ما از چارچوب براساس بهینه سازی پلی هدرال برای تشخیص تمام تبدیل های چرخه ی درست استفاده می کنیم. جدول 3 ارتباطات به اشتراک گذاری داده را بین پیمایش های چرخه و آرایه ها نشان می دهد. ارتقای حافظه ی محلی در هر حلقه ی درست برنامه ریزی شده، هر هنگام که برای کاهش حجم کل ارتباطات کاربردی باشد مورد استفاده قرار می گیرد.

نسبت CTC. نسبت محاسبه به ارتباط (CTC) برای توضیح عملیات های محاسباتی بر دسترسی به حافظه مورد استفاده قرار می گیرد. بهینه سازی بازاستفاده ی داده، تعداد کل دسترسی به حافظه را کاهش می دهد. نسبت محاسبه بر ارتباط کد نشان داده شده در شکل 9 می تواند با معادله ی (4) محاسبه شود به گونه ای که ain، aout، awght و Bin،Bout،Bwght نشان دهنده ی تعداد حرکات و اندازه های بافر حافظه مورد دسترسی قرار گرفته به ترتیب برای نقشه و اوزان امکانات ورودی/خروجی است.



به گونه ای که



بدون باز استفاده ی داده ی *output\_fm*



با باز استفاده ی داده ی *output\_fm*



با داشتن یک چرخه ی برنامه ی خاص و یک مجموعه از اندازه ی تاپل کاشی (Tm، Tn، Tr، Tc)، نسبت محاسبه به ارتباط می تواند با فرمول بالا محاسبه شود.

3.4 جستجوی فضای طراحی

همان گونه که در بخش 3.2 و بخش 3.3 اشاره شد، با داشتن یک چرخه ی خاص برنامه ریزی و تاپل اندازه ی کاشی (Tm، Tn، Tr، Tc)، سقف محاسباتی و نسبت محاسبه به ارتباطات متغیرهای طراحی می تواند محاسبه شود. با شمارش تمام نظم های چرخه ی ممکن و اندازه های کاشی یک سری از کارایی های محاسباتی و جفت های نسبت محاسبه به ارتباط ساخته می شود. شکل 8(a) نشان دهنده ی تمام راه حل های درست برای لایه ی 5 از مثال کاربرد CNN در سیستم هماهنگی مدل قانون بندی است. محور x نشان دهنده ی نسبت محاسبه به ارتباطات یا نسبت عملیات ممیز شناور بر دسترسی بایت DRAM است. محور y نشان دهنده ی کارایی محاسباتی (GFLOPS) است. شیب خط بین هر نقطه و نقطه ی مبدا (0،0) نشان دهنده ی نیازمندی حداقلی پهنای باند مورد نیاز برای این پیاده سازی است. برای نمونه، نیازمندی پهنای باند حداقلی متعلق به طراحی P برابر با شیب خط P’ است.

در شکل 8 (b)، خط سقف پهنای باند و سقف محاسباتی با مشخصات پلتفرم توضیح داده می شود. هر نقطه در طرف چپ از خط سقف پهنای باند نیازمند یک پهنای باند بالاتر از آنچه که پلتفرم می تواند ارایه دهد است. برای نمونه، هر چند پیاده سازی A به بالاترین کارایی محاسباتی دست پیدا کرده است، پهنای باند حافظه ی مورد نیاز نمی تواند توسط پلتفرم هدف تامین شود. کارایی واقعی قابل دستیابی بر روی پلتفرم مقدار بعد قائم A’ است. بنابراین طراحیهای پشتیبانی شده توسط پلتفرم به عنوان یک مجموعه شامل آنهایی که در طرف راست خط سقف پهنای باند هستند توضیح داده می شود، همان هایی که نمایش طراحی های سمت چپ هستند.

