به نام خدا

طراحی کامپیوتری سیستم‌های دیجیتال - پاییز ۱۴۰۳

پروژه پنجم : طراحی و پیاده‌سازی واحد پردازشی

Eyeriss[1] شتاب‌دهنده (**Processing Element**)

طراحان: مهدی محمدی نسب - مهدی ابراهیم سلطانی

هدف پروژه :

در این تمرین قصد داریم طراحی ساختار واحدهای پردازش‌کننده شتاب‌دهنده Eyeriss را، که قسمتی از آن در تمرین ۴ پیاده‌سازی شد، کامل کنیم. بنابراین در پیاده‌سازی این تمرین، ماژول پیاده‌سازی شده در تمرین قبلی مورد نیاز است.

مقدمه:

پیش‌تر با شبکه‌های عصبی پیچشی، و شتاب‌دهنده Eyeriss آشنا شده‌اید. در ابتدا، معماری این شتاب‌دهنده و روند محاسبات به تفصیل توضیح داده می‌شود. در ادامه، جزئیات مربوط به هر بخش واحد پردازشی که نیازمند پیاده‌سازی است معرفی شده و نکات آن ذکر می‌گردد.

معماری Eyeriss:

ساختار کلی Eyeriss، به صورت یک آرایه ۱۲*۱۴ از واحدهای پردازشی است، که از طریق ساختاری به نام شبکه روی تراشه[2] به بافر سراسری[3] یا واحدهای همسایه متصل می‌شوند.

هر کدام از واحدهای پردازش‌کننده، توانایی انجام کانولوشن یک بعدی را دارد. به عبارتی دیگر، این واحد با دریافت یک ماتریس تک سطری فیلتر، و یک ماتریس تک سطری ورودی، نتیجه کانولوشن را در چند مرحله ارائه می‌دهد. کانولوشن دو بعدی و سه بعدی را می‌توان مجموعه‌ای از کانولوشن‌های یک بعدی در نظر گرفت. به همین دلیل، با داشتن یک واحد پردازشی، توانایی انجام این کانولوشن ها نیز میسر می‌باشد. اما برای افزایش سرعت محاسبات، و کاهش بار پردازشی، از مجموعه‌ای از این واحدها برای انجام عملیات کلی در شتاب‌دهنده Eyeriss استفاده شده است.

[1] Accelerator
[2] Network-On-Chip
[3] Global Buffer

ساختار واحدهای پردازشی (PE)

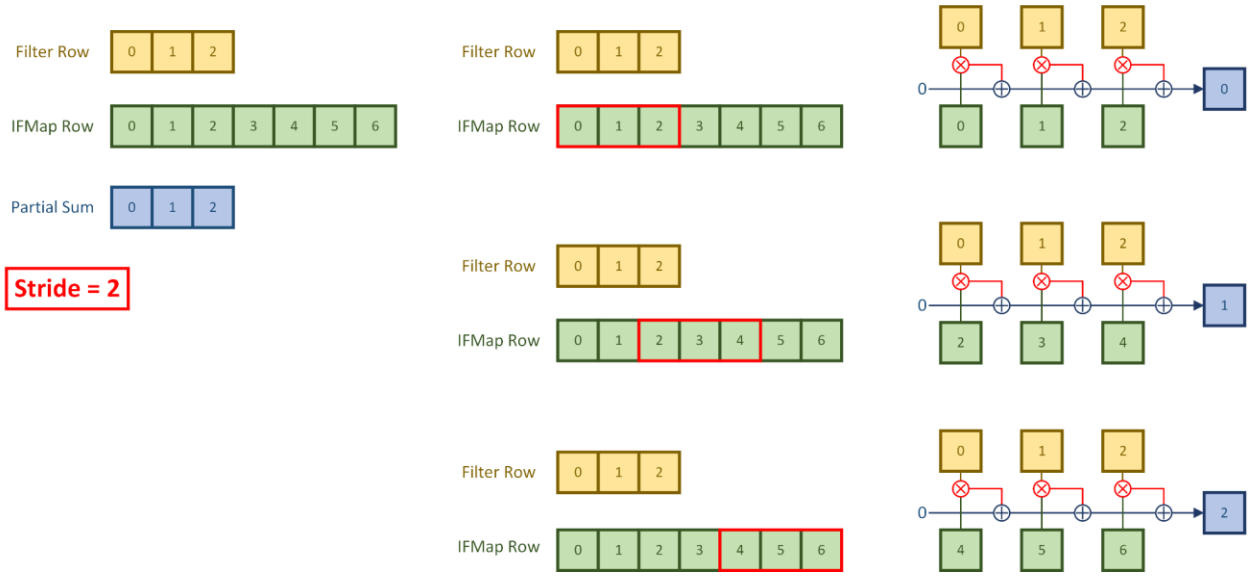
به طور کلی در ساختار PE، بافرهای ورودی و خروجی وظیفه handshaking و انتقال داده به و یا دریافت آن از خارج PE را دارند و قسمت داخلی، داده ها را از داخل این بافرها دریافت می‌کند. بافرهای IFMap و Filter، به ترتیب وظیفه دریافت داده‌های ماتریس ورودی و فیلتر را از خارج PE بر عهده دارند. در ادامه این داده‌ها به حافظه‌های ScratchPad متناظر خود انتقال یافته و در آن ذخیره می‌شوند.

