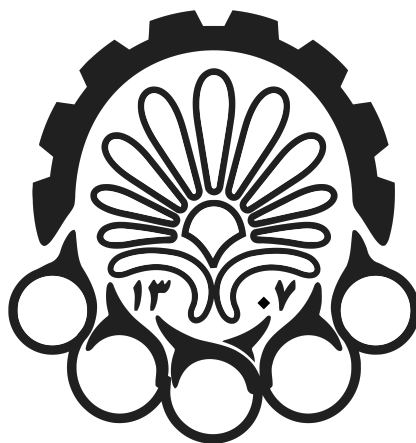


طراحی سیستم‌های قابل بازیگر بندی دکتر صاحب‌الزمانی



دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)
دانشکده مهندسی کامپیوتر

رضا آدینه پور ۴۰۲۱۳۱۰۵۵

تمرین سری سوم

۱۱ آذر ۱۴۰۳

سوال اول

با ذکر دلیل بیان کنید جملات زیر صحیح هستند یا خیر:

۱. CGRA ها از بلوک‌های محاسباتی در سطح کلمه استفاده می‌کنند در حالی که FPGA ها از بلوک‌های منطقی در سطح بیت استفاده می‌کنند.

پاسخ

درست.

در CGRA ها بلوک‌های محاسباتی بزرگتر از FPGA ها و در سطح کلمه هستند و می‌توانند داده‌های بزرگتری را پردازش کنند.

۲. Eyeriss یک CGRA است که برای پردازش تصویر طراحی شده است.

پاسخ

نادرست.

گزاره آنکه CGRA است صحیح است. اما گزاره اینکه برای کاربردهای پردازش تصویر طراحی شده است نادرست است. این تراشه با هدف شتابدهی شبکه‌های عصبی کانولوشنی ارائه شده است. درست است که یکی از کاربردهای عمده شبکه‌های CNN در زمینه تصاویر است اما هدف اصل این تراشه پردازش تصویر نمی‌باشد.

۳. HyCUBE امکان ارتباط بین واحدهای عملیاتی دور از هم را در یک سیکل فراهم می‌کند.

پاسخ

درست.

یک معماری CGRA است که از یک شبکه ارتباطی هیبریدی برای ایجاد ارتباط سریع بین واحدهای عملیاتی در فواصل دور از هم استفاده می‌کند.

۴. در سیستم‌های قابل پیکربندی مجدد استاتیک، زمان اجرا، توالی محاسبات و پیکربندی مجدد از قبل در زمان کامپایل مشخص می‌شود.

پاسخ

درست.

در این سیستم‌ها زمان اجرا، توالی محاسبات و پیکربندی مجدد، پیش از زمان کامپایل تعیین و مشخص می‌شود.

۵. پیکربندی مجدد پویا امکان تنظیم رفتار سیستم را در حین پردازش فعال وظایف فراهم می‌کند.

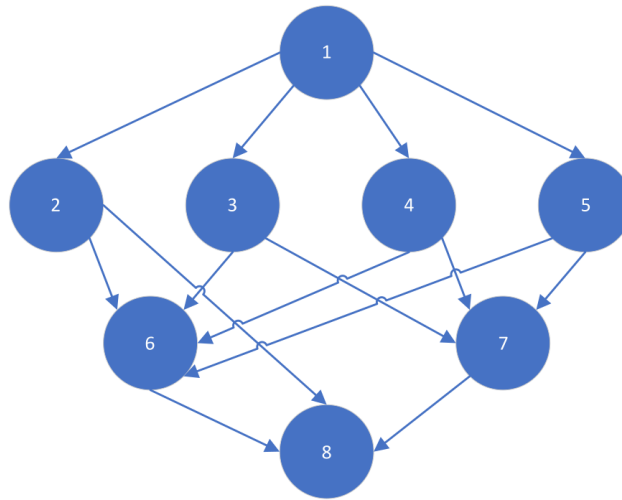
پاسخ

درست.

سیستم‌هایی که این قابلیت را دارند، می‌توانند با توجه به محیط و وظیفه خواسته شده از آن‌ها بدون توقف، پیکربندی خود را تغییر دهند و با وظایف جدید سازگار شوند.

سوال دوم

شکل زیر یک DFG است که می‌بایست به صورت بهینه بر روی یک 2×2 CGRA نگاشت شود.