ما این فضای طراحی های پشتیبانی شده توسط پلتفرم را جستجو می کنیم و یک مجموعه از پیاده سازی های با بالاترین کارایی می تواند به دست آید. اگر این مجموعه تنها شامل یک طراحی باشد، این طراحی نتیجه ی نهایی ما از جستجوی فضای طراحی است. هرچند یک وضعیت عمومی تر این است که ما بتوانیم چندین همتا را درون این مجموعه بیابیم، مثلا نقطه ی C، D و تعدادی دیگر در شکل 8 (b). ما نقطه ای را با بالاترین ارزش CI بر می گزینیم زیرا این طراحی نیازمند کمتریم پهنای باند است.

این معیار انتخاب از این حقیقت نشات می گیرد که ما می توانیم از پورت های ورودی/خروجی کمتر، LUT ها و ارتباطات سیم بندی شده ی کمتر و غیره برای موتور انتقال داده با نیازمندی های پهنای باند کمتر استفاده کنیم. بنابراین نقطه ی C طراحی انتخاب شده ی نهایی ما در این مورد برای لایه ی 5 است. نیازمندی پهنای باند آن 2.2 GB/s است.

3.5 طراحی شتاب دهنده ی CNN چندلایه

در بخش قبلی، ما در مورد چگونگی پیدا کردن پارامترهای پیاده سازی بهینه برای هر لایه ی کانولوشنی بحث می کنیم. در یک کاربرد CNN، این پارامترها ممکن است بین لایه های مختلف متفاوت باشند. جدول 4 نشان دهنده ی فاکتورهای بهینه ی باز شدن است (Tm و Tn) برای تمام لایه های مثال کاربرد CNN است (شکل 2 را ببینید).

طراحی یک شتاب دهنده ی برای پشتیبانی با چند لایه ی کانولوشنی با عوامل بازشدن متفاوت ممکن است چالش برانگیز باشد. ساختارهای سخت افزاری پیچیده برای بازتنظیم موتور محاسباتی و ارتباطات درونی مورد نیاز است.

یک راه جایگزین، طراحی یک معماری سخت افزار با عوامل یکسان در تمام لایه های کانولوشنی است. ما تمام راه حل های درست برای انتخاب پارامترهای طراحی جهانی بهینه را شمارش کردیم. شتاب دهنده ی CNN با ضرایب یکسان شده ی بازشدن برای طراحی و پیاده سازی آسا است، اما ممکن است برای تعدادی از لایه ها غیر بهینه باشد. جدول 4 نشان دهنده ی این است که با ضرایب بازشدن سیکسان شده ((64،7))، تنزل در مقایسه با سیکلهای اجرای کلی هر لایه ی کانولوشنی بهینه سازی شده 5% است. با این تحلیل، شتاب دهنده ی CNN با ضرایب یکسان شده ی بازشدن در میان لایه های کانلوشنی در آزمایش ما انتخاب شده اند. حد بالای اندازه ی فضای شمارش، 98 هزار طرح درست است که می تواند در 10 دقیقه در یک لپتاپ معمولی به پایان برسد.

4. جزییات پیاده سازی

این بخش توضیح دهنده ی پیاده سازی با جزییات از راه حل ماست.

4.1 بررسی سیستم

شکل 10 نشان دهنده ی یک بررسی از پیاده سازی ما است. کل سیستم درون یک چیپ FPGA قرار می گیرد و از یک DRAM از نوع DDR3 برای ذخیره ی خارجی استفاده می کند. MicroBlaze یک هسته ی پردازشی نرم RISC توسعه داده شده برای Xilinx FPGAs ، برای کمک به شروع شتاب دهنده ی CNN، ارتباطات با پردازنده ی میزبان و اندازه گیری زمان مورد استفاده قرار گرفته است. باس AXI4lite برای انتقال دستور و باس AXI4 برای انتقال داده است. شتاب دهنده ی CNN به عنوان یک IP بر روی باس AXI4 کار می کند. دستورات را دریافت می کند و پارامترهای تنظیماتی را از طریق باس AXI4lite از MicroBlaze می گیرد و موتور شخصی سازی شده ی انتقال داده از طریق اینترفیس FIFO ارتباط برقرار می کند. این موتور انتقال داده می تواند از طریق باس AXI4 به حافظه ی خارجی دسترسی داشته باشد. مکانیزم وقفه بین MicroBlaze و شتاب دهنده ی CNN برای ممکن کردن اندازه گیری دقیق زمان، فعال می شود.