پس از ذخیره شدن داده‌ها، هر کدام از درایه‌های متناظر که می‌بایست در هم ضرب شوند، از روی حافظه‌ها خوانده شده، وارد pipeline می‌شوند تا ضرب و جمع آنها انجام شود. در نهایت، پس از انجام تعداد مشخصی ضرب و جمع و محاسبه یکی از درایه‌های خروجی، این داده به Psum ScratchPad (آرایه‌ای از رجیسترها) منتقل می‌شود. سپس خروجی‌های مربوط به انجام عملیات یک سطر کامل با فیلترهای متناظر آن‌ها از طریق بافر خروجی به بیرون منتقل می‌شوند. در نهایت این خروجی‌ها در حافظه سراسری نوشته شده یا برای انجام ادامه محاسبات به واحد دیگری منتقل شود[4]. در ادامه، نحوه محاسبات کانولوشن یک بعدی شرح داده شده‌است.

مراحل انجام کانولوشن یک بعدی:

شکل زیر ترتیب و نحوه انجام محاسبات مربوط به یک کانولوشن تک بعدی را نمایش می‌دهد. برای این کار باید پنجره‌ای به اندازه طول فیلتر روی ردیف ورودی قرار دهیم. سپس درایه‌های مشخص شده توسط پنجره را در درایه متناظر آن در فیلتر ضرب کرده و نتایج را با یکدیگر جمع می‌کنیم. در نهایت به این ترتیب درایه اول مربوط به Partial sum محاسبه می‌شود. در اینجا مفهومی تعریف می‌شود تحت عنوان Stride که مشخص می‌کند در هر مرحله پنجره قرار داده شده بر روی ورودی چند خانه به جلو حرکت کند. در مثال زیر میزان Stride برابر ۲ است که موجب شده پنجره به اندازه دو خانه به جلو بلغزد. به همین ترتیب درایه‌های دوم و سوم خروجی نیز محاسبه می‌شوند.

