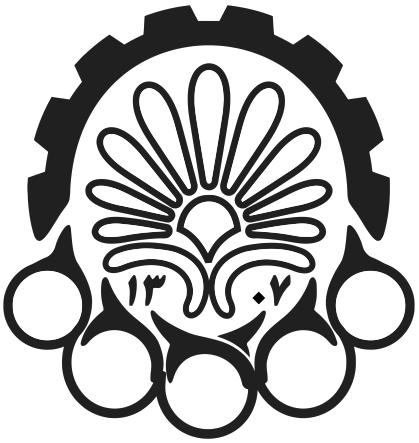
طراحی سیستم‌های قابل بازپیکربندی

دکتر صاحب‌الزمانی



دانشگاه صنعتی امیرکبیر

(پلی تکنیک تهران)

دانشکده مهندسی کامپیوتر

رضا آدینه پور ۴۰۲۱۳۱۰۵۵

تمرین سری سوم

۱۲ آذر ۱۴۰۳

طراحی سیستم‌های قابل بازپیکربندی

تمرین سری سوم

رضا آدینه پور ۴۰۲۱۳۱۰۵۵

دانشکده مهندسی کامپیوتر

سوال اول

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

۱. CGRAها از بلوک‌های محاسباتی در سطح کلمه استفاده می‌کنند در حالی که FPGAها از بلوک‌های منطقی در سطح بیت استفاده می‌کنند.

پاسخ

درست.
در CGRA ها بلوک‌های محاسباتی بزرگتر از FPGA ها و درسطح 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 نا درست 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 در زمان کامپایل تعیین و مشخص می‌شود.

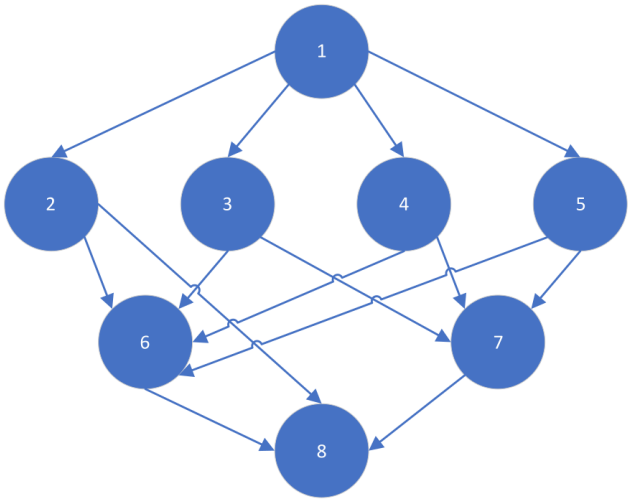
۵. پیکربندی مجدد پویا امکان تنظیم رفتار سیستم را در حین پردازش فعال وظایف فراهم می‌کند.

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 2x2 CGRA نگاشت شود.

