به نام خدا

گزارش درس مدار های منطقی برنامه پذیر

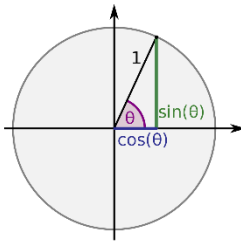
زهرا عربی 9523083 – سپیده زهدی 9523055

شرح پروژه:

الگوریتم cordic یک الگوریتم رایج برای محاسبه ی توابعی نظیر مثلثاتی و هایپربولیک می باشد.در این پروژه می خواهیم این الگوریتم را برای محاسبه ی سینوس و کسینوس زاویه ی ورودی در نظر بگیریم.

برای این کار علاوه بر کلاک و سیگنال ریست نیاز به دو ورودی زاویه و سیگنال استارت داریم.طبق سوال سیگنال استارت در حالت عادی صفر می باشد و هنگامی که بخواهیم زاویه ای را به عنوان ورودی بدهیم مقدارش یک می شود.برای داشتن حالت pipeline بهتر بود که این سیگنال را سنکرون با کلاک بگیریم.چراکه هر گونه تغییرات در کلاک بعدی اعمال شود و حالتی از دست نرود.

این الگوریتم بر اساس چرخش های متوالی یک زاویه می باشد.برای این کار باید یک نقطه ی شروع اولیه برای آن در نظر بگیریم و سپس طبق فرمول زیر در هر مرحله مختصات جدید را محاسبه کنیم و زاویه ی به دست آمده در هر مرحله را به زاویه ی مطلوب مقایسه کنیم این کار را به صورت متوالی تا جایی انجام می دهیم که یا تعداد مراحل یعنی 16 مرحله تمام شود یا جاییکه زاویه به مقدار مطلوب رسیده باشد.



$$x[i+1] = x[$$

$$y[i+1] = y[i \quad \begin{bmatrix} x_R \\ y_R \end{bmatrix} = cos(\theta) \begin{bmatrix} 1 & -tan(\theta) \\ tan(\theta) & 1 \end{bmatrix} \begin{bmatrix} x_{in} \\ y_{in} \end{bmatrix}$$

$$z[i+1] = z[i] - \sigma_i tan^{-1}(2^{-i})$$

برای map کردن اعداد عشاری به اعداد باینری (17 بیت ورودی) ابتدا یک بیت به عنوان بیت علامت در نظر گرفته ایم و دو بیت بعدی را برای اعداد صحیح اختصاص داده ایم و بقیه ی بیت ها را یعنی 14 بیت را برای قسمت اعشاری در نظر گرفته ایم.زاویه ها را برای نمایش در این فرمت باید به رادیان محاسبه کرد زیرا باید قسمت صحیح آن کمتر از 4 باشد.

برای قسمت اعشاری از یک تبدیل ساده استفاده شده است:

$$0 \rightarrow dec \rightarrow 1$$

$$0 \rightarrow bin \rightarrow 2^{14}$$

بنابر این برای زاویه های مثبت این تبدیل به صورت زیر می باشد:

$$\frac{\pi}{2} \rightarrow 00110010001111011$$

$$\frac{\pi}{4} \rightarrow 00011000111101100$$

برای زاویه های منفی مکمل 2 در نظر گرفته ایم:

$$-\frac{\pi}{2} \rightarrow 11001101110000101$$

با توجه به فرمول ذکر شده باید در هر مرحله مقدار $\tan^{-1}(2^{-i})$ محاسبه گرد که برای سادگی کار آن را برای 16 مرحله حساب کرده و در جدولی به نام lookup table قرار میدهیم.توجه داشته باشید که از سطر آخر این جدول استفاده نمی شود زیراکه خروجی مرحله قبل همان خروجی کل ماژول می باشد.

$$\mathrm{i} = 0 \rightarrow \tan^{-1}(1) = 45° = \frac{\pi}{4} \rightarrow 00011000111101100$$

و به همین صورت برای بقیه ی زوایا و مراحل این کار را تکرار میکنیم.

برای نوشتن کد از یک پروسس استفاده شده که در ابتدای پروسس باید سیگنال ریست را چک کنیم که اگر این سیگنال ۱ شده باشد تمامی متغیر ها و خروجی ها باید صفر گردد.

