امیرحسین عسکری - ۹۷۳۱۰۴۳

فاطمه صحرایی - ۹۷۳۱۱۲۹

در این پروژه، از فاز اول تا فاز سوم، با چالش های مختلفی مواجه شدیم. برخی از این چالش‌ها بدلیل عدم آشنایی کافی ما با ابزار و برخی دیگر از سر بی‌دقتی بود!

فاز اول

در فاز اول، با چندان چالشی مواجه نشدیم و میتوان گفت بزرگترین مسئله ما این بود که بفهمیم باید IP یِ CORDIC را چگونه پیکربندی کنیم تا به درستی جذر بگیرد.

فاز دوم

در فاز دوم، چالش‌ها متنوع‌تر شدند چراکه پا فراتر گذاشتیم و همه سوالات امتیازی را پیاده کردیم! و البته این پیاده‌سازی امتیازی‌ها به کمک‌مان هم آمد زیرا نمی‌دانستیم واقعا برای FSM کدام سیگنال‌ها باید تنظیم شوند اما به لطف CRC، نیاز به سیگنال Enable برای تمام component ها پیدا کردیم و توانستیم در FSM این سیگنال‌ها را کنترل کنیم. علت اینکه برای CRC نیاز به کنترل سایر component ها داشتیم این بود که در CRC یک تعداد کلاک مشخصی زده می‌شد و پس از آن خروجی حاضر بود و تا آمدن ورودی جدید، این خروجی حفظ می‌شد. اگر ورودی invalid بود، باعث می‌شد آن تعداد کلاک معین و محدود زده شود و دیگر خروجی با ورودی جدید و valid تغییر نکند. برای همین لازم داشتیم تا بگوییم هر component از چه زمانی به بعد می‌تواند ورودی اش را بخواند و کلاک بزند. اما این همه ماجرا نبود! در CRC حتی پس از اینکه به آن می‌گفتیم از چه زمانی به بعد ورودی‌اش معتبر است، بازهم خروجی درستی نداشتیم و دلیلش این بود که CORDID square، بدون توجه به اینکه ورودی واقعا چیست، تا زمان آماده شدن خروجی، صفر نشان میداد و این برای CRC یک ورودی معتبر تلقی می‌شد!! برای حل این مشکل، به FSM قسمت فعال کردن CRC گفتیم چک کند آیا ورودی صفر است یا نه، اگر صفر نیست فعال شود. برای ورودی‌ای هم که واقعا صفر بود، مقدار اولیه خروجی پورت‌های crc_out و hamming از filter_px را برابر با صفر قرار دادیم. تقریبا در اینجا با کمی کدزنی‌های اضافه، فاز ۲ به پایان رسید و وارد فاز ۳ شدیم.

