TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 01/15 pp565–573
DOI: 10.26599/TST.2020.9010048
Volume 26, Number 5, October 2021

Design and Tool Flow of a Reconfigurable Asynchronous Neural Network Accelerator

Jilin Zhang, Hui Wu, Weijia Chen, Shaojun Wei, and Hong Chen*

Abstract: Convolutional Neural Networks (CNNs) are widely used in computer vision, natural language processing, and so on, which generally require low power and high efficiency in real applications. Thus, energy efficiency has become a critical indicator of CNN accelerators. Considering that asynchronous circuits have the advantages of low power consumption, high speed, and no clock distribution problems, we design and implement an energy-efficient asynchronous CNN accelerator with a 65 nm Complementary Metal Oxide Semiconductor (CMOS) process. Given the absence of a commercial design tool flow for asynchronous circuits, we develop a novel design flow to implement Click-based asynchronous bundled data circuits efficiently to mask layout with conventional Electronic Design Automation (EDA) tools. We also introduce an adaptive delay matching method and perform accurate static timing analysis for the circuits to ensure correct timing. The accelerator for handwriting recognition network (LeNet-5 model) is implemented. Silicon test results show that the asynchronous accelerator has 30% less power in computing array than the synchronous one and that the energy efficiency of the asynchronous accelerator achieves 1.538 TOPS/W, which is 12% higher than that of the synchronous chip.

Key words: Convolutional Neural Network (CNN) accelerator; asynchronous circuit; energy efficiency; adaptive delay matching; asynchronous design flow

1 Introduction

The Convolutional Neural Network (CNN) is a class of deep neural networks widely used in computer vision and natural language processing[1, 2]. In many real-world applications such as robotics, self-driving car, and augmented reality, recognition tasks need to be carried out in a timely fashion on a computationally limited platform, which makes energy efficiency become an increasingly critical indicator of CNN accelerators[3]. With sparse MobileNet and reconfigurable architecture, Eyeriss v2[4] in a 65 nm Complementary Metal Oxide Semiconductor (CMOS) process achieves a throughput of 1470.6 inferences/s and 2560.3 inferences/J at a batch size of 1. With butterfly structure, the NPU[5] achieves 6.9 TOPS and 3.5 TOPS with 75% zero-weights for 55 and 33 kernels, respectively. The LNPU[6], fabricated in 65 nm CMOS technology, can achieve 5.84 TFLOPS/W with 8.2 fps and 2.74 TFLOPS/W with 32.8 fps on the VGG16 Conv-layer inference benchmark at FP8 precision by maintaining training accuracy with a fine-grained mixed precision of FP8-FP16.

However, most CNN accelerators are designed with synchronous circuits. Asynchronous circuits have the main potential advantages in low power consumption, high-performance speed, and no clock distribution problems[7, 8]. Many successful asynchronous chips prove the advantages. Gageldonk et al.[9] implemented an asynchronous 80C51 microcontroller, which showed a power advantage of a factor 4 compared with a

• Jilin Zhang, Hui Wu, Weijia Chen, Shaojun Wei, and Hong Chen are with the Institute of Microelectronics, Tsinghua National Laboratory for Information Science and Technology, and Beijing Engineering Center of Technology and research on Wireless Medical and Health System, Tsinghua University, Beijing 100084, China. E-mail: hongchen@tsinghua.edu.cn.
* To whom correspondence should be addressed.
Manuscript received: 2020-08-31; accepted: 2020-10-09

synchronous implementation in the same technology. Moreover, asynchronous circuits are characterized by their local data or control-driven flow of operations, which differs from the global clock-driven flow of synchronous designs. This character enables the different blocks of asynchronous circuits to operate at individual ideal "frequencies" or in event idle mode as needed, consuming energy only when and where needed. Clock gating has a similar goal-enabling register only when needed but does not address the power drawn by the centralized control and clock tree buffers[10]. In some event-driven applications, such as Spiking Neural Networks (SNN), asynchronous circuits can control circuits in an event-driven manner more naturally than clock gating. As a result, asynchronous logic has been advocated as a means of reducing power consumption in many applications[11, 12]. IBMs TrueNorth, which successfully implements SNN on a chip, has a low power requirement of 65 mW[13], and the energy consumption in every state change of the neuron in Intel's Loihi is only 52–81 pJ[14].

