)	FAST	clk_BUFGP	0.000
z_out<11>	11.107(R)	SLOW	4.675(R)	FAST	clk_BUFGP	0.000
z_out<12>	11.306(R)	SLOW	4.781(R)	FAST	clk_BUFGP	0.000
z_out<13>	10.963(R)	SLOW	4.550(R)	FAST	clk_BUFGP	0.000
z_out<14>	10.803(R)	SLOW	4.365(R)	FAST	clk_BUFGP	0.000
z_out<15>	11.296(R)	SLOW	4.636(R)	FAST	clk_BUFGP	0.000

جدول ۴ : Clock clk to pad در پیاده سازی

Destination	Max (slowest) clk (edge) to PAD	Process Corner	Min (fastest) clk (edge) to PAD	Process Corner	Internal Clock(s)	Clock Phase
done	9.822(R)	SLOW	5.454(R)	FAST	clk_BUFGP	0.000
z_out<0>	8.183(R)	SLOW	4.431(R)	FAST	clk_BUFGP	0.000
z_out<1>	8.132(R)	SLOW	4.401(R)	FAST	clk_BUFGP	0.000
z_out<2>	8.167(R)	SLOW	4.420(R)	FAST	clk_BUFGP	0.000
z_out<3>	8.024(R)	SLOW	4.336(R)	FAST	clk_BUFGP	0.000
z_out<4>	7.937(R)	SLOW	4.285(R)	FAST	clk_BUFGP	0.000
z_out<5>	8.155(R)	SLOW	4.413(R)	FAST	clk_BUFGP	0.000
z_out<6>	8.116(R)	SLOW	4.431(R)	FAST	clk_BUFGP	0.000
z_out<7>	8.177(R)	SLOW	4.469(R)	FAST	clk_BUFGP	0.000
z_out<8>	9.836(R)	SLOW	5.479(R)	FAST	clk_BUFGP	0.000
z_out<9>	9.701(R)	SLOW	5.197(R)	FAST	clk_BUFGP	0.000
z_out<10>	9.802(R)	SLOW	5.138(R)	FAST	clk_BUFGP	0.000
z_out<11>	10.034(R)	SLOW	5.304(R)	FAST	clk_BUFGP	0.000
z_out<12>	9.844(R)	SLOW	5.198(R)	FAST	clk_BUFGP	0.000
z_out<13>	9.945(R)	SLOW	5.238(R)	FAST	clk_BUFGP	0.000
z_out<14>	10.056(R)	SLOW	5.201(R)	FAST	clk_BUFGP	0.000
z_out<15>	10.177(R)	SLOW	5.184(R)	FAST	clk_BUFGP	0.000

۴-۱۱-۳ Clock to Setup on destination clock clk

جدول ۵ : Clock to Setup on destination clock clk

Source Clock	Src:Rise | Dest:Rise	Src:Fall | Dest:Rise	Src:Rise | Dest:Fall	Src:Fall | Dest:Fall
clk	4.160			

۴-۱۱-۴ Pad to Pad

یک مسیر که از یک عنصر ساعت دار شروع می‌شود و به یک عنصر ساعت دار ختم می‌شود. ورودی، یک عنصر ساعت دار خارج از FPGA است که خصوصیات آن توسط set_input_delay شرح داده شده است. به طور مشابه خروجی، عنصر ساعت دار خارج از FPGA است که خصوصیات آن توسط set_output_delay شرح داده شده است.

اگر از طریق FPGA یک اتصال ترکیبی وجود داشته باشد، فقط در صورتی که ورودی دارای set_input_delay باشد و خروجی دارای یک دستور set_output_delay باشد، تبدیل به " path" می‌شود. در این صورت، مسیر مانند سایر مسیرهای زمان بندی دیگر زمان بندی می‌شود و به همراه تمام مسیرهای زمان بندی دیگر مرتب می‌شوند. این روش "correct" برای محدود کردن اتصال ترکیبی از طریق FPGA است.

