بسمه تعالی

Codesign Hw2

سید محمدرضا حسینی 97243129

سید عباس میرقاسمی 97243068

# ۱ مقدمه

در این گزارش به طراحی و پیاده سازي یک FSMD یا Finite State Machine with Datapath براي محاسبه حاصل ضرب گفته شده در صورت سوال میپردازیم. به طور خلاصه براي دستیابی به یک Datapath مناسب ابتدا لازم است که یک CFG و در ادامه یک DFG از روي آن تولید شود. در یک data edge ،DFG هاي روي آن به شکلی ارتباطات Datapath ما را تشکیل می دهند و بدین صورت Datapath ما به طور خاص در دسترس است؛ همچنین control edge هاي CFG براي واحد کنترل این Datapath استفاده می شود.

هر variable در زبان C به یک رجیستر تبدیل می شود و در این حالت نیاز به یک multiplexer براي انتخاب خروجی هاي variable جهت نوشته شدن در رجیستر لازم است. FSMD نیز به طور ضمنی شکلی از DFG ایجاد شده با در نظر گرفتن run ها در state هاي مختلف می باشد.

ابزار gezel یک ابزار کاربردي براي ملموس سازي توصیف سخت افزار و نرم افزار در کنار هم می باشد و در این گزارش به پیاده سازي FSMD ذکر شده به وسیله این ابزار می پردازیم .

# ۲ نحوه پیاده سازي پروژه با ابزار Gezel

در ابزار Gezel متغیر ها می توانند به صورت sig یا reg مورد استفاده قرار بگیرند که تفاوت آنها مانند تفاوت آنها در زبان HDL است .

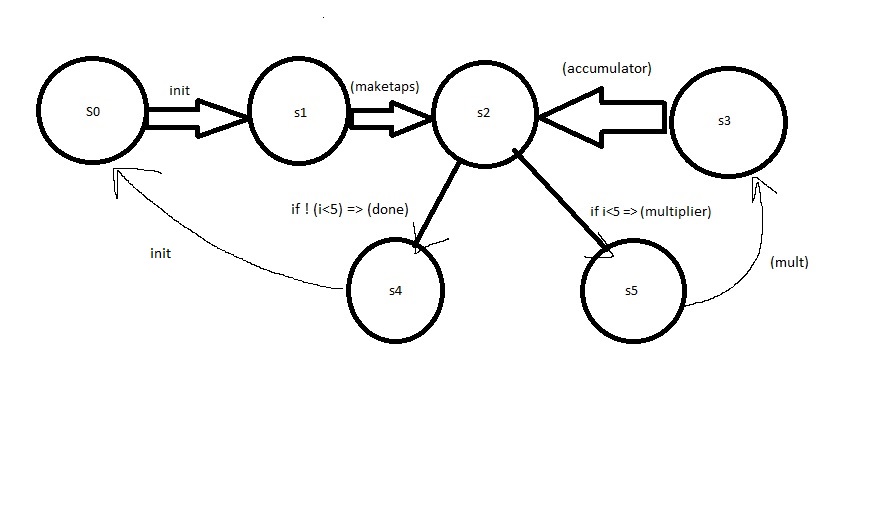
(رجیستر ها صرفا می توانند دو مقدار داشته باشند و)... در صورت استفاده از یک “Always” Statement مقادیر داخل آن با هر کلاك سیستم تغییر می کنند؛ نکته اي که اینجا مد نظر است، عدم دسترسی مستقیم به کلاك توسط ماست و کلاك به صورت مستقل توسط سیستم تنظیم شده و تغییر می کند. always و sfg که به ترتیب SFG "بی نام" و "با نام" در این ابزار نام دارند در حقیقت مجموعه اي از دستورات ریاضی و منطقی هستند که در کلاك هاي مختلف سیستم اجرا می شوند و تفاوت آنها در این است که SFG هاي با نام بر خلاف بی نام ها صرفا در کلاك هایی که controller دستور اجرا شدن آنها را صادر کند، execute می شوند. با در کنار هم قرار گرفتن این SFG ها Datapath ما شکل می گیرد.