In our previous work, we implemented asynchronous and synchronous CNN accelerators with Xilinx FPGA VC707 to maximize the advantages of asynchronous circuits and compare their performance with synchronous ones[15]. In the present study, we implemented asynchronous CNN accelerators with a TSMC 65 nm CMOS process to further verify the performance of asynchronous accelerators. The implemented CNN accelerators contain a Computing Array (CA) that consists of six computing cores (CEs), one Pooling Unit (PU), and one matrix multiplication module. The CA has no global clock, and the data are driven by the local pulse signals from the Click elements in the asynchronous pipeline. We also implemented a synchronous accelerator with the same technology for comparison. The difference between asynchronous and synchronous accelerators is that the asynchronous one uses a Click-based pipeline instead of a global clock.

The rest of the paper is organized as follows. Section 2 introduces the handshake protocol, Click element, and Click-based asynchronous pipeline. In Section 3, the design of the asynchronous accelerator is discussed in detail. The design flow to adopt commercial EDA tool for asynchronous circuit design is illustrated in Section 4. The ASIC implementation and test results are discussed in Section 5, and the paper is concluded in Section 6.

2 Click-Based Asynchronous Circuits

2.1 Handshake protocol

In asynchronous circuits, the handshake protocol is used to send and receive data instead of the global clock adopted in synchronous circuits. As shown in Fig. 1, with the handshake signals *req* and *ack*, the blocks C_n and C_{n+1} exchange bundled data, in which the data signals use normal Boolean levels to encode information, and separate request and acknowledge wires are bundled with the data signals[7]. Only when needed and the data are ready, the sender C_n sends a request signal to remind the receiver C_{n+1} that the data are valid, and then C_{n+1} receives the data and sends an acknowledgment signal indicating that the receiving is completed. Then, the sender C_n can send another request signal.

2.2 Click element

Click is an asynchronous control element that adopts the two-phase handshake protocol, which outperforms the four-phase protocol in speed and power[16]. In addition, the "*fire*" impulse of the Click element can be regarded as a local clock allowing Click-based circuits to be designed with commercial EDA tools. The schematic of the Click element is shown in Fig. 2, in which either the rising or falling edge of the "*in_req*" signal can cause a pulse signal called "*fire*", which is used as a local clock to drive the D flip-flop to capture the data for computation. The transition of "*in_req*" signal can be delivered to "*out_req*", which becomes the "*in_req*" signal for the next stages. The width of the pulse "*fire*", which is decided by the delay lines, should be designed carefully to meet the setup and hold requirements of D flip-flop.

Fig. 1 Asynchronous handshake protocol.

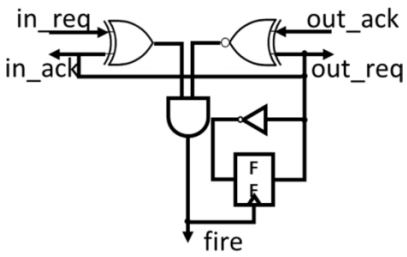

Fig. 2 Click element[16].

2.3 Comparison of asynchronous and synchronous pipelines

We consider two-stage pipelines as examples to compare the speed between asynchronous and synchronous pipelines that are widely used in circuit design. The synchronous pipeline is comprised of three DFFs, as shown in Fig. 3, in which the delay of the combinational logics CL1 and CL2 between the DFFs are $D1$ and $D2$, respectively. As shown in Fig. 3, three Click elements are connected in series. We suppose the data path in the asynchronous pipeline is the same as that in the synchronous pipeline and that $D1$ is larger than $D2$. Between each stage of Click, delay lines (not shown) are used to match the delay of the combinational logic in the data path, which makes the interval of the fire signals almost the same as the delay of the corresponding data path and speeds up the circuits.