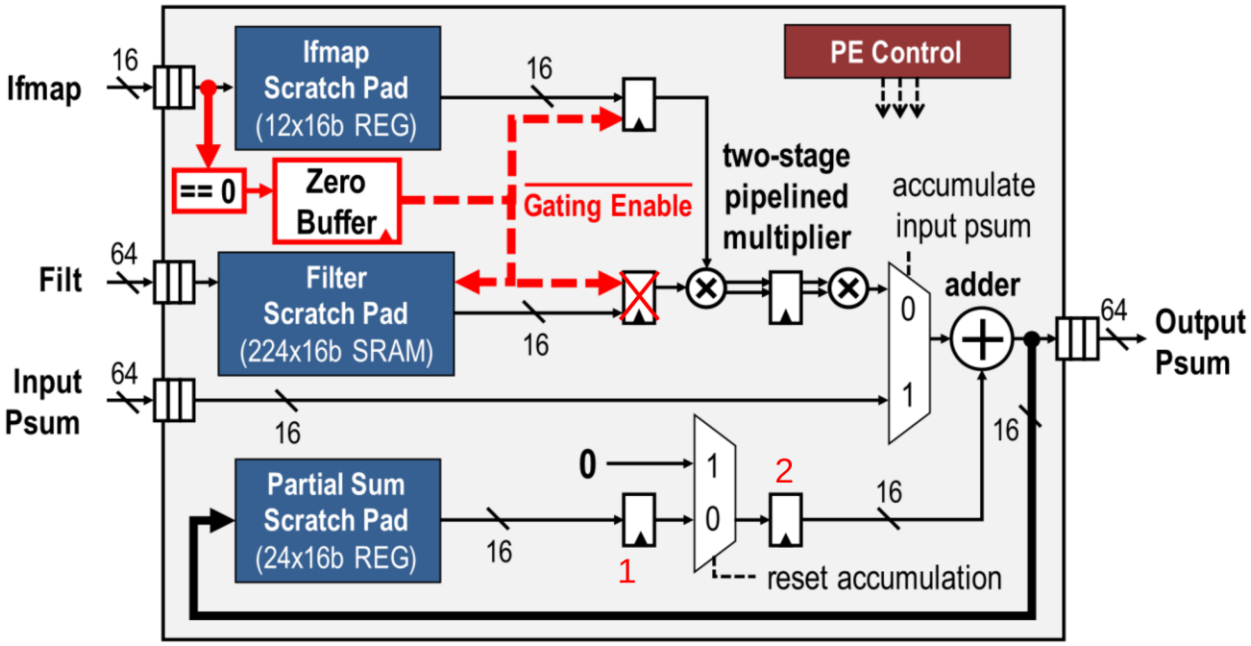
[4] علت آنکه خروجی واحد، Psum یا Partial-Sum نامیده شده بدین جهت است که در صورتی که محاسبات برای کانولوشن دو یا سه بعدی باشد، خروجی بدست آمده تنها بخشی از محاسبات ضرب و جمع برای درایه متناظر است، و می بایست سطرهای دیگر از ماتریس های ورودی و فیلتر نیز در یکدیگر ضرب شده و نتیجه آن با نتیجه حاصل از PE جمع گردد. لذا به گونه‌ای محاسبات انجام شده برای یک درایه خروجی در PE، جزئی از محاسبات کامل است.



شکل ۱

پیاده‌سازی:

شما در تمرین قبل بخشی از PE را پیاده‌سازی کردید. در این تمرین آن را تکمیل خواهید کرد. شماتیک کلی PE به صورت کامل در شکل زیر آمده است. شما بخش Psum Scratch Pad و کنترلر مربوط به آن را به طراحی قبلی خود اضافه خواهید کرد. توضیحات بیشتر مربوط به شکل در ادامه آورده شده است.



شکل ۲

۱. حافظه‌های **ScratchPad**

برای این قسمت به تمرین قبل مراجعه کنید.

۲. کنترل‌کننده خواندن از بافرها (**Buffer Read Controller**)

برای این قسمت به تمرین قبل مراجعه کنید.

۳. کنترل‌کننده‌ی نوشتن در بافر خروجی:

برای اینکه پس از اتمام محاسبات، Psum محاسبه شده به بافر خروجی منتقل شود، نیازمند واحدی هستیم تا با بررسی شرایط، داده خروجی را به بافر منتقل کند. این واحد، ورودی done به معنای اتمام محاسبه یک Psum را از واحد کنترلی اصلی دریافت می‌کند و بررسی می‌کند که آیا امکان نوشتن در بافر وجود دارد یا خیر. چنانچه قابلیت نوشتن در بافر خروجی وجود داشته باشد، محتوای رجیستر Psum با فعال‌سازی دستور نوشتن بافر خروجی، در آن نوشته می‌شود. در صورتی که امکان نوشتن وجود نداشته باشد، این واحد سیگنال خروجی stall را به واحد کنترل اصلی می‌دهد، تا از تغییرات و یا نابودی مقدار Psum جلوگیری کند.

- این واحد کنترلی را که در تمرین قبلی پیاده سازی کردید، در صورت نیاز برای این تمرین اصلاح کنید.

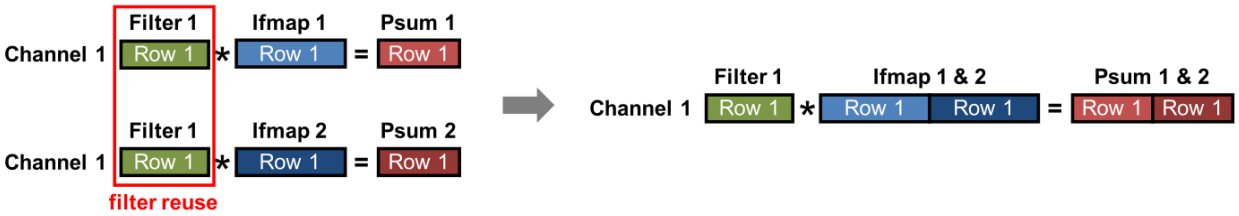
٤. تولیدکننده آدرسِ خواندن از **ScratchPad** ها :

برای این قسمت نیز به تمرین قبل مراجعه کنید.