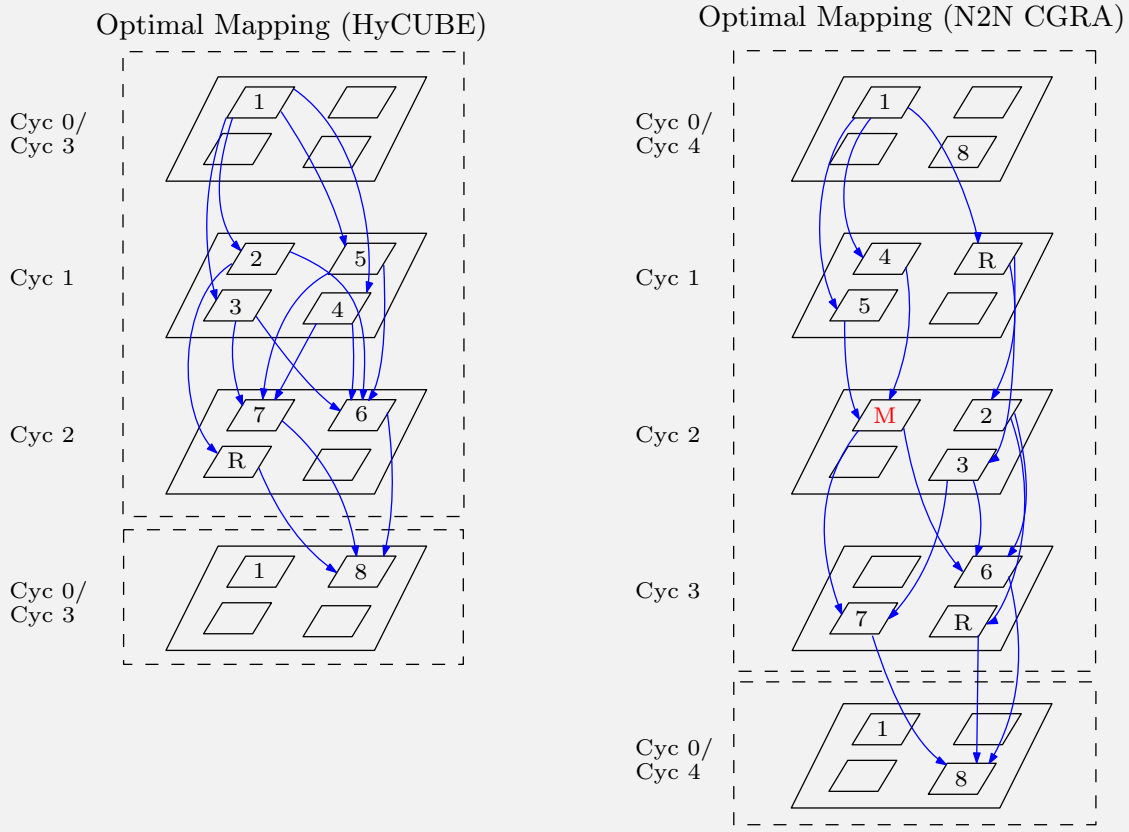


شکل ۱: DFG

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

۱. نحوه نگاشت خود را شرح دهید.

پاسخ



شکل ۲: Optimal Mapping

پاسخ

با توجه با شکل بالا (سمت راست) باتوجه به اینکه گره شماره ۶ به ۴ گره وابسته است، به‌صورت مستقیم نمی‌توان مسئله را حل نمود. برای حل آن گره‌های ۴ و ۵ را باهم ترکیب می‌کنیم و نام آن را M می‌گذاریم. با این روش مسئله را می‌توان به روش N2N حل نمود. در روش دوم که در شکل سمت چپ آورده شده است محدودیتی نداریم و به سادگی می‌توان 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 زمان‌ـ 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 زمان‌ـاجرا را می‌توان به‌صورت 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 زمان‌ـاجرا: 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 به‌طور پویا تغییر کند تا بهترین عملکرد را در پاسخ به شرایط لحظه‌ای فراهم کند. این امکان به سیستم‌ها اجازه می‌دهد که به‌طور 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 زمان‌ـاجرا: 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 به‌طور 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 کانولوشن 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 به‌صورت 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 600K Spartan-IIE 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

کاربردهای CGRA در حوزه امنیت، یادگیری عمیق و سلامتی (از هر مورد ۱ نمونه کاربرد) را با ذکر مثال (مانند COBRA و Eyeriss) با ارجاع به منابع شرح دهید.

پاسخ

۱. **یادگیری عمیق:**

یادگیری عمیق، به‌ویژه شبکه‌های عصبی کانولوشنی (CNN ها) که از نظر محاسباتی مرتب و منظم هستند، تبدیل به هدفی برای CGRAهای شده‌اند. در اینجا هدف این است که عمومیت و قابلیت پیکربندی مجدد CGRAهای سنتی محدود شود تا با الگوهای محاسباتی CNN هماهنگ شود و به جای آن، منطق اضافی برای پشتیبانی از عملیات خاص برای بارهای کاری یادگیری عمیق (مثل فشرده‌سازی، Multi Scaling و ...) به کار رود. همچنین، این معماری‌ها معمولاً از نمایش‌های عددی کوچکتر (یا ترکیبی) پشتیبانی می‌کنند، چون یادگیری عمیق معمولاً با محاسبات با دقت پایین سازگار است [۲].

معماری DT-CGRA [۴]، [۳] طراحی CGRA با واحدهای پردازشی نسبتاً درشتی دارد که شامل حداکثر سه دستور ضرب و جمع می‌شود. این واحدهای پردازشی همچنین شامل خطوط تأخیر قابل برنامه‌ریزی هستند تا داده‌های زمانی نزدیک به هم راحت‌تر نقشه‌برداری شوند. داده‌ها در داخل RCها از طریق توکن‌ها همگام‌سازی می‌شوند. پشتیبانی از پترن‌های دسترسی رایج در یادگیری عمیق (مثل stride، type و ...) از طریق واحدهای Stream-Buffers های شخصی‌سازی شده که به‌صورت VLIW قابل برنامه‌ریزی هستند، فراهم می‌شود و دسترسی به حافظه خارجی را تولید می‌کنند.

