Designing an Energy-Efficient Fully-Asynchronous Deep Learning Convolution Engine

Mattia Vezzoli *Yale University* Lukas Nel *Yale University* Kshitij Bhardwaj *Lawrence Livermore National Lab* Rajit Manohar *Yale University* Maya Gokhale *Lawrence Livermore National Lab*

***Abstract*—In the face of exponential growth in semiconductor energy usage, there is a significant push towards highly energy-efficient microelectronics design. While the traditional circuit designs typically employ clocks to synchronize the computing operations, these circuits incur significant performance and energy overheads due to their data-independent worst-case operation and complex clock tree networks. In this paper, we explore asynchronous or clockless techniques where clocks are replaced by request, acknowledge handshaking signals. To quantify the potential energy and performance gains of asynchronous logic, we design a highly energy-efficient asynchronous deep learning convolution engine, which uses 87% of total DL accelerator energy. Our asynchronous design shows $5.06\times$ lower energy and $5.09\times$ lower delay than the synchronous one.**

I. Introduction

Since 2010, semiconductor energy use has doubled every three years, and by 2030, semiconductors could consume nearly 20% of planetary energy production [1]. The U.S. Department of Energy has set a goal of nearly $1000\times$ improvement in energy efficiency of chips in the next 20 years to help grow the economy as well as tackle the climate crisis [1].

Asynchronous design is a rapidly growing alternative to traditional synchronous design, especially for neuromorphic computing, due to its potential for low-power, high-speed, increased reliability, and reduced electromagnetic interference [2]. In asynchronous circuits, the elimination of clock and the associated clock distribution circuit, along with their data-driven computation have shown significant energy improvements over synchronous designs [3]. Despite these advantages, the lack of asynchronous design tools and libraries has limited its widespread adoption [3]. While asynchronous design has been used for spiking neural networks, there is very limited work on using it for widely used deep learning (DL) accelerators [4]. DL accelerators are deployed for a variety of autonomous tasks in low-power, near-sensor edge applications, and their training consumes significant data center cycles.

In this work, we design the heart of a DL accelerator – a convolution engine which contributes 87% of the total DL accelerator energy [5] – and compare asynchronous and synchronous mappings generated automatically through the ACT [6] EDA flow. We employ two asynchronous protocol libraries: (i) quasi delay-insensitive (QDI) [4], which uses a four-phase handshake protocol where a single communication requires a set and a reset phase of request (req) and acknowledgement (ack) signals; and (ii) MOUSETRAP [7] which uses a two-phase protocol where only a single transition on req and ack wires constitutes a communication. In both libraries, we use single-rail data encoding in which the validity of data is enforced by a timing constraint: a request control signal corresponding to the data should arrive only after the data are stable. We evaluate both these libraries as they have interesting tradeoffs: e.g., QDI is more robust than MOUSETRAP but can incur more energy overheads [3]. The resulting asynchronous implementations are compared with the clocked and clock-gated synchronous versions that use ready/valid flow control. Our results show that the MOUSETRAP design achieves $5.06\times$ lower energy and $5.09\times$ lower delay than the equivalent synchronous design (without clock gating). All the implementations are generated through the ACT EDA flow [8] from a single architecture description, and are synthesized at a pre-layout level to generate performance and energy metrics. Using a single architecture description ensures fair comparison, focused on the differences across synchronous and asynchronous circuit technology families.

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 (IM: LLNL-CONF-857644).