4.2 موتور محاسباتی

بخش موتور محاسباتی در شکل 11 نشان دهنده ی یک بلوک دیاگرام از پیاده سازی ماست. آنها براساس تحلیل ما در بخش 3 طراحی شده اند. حلقه های بازشدن دو سطحی (Tm، Tnدر شکل 2) به عنوان یک موتور محاسباتی در حال اجرای همزمان پیاده سازی می شوند. یک چند ساختار درخت شکل مانند آن که در شکل 7 وجود دارد مورد استفاده قرار می گیرد. برای بهترین طراحی میان لایه ای در مورد ((64،7)=( Tm، Tn))، موتور محاسباتی به عنوان یک چند ساختار درخت شکل با 7 ورودی از نقشه های امکانات ورودی و 7 ورودی از اوزان و یک ورودی از بایاس، که در بافرهای نقشه های امکانات خروجی ذخیره شده است، پیاده سازی می شود. 64 چند ساختار برای چرخه ی بازشدن Tm کپی می شوند. یک بررسی در بخش موتور محاسباتی شکل 11 یافت می شود.

4.3 زیرسیستم های حافظه

بافرهای روی چیپ بر اساس یک ایده ی اساسی از بافر دوگانه ساخته می شوند، که در آن بافرهای دوگانه به یک صورت پینگ پونگی به کار می آیند تا زمان انتقال داده را با محاسبات همپوشانی دهند. بنابراین آنها به 4 مجموعه سازماندهی می شوند: دو برای نقشه های امکانات ورودی و اوزان و دو برای نقشه های امکانات خروجی. ما در ابتدا سازمان دهی هر مجموعه بافر را معرفی می کنیم و در ادامه مکانیزم انتقال داده ی پینگ پونگی را توضیح می دهیم. هر مجموعه بافر شامل چندین بانک بافر مستقل است. تعداد بانک های بافر در هر مجموعه ی ورودی بافر برابر با Tn (اندازه ی کاشی input\_fm) است. تعداد بانک های بافر در هر مجموعه بافر خروجی برابر با Tm (اندازه ی کاشی output\_fm) است.

مجموعه بافر دوگانه برای درک اجرای پینگ پونگی استفاده می شوند. برای ساده سازی بحث، ما از یک مورد ثابت در شکل 9 برای نمایش مکانیزم عمل پینگ پونگی استفاده می کنیم. کد را در شکل 9 ببینید. عمل "بیرون بار" تنها بعد از  بار عملیات بارگزاری روی می دهد. اما مقدار داده در هر انتقال output\_fm بزرگتر از input\_fm به یک نسبت  است. برای افزایش استفاده از پهنای باند، ما دو کانال مستقل را پیاده سازی می کنیم، یکی برای عمل بارگزاری و دیگری برای عمل برون بری بار.

شکل 12 نشان دهنده ی زمان بندی چندین محاسبه و فاز انتقال داده است. برای فاز اول، موتور محاسبه مجموعه بافر وروردی 0 را محاسبه می کند در حالی که داده ی فاز بعدی را به مجموعه بافر ورودی 1 کپی می کند. فاز بعدی عمل برعکس را انجام می دهد. این عملیان پینگ پونگی نقشه و اوزان امکانات ورودی است. هنگام که تعداد  از فازهای محاسبه و کپی داده انجام شد، نقشه امکانات خروجی منتج شده به درون DRAM نوشته می شود. عملیات برون بری بار، نتایج درون مجموعه بافر خروجی 0 را در مدت  فاز خارج می کند تا اینکه داده های موقت بازاستفاده شده در مجموعه بافر خروجی 1، نتایج جدید را بسازند. این عملیات پینگ پونگی نقشه های امکانات خروجی است. توجه کنید که این دو کانال مستقل برای بارگزاری و ذخیره ی مکانیزم عملیات، برای هر موقعیت بازاستفاده ی داده ی دیگری در این چارچوب کار می کنند.