همچنین در مثالی دیگر Cambricon [۱] یک معماری CGRA است که به طور خاص برای شتاب‌دهی محاسبات شبکه‌های عصبی طراحی شده است. این معماری به ویژه در دستگاه‌های موبایل و سیستم‌های embedded استفاده می‌شود و به دلیل مصرف پایین انرژی و عملکرد بالای آن، در اجرای مدل‌های یادگیری عمیق در زمان واقعی مفید است. Cambricon برای پردازش‌های پیچیده مانند تشخیص اشیاء و پردازش زبان طبیعی بهینه‌سازی شده است

۲. **امنیت:**

AES یکی از استانداردهای معروف و پرکاربرد برای رمزگذاری داده‌ها است که در بسیاری از سیستم‌های امنیتی استفاده می‌شود. از آنجا که AES نیاز به پردازش‌های ریاضی پیچیده‌ای دارد، استفاده از معماری‌های موازی می‌تواند زمان پردازش را به طور چشمگیری کاهش دهد. در [۵]، به عنوان یک معماری قابل برنامه‌ریزی و منعطف معرفی می‌شود که می‌تواند عملیات مختلفی از جمله الگوریتم‌های رمزنگاری را با بهره‌وری انرژی بالا و کارایی بالا پردازش کند. چالش اصلی در طراحی سیستم‌های رمزنگاری سخت‌افزاری، دستیابی به تعادل میان سرعت پردازش بالا و انعطاف‌پذیری است. بسیاری از معماری‌ها یا سرعت بالایی دارند ولی انعطاف‌پذیری کمی دارند، یا برعکس. این محدودیت‌ها باعث می‌شود که بهره‌وری سخت‌افزار پایین باشد و استفاده از آن‌ها در کاربردهای چندگانه (multi-domain) با مشکلاتی مواجه شود. برای حل این مشکل، نویسندگان به CGRA روی آورده‌اند که به دلیل ویژگی‌های خاص خود، می‌تواند هم سرعت پردازش بالا و هم انعطاف‌پذیری خوبی را فراهم کند. UECP بر اساس معماری CGRA طراحی شده و با استفاده از دو تکنیک بهینه برای افزایش عملکرد و کارایی سخت‌افزار بهینه شده است:

- آرایه پردازش پایپلاین شده 4×4 (PEA): این آرایه پردازشی پایپلاین‌شده به UECP اجازه می‌دهد تا عملیات‌های رمزنگاری را به صورت موازی و با سرعت بالا انجام دهد.
- واحد حسابی منطقی قابل تنظیم (C-ALU): C-ALU به UECP اجازه می‌دهد که عملیات‌های ریاضی و منطقی مختلف مورد نیاز برای الگوریتم‌های رمزنگاری را به طور دینامیک و بسته به نیاز برنامه تنظیم کند. این امکان انعطاف‌پذیری بالایی به سیستم می‌دهد تا بتواند به راحتی برای انواع مختلف الگوریتم‌ها،

پاسخ

پیکربندی شود.

۳. **سلامت:** در زمینه بیوانفورماتیک، معماری‌های CGRA برای پردازش داده‌های پیچیده و محاسبات سنگین مورد استفاده قرار می‌گیرند که در تحلیل‌های ژنومیک و مدل‌سازی‌های بیولوژیکی کاربرد دارند. یک مثال از این نوع کاربرد، معماری CGRA در پردازش داده‌های ژنومیک و شبیه‌سازی‌های بیولوژیکی است. در [۶] معماری CGRA به‌گونه‌ای طراحی شده است که می‌تواند چهار الگوریتم محبوب بیوانفورماتیک را به طور کامل پردازش کند:

- Needleman-Wunsch برای هم‌ترازی دنباله‌های پروتئینی.
- Smith-Waterman برای هم‌ترازی دنباله‌های DNA.
- HMMER برای شبیه‌سازی مدل‌های مارکوف مخفی (HMM) در توالی‌های زیستی.
- Maximum Likelihood برای تحلیل‌های فیلوژنتیکی.