در بخش datapath filter با توجه به تمپلیت گفته شده در آخر pdf سوالات ، ورودی و خروجی رو مشخص کرده و سپس رجیستر های مورد نیاز را برای محاسبه حاصلضرب را تعریف کردیم . دلیل استفاده از نوع دیتای TC این است که در اعداد گفته شده اعداد منفی وجود دارد و استفاده از ns موجب عوض شدن حاصلضرب خواهد شد .

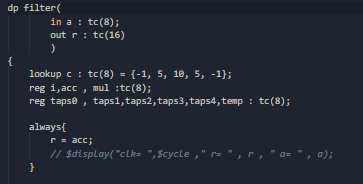
در ادامه 7 sfg ( شامل یک sfg ، Always و شش sfg با نام استفاده میکنیم . در always مقدار r را برابر رجیستر acc قرار میدهیم تا هنگام محاسبه نهایی حاصل بر روی خروجی نشان داده شود . در sfg init مقادیر اولیه به reg ها داده میشود با توجه به اینکه در صورتی که به taps ها مقدار داده نشود ، مقدار آن صفر در نظر گرفته میشود برای اطمینان از حاصلضرب به آن ها مقدار دلخواه دادیم . در maketaps عملیات انجام گرفته در حلقه اول کد c که یک شیفت به راست است انجام میشود . در ادامه به sfg multiplier میرویم در ابتدا چون میخواهیم فقط یک ضرب کننده داشته باشیم با یک سری ternary operation و با توجه به یک رجیستر I که خاصیت یک شمارنده را دارد یکی از taps ها انتخاب میشود و به sfg mult که برای به دست آوردن حاصلضرب taps در c است میرویم . سپس به sfg accumulator میرویم و حاصضلرب به دست آمده را در acc میریزیم . در این مرحله به sfg ، multiplier برگشته و با توجه به مقدار I ، یا با sfg done که sfg نهاییست رفته یا اینکه دوباره عملیات ذکر شده در بالا را تکرار میکنیم تا در آخر به حالت نهایی برسیم .

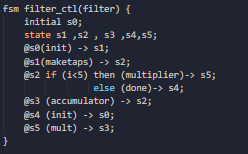
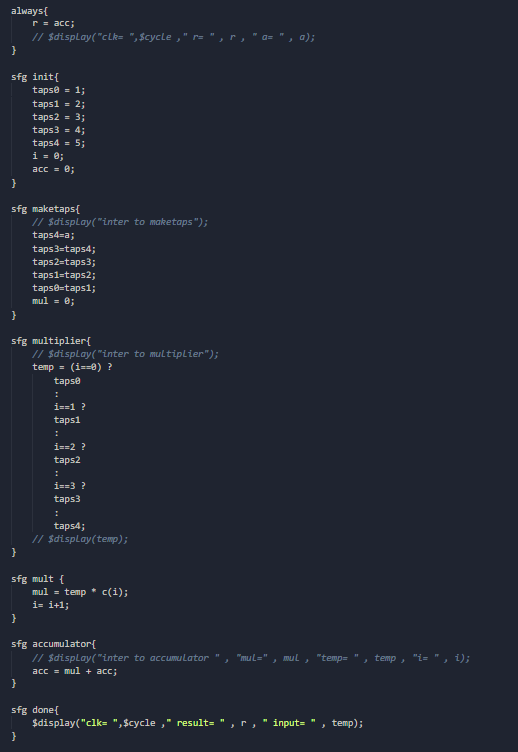
بخش filter\_ctl نیز برای برای کنترل حالات و state های پیاده سازی شده و ترتیب sfg های گفته شده در بالا را با توجه به fsm طراحی شده ، مینویسیم . این fsm از یک initial state و 5 state دیگر تشکیل شده است که نحوه اجرای آن به صورت ترکیبی در بخش بالا ذکر شده است .