جدول ۶ : Pad to Pad در سنتز

Source Pad	Destination Pad	Delay
x_in<15>	z_out<8>	7.334
x_in<15>	z_out<9>	7.618
x_in<15>	z_out<10>	7.425
x_in<15>	z_out<11>	8.134
x_in<15>	z_out<12>	8.510
x_in<15>	z_out<13>	8.167
x_in<15>	z_out<14>	8.007
x_in<15>	z_out<15>	8.500
y_in<15>	z_out<8>	9.250
y_in<15>	z_out<9>	9.534
y_in<15>	z_out<10>	9.341
y_in<15>	z_out<11>	9.923
y_in<15>	z_out<12>	10.181
y_in<15>	z_out<13>	9.838
y_in<15>	z_out<14>	9.678
y_in<15>	z_out<15>	10.171

جدول ۷ : Pad to Pad در پیاده سازی

Source Pad	Destination Pad	Delay
x_in<15>	z_out<8>	12.364
x_in<15>	z_out<9>	12.229
x_in<15>	z_out<10>	12.330
x_in<15>	z_out<11>	12.562
x_in<15>	z_out<12>	12.372
x_in<15>	z_out<13>	12.472
x_in<15>	z_out<14>	12.520
x_in<15>	z_out<15>	12.635
y_in<15>	z_out<8>	8.109
y_in<15>	z_out<9>	7.974
y_in<15>	z_out<10>	8.130
y_in<15>	z_out<11>	8.364
y_in<15>	z_out<12>	8.213
y_in<15>	z_out<13>	8.313
y_in<15>	z_out<14>	8.453
y_in<15>	z_out<15>	8.570

۴-۱۲ گزارش نتایج

جدول ۸ : *Synthesis Options Summary*

Target Device	xc6slx150-3-fgg484
Top Module Name	interface
Optimization Goal and Effort	Speed 1

جدول ۹ : *Register Report*

Registers	Used
# Flip-Flops	633

جدول ۱۰ : *Design Summary*

Primitive and Black Box	Used
# S :	5228
# GND	1
# INV	204
# LUT1	46
# LUT2	435
# LUT3	1354
# LUT4	315
# LUT5	62
# LUT6	269
# MUXCY	1249
# VCC	1
# XORCY	1292

# FlipFlops/Latches : # FDC # FDCE	633 629 4
# Clock Buffers : # BUFGP	1 1
# IO Buffers : # IBUF # OBUF	51 34 17

جدول ١١ : *Slice Logic Utilization*

Slice Logic	Used	Utilization
# of Slice Registers : # used as Flip Flops	633 out of 184,304 633	1%
# of Slice LUTs : # used as logic # using O6 output only # using O5 output only # using O5 and O6 # used exclusively as route-thrus # with same-slice carry load	2,095 out of 92,152 2,047 out of 92,152 1,411 30 606 48 48	2% 2%

جدول ١٢ : *Slice Logic Distribution*

Slice Logic	Used	Distribution
# of occupied Slices	595 out of 23,038	2%
# of MUXCYs used	1,348 out of 46,076	2%

# of LUT Flip Flop pairs used	2,096	
# with an unused Flip Flop	1,616 out of 2,096	77%
# with an unused LUT	1 out of 2,096	1%
# of fully used LUT-FF pairs	479 out of 2,096	22%

جدول ۱۳ : *IO Utilization*

IO	Used	Utilization
# of bonded IOBs	52 out of 338	15%

جدول ۱۴ : *Specific Feature Utilization*

Specific Feature	Used	Utilization
# of BUFG/BUFGMUXs	1 out of 16	6%
# used as BUFGs	1	

جدول ۱۵ : *Timing Report*

Minimum period	5.580ns
Maximum Frequency	179.211MHz
Minimum input arrival time before clock	9.888ns
Maximum output required time after clock	5.076ns
Maximum combinational path delay	7.148ns

