طراحی فیلتر نویز تصویر به کمک ترکیبی از مدل‌سازی‌های زبان VHDL

پارسا حجابی

چکیده

در این نوشـــتار نحوه طراحی یک مدار که دارای بخش‌های مختلفی از جمله بخش حافظه، بخش پردازش و بخش انتقال داده می‌باشد و البته اتصالات میان آن‌ها را به کمک ترکیبی از مدل‌سازی ساختاری، dataflow و رفتاری نشان می‌دهیم.

کلمات کلیدی

Structural، Behavioral، Dataflow، RAM، Filter، Apply Data

۱- مقدمه

هر تصـــویر از پیکســـل‌هایی تشـــکیل شـــده که هر کدام از پیکســـل‌ها عددی بین ۰ تا ۲۵۶ دارند. برای کاهش نویز تصـــویر می‌توان از فیلترهای متفاوتی استفاده کرد.
نحوه عملکرد یکی از این فیلترها به این صـــورت اســـ ت‌که یک پنجره سه در سه را روی پیکسل‌های تصویر حرکت می‌دهند و با توجه به مقادیر پیکســـل‌های اطراف پیکســـ ل مرکزی و گرفتن میانگینی از آن مقادیر مقدار پیکسل مرکزی را مشخص می‌کنند.
در این تمرین هدف پیاده‌ســـازی مداری اســـت که یک عکس با ابعاد ۳۰ پیکسل در ۲۰ پیکسل را از یک فایل txt بخواند و روی آن این فیلتر را پیاده‌سازی نماید.

۲- مطالب اصلی

۲-۱- مشخص کردن بخش‌های اصلی

این مدار ۳ بخش اصـــلی خواهد داشـــت. یک بخش برای خواندن تصویر از فایل، یک بخش برای ذخیره کردن پیکسل‌های تصـــویر و بخش دیگر برای پردازش و انجام عملیات فیلتر روی تصویر.
همچنین برای اتصالات نیاز به ۳ عدد مالتی پلکسر است که یکی از آن‌ها تک بیتی، یکی ۱۱ بیتی برای آدرس‌ها و دیگری ۸ بیتی برای داده‌ها می‌باشد.

۲-۲- پیاده‌سازی مدل

پس از مشـــخص کردن بخش‌های اصـــلی، کد هر کدام از بخش‌ها باید درون یک فایل مجزا نوشـــته شـــود. بنابراین در کل پروژه ما دارای ۷ بخش خواهد بود که یکی از آن‌ها برای نوشـــتن تست می‌باشد.

۲-۲-۱- مالتی‌پلکسر

هر کدام از مالتی پلکسرها به وسیله مدل‌سازی Dataflow نوشـته شـده‌اند و برای هر کدام از مالتی پلکسـرها به دلیل متغیر بودن اندازه داده‌های ورودی و خروجی یک فایل جدا نوشته شد.

۲-۲-۲- بخش حافظه یا RAM

برای این بخش یک type جدید تعریف شـد که در واقع یک آرایه ۲۰۴۸ تایی از vector های ۸ تایی می‌باشـــد. این بخش توسط مدل‌سازی Behavioral نوشته شد و در آن یک process حساس به کلاک گذاشـته شد که در لبه بالارونده کلاک اگر خط RE آن یک باشـد داده را از آدرسـی که در ورودی آن داده شـده می‌خواند و اگر خط WR آن یک باشـد داده را در آدرس ورودی می‌نویسد.

۲-۲-۳- بخش انتقال داده

کد این بخش به ما داده شـــده بود که با اضـــافه کردن ۲ سیگنال خروجی Address که ۱۱ بیتی است و WR که ۱ بیتی می‌باشد و تغییرات دیگر کد آن را کامل کردیم.

در انتهای کد این بخش با حساب کردن آدرس هر داده با فرمول ستون آن داده ضربدر سطر آن داده منهای یک آدرسی که باید آن داده در بخش حافظه قرار گیرد را محاسبه می‌کنیم و سپس خط WR را ۱ می‌کنیم تا در حافظه بنویسد.