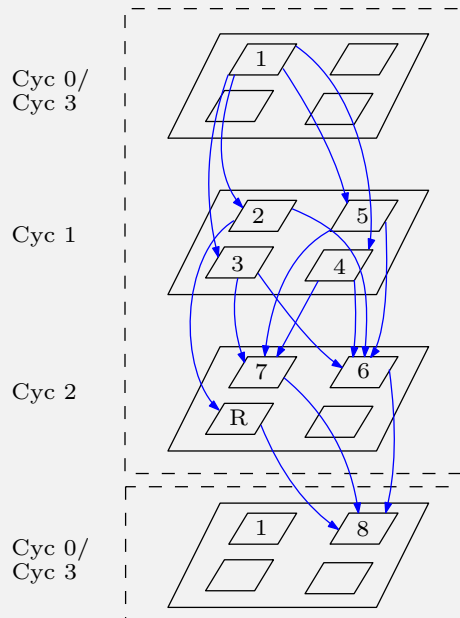
شکل ۱: DFG

هر گره تنها شامل یک عملیات است و شماره هر گره داخل آن درج شده است.

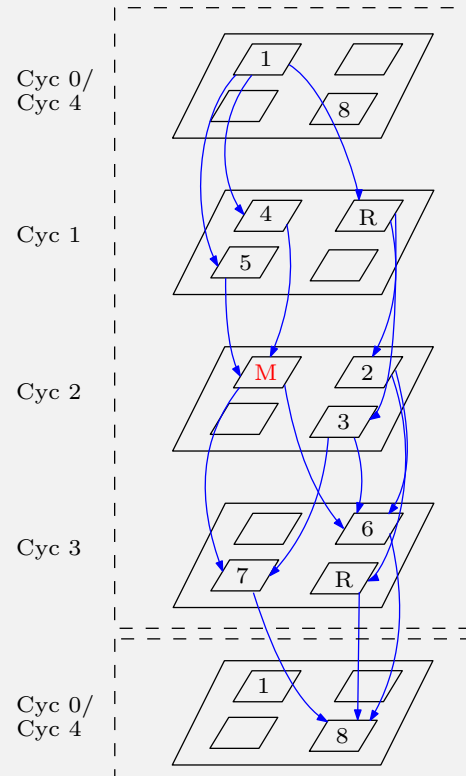
۱. نحوه نگاشت خود را شرح دهید.

پاسخ

Optimal Mapping (HyCUBE)



Optimal Mapping (N2N CGRA)



شکل ۲: Optimal Mapping

پاسخ

با توجه با شکل بالا (سمت راست) با توجه به اینکه گره شماره ۶ به ۴ گره وابسته است، به صورت مستقیم نمی توان مسئله را حل نمود. برای حل آن گره های ۴ و ۵ را با هم ترکیب می کنیم و نام آن را M می گذاریم. با این روش مسئله را می توان به روش N2N حل نمود. در روش دوم که در شکل سمت چپ آورده شده است محدودیتی نداریم و به سادگی می توان مسئله را حل نمود

۲. مقدار Initiation Interval را گزارش نمایید.

پاسخ

(آ) برای N2N: ۴

(ب) برای HyCUBE: ۳

سوال سوم

تفاوت‌های اصلی بین سیستم‌های قابل پیکربندی مجدد استاتیک آفلاین و سیستم‌های قابل پیکربندی مجدد پویای زمان-اجرا را با تمرکز بر نحوه تعریف توالی محاسبات و قابلیت‌های پیکربندی مجدد توضیح دهید. یک مقاله را که از این قابلیت‌های FPGA استفاده کرده است بررسی کنید و خلاصه آن را در دو پاراگراف گزارش نمایید.