جدول ۱۶ : وضعیت پروژه در پایان پیاده سازی

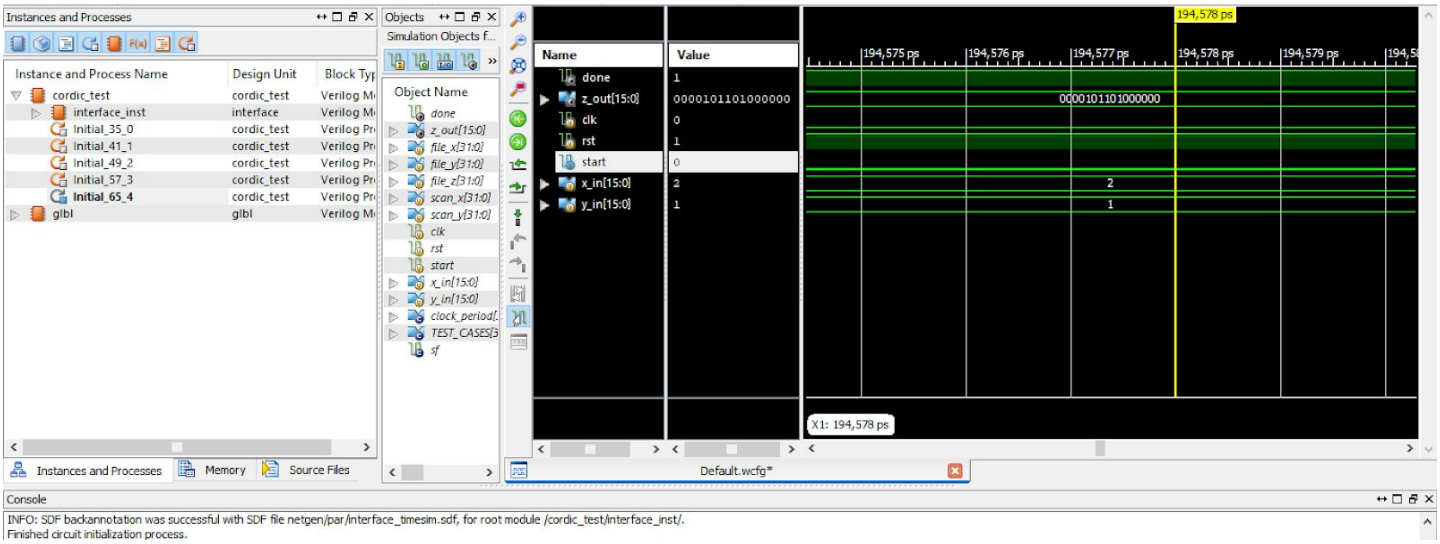
interface Project Status (08/08/2020 - 09:43:06)			
Project File:	Final.xise	**Parser Errors:**	No Errors
Module Name:	interface	**Implementation State:**	Placed and Routed
Target Device:	xc6slx150-3fgg484	• **Errors:**	No Errors
Product Version:	ISE 14.7	• **Warnings:**	98 Warnings (0 new)
Design Goal:	Balanced	• **Routing Results:**	All Signals Completely Routed
Design Strategy:	Xilinx Default (unlocked)	• **Timing Constraints:**	All Constraints Met
Environment:	System Settings	• **Final Timing Score:**	0 (Timing Report)

جدول ۱۷ : *Performance Summary*

Performance Summary	
Final Timing Score:	0 (Setup: 0, Hold: 0)
Routing Results:	All Signals Completely Routed
Timing Constraints:	All Constraints Met

* Post-implementation timing simulation (امتیازی)