4.4 موتور انتقال داده ی خروجی

هدف از استفاده از موتور انتقال داده ی خروجی در دو مورد است: 1) می تواند انتقال داده ی بین شتاب دهنده ی و حافظه ی خارجی را ممکن کند، 2) می تواند شتاب دهنده ی ما را از چندین امکانات مربوط به پهنای باند با پلتفرم و ابزار خاص متماز کند. شکل 13 نشان دهنده ی یک آزمایش با باس پهنای باند AXI4 در Vivado 2013.4 است. در این دو شکل ما دو پارامتر را تنظیم می کنیم، پهنای بیت باس AXI را به پهنای باند خارجی کنترل DRAM و کنترل DRAM در بالاترین تنظیماتشان قرار می دهیم در حالی که تعداد اینترفیس های IP-AXI و پهنای بیت هر IP را تغییر می دهیم. در شکل 13(a)، افزایش در پهنای بیت IP-AXI هیچ تاثیری را بر روی پهنای باند (400 MB/s زیر فرکانس 100 MHz) ندارد. در شکل 13(b)، با IPهای افزوده ی بیشتری به باس AXI، پهنای باند آن تقریبا به صورت خطی افزایش می یابد و بالاترین پهنای باند در حدود 4.5 GB/s است. در طراحی شتاب دهنده ی CNN ما، یک پهنای باند حداقلی 1.55 GB/s مورد نیاز است. بنابراین 4 اینترفیس IP برای این طراحی براساس شکل 13 کافی است. ما دو اینترفیس AXI-IP را در موتور انتقال داده ی 0 و دو تا را در موتور انتقال داده ی 1 به گونه ای که در شکل 10 نشان داده شده است مورد استفاده قرار می دهیم.

5. ارزیابی

در این بخش ما ابتدا مشخصات محیطی آزمایشمان را توضیح می دهیم سپس نتایج آزمایشی جامع ارایه شده است.

5.1 تنظیمات محیطی

طراحی شتابدهنده با Vivado HLS (v2013.4) پیاده سازی شده است. این ابزار امکان پیاده سازی شتابدهنده را با زبان C و استخراج RTL را به عنوان یک هسته ی IP ویوادو ممکن می کند. کد C متعلق به طراحی CNN ما با اضافه کردن پراگمای تعریف شده با HLS موازی سازی می شود و نسخه ی موازی سازی شده با ابزار تحلیل زمان بندی ارزیابی می شود. شبیه سازی سریع پیش از سنتز با شبیه سازی C این ابزار و هم شبیه ساز C/RTL انجام می شود. گزارش های منابع پیش سنتز برای جستجوی فضای طراحی و تخمین کارایی مورد استفاده قرار می گیرد. RTL استخراج شده سنتز می شود و در ویوادو (v2013.4) پیاده سازی می شود.

پیاده سازی ما براساس برد VC707 است که دارای یک چیپ Xilinx FPGA از نوع Virtex7 485t است. فرکانس عملیاتی آن 100 MHz است. پیاده سازی نرم افزار بر روی Intel Xeon CPU E5-2430 (@2.20GHz) با 15MB کش اجرا می شود.

5.2 نتایج تجربی

در این زیربخش ما در ابتدا استفاده از منابع را گزارش می دهیم. سپس ما پیاده سازی نرم افزاری (بر روی پردازنده) را با شتابدهنده مان بر روی FPGA مقایسه می کنیم. در نهایت مقایسه ی بین پیاده سازی ما و روش FPGA موجود ارایه می شود.