پاسخ

تفاوت‌های اصلی بین سیستم‌های قابل پیکربندی مجدد استاتیک آفلاین و سیستم‌های قابل پیکربندی مجدد پویای زمان-اجرا را می‌توان به صورت زیر دسته‌بندی نمود:

۱. توالی محاسبات:

- سیستم‌های استاتیک آفلاین: در این سیستم‌ها، توالی محاسبات و پیکربندی‌های مربوط به آن‌ها در زمان کامپایل و پیش از اجرای برنامه تعریف می‌شود. یعنی کل پیکربندی سیستم از پیش مشخص است و نیازی به تغییر آن در زمان اجرا نیست. به عبارت دیگر، توالی محاسبات به صورت ثابت است و تغییر نمی‌کند.
- سیستم‌های پویای زمان-اجرا: در این سیستم‌ها، توالی محاسبات و پیکربندی‌ها می‌توانند در طول زمان اجرا تغییر کنند. به این معنا که در حین انجام پردازش‌ها، ممکن است سیستم به طور پویا تغییر کند تا بهترین عملکرد را در پاسخ به شرایط لحظه‌ای فراهم کند. این امکان به سیستم‌ها اجازه می‌دهد که به طور مؤثرتر از منابع خود استفاده کنند و به نیازهای مختلف پردازش پاسخ دهند.

۲. قابلیت‌های پیکربندی مجدد:

- سیستم‌های استاتیک آفلاین: پیکربندی سخت‌افزار در این سیستم‌ها در زمان کامپایل و پیش از اجرا تعیین می‌شود. این سیستم‌ها هیچگونه تغییر پیکربندی در زمان اجرا ندارند و همه‌ی تنظیمات از ابتدا مشخص و ثابت هستند.
- سیستم‌های پویای زمان-اجرا: در این سیستم‌ها، پیکربندی سخت‌افزار می‌تواند در حین اجرا تغییر کند. این سیستم‌ها قابلیت پیکربندی مجدد پویا دارند، که این امکان را فراهم می‌کند که سخت‌افزار به طور مداوم و در پاسخ به نیازهای پردازشی مختلف تنظیم شود. این ویژگی به ویژه در کاربردهایی که نیاز به انعطاف‌پذیری و کارایی بالا دارند مفید است.

در ادامه با توجه به اینکه موضوع تمرین عملی پیاده سازی کانونولوشن بود، مقاله [۱] را انتخاب کردم و در ادامه سعی می‌کنم کلیت این کار را در دو پاراگراف شرح دهم.

مقاله به بررسی استفاده از پیکربندی مجدد پویا در پردازش تصویر و سیگنال می‌پردازد. این تحقیق به طور خاص روی تخصیص منابع سخت‌افزاری به صورت پویا تمرکز دارد، به گونه‌ای که بر اساس سطح نویز در یک لحظه خاص، منابع سخت‌افزاری به حداقل مورد نیاز برای اجرای عملیات پردازش تصویر تخصیص داده می‌شود. در این سیستم‌ها، پیکربندی مجدد می‌تواند سرعت تغییرات پیکربندی و تعداد پیکربندی‌های ممکن را مشخص کند، این رویکرد به الگوریتم‌های پردازش‌های تصویر کمک می‌کند تا با توجه به تغییرات سیگنال تصویر، میزان محاسبات و نیاز به حافظه تغییر کند. در این مقاله به توصیف الگوریتمی برای فیلتر کردن تصویر می‌پردازد. این پیاده‌سازی بر روی FPGA مدل Xilinx Spartan-IIIE 600K انجام شده است. مقاله نشان می‌دهد که سیستم قادر است میزان محاسبات و حافظه مورد استفاده را در مقیاس‌های زمانی ریز (میلی‌ثانیه) و درشت (ثانیه) به طور پویا تنظیم کند. نتایج تجربی نشان می‌دهند که زمان اجرای فیلتر تصویر روی FPGA تا ۴۰۰ برابر سریع‌تر از پیاده‌سازی نرم‌افزاری روی پردازنده ۸.۲ گیگاهرتزی Pentium است، که این خود نشان‌دهنده مزیت‌های قابل توجه پیکربندی مجدد پویا در FPGA برای کاربردهای پردازش تصویر است.

*


References

- [1] FPGA Based Hardware Implementation of Image Filter With Dynamic Reconfiguration Architecture [\[Link\]](#)

سوال چهارم

سوال پنجم - پروژه عملی

در این پروژه، با نگاه به پروژه قبلی، بخش کانولوشن، یک سیستم پردازش تصویر طراحی می‌گردد. در این سیستم ورودی مربوط به دیتاست MNIST با سایز ۲۸ در ۲۸ بوده و ۳ فیلتر کانولوشن به ابعاد ۳ در ۳ به صورت پشت سر هم بر روی تصویر اعمال می‌شود. مقادیر موجود در ماتریس‌های کانولوشن به صورت تصادفی انتخاب شده و به عنوان ورودی به تابع کانولوشن داده می‌شود (در کد Fix نشده باشد) و خروجی با نمونه نرم‌افزاری مورد بررسی قرار می‌گیرد.