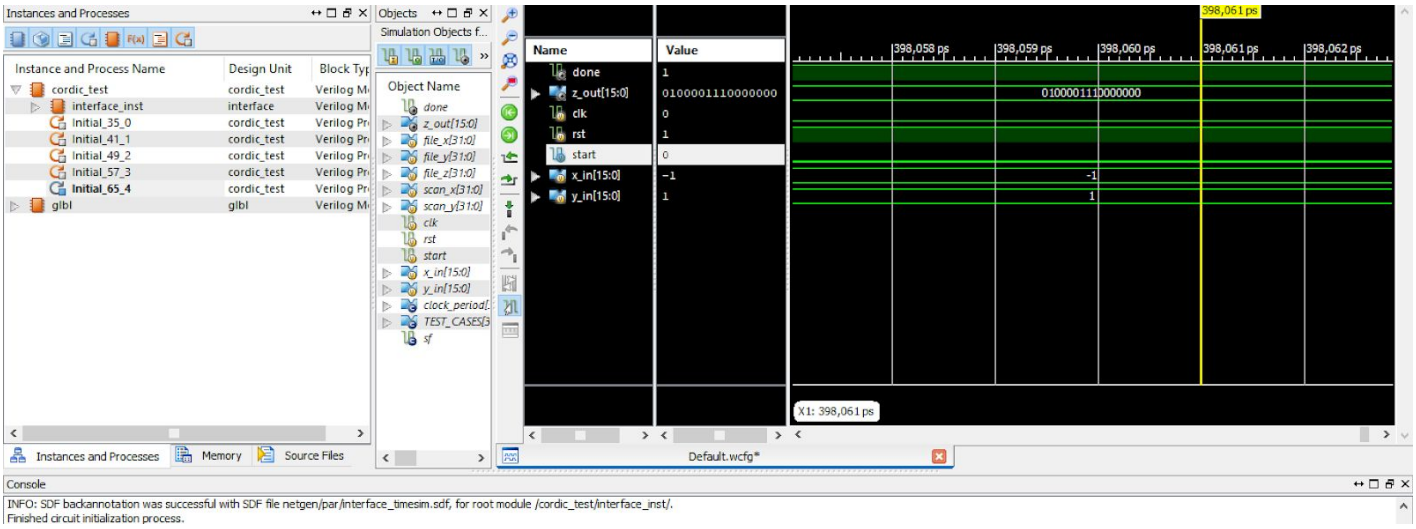
پس از پایان مراحل سنتز و پیاده سازی، مدل شبیه سازی پس از پیاده سازی[20] را به کمک نرم افزار تولید کردیم. سپس view را در نرم افزار به simulation تغییر دادیم و آن را در حالت Post-Route قرار دادیم. در نهایت توسط ISim شبیه سازی را در محیط انجام دادیم.



شکل ۳۱ : نتیجه شبیه‌سازی پس از پیاده سازی ۱-

همانگونه که در تصویر مشاهده می‌کنید مولفه‌های X و Y بردار به ترتیب اعداد 2 و 1 هستند. زاویه‌ای که این بردار با محور X می‌سازد عدد 22.5 درجه است که دقت آن در حالت باینری 7 بیت اعشار است.

[20] Post-Place & Route Simulation Model



شکل ۳۲ : نتیجه شبیه‌سازی پس از پیاده سازی ۲-

همانگونه که در تصویر مشاهده می‌کنید مولفه‌های X و Y بردار به ترتیب اعداد 1- و 1 هستند. زاویه‌ای که این بردار با محور X می‌سازد عدد 135 درجه است که دقت آن در حالت باینری 7 بیت اعشار است.

لازم به ذکر است که این نتایج با نتایج حاصل از شبیه سازی اولیه مدار که در بخش های قبلی گفته شد تطابق کامل دارد.