First, we analyze the throughput and delay of the three-stage synchronous and asynchronous pipelines, as shown in Fig. 3, and the results are shown in Table 1. When the input data flow continuously, the throughputs of the two pipelines are both $1/D1$. When the input data flow intermittently, the delay of the asynchronous pipeline is $D1+D2$, which is shorter than that of the synchronous pipeline (i.e., $2\times D1$). The clock period of the synchronous circuits should be the worst case, that is, the longest delay $D1$. In a word, the speed of the asynchronous pipeline is faster than that of the synchronous pipeline. In addition, asynchronous circuits have no standby power consumption because of their event-driven feature, no power consumption from the clock trees, no global clock skew, no jitter problem, and so on.

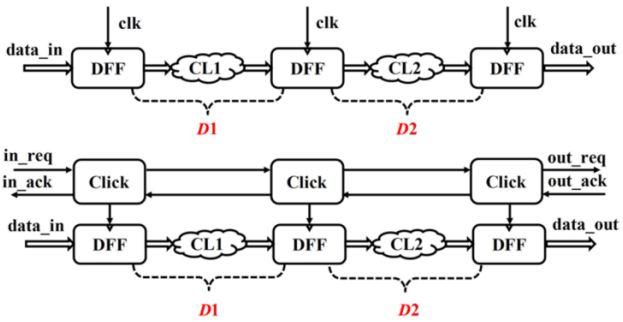

Fig. 3 Synchronous (upper one) and asynchronous (lower one) pipelines.

Table 1 Comparison of asynchronous and synchronous pipeline.

Performance	Throughput of continuous input	Delay of intermittent input
Synchronous pipeline	$1/D1$	$2\times D1$
Asynchronous pipeline	$1/D1$	$D1+D2$

Clock gating is widely used to mitigate the power drawbacks of global clock trees in synchronous circuits. However, it is generally performed at a very coarse level and requires complex control circuitry to ensure proper operation of all circuits[13]. Although the asynchronous pipeline has many advantages, it is not widely adopted because of the lack of commercial Electronic Design Automation (EDA) tool support, which makes the correct operation of the circuits hard to guarantee.

3 Design of the Asynchronous Accelerator

3.1 Architecture of the accelerator

The top-level architecture of the asynchronous CNN accelerator is shown in Fig. 4, in which the weight data are stored in the on-chip memory and the input data are transmitted to the chip via a universal asynchronous receiver/transmitter serial port. The configuration information from the controller is considered in the computation array, including six cores together with the register array and a PU, which determines the calculation mode of the CA, the direction of data flow in the register arrays for input data reuse, the activation function of Processing Element (PE) in each core, and pooling way and size. The CA contains six CEs for convolution, a PU, and a matrix multiplication for the computation of the fully connected layer. The results are stored in the output buffer.

3.2 Design of PE and PU

The basic computing modules in the CA are PEs and PU. The designed PE circuits are shown in Fig. 5, in which a three-stage asynchronous pipeline includes three Clicks connected in series. Between each stage of the control path, delay lines (not shown in Fig. 5) are inserted to match the corresponding combinational logic

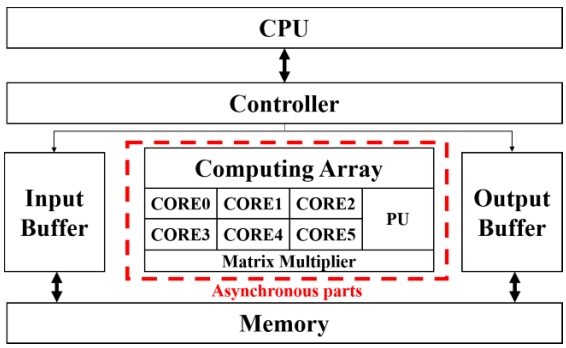

Fig. 4 Top-level architecture of the accelerator.