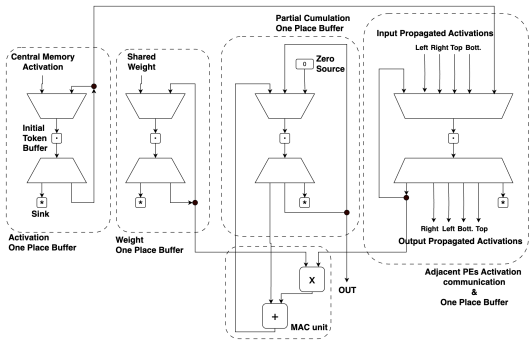


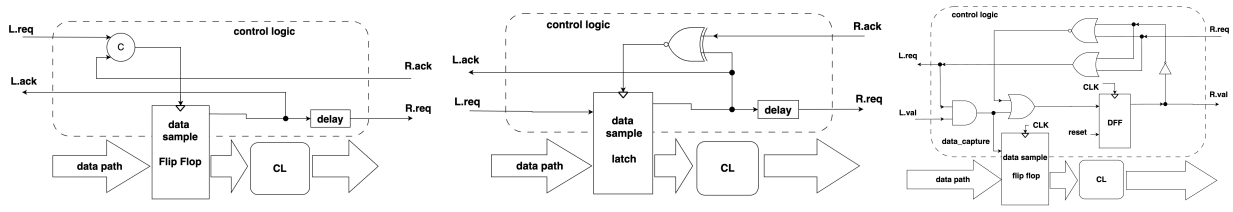
Fig. 1: Target MAC architecture

II. Convolution Engine

a) The convolution engine: Figure 1 shows the MAC unit of our target 70×70 MAC array [9]. The engine also includes buffers for input activation, weights and partial sum, and logic to communicate with adjacent MACs. It uses the output stationary dataflow, widely used in DNN accelerators [9].

b) Control logic: Synchronous and asynchronous pipeline control libraries are used to generate multiple implementations of the convolution engine. Figure 2 shows three pipeline control designs. The control logic activates the data sample flipflop or latch, which in turn forwards data into the datapath combinational logic (CL). The QDI control is based on a C-element which is an asynchronous register whose output is de-asserted if both inputs are low and asserted if both inputs are high, otherwise it holds the state. The MOUSETRAP control is the simplest, involving a single XNOR gate. In terms of the datapath registers also, MOUSETRAP uses more lightweight latches compared to the flip flops used in the other two designs. The asynchronous pipelines rely on additional delay in the control path to match the combinational logic delay such that the request transitions

979-8-3503-4859-0/DATE24/© 2024 EDAA



(a) Asynchronous QDI (b) Asynchronous MOUSETRAP (c) Synchronous Control

Fig. 2: Pipeline control logic families

only after the data is stable. The delay is automatically inserted by our asynchronous design tool flow.

The synchronous design has been augmented with flow control (ready/valid signals) so it is functionally equivalent to the asynchronous designs as the latter support flow control inherently (through local handshaking). For synchronous clock gating (not shown), local ready/valid signals are used to gate datapath flip-flops. The synchronous control logic is significantly more complex than either asynchronous ones.

c) Design tools: We use the ACT tool flow to compile the convolution engine. The convolution engine is described in a low-level dataflow circuit description language, which is synthesized at pre-layout level into the four pipeline control variants using the ACT compiler. We extended the ACT tools to include a *dataflow mapper* utility (`dflowmap`) that translates dataflow building blocks into the four simulation models. Each simulation model is run in the `actsim` simulation engine. `actsim` supports back-annotation of energy and delay metrics from SPICE simulations of individual circuit components and datapath logic (obtained from Cadence Genus in our case in 28nm). SRAMs are used to store activations and weights. SRAM read/write delay, energy, and leakage power were obtained from the Asynchronous Memory Compiler (AMC) [10], which are then included in the back-annotation configuration file for `actsim`. For synchronous memories, these metrics are taken from the foundry datasheet. Using these metrics, the simulation engine can determine overall energy and delay of the convolution engine. This process is analogous to back-annotated simulation supported by commercial tools.

During the simulation and testing phase, we utilized a 70×70 input activation matrix and 3×3 kernel matrix. To verify the correctness of the engine's computations, we compared the generated output matrix against a Pytorch convolution.

