گزارش پیاده سازی تکلیف 6 هم طراحی سخت افزار و نرم افزار

غزل کلهری 97243058

عرفان مشیری 97243066

هدف از انجام این تکلیف، شبیه سازی سیستم سخت افزار – نرم افزار با استفاده از محیط GEZEL برای محاسبه سینوس و کسینوس زوایای مختلف بر اساس الگوریتم CORDIC بود. بدین ترتیب که برنامه داده شده در صورت سوال را به دو بخش نرم افزاری و سخت افزاری تفکیک کرده و از طریق پردازنده ARM ارتباط این دو بخش را برقرار کردیم. در ادامه توضیحات هریک از بخش ها و سپس نتایج بدست آمده در دو محیط و مقایسه آنها با نتایج حاصل از اجرای برنامه اصلی آورده شده است.

1. بخش نرم افزار:

بغیر از بخشی از برنامه اصلی (main.c) که مسئول عملیات محاسبه سینوس و کسینوس است و برای سرعت عمل بیشتر در کد سخت افزار آورده شده است، تمامی بخش های دیگر در کد نرم افزار پیاده سازی شده است. بدین ترتیب که یک فایل cordic.c داریم که تابع main آن مشابه با برنامه اصلی بوده و زمانیکه به بخش محسابات میرسد تابع cordic_driver را فراخوانی میکند و مقادیر sin, cos تابع main را بصورت رفرنس به آن پاس میدهد تا به بخش سخت افزار رفته و پس از محاسبه مقادیر، نتایج آنها برگردانده شود.

تابع cordic_driver ورودی های theta، iterations و sin, cos را گرفته و مقادیر آنها را در پوینترهایی که برای ارتباط با پردازنده ARM تعریف کرده ایم ست میکند. سپس منتظر میماند تا مقادیر خروجی از این پوینترها که سینوس و کسینوس مد نظر ما هستند از طرف سخت افزار مقدار دهی شوند، سپس آنها را در ورودی های sin, cos خودش که بصورت رفرنس گرفته بود ست میشوند تا در ادامه بخش نرم افزار مورد استفاده قرار گیرند.

```
// connection with ARM core
volatile unsigned int *channel_exists = (int *) 0x80000000;
volatile unsigned int *channel_theta = (int *) 0x80000004;
volatile unsigned int *channel_itr = (int *) 0x80000008;
volatile unsigned int *channel_cos = (int *) 0x8000000C;
volatile unsigned int *channel_sin = (int *) 0x80000016;
```

همچنین در این بخش از متغیر exists برای اینکه سخت افزار بداند ورودی های نرم افزار پاس داده شده اند و زمان محسابات فرا رسیده است استفاده میکنیم. وقتی مقدار آن 1 میشود بدین معنا است که ورودی ها بطور کامل آماده هستند و وقت شروع کار سخت افزار است.

نکته: نتیجه بخش نرم افزار در انتهای کد با دستور printf نمایش داده میشود تا از صحت گرفتن نتایج از بخش سخت افزار مطمئن شویم.

2. بخش سخت افزار:

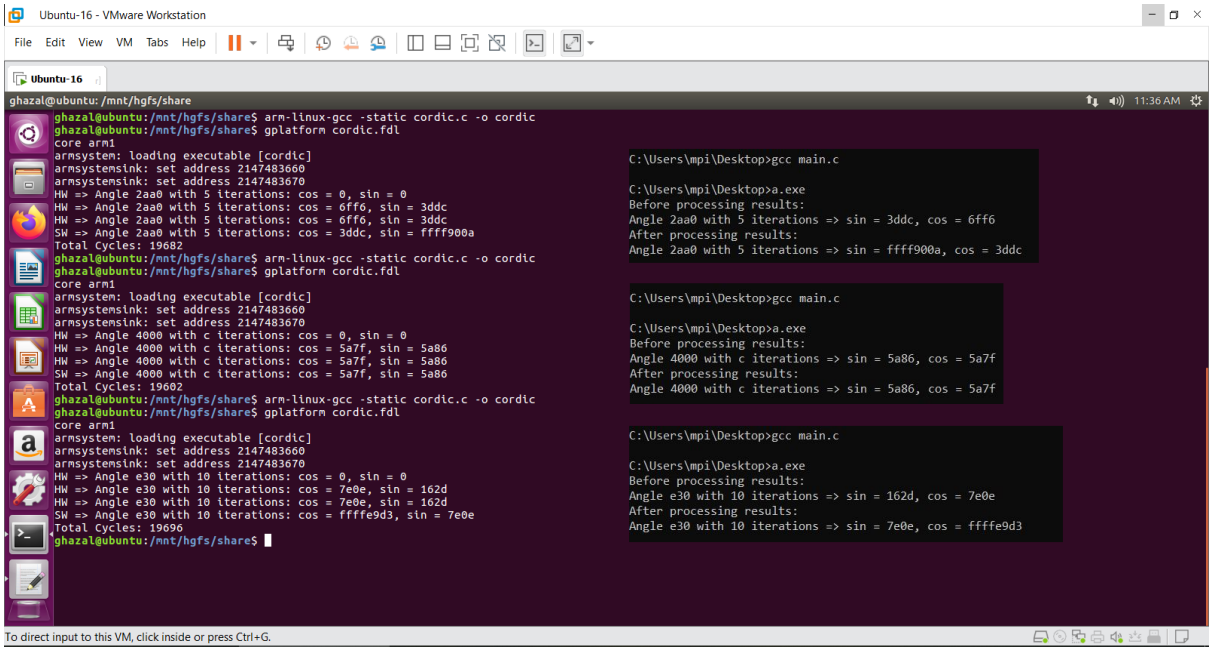
ابتدا تعدادی ipblock تعریف میکنیم که برای ارتباط با بخش نرم افزار مورد استفاده قرار گیرد و آدرس هر ipparm را با متغیر نظیر به نظیر آن در بخش نرم افزار مساوی قرار میدهیم. در ادامه dp cordic را داریم که از تعدادی sfg برای انجام محاسبات ریاضی تشکیل میشود و fsm cordic_controller را داریم که شروط و ترتیب اجرای sfg های cordic را کنترل میکند.

cordic ما شامل ورودی های گرفته شده از بخش نرم افزار، آرایه atantable و تعدادی رجیستر و سیگنال برای انجام محاسبات بر روی زوایای ورودی است. sfg capture مقداردهی های اولیه متغیرهای مارا انجام میدهد. sfg iterate محاسبات ریاضی را مدیریت کرده (بر اساس برنامه اصلی) و و نتیجه حاصل از محاسبات توسط sfg write در خروجی های cordic ست میشود تا به بخش نرم افزار برگردد. sfg jump_over نیز کار خاصی انجام نمیدهد و برای مدیریت استیت هایی که نیاز به پرش دارند گذاشته شده است.

cordic_controller شامل 3 استیت است. استیت s0 یا initial به شرط آماده بودن ورودی های بخش نرم افزار، برای مقداردهی اولیه capture را فراخوانی میکند، در غیر اینصورت به خودش برمیگردد مادامیکه ورودی ها حاضر نباشند. استیت s1 با چک کردن شرط ورود به حلقه iterate را فراخوانی میکند تا محاسبات شروع شود و در صورت عدم برقراری شرط حلقه به s2 میرود تا عملیات پایان یابد. با تمام شدن حلقه، write در s2 فراخوانی شده تا نتیجه محاسبات در خروجی های cordic نوشته شود.

در نهایت در dp tb با دستور use کلیه ipblock های تعریف شده را به همراه dp cordic بکار گرفته میشوند. مقادیر ورودی گرفته شده از نرم افزار از طریق ipblock هایی که ورودی out دارند در سیگنال های تعریف شده ست شده و به cordic پاس داده میشوند. در طرف مقابل خروجی های گرفته شده از cordic از طریق ipblock هایی که ورودی in دارند به بخش نرم افزار برگردانده میشوند.

3. نتایج:



سه ورودی داده شده در بالا دوبار، یکبار با اجرای main.c که صورت سوال بود (سمت راست) و بار دیگر با استفاده از برنامه ای که نوشتیم (سمت چپ) با دستور gplatform شبیه سازی و تست شده اند. همانطور که در سمت چپ مشاهده میشود، به ازای هر ورودی دو خروجی با کد HW, SW داریم که دستور چاپ هریک در دو بخش نرم افزار و سخت افزار مربوطه اجرا شده و نتیجه یکسانی را به ما نمایش میدهد.

ورودی اول: theta = -60, iterations = 5

ورودی دوم: theta = 45, iterations = 12

ورودی سوم: theta = 100, iterations = 300