فاز سوم

در فاز ۳، چالش اول و اصلی‌مان دانلود فایل تصویری بود :) چندباری سرور دانشگاه قطع شد و فایل ناقص ماند. پس از دانلود و تماشای فایل، شروع به اضافه کردن IP کردیم. اولین چالش در فایل IP این بود که ویوادو، همه چیز را به Verilog تولید کرد که چندلحظه بعد راه آن را فهمیدیم و حل شد، اما چالش دوم و مهم‌تر این بود که ویوادو سایر IP های بکار رفته در پروژه مثل CORDIC را نمیشناخت و مجبور بودیم آن‌ها را دستی اضافه کنیم و زمان زیادی می‌برد تا هردو IP را اضافه کند. بعلاوه ما چون می‌خواستیم بخش امتیازی را بزنیم، این زمان ۴ برابر هم می‌شد. پس از آن، چون اکثر ورودی و خروجی‌های مدار اصلی را عددصحیح نوشته بودیم، باید همه اینها را در رجیسترها هندل میکردیم. مشکل دیگر این بود که ابتدا اشباهی فکر کردیم در فیلتر لاپلاس Gx و Gy یکی هستند اما پس از اینکه مدار را پیاده کردیم متوجه این اشتباه شدیم و اصلاح آن طول کشید. در فایل پروژه اصلی پس از آنکه از این IP ها در بلوک دیگرام نمونه گرفتیم، Export hardware را زدیم اما با ارور موجود نبودن فایل generate شده مواجه شدیم که بعد فهمیدیم در ویوادو جدید، حتما باید ابتدا بلوک دیاگرام generate شود و سپس Export بشود. فرایند generate شدن آن بسیار طولانی و طاقت‌فرسا بود و هربار که میخواستیم نکته کوچکی را در IP ها ادیت کنیم، زمان زیادی می‌برد تا بتوانیم کار را ادامه دهیم. یکی از مشکلاتی که در این حین با آن مواجه شدیم این بود که طبق ویدیو پس از ساختن IP آن را می‌بست اما ما چون قصد ادیت کردن آن را داشتیم باید آن را باز نگه می‌داشتیم اما آن را بستیم و هرچقدر گشتیم نتوانستیم پیدایش کنیم :) آخر سر مجبور شدیم یک پروژه جدید باز کنیم و همه چیز را از اول بسازیم. پس از آن، وقتی SDK را باز کردیم، SDK با فایل سخت‌افزاری مچ نشد زیرا در مسیر پروژه اسم یکی از فایل‌ها space داشت و گویا space برای آن قابل درک نبود و مجبور شدیم باز هم پروژه را جابجا کنیم و به مسیر بدون space ببریم! در نهایت پس از همه این موارد، مشغول شبیه‌سازی شدیم که با ارور عجیبی مواجه شدیم که میگفت IP ها قادر به شناختن ECC نیستند! در ابتدا دلیل آن را نفهمیدیم اما پس از مدتی متوجه شدیم پس از هر بار ادیت کردن یک IP باید آن را مجددا بسازیم و آپدیت را در فایل اصلی اعمال کنیم. بعد از همه اینها، شبیه‌سازی را انجام دادیم که ابتدا با ورودی 0 و خروجی U مواجه شدیم و این یعنی در نرم‌افزار توابع ما موفق نشده‌بودند به سخت‌افزار ورودی بدهند. بعد از گشتن فهمیدیم ریستی که در ویدیو ۰ داده می‌شد، در واقع active low بوده و به هردلیلی که در ویدیو کار کرد(احتمالا چون مدار اصلی ریست نداشت) برای ما کار نکرده بود و آن را ۱ کردیم. پس از این قسمت شبیه‌سازی برای تصویر 3x3 با موفقیت انجام شد.

در شبیه‌سازی برای تصویر 8x8 چالش ما تبدیل کردن int و hex به یکدیگر بود که پس از سرچ متوجه شدیم می‌توانیم ورودی تابع را بصورت int هم بدهیم و با چندبار محاسبه فهمیدیم چگونه باید int در جایگاه درستی قرار بگیرد تا در سخت‌افزار بدرستی ترجمه شود. پس از این قسمت تقریبا کار ما به پایان رسید و کد نرم‌افزاری را در حدود ۱۰ دقیقه زدیم (:)))))) و نتایج آن را با شکل موج مقایسه کردیم:

در سخت‌افزار خروجی اول پس از ۹۰ نانوثانیه(با کلاک ۱۰ نانوثانیه) و خروجی کل پس از ۹۰ + ۱۰*۳۵ (بدلیل پایپ لاین)، آماده شد.

در نرم‌افزار برای فیلتر Sobel، ۰.۰۰۰۳۳۷ و برای فیلتر Prewitt، ۰.۰۰۰۳۲۰ ثانیه طول کشید.

چالش نهایی هم این بود که چطور این همه چالش را طوری بنویسیم که ۴ صفحه شود اما بعد دیدیم حداکثر ۴ صفحه بود نه حداقل :))