قرارگیری و مسیریابی توسط مجموعه ابزار ویوادو انجام شد. بعد از آن استفاده از منابع پیاده سازی مان گزارش می شود به گونه ای که در جدول 6 نشان داده شده است. ما می توانیم بگوییم که شتابدهنده ی CNN ما تقریبا به صورت کامل از منابع سخت افزاری FPGA استفاده کرده است.

مقایسه ی کارایی بین شتابدهنده ی ما و همتای نرم افزاری آن در جدول 7 نشان داده شده است. ما شتابدهنده ی میان لایه ای پیشنهاد داده شده را برای مقایسه انتخاب می کنیم. پیاده سازی نرم افزاری در 1 رشته و 16 رشته با استفاده از gcc با گزینه ی بهینه سازی -O3 به ترتیب انجام می شود. در کل پیاده سازی براساس FPGA ما به 17.42 برابر افزایش سرعت بیشتر از پیاده سازی با 1 رشته دست یافت. همچنین به 4.8 افزایش سرعت بیشتر از پیاده سازی نرم افزاری با 16 رشته دست یافت. کارایی کلی شتاب دهنده ی ما به 61.62 GFLOPS رسید.

شکل 14 نشان دهنده ی یک تصویر از پیاده سازی بر روی برد ما است. یک اندازه گیر توان به آن وصل شد تا کارایی توان در حال اجرای آن را اندازه بگیرد، که 18.6 وات است. توان طراحی گرمایی پردازنده 95 وات است. بنابراین ما می توانیم یک تخمین تقریبی از توان نرم افزار و پیاده سازی FPGA داشته باشیم. جدول 8 نشان می دهد که نسبت انرژِی مصرف شده بین پیاده سازی نرم افزاری و پیاده سازی FPGA حداقل 24.6 برابر است. پیاده سازی FPGA انرژی کمتری را نسبت به همتای نرم افزاری خود مصرف می کند.

در جدول 5، چندین شتابدهنده ی CNN براساس FPGA لیست و با پیاده سازی ما در این کار مقایسه شده است. به دلیل اینکه روشهای قبلی از GMACS (گیگا ضرب کننده و جمع کننده بر ثانیه) استفاده می کند و ما از GFLOPS (گیگا عملیات ممیز شناور بر ثانیه) به عنوان معیار کارایی استفاده می کنیم، در ابتدا باید تمام اعداد منتج شده را به GOPS (گیگا عملیات بر ثانیه) تبدیل کنیم تا یک معیار یکسان را استفاده کرده باشیم. توجه کنید که هر عملیات ضرب-جمع شامل دو عملیات عدد صحیح است. به گونه ای که در نهمین ردیف از جدول 5 نشان داده شده است، شتابدهنده ی ما دارای یک خروجی 61.61 GOPS است که نسبت به سایر روش ها حداقل 3.62 برابر بهتر است.

به دلیل اینکه کارهای مختلف از فرصت های موازی سازی متفاوتی استفاده می کنند و پلتفرم های FPGA مختلف را استفاده می کنند، بسیار سخت است که یک مقایسه ی سرراست را بین آنها داشت. برای انجام یک مقایسه ی عادلانه، ما در ادامه تنایج چگالی کارایی را در جدول 5 ارایه می دهیم. به عنوان GOPS متوسط بر واحد سطح (تکه) تعریف می شود که می تواند نشان دهنده ی بهره وری یک طراحی باشد بدون توجه به پلتفرم FPGA مورد استفاده برای پیاده سازی آن. همان طور که در آخرین ردیف از جدول 5 نشان داده شده است، پیاده سازی ما به بالاترین چگالی کارایی دست یافته است که 1.8 برابر بهتر از بهترین بعد از خود است. به علاوه