این الگوریتم‌ها معمولاً نیاز به پردازش‌های سنگین و پیچیده دارند که با استفاده از معماری CGRA می‌توانند به صورت موازی و بهینه پردازش شوند. این پلتفرم طراحی شده به‌طور خاص برای نیازهای محاسباتی این الگوریتم‌ها بهینه شده است و باعث افزایش کارایی و سرعت در پردازش داده‌ها می‌شود.

*

References

[1] J. Chen et al., ”Cambricon: An Efficient and Flexible Architecture for Deep Learning,” 2016 [Link]

[2] M. Courbariaux, Y. Bengio, and J.-P. David, ‘‘Training deep neural networks with low precision multiplications,’’ 2014, arXiv:1412.7024. [Link]

[3] X. Fan, H. Li, W. Cao, and L. Wang, ‘‘DT-CGRA: Dual-track coarse grained reconfigurable architecture for stream applications,’’ in Proc. 26th Int. Conf. Field Program. Log. Appl. (FPL), Aug. 2016, pp. 1–9.

[4] X. Fan, D. Wu, W. Cao, W. Luk, and L. Wang, ‘‘Stream processing dual track CGRA for object inference,’’ IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 26, no. 6, pp. 1098–1111, Jun. 2018.

[5] A. G. et al., ”AES Encryption Accelerator with CGRA Architecture,” 2017 [Link]

[6] A Coarse-Grained Reconfigurable Processor for Sequencing and Phylogenetic Algorithms in Bioinformatics [Link]

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.

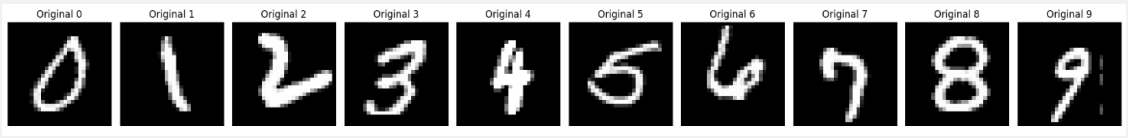
در گزارش ارسالی علاوه بر شرح مراحل کار با فرض استفاده از Zynq7010 میزان سرعت و تأخیر اولیه را گزارش نمایید. همچنین با فرض امکان گسترش که برای پردازش موازی چه تعداد از بلوک طراحی شده شما در این FPGA قابل به کارگیری به صورت همزمان خواهد بود؟

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 به‌صورت رندوم از هر رقم یکی را انتخاب می‌کنیم و آن را نمایش می‌دهیم:



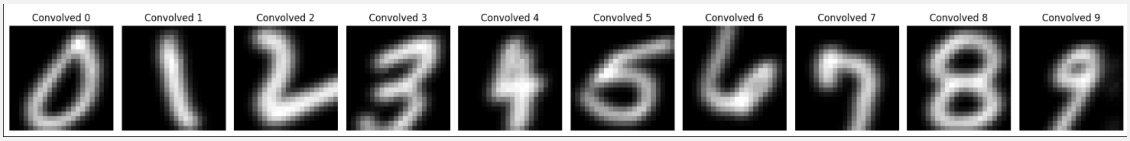
شکل ۳: اعداد رندوم انتخاب شده

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

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

$$\left[\begin{bmatrix}0&1&0\\1&0&0\\0&0&0\end{bmatrix},\begin{bmatrix}1&0&1\\0&1&1\\1&1&1\end{bmatrix},\begin{bmatrix}0&1&0\\1&0&1\\0&0&0\end{bmatrix}\right]$$

خروجی تصاویر کانوالو شده به‌صورت زیر ارائه می‌شود:



شکل ۴: خروجی تصاویر کانوالو شده

همچنین تصاویر خروجی را نیز در فایل `txt`. ذخیره می‌کنیم.

پاسخ

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 کرنل ها در هر مرحله از کانولوشن، از ابعاد تصویر کاسته می‌شود و نهایتا 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 تیاز به تصویر ورودی صفر اضافه می‌کنیم. (Zero Padding).