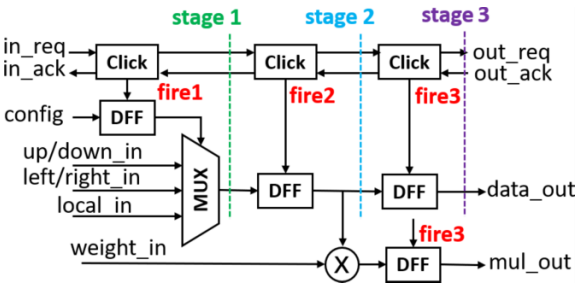

Fig. 5 Processing element circuits.

delay. In the absence of a request signal, the whole asynchronous circuits are completely turned off and have no dynamic power. Given that different stages have different delay times, the time intervals of the signal "*fires*" are also different. In stage 1, the configuration information that decides the direction of the input data is read. The multiplication of the input map and the weights is performed in stage 2. In stage 3, the input data are stored temporarily so that the neighbor PE can reuse it.

As illustrated in Fig. 6, the PU circuits comprise four Clicks, in which all the request signals from the 25 PEs (PE1-PE25) are the input of a Muller C element[17] used for completion detection in stage 4. The output of Muller C is high (low) only when all the inputs are high (low); otherwise, it remains unchanged. A 25-input Muller C is used for completion detection, that is, only when all the request signals are valid will the Muller C generate a valid signal as the request signal to the next stage, which ensures that all the multiplication results of the PEs are ready and then the PU starts to work.

As mentioned above, the activation function, the pooling method (maximum or average), and the pooling size can be dynamically modified according to the configuration information. The 25 PEs together with 1 PU form a seven-stage Click-based asynchronous pipeline, which is easy to be split into a smaller pipeline or reassemble into a larger one. In other words, we can cut off any stage in the pipeline and connect it to the control path of the other computation modules.

3.3 Design of the computing core

The 25 PEs and 1 PU, together with the 5×5 register array, form a computing core (Fig. 7), in which each PE has the function of both multiplication and data storage. As a result, each PE can receive the data from its neighbor PE from any direction (such as top, bottom, left, and right). According to the requirement of different layers or CNN models, we can configure the data path of the core in real-time. The handshake protocol instead of a global clock is adopted in the core to ensure correct data flow between PEs. Given the "event-driven" feature of the asynchronous handshake protocol, a PE will be completely turned off when it does not receive a request signal to save power. All the calculation results from multipliers in PEs are added up by the adder in stage 4, and pooling calculation is completed in stages 5 to 7.

As shown in Fig. 8, the CA, including six CEs, uses a Muller C element for completion detection. The convolution computation with a larger size for

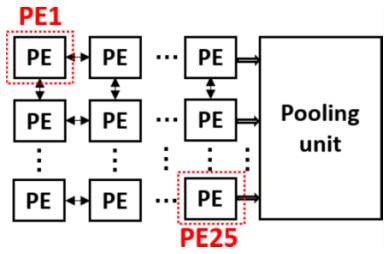

Fig. 7 Computing core.

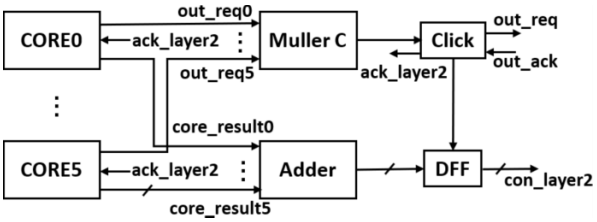

Fig. 8 Part of the schematic of computing array.

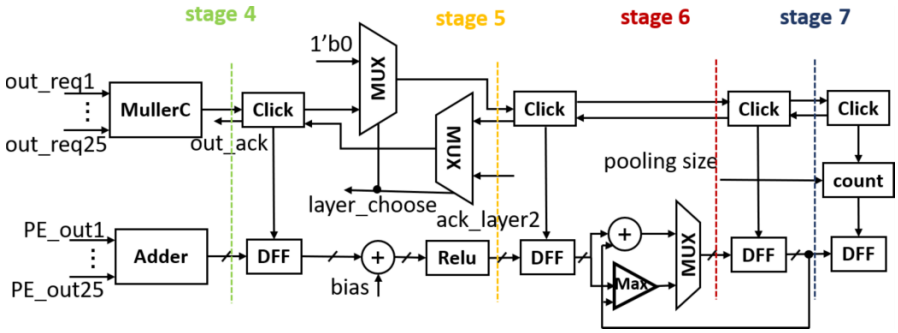

Fig. 6 Pooling unit circuits.