III. Results

Figure 3 shows the energy and delay comparisons of two asynchronous and two synchronous implementations: asynchronous MOUSETRAP, asynchronous QDI, synchronous and synchronous with clock gating. The Figure shows that MOUSETRAP delivers $5.06\times$ energy improvement and $5.09\times$ lower delay than the synchronous implementation. These advantages are due to: (i) MOUSETRAP's latch-based design, which results in reduced energy consumption and latency compared to flip-flop-based synchronous pipelines; (ii) the data-driven operation of asynchronous pipelines, where components activate only as needed, significantly reducing idle power consumption; and (iii) the use of two-phase handshaking protocol leads to lower energy cost compared to the QDI design which uses the four-phase protocol. Clock gating is able to improve energy but only very slightly as it comes with its own added overhead of extra circuitry.

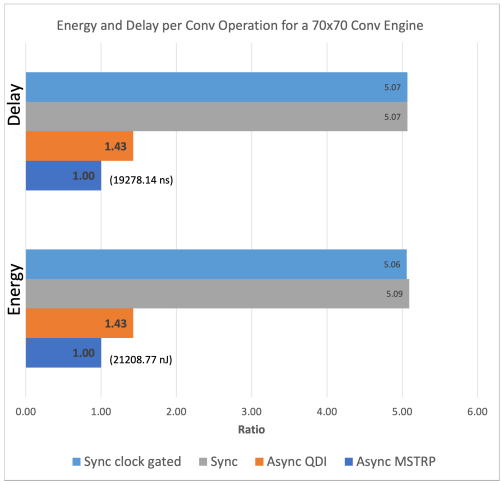


Fig. 3: Energy and delay per convolution operation of tested convolution engine, normalized to the MOUSETRAP design.

IV. Conclusions

In this paper, we quantitatively evaluate asynchronous design applied to the main compute logic of deep neural network accelerators, the convolution engine. Using an automated tool flow for asynchronous logic, we compare two asynchronous and two synchronous implementations for energy and speed. Our results show that the MOUSETRAP asynchronous control protocol yields the most energy-efficiency and highest speed performance. Our evaluation indicates that using asynchronous logic may yield significant benefit toward energy reduction of ubiquitous deep learning accelerators.

References

[1] DOE, "Semiconductor Industry Energy Efficiency Scaling (EES2) Goal Technical Workshop." https://www.energy.gov/eere/amo/events/semiconductor-industry-energy-efficiency-scaling-ees2-goal-technical-workshop.

[2] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *Ieee Micro*, vol. 38, no. 1, pp. 82–99, 2018.

[3] S. M. Nowick and M. Singh, "Asynchronous design—part 1: Overview and recent advances," *IEEE Design & Test*, vol. 32, no. 3, pp. 5–18, 2015.

[4] S. Xiao, W. Liu, J. Lin, and Z. Yu, "A data-driven asynchronous neural network accelerator," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 9, pp. 1874–1886, 2020.

[5] D. Li, X. Chen, M. Becchi, and Z. Zong, "Evaluating the energy efficiency of deep convolutional neural networks on cpus and gpus," in *2016 IEEE international conferences on big data and cloud computing (BDCloud), social computing and networking (SocialCom), sustainable computing and communications (SustainCom)(BDCloud-SocialCom-SustainCom)*, pp. 477–484, IEEE, 2016.

[6] S. Ataei, W. Hua, Y. Yang, R. Manohar, Y.-S. Lu, J. He, S. Maleki, and K. Pingali, "An open-source eda flow for asynchronous logic," *IEEE Design & Test*, vol. 38, no. 2, pp. 27–37, 2021.

[7] M. Singh and S. M. Nowick, "Mousetrap: Ultra-high-speed transition-signaling asynchronous pipelines," in *Proceedings 2001 IEEE International Conference on Computer Design: VLSI in Computers and Processors. ICCD 2001*, pp. 9–17, IEEE, 2001.

[8] R. Manohar *et al.*, "Asynchronous VLSI Design Tools." https://github.com/asyncvlsi/, 2022.

[9] Z. Du, R. Fasthuber, T. Chen, P. Ienne, L. Li, T. Luo, X. Feng, Y. Chen, and O. Temam, "Shidiannao: Shifting vision processing closer to the sensor," pp. 92–104, 2015.

[10] S. Ataei and R. Manohar, "Amc: An asynchronous memory compiler," in *2019 25th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*, pp. 1–8, IEEE.