سپس با هر لبه ی بالارونده ی کلاک، مراحل state machine را چک میکنیم. مراحل گفته شده به شرح زیر می باشد:

1. Idle
در این مرحله اتفاقی نمی افتد و ماژول منتظر سیگنال استارت می باشد و تا زمانی که سیگنال استارت 0 باشد در این مرحله باقی می ماند.
2. Cs_sign
در این مرحله باید علامت زاویه ی ورودی چک گردد.اگر زاویه ی ورودی در ناحیه ی 3 و 4 باشد علامت sign_ch یک می شود و اگر درناحیه ی 1 و 2 باشد، صفر می شود. بعد از مشخص کردن بیت علامت زوایای منفی را مثبت میکنیم (یعنی تمامی زوای را در نیمکره ی بالایی در نظر میگیریم.) سپس باید مختصات اولیه را برای شروع بدهیم.در کد پروژه xoc به عنوان cos و xos به عنوان sin درنظر گرفته شده اند. برای نقطه ی شروع باید زاویه ی 45 درجه در نظر گرفته شود.با توجه به اینکه شعاع دایره را 1 فرض میکنیم سینوس و کسینوس 45 درجه معادل با $0.7 = \frac{\sqrt{2}}{2}$ می باشد که با توجه به مپینگ بالا مقدار باینری آن معادل زیر می باشد:

$$0.7 * \ 2^{14} \approx 11469 \rightarrow (000)\ \&\ (0010\ 1100\ 1100\ 1101)$$

اکنون چک میکنیم که زاویه ی در نیمکره ی بالایی در کدام ربع است.در این حالت اگر زاویه در ربع دوم باشد با کم کردن مقدار 180 از آن یا مقدار 90 از زاویه آن را به ربع اول map میکنیم.(یعنی تمامی محاسبات ما در ربع اول انجام میشود بدون اینکه سینوس یا کسینوس جا به جا شود.)
در این مرحله با توجه به اعداد خروجی دریافتیم که باید offset برای خروجی در نظر بگیریم تا حاصل محاسبه شده درست باشد که این آف ست با توجه به مثبت یا منفی بودن ورودی به زاویه ی ورودی برای سینوس یا زاویه ی ورودی برای کسینوس اضافه یا کم میشود. از مقدار محاسبه شده در این مرحله باید در مرحله ی آخر که به خروجی اضافه یا از آن کم بشود.
3.initial

در مرحله ی قبل زوایای رند 0، 90 و 180 را در نظر نگرفتیم.این زوایای رند را در این مرحله به صورت جدا مقدار دهی میکنیم زیرا اگر مراحل به صورتی پیش برود که زاویه ی خروجی مرحله ای به یکی ازین زوایای رند ختم شود،دیگر نیازی به پیش روی نیست و همان عدد را به عنوان خروجی کلی باید در نظر بگیریم.

compute.4

در این مرحله دو مرتبه محاسبه میکنیم ابتدا برای کسینوس محاسبات را انجام می دهیم و سپس دقیقا همان مراحل را برای سینوس تکرار میکنیم.

برای شروع محاسبه ی کسینوس دو شرط باید چک شود. اولی اینکه فلگ مربوط به محاسبه س کسینوس ۰ باشد زیرا در صورتی که محاسبه به اتمام رسیده باشد این فلگ ۱ می شود.(مثلا قبل از ۱۶ مرحله به یک زاویه ی رند اشاره شده رسیده باشد.)و دومین شرط تعداد مراحل است که باید از ۰ تا iteration-1 مرحله داشته باشیم.طبق فرمول گفته شده خروجی مرحله ی قبل ورودی مرحله ی جدید می شود.ورودی سینوس باید در 2^{-i} ضرب شود و با توجه به علامت زاویه با مقدار قبلی جمع یا تفریق بشود.بنابراین خروجی مرحله ی قبل را در متغیر جدیدی میریزیم و آن را شیفت میدهیم.

برای شیفت دادن از جمع یا ۱۷ صفر استفاده شده و در این مرحله نمیتوان از & استفاده کرد زیرا طول ارایه متغیر در نظر گرفته شده است.این عمل شیفت را هم برای ورودی سینوس و هم ورودی کسینوس انجام میدهیم و سپس با توجه به علامت زاویه دو دسته فرمول برای ان ها در نظر میگیریم.

روند بالا را برای سینوس نیز تکرار میکنیم.