۵ نتیجه گیری

در این گزارش ابتدا به توضیح تاریخچه و کاربردهای الگوریتم CORDIC پرداختیم. برای تست صحت عملکرد سیستم، مدل طلایی به زبان پایتون طراحی شد.
در ادامه‌ی گزارش توضیحاتی در باب پایه ریاضی و جبری اعمال شده در الگوریتم مورد استفاده بیان گردید.
در مقایسه‌های انجام شده با مدل طلایی به دقت بسیار بالایی رسیدیم به طوری که تنها تفاوت خروجی سیستم با نتایج مدل طلایی در حداکثر یک بیت کم ارزش بود، که آن هم تنها در زمان‌هایی که نمی‌توان محاسبه را با دقت کامل انجام داد رخ می‌دهد که در گرد کردن عدد، مدل طلایی دقت بالاتری داشت.
در ادامه نکاتی در باب اسکریپت‌های نوشته شده به زبان پایتون و TCL و نحوه‌ی استفاده از آن‌ها به منظور خودکار کردن مراحل تست و شبیه‌سازی بیان گردید.
در انتها این سیستم را با استفاده از نرم افزار Xilinx سنتز، پیاده‌سازی و تست کردیم

لازم به ذکر است که تعداد کلاک های اجرایی عملیات محاسبه زاویه در مدار طراحی شده 18 کلاک است.

چند ویژگی از نحوه پیاده سازی این الگوریتم مشهود هستند:

- **سادگی** کلی الگوریتم، به علت ساده بودن عملیات ریاضی صورت گرفته در آن (جمع و شیفت)
- **سرعت مناسب** الگوریتم، به علت ساختار pipeline مانند آن
- **مصرف انرژی کم**

مراجع

[1] *Xilinx - Timing Report - URL:*
https://www.xilinx.com/support/documentation/sw_manuals/xilinx10/isehelp/pta_p_ttc-timing-report-view.htm
[2] *Xilinx - Synthesis - URL:*
https://www.xilinx.com/support/documentation/sw_manuals/xilinx11/ise_c_using_xst_for_synthesis.htm
[3] *Xilinx - Synthesis - URL:*
https://forums.xilinx.com/t5/Synthesis/What-s-the-difference-between-Primitive-and-Black-Box-Usage-and/td-p/312167
[4] *Xilinx - Crossing Clock Domains - URL:*
https://forums.xilinx.com/t5/Timing-Analysis/how-to-list-the-paths-of-Crossing-clock-domains-with-vivado/td-p/514353
[5] *Xilinx - Data Sheet Report - URL:*
https://www.xilinx.com/support/documentation/sw_manuals/xilinx11/pta_p_ar_data_sheet_report.htm
[6] *Xilinx - Clock to Pad - URL:*
https://www.xilinx.com/support/documentation/sw_manuals/xilinx10/isehelp/pce_c_clock_to_pad_top.htm#:~:text=A%20clock%20to%20pad%20time,flop%20to%20the%20output%20pad.
[7] *Xilinx - Documentation - URL:*
https://www.xilinx.com/support/documentation/sw_manuals/xilinx11/ise_c_using_xst_for_synthesis.htm
[8] *Xilinx - Documentation - URL:*
https://www.xilinx.com/support/documentation/sw_manuals/xilinx11/xst.pdf
[9] *Xilinx - Documentation - URL:*
https://www.xilinx.com/support/documentation/sw_manuals/xilinx11/xst.pdf
[10] *Wikipedia - CORDIC Algorithm - URL:*
https://en.wikipedia.org/wiki/CORDIC
[11] *IJCSMC - Implementation of Fast CORDIC Algorithm for Embedded Application - URL:*
https://ijcsmc.com/docs/papers/September2017/V6I9201715.pdf
[12] Andraka, R., 1998, March. A survey of CORDIC algorithms for FPGA based computers. In *Proceedings of the 1998 ACM/SIGDA sixth international symposium on Field programmable gate arrays* (pp. 191-200).
[13] Lakshmi, B. and Dhar, A.S., 2010. CORDIC architectures: a survey. *VLSI design*, *2010*.
[14] Yadav, P. and Singh, K., A Review Paper on CORDIC Algorithm and Its Applications for Current Technology. *InternationalJournal of Science and Research (IJSR)*.
[16] Sudha, J., Hanumantharaju, M.C., Venkateswarulu, V. and Jayalaxmi, H., 2012. A novel method for computing exponential function using cordic algorithm. *Procedia Engineering*, *30*, pp.519-528.