the deeper layer can be realized in the array if more PEs are included. If we want to perform convolution computation for the deeper layer with a larger size, we must add more PEs in the CEs and adopt a Muller C element to connect them and set up an eight-stage asynchronous pipeline. In addition, the matrix multiplication module completes the computation for the fully connected layer.

4 Chip Design Methodology

4.1 Design flow

We propose a design flow (Fig. 9) to implement Click-based asynchronous circuits. The first step is "Hardware description of Asyn. circuits", in which the Click-based asynchronous circuits are described with Verilog codes. The second step is "Synthesis with ADM", in which Design Compiler (DC) is used to synthesize the Verilog codes to generate a netlist of the circuits. During synthesis, we put forward an Adaptive Delay Matching (ADM) method. After synthesis, physical design of the circuits including "Place and Route" is implemented with Encounter Digital Implementation (EDI). Then, the Design Rule Check, Layout Versus Schematic check, and Static Timing Analysis are carried on.

4.2 Synthesis with the method of ADM

We first design the Click elements in the gate level by using the standard cells in the library and take the fire signals generated by the Click elements as the local clocks. Hence, we can adopt the command "*create_clock*" to create clocks, with which we can synthesize the Click-based asynchronous circuits with DC. The synthesis process for asynchronous circuits is different from that for synchronous circuits in the following aspects: (1) the Click element should not be modified to guarantee its correct operation once it is instantiated; (2) fire signals are treated as local clocks; (3) delay matching is needed; (4) delay lines should be inserted carefully between the Click elements to facilitate hazard-free circuit operations. The approach of creating clocks and delay matching for the asynchronous circuits is discussed in Section 4.3.

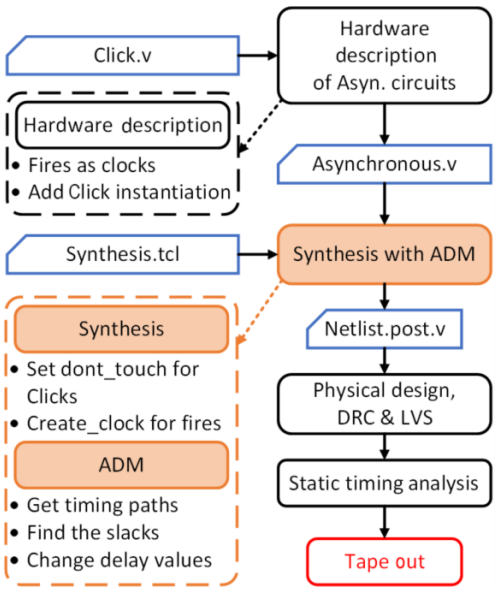

Fig. 9 Design flow for Click-based asynchronous circuits.

4.3 ADM

Delay matching is essential to meet the timing constraint for asynchronous bounded-data circuits, such as Click-based asynchronous circuits. ADM according to corresponding data paths is critical to ensure the function and high performance of the circuits. We could insert delay lines with the command "*set_min_delay*". However, the most critical step is to determine the length of the inserted delay lines. Without a global clock, DC tools cannot obtain the timing path, making the delay matching impossible. ADM is used to solve this problem, as shown in Fig. 10, with which we can perform delay matching adaptively in asynchronous circuits.