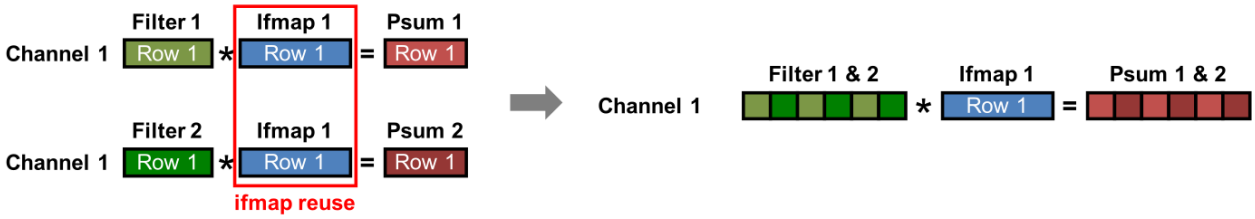
٥. واحد کنترل اصلی:

این بخش، علاوه بر عملیات تمرین قبل، باید برای کنترل بخش Psum نیز بسط داده شود. به ماژول PE یک سیگنال ورودی mode اضافه می‌شود که سه حالت را کنترل می‌کند:

- حالت فیلتر مشترک: در این حالت یک سطر از یک کانال از یک فیلتر در حافظه مربوط قرار می‌گیرد. در حافظه ورودی‌ها نیز دو ورودی متفاوت (ولی با شماره سطر و کانال یکسان) از دو ورودی ifmap متفاوت قرار می‌گیرند. بنابراین دو خروجی Psum متفاوت از آن‌ها تولید خواهند شد. در شکل زیر می‌توان این فرآیند را مشاهده کرد:



- حالت فیلترهای متفاوت برای یک ورودی: در این حالت سطر کانال i-ام (با شماره سطر یکسان) از فیلترهای متفاوت (n فیلتر) در حافظه فیلترها قرار خواهند گرفت. در حافظه مربوط به ورودی‌ها نیز یک سطر از یک ifmap قرار خواهد گرفت. در این حالت، از آنجا که عناصر فیلترها به صورت یک درمیان در کنار هم قرار می‌گیرند، خروجی هر ضرب و جمع به صورت یک درمیان در Psum رجیسترهای متفاوت قرار خواهند گرفت. نحوه انجام این فرایند در شکل زیر قابل مشاهده است. توجه کنید که نحوه یک در میان شدن فیلترها از بیرون (تست بنچ) به PE داده می‌شود؛ بنابراین درون واحد پردازشی با آن به عنوان یک فیلتر عادی برخورد می‌شود و تنها تعیین کننده نحوه ذخیره، سیگنال mode است.



- حالت کانال‌های متفاوت از یک فیلتر و سطر: در این حالت در حافظه فیلترها سطر i-ام از یک فیلتر با کانال‌های (n کانال) متفاوت قرار خواهد گرفت. در حافظه ورودی‌ها نیز سطر i-ام از یک ورودی با کانال‌های متفاوت قرار خواهند گرفت. عناصر هر دو فیلتر و ورودی به صورت یک در میان قرار گرفته و ذخیره می‌شوند (همانند حالت قبل از تست بنچ کنترل خواهد شد). در نهایت، عملیات یک کانولوشن عادی انجام خواهد شد و نتیجه ضرب و جمع حاصل در یک رجیستر از Psum قرار خواهد گرفت. شکل زیر این فرایند را نشان می‌دهد.

کنترلر باید با توجه به mode، در یکی از این سه حالت قرار گرفته و عملیات را انجام دهد.

۶. تکمیل مسیرداده :

حال با قرار دادن رجیسترها، ضرب کننده و جمع‌کننده مسیرداده را تکمیل کنید.