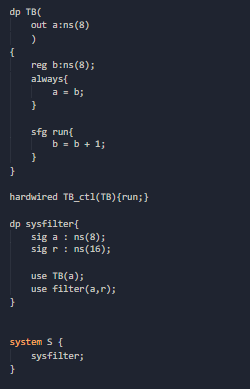
در پایان با ایجاد یک sysfilter و نوشتن یک TB برای اطمینان از صحت عمکرد سیستم کار به اتمام میرسد .



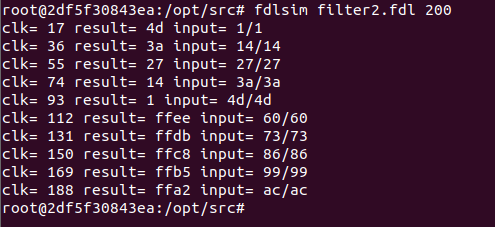
بخش DP :



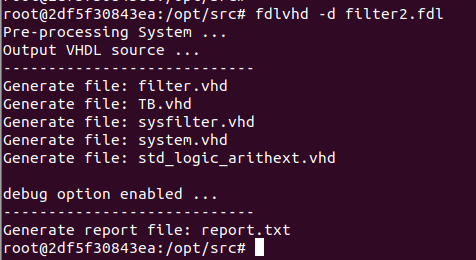




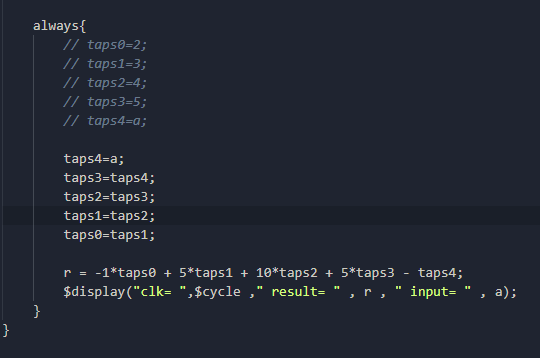
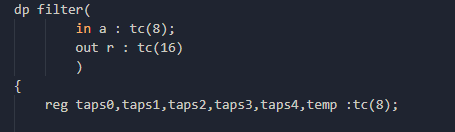
نتبجه اجرا در gezel :

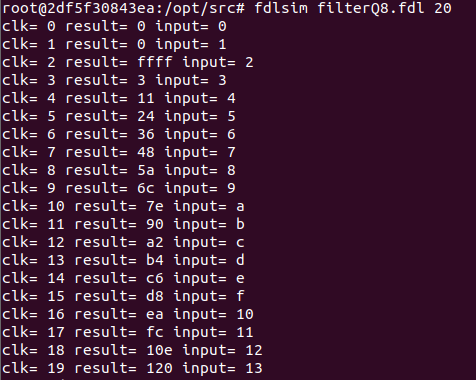
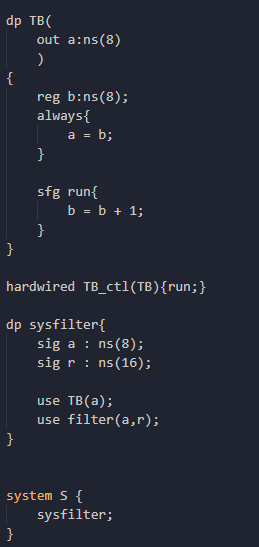


در ادامه با استفاده از دستور fdlvhd کد vhdl حاصل از کد gezel استخراج میشود که در کنار گزارش قرار دارند.



سوال 8 : در این سوال باید در یک کلاک همه ی عملیات ها انجام شود پس به همین دلیل فقط به یک sfg بدون نام always احتیاج داریم





تست بنچ و نتیجه اجرای این کد که برای سوال 8 است در بالا مشخص است