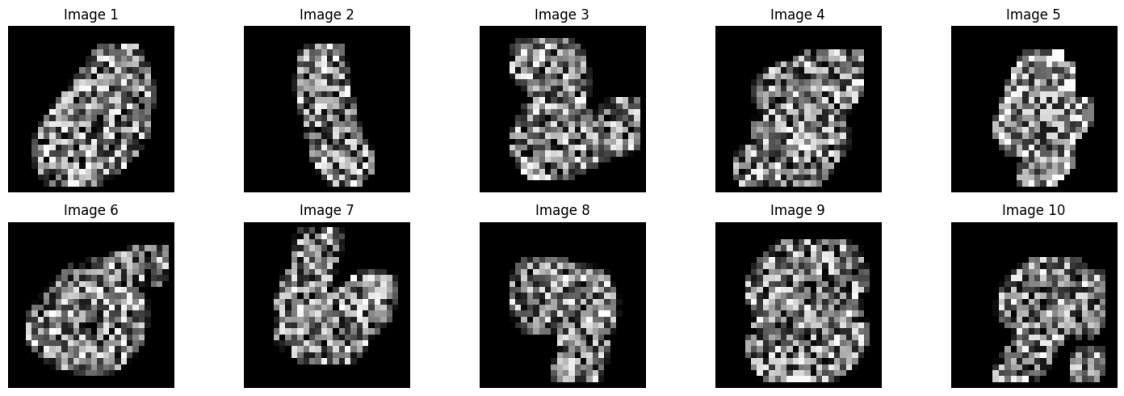
نهایتا پس از انجام محاسبات و مشاهده مشاهده پایان محاسبات در نرم‌افزار «شکل ۵»، به کد نوشته شده در فایل `txt2img.ipynb` می‌رویم تا به‌ترتیب تصاویر پردازش شده‌ای را که در فایل تکست ذخیره کرده‌ایم را بخوانیم و آن‌ها را نمایش دهیم تا بتوانیم با کد نرم‌افزاری مقایسه کنیم.

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

```

شکل ۵: اتمام Simulation

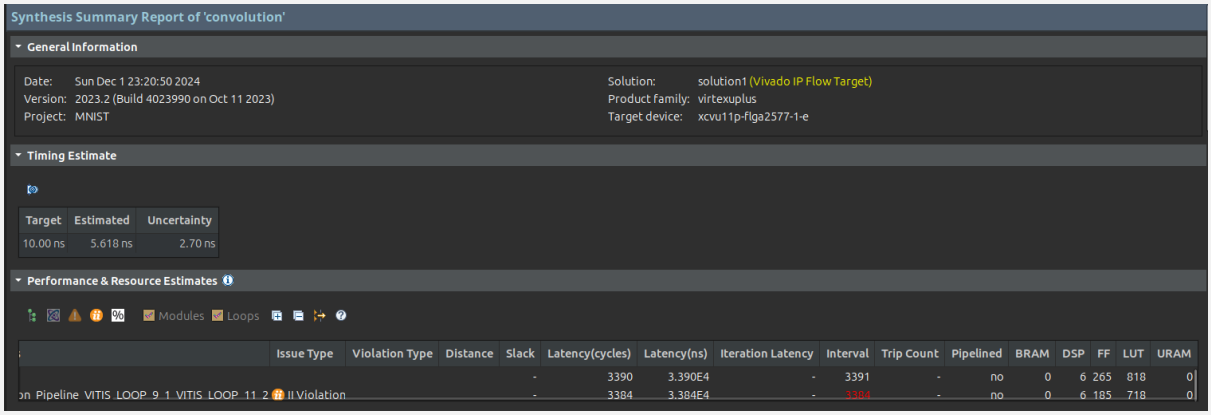
خروجی تصاویر کانوالو شده در ماژول HLS به‌صورت زیر ارائه می‌شود:



شکل ۶: تصاویر خروجی ماژول HLS

همچنین ریپورت سنتز ماژول کانولوشن نیز به‌صورت زیر ارائه می‌شود:

پاسخ



شکل ۷: ریپورت 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 درنظر گرفتن شرایط طراحی و تعداد منابع مصرفی، بالغوه می‌توانیم تا سقف تعداد منابع تراشه از ماژول طراحی شده قرار دهیم اما موضوعی که مارا محدود می‌کند، میزان Interval گزارش شده است که مقدار بالاییست و ما را در موازی سازی ماژول ها با یکدیگر مجدود می‌کند.
برای رفع این مشکل، یکی از راه‌های ساده بدین صورت است که ابعاد ماژول کانولوشن را به 3×3 کاهش دهیم. در این‌صورت، همزمان می‌توانیم تمام تصویر را با ماژول های کانولوشن 3×3 پر کنیم و با سطح موازی سازی بالایی عملیات کانولوشن را انجام دهیم.