سپس در اخر این state باید وضعیت چک بشود که آیا باید در همین مرحله بماند یا به مرحله ی بعد برود.زمانی به مرحله ی بعد میرود که یا فلگ مربوط به سینوس و کسینوس ۱ شده باشد یا تعداد مراحل به اتمام رسیده باشد.

ex_finish.5

در این مرحله ابتدا خروجی حاصل از مرحله ی قبل را مثبت میکنیم.(اگر مثبت باشد که عملی روی آن انجام نمیگیرید ولی اگر منفی باشد آن را با مکمل ۲ مثبت میکنیم.) سپس بیت علامت آن را چک میکنیم.برای کسینوس بیت علامت sign_chکافی است زیرا cos(-x) = cos(x) اما برای سینوس علاوه بر بیت علامت sign_ch باید بیت علامت ssign را که در مراحل دادن مقدار اولیه ست شده بود را در نظر بگیریم.

اگر علامت منفی بود دوباره خروجی را مکمل ۲ را محاسبه کرده با اضافه کردن بیت علامت ان را به عنوان سینوس یا کسینوس خروجی قرار میدهیم.

مقدار آف ست باید در این مرحله اضافه میشد که به دلیل مشکل به وجود آمده در تست بنچ نتوانستیم اضافه کنیم.برای همین جواب خروجی مقدار بسیار کمی با جواب اصلی متفاوت می شود.

finish.6

در این مرحله یعنی خروجی خواسته شده محاسبه شده و باید سیگنال خروجی done صفر شود.و بعد از آن باید به حالت idle بازگردد.

تست بنچ:

در تست بنچ دو زاویه ۶۰ درجه و ۶۰- را به عنوان ورودی داده ایم.

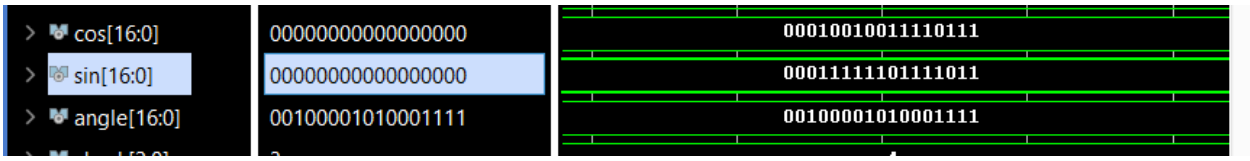
$$\frac{\pi}{3} = 1.04 \rightarrow 0.04 * 2^{14} \approx 655 = (00001010001111)_2 \rightarrow \frac{\pi}{3} = (00100001010001111)_2$$

$$-\frac{\pi}{3} \rightarrow 2th\ comp. \frac{\pi}{3} \rightarrow (11011110101110001)_2$$

```
wait for  150 ns;

    angle <= "0010000101000111"; --(pi/3)
    wait for clk_period/2;
    angle <= "1101111010111000"; --(-pi/3)
    wait for 30*clk_period;
wait;
end process;
```

نتایج اجرا را به صورت زیر میبینیم:



cos:

000 -> +0.

10010011110111 -> 9463 -> $\frac{9463}{2^{14}}$= 0.57

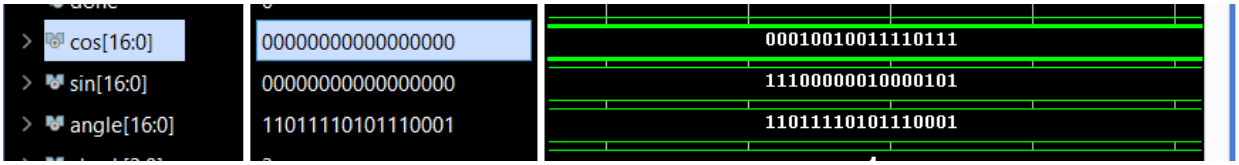
$$\cos\left(\frac{\pi}{3}\right) = +0.57$$

sin:

000 -> +0.

11111101111011 -> 16251 -> $\frac{16251}{2^{14}}$= 0.89

$$\sin\left(\frac{\pi}{3}\right) = +0.89$$



همانگونه که میبینیم مقدار کسینوس تغییری نکرده و همان مقدار قبلی می باشد اما مقدار سینوس تغییر کرده

sin:

1 -> -

11100000010000101 -> 2th comp. -> 00011111101111011

11111101111011 -> 16251 -> $\frac{16251}{2^{14}}$= 0.89

$$\sin\left(\frac{\pi}{3}\right) = -0.89$$