- برای شبیه‌سازی فعالیت ضرب‌کننده دو مرحله‌ای، از یک ضرب کننده ساده و یک رجیستر استفاده کنید.
- در اتصال‌ها، به تطبیق عرض بیت‌ها دقت کنید.
- توجه شود که عرض بیت تمامی ماژول‌های نوشته‌شده در این بخش مانند ضرب‌کننده یا رجیستر، باید پارامتری باشند.

در شکل PE (شکل ۲)، مسیر داده به صورت کلی نشان داده شده است. آن را با جزئیات بیشتر با در نظر گرفتن نکات زیر تکمیل کنید:

- در صورت نیاز، می‌توانید یکی از رجیسترهای ۱ یا ۲ مشخص شده در شکل ۲ را حذف کنید.
- در صورت نیاز رجیستری را که روی آن با ضربدر خط کشیده شده است، حذف کنید.

- واحد پردازشی در دو حالت قرار می‌گیرد. حالت اول محاسبه کانولوشن به طریق ذکر شده در قسمت قبل است. حالت دوم جمع ورودی‌های psum با مقادیر داخل Psum ScratchPad است. در این حالت ماژول‌های دیگر عملیاتی انجام نمی‌دهند. پس، یک سیگنال ورودی یک بیتی را در نظر بگیرید؛ به طوری که PE را در یکی از این دو حالت قرار می‌دهد. در حالت Psum، ورودی‌های Psum به ترتیب با رجیسترهای Psum ScratchPad جمع می‌شوند. یعنی شمارنده آدرسی برای این حافظه وجود دارد؛ به طوری که به ازای خواندن هر ورودی از input Psum، مقدار رجیستر i-ام از حافظه را خوانده و این مقادیر با یکدیگر جمع شده و بعد از هر خواندن شمارنده آدرس به روز می‌شود. همچنین خروجی مستقیم در بافر خروجی نوشته می‌شود. توجه داشته باشید که با تغییر حالت آدرس ها را reset کنید تا جمع مقادیر با هم به صورت درست انجام شود.
- مالتی پلکسر قرار گرفته پس از Psum ScratchPad به ازای هر عملیات کانولوشن جدید و برای هر Psum رجیستر، برای بار اول مقدار صفر را (ورودی دوم) به خروجی می‌دهد.
- قسمت *zero buffer* که با رنگ قرمز در شکل مشخص شده، در صورت پیاده‌سازی نمره امتیازی خواهد داشت.

سایر نکات

- انجام این تمرین به صورت گروه های دونفره در دو فاز خواهد بود:

۱. در فاز اول `controller` و `datapath` را طراحی کرده و در موعد تعیین شده برای فاز اول داخل سایت بارگذاری کنید.

۲. در فاز دوم `controller` و `datapath` طراحی شده در فاز اول را با زبان verilog مدل سازی و در Modelsim شبیه‌سازی کرده و در موعد معین برای فاز دوم در داخل سایت بارگذاری کنید.

- برای فاز دوم تمرین لازم است فایل های `Testbench` و `HDL` خود را مطابق توضیح داده شده در `trunk` در `subdirectory` های `trunk/doc` بارگذاری کنید. همچنین اطمینان حاصل کنید که با اجرای `trunk/sim/sim_top.tcl` تست پنج شما اجرا می شود. برای اجرای این اسکریپت می توانید از دستور زیر در Modelsim استفاده کنید:

<do <sim_file <<

- فایل ها و گزارش خود را تا قبل از موعد تحویل هر فاز، با نام CAD_HW5_P1_<SID>.zip و CAD_HW5_P2_<SID>.zip به ترتیب در محل های مربوطه در صفحه درس آپلود کنید.

- برای آزمودن کد خودتان در این تمرین تست بنچ‌های مربوطه را خود شما طراحی و پیاده سازی می کنید، اما با توجه به اینکه تمارین بعدی درس مبتنی بر ادامه دادن این تمرین هستند حتما در طراحی و پیاده‌سازی خود به قابلیت مقاومت در برابر تغییر و پارامتری بودن ورودی‌ها و خروجی‌ها توجه لازم را داشته باشید.

- نام گذاری صحیح متغیرها، تمیزی کد و توضیحات و پارامتری بودن ورودی‌های ماژول‌ها می تواند تا حدودی کاستی‌های کد را در بخش‌های دیگر جبران کند.

- هدف این تمرین یادگیری شماست! در صورت کشف تقلب، مطابق با قوانین درس برخورد خواهد شد.

موفق باشید