برای اطلاعات بیشتر می‌توانید از [این لینک](#) استفاده کنید.

در گزارش ارسالی علاوه بر شرح مراحل کار با فرض استفاده از Zynq7010 میزان سرعت و تأخیر اولیه را گزارش نمایید. همچنین با فرض امکان گسترش که برای پردازش موازی چه تعداد از بلوک طراحی شده شما در این FPGA قابل به کارگیری به صورت همزمان خواهد بود؟

پاسخ

این پروژه را به دو قسمت نرم‌افزاری و سخت‌افزاری تقسیم می‌کنیم و در انتها پاسخ‌های این دو قسمت را باهم مقایسه می‌کنیم.

۱. فاز نرم‌افزاری:

در این فاز ابتدا دیتاست MNIST را لود کرده و به صورت رندوم از هر رقم یکی را انتخاب می‌کنیم و آن را نمایش می‌دهیم:



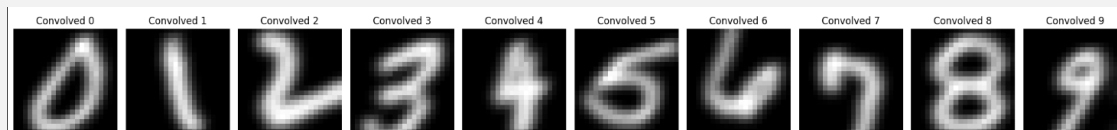
شکل ۳: اعداد رندوم انتخاب شده

ضمن اینکه مقادیر پیکسل‌های این ۱۰ عدد تصویر را برای استفاده در فاز سخت‌افزاری، به صورت ماتریس‌های 28×28 در فایل txt ذخیره می‌کنیم.

سپس با استفاده از ۳ کرنل زیر، عملیات کانولوشن را برای تمامی وروی‌ها به صورت متوالی انجام می‌دهیم.

$$\begin{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \end{bmatrix}$$

خروجی تصاویر کانوالو شده به صورت زیر ارائه می‌شود:



شکل ۴: خروجی تصاویر کانوالو شده

همچنین تصاویر خروجی را نیز در فایل txt ذخیره می‌کنیم.

۲. فاز سخت‌افزاری:

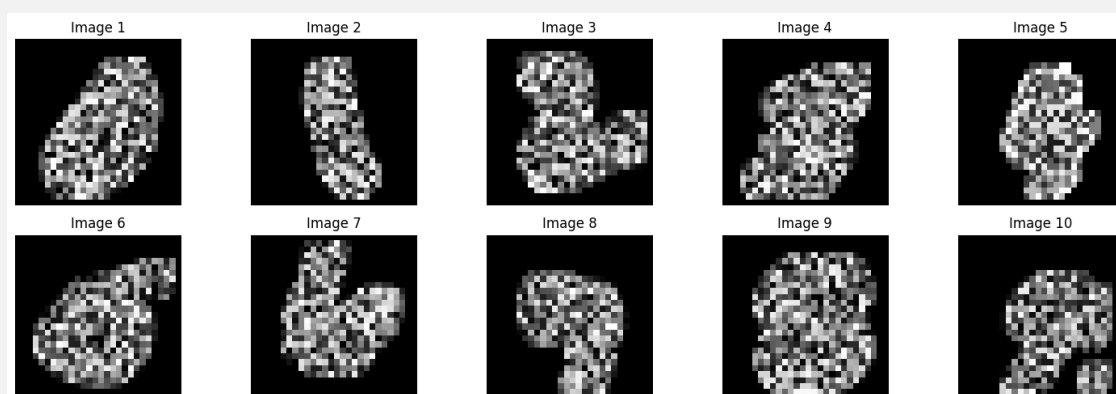
در این فاز ابتدا ماژول کانولوشن را در HLS طراحی می‌کنیم و سپس در فایل Testbench ورودی‌های تکست ذخیره‌شده در فاز نرم‌افزاری را به‌ترتیب می‌خوانیم و هر سه کرنل را به‌صورت پشت‌سرهم بر روی آن‌ها اعمال می‌کنیم. با توجه به ابعاد کرنل‌ها در هر مرحله از کانولوشن، از ابعاد تصویر کاسته می‌شود و نهایتاً ابعاد تصویر خروجی پس از انجام هر سه کانولوشن می‌بایست 22×22 باشد اما در این کد، ابعاد ورودی و خروجی را 28×28 گرفته ایم. و پس از هر مرحله کانولوشن، به تعداد مورد نیاز به تصویر ورودی صفر اضافه می‌کنیم. (Zero Padding).