First, we generate local clocks for asynchronous circuits. In general, the DC tools consider that two clocks are synchronous if they share a common source and have a phase relationship. The fire signals in the Click-based asynchronous circuits have latency equal to the delay between the Click elements. As a result, we can treat all the fire signals as local clocks with

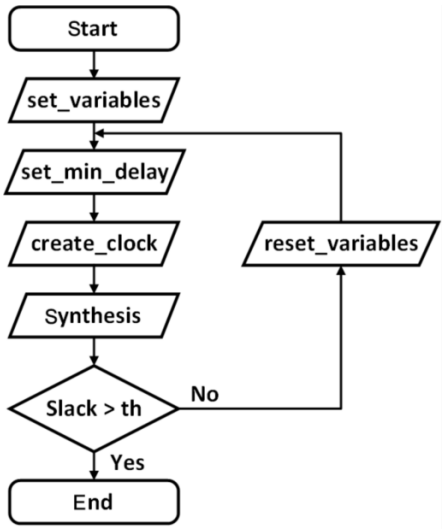

Fig. 10 Method of adaptive delay matching.

a phase relationship. Therefore, we create clocks for the fire signals and describe their relationship using the command "*create_generated_clock*". In this way, all the timing paths are checked, and the combinational logic are optimized according to the latency of the fire signals by DC. During the synthesis, the worst corner is used to perform the delay matching to guarantee timing convergence.

Second, the proposed method of ADM not only determines the timing paths in asynchronous circuits but also tries to identify the length of delay lines to be inserted between the Click elements to optimize speed. Considering that the exact delay values are undetermined before synthesis, we first create a delay variable with a small non-zero value, such as 0.1 ns, by using the command "*set_min_delay*".

Finally, we synthesize the circuits and run the command "*for_in_collection*" to obtain the slack values of each timing path from the report of command "*report_timing*" cyclically. Taking the parasitic estimation after Place and Route into account, only if all the slacks reported have a margin (noted as th) of +10% with respect to the delay of the corresponding timing paths will the synthesis be finished. If not, the delay variable of each stage will be reset to a new value, which is the delay value of the current timing path plus the threshold, and the circuit will be resynthesized. The synthesis process will be finished until the delay lines for each stage meet the timing constraint. In general, our design shows that if the initial value is larger than zero and smaller than 5 ns, then only two iterations of the synthesis process are needed.

The proposed delay matching method is fully automatic during synthesis. This ADM method for the Click-based asynchronous circuits can also be used to perform static analysis of the asynchronous circuits. For complex asynchronous pipeline structures including fork, join, split, or merge, ADM can also be used to perform delay matching with the created clocks and control circuits for different handshaking protocols[8], which controls the data flow to guarantee the correctness of circuits.

5 Chip Test Results and Discussion

The asynchronous and synchronous CNN accelerators are implemented in a TSMC 65 nm process, and the die micro photo of the chips is shown in Fig. 11, including six CEs, each of which contains 5×5 PEs,

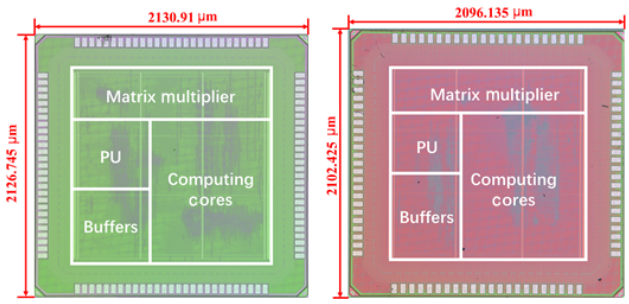

Fig. 11 Die micro photo of the asynchronous (left) and synchronous (right) CNN accelerators.

a PU, and a matrix multiplication. The area of the asynchronous accelerator is slightly larger than that of the synchronous one because of the Click elements used in the asynchronous accelerator. As mentioned above, the commercial EDA tool like DC cannot capture the timing path between each asynchronous pipeline because of the lack of a global clock in the asynchronous circuits, making delay matching for each pipeline challenging.

To avoid inserting the delay lines manually, we adopt the ADM method, with which all the timing paths in the asynchronous pipelines can be captured by DC so that the length of delay lines to be inserted between Click elements can be calculated. We use the command "*set_min_delay*" when the circuits are synthesized to insert delay lines between Click elements to match the delay of the combinational logic between each pipeline. In this way, the delay lines to be inserted in each pipeline are adaptive to its corresponding combinational logic, which saves unnecessary delay lines and power of the circuits as a result. With the ADM method, accurate static timing analysis on the asynchronous circuits can be performed by PrimeTime to ensure the correct timing of asynchronous circuits.