۲-۲-۴- بخش فیلتر نویز

این بخش کلیدی‌ترین و سنگین‌ترین بخش کد می‌باشد. الگوریتم این بخش کد من به این صورت است که من در لبه بالارونده کلاک اگر Ready صفر بود یعنی داده ما حاضر نبود کار اصلی را آغاز می‌کنم.
یک متغیر mode در نظر گرفته شده که در صورتی که صفر باشد آدرس ۹ خانه به همراه x و y خانه‌ای که روی آن هستیم حساب می‌شود و سپس چک می‌شود که کدام یک از این ۹ خانه در ماتریس اصلی تصویر قرار نمی‌گیرند. برای مثال اگر x نقطه‌ای که روی آن هستیم صفر باشد به این معنا است که ما سمت چپ نداریم یا به عبارت دیگر خانه‌های ۰ و ۳ و ۶ از آن مربع سه در سه در ماتریس اصلی تصویر ما قرار نمی‌گیرند بنابراین یک متغیر به اسم leftCons را یک می‌کنیم تا در بخش حساب کردن داده از این ۳ خانه داده‌ای نخوانیم و به جای آن‌ها صفر بگذاریم. پس از حساب کردن آدرس هر ۹ خلنه ما روی آدرس خروجی این بخش آدرس خانه شماره ۰ آن ۹ خانه را قرار می‌دهیم تا در یک کلاک بعد داده آن خانه از حافظه خوانده شود و روی Data_in این بخش قرار گیرد. وقتی تمام این کارها انجام شد mode را یک می‌کنیم تا در ۸ کلاک بعدی دیگر وارد این بخش نشویم.
در صورتی که mode یک باشد وارد فاز پردازش می‌شویم. در این فاز آدرس هر ۹ خلنه را داریم بنابراین در هر کلاک باید آدرس را روی خروجی آدرس این بخش قرار دهیم تا در کلاک بعدی مقدار آن آدرس از حافظه خوانده شود و روی Data_in این بخش قرار گیرد. در صورتی که آن آدرس در تصویر اصلی ما قرار نگرفته بود ما داده آن ناحیه را صفر می‌کنیم تا در عملیات خنثی باشد. سپس وقتی هر ۹ خانه کارش انجام شد با گذاشتن یک لوپ خلنه‌های متناظر داده را در اعداد داده شده ضرب می‌کنیم و نتیجه را در سیگنال newPix ذخیره می‌کنیم. حال وقت آن است که مقدار newPix روی Data_out این بخش قرار گیرد و Write_Enable را برای حافظه یک کند تا این حافظه در آدرس تعیین شده ذخیره گردد.

۲-۲-۵- بخش top_module

در این بخش باید به کمک مدل‌سازی Structural تمامی بخش‌های گذشته را به همدیگر متصل کنیم. در واقع از تمامی componentهای مورد بحث instance ایجاد کنیم و اتصالات میان آن‌ها را به کمک port map وصل کنیم.

۲-۲-۶- بخش تست

در این بخش نیز یک instance از بخش top_module می‌سازیم و با صفر کردن سیگنال ریست پس از ۱۰ نانوثانیه و ایجاد یک کلاک با پریود ۱۰ نانوثانیه‌ای کل مدار خود را تست کنیم.

۳- نتیجه

می‌توان به وضوح انعطاف پذیری در vhdl را مشاهده کرد. این زبان دست طراح را باز می‌گذارد تا مدار پیچیده و سخت خود را به بخش‌های بسیار کوچک تبدیل کند و آن‌ها را به هر شکلی که خودش می‌خواهد پیاده کند و سپس با سیم بندی‌های ساده از همان بخش‌های بسیار کوچک یک مدار پیچیده را ایجاد کند.

مراجع

[1]