نهایتاً پس از انجام محاسبات و مشاهده مشاهده پایان محاسبات در نرم‌افزار «شکل ۵»، به کد نوشته شده در فایل txt2img.ipynb می‌رویم تا به‌ترتیب تصاویر پردازش شده‌ای را که در فایل تکست ذخیره کرده‌ایم را بخوانیم و آن‌ها را نمایش دهیم تا بتوانیم با کد نرم‌افزاری مقایسه کنیم.

```
1 INFO: [SIM 2] ***** CSIM start *****
2 INFO: [SIM 4] CSIM will launch GCC as the compiler.
3 make: 'csim.exe' is up to date.
4 Final output for digit 0 saved!
5 Final output for digit 1 saved!
6 Final output for digit 2 saved!
7 Final output for digit 3 saved!
8 Final output for digit 4 saved!
9 Final output for digit 5 saved!
10 Final output for digit 6 saved!
11 Final output for digit 7 saved!
12 Final output for digit 8 saved!
13 Final output for digit 9 saved!
14 INFO: [SIM 1] CSim done with 0 errors.
15 INFO: [SIM 3] ***** CSIM finish *****
16
```

شکل ۵: اتمام Simulation

خروجی تصاویر کانوالو شده در ماژول HLS به‌صورت زیر ارائه می‌شود:



شکل ۶: تصاویر خروجی ماژول HLS

همچنین رپورت سنتز ماژول کانولوشن نیز به‌صورت زیر ارائه می‌شود:

پاسخ

Synthesis Summary Report of 'convolution'

General Information

Date:Sun Dec 1 23:20:50 2024

Version:2023.2 (Build 4023990 on Oct 11 2023)

Project:MNIST

Solution:solution1 (Vivado IP Flow Target)

Product family:virtexuplus

Target device:xcvu11p-flga2577-1-e

Timing Estimate

TargetEstimatedUncertainty

10.00 ns5.618 ns2.70 ns

Performance & Resource Estimates ⓘ

ModulesLoops

	Issue Type	Violation Type	Distance	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSP	FF	LUT	URAM
				-	3390	3.390E4	-	3391	-	no	0	6	265	818	0
on Pipeline VITIS_LOOP_9_1 VITIS_LOOP_11_2 ⚠️ II Violation				-	3384	3.384E4	-	3384	-	no	0	6	185	718	0

شکل ۷: رپورت سنتز HLS

بر اساس اطلاعات گزارش شده پس از سنتز، این ماژول ۲۵۶ عدد Flip Flop، ۸۱۸ عدد LUT و ۶ عدد DSP استفاده کرده است. همچنین Latency ماژول ۳۳۹۰ سیکل کلاک و Interval، ۳۳۹۱ گزارش شده است. مطابق با منابع موجود در تراشه ZYNQ 7010 که در جدول زیر آورده شده است، با فرض ثابت در نظر گرفتن شرایط طراحی و تعداد منابع مصرفی، بالغه می‌توانیم تا سقف تعداد منابع تراشه از ماژول طراحی شده قرار دهیم اما موضوعی که ما را محدود می‌کند، میزان Interval گزارش شده است که مقدار بالاییست و ما را در موازی سازی ماژول ها با یکدیگر محدود می‌کند.

برای رفع این مشکل، یکی از راه‌های ساده بدین صورت است که ابعاد ماژول کانولوشن را به 3×3 کاهش دهیم. در این صورت، همزمان می‌توانیم تمام تصویر را با ماژول های کانولوشن 3×3 پرکنیم و با سطح موازی سازی بالایی عملیات کانولوشن را انجام دهیم.