LeNet-5 is implemented on our accelerator, and the chip test platform is shown in Fig. 12, in which the test chip recognizes the input picture and sends the results to the Field-Programmable Gate Array (FPGA) board to display. As shown in Fig. 12, the picture of number five is correctly recognized by the test chip. Figure 13 illustrates the waveforms of the request and the fire signals from the test chip. Figure 13 shows that each transition of the request signal "*req*" triggers the Click to generate a fire signal "*fire1*" and the fire signal transfers to the next stage to generate another fire signal "*fire2*", and so on. The matched delay lines determine the time intervals of the fire signals. The test results show that the time intervals between "*fire1*" and "*fire2*", and "*fire2*"

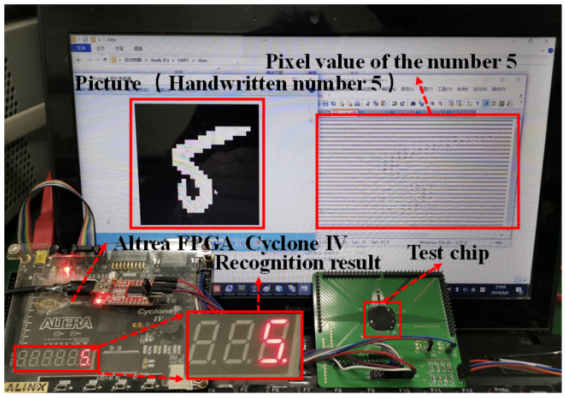

Fig. 12 Chip test platform.

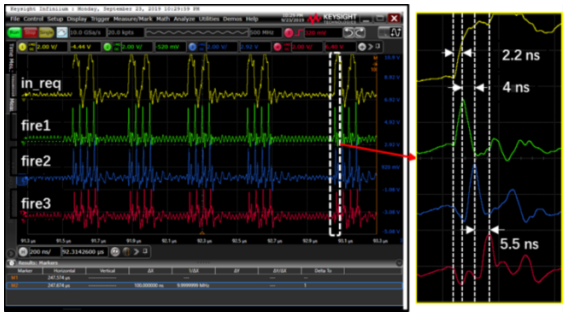

Fig. 13 Test results of the asynchronous CNN accelerator test chip.

and "*fire3*" are 4 and 5.5 ns, respectively.

The synchronous CNN accelerator chip has the same architecture with the asynchronous one and only differs in that the asynchronous circuits adopt a Click-based pipeline when designing the CA. The area and performance comparison of the two chips is shown in Table 2, from which we can see that the area of the asynchronous accelerator is almost the same as that of the synchronous one because the area of Click elements only accounts for 1.65% of the total area, which means that the Click elements only consume a small amount of static power. The peak performance of the asynchronous accelerator is 60.9 GOPS at 150 MHz, which is 1.25 times higher than that of the synchronous one. The number of clock tree buffers of the asynchronous accelerator is 1076, which is 84.7% of the synchronous one because the asynchronous CA saves 194 clock tree buffers, saving 11% power in the asynchronous accelerator compared with synchronous one.

Table 2 Test results of asynchronous and synchronous accelerators chips.

	Asynchronous	Synchronous
Area (mm^2)	1.96×1.96	1.93×1.93
Click element gate	30 177	0
Clock tree buffers	1076	1270
Performance (GOPS)	60.9@150 MHz	48.4@120 MHz
static power (mW)	8.0	7.76
Standby power (mW)	24.0@ (0.8 V, 100 MHz)	28.8@ (0.8 V, 100 MHz)
Power in working mode (mW)	26.4@ (0.8 V, 100 MHz)	29.6@ (0.8 V, 100 MHz)
Energy efficiency (TOPS/W)	1.538	1.37

To analyze further the power of CA in the synchronous and asynchronous accelerators, we test the chips in two modes. One is the standby mode, in which the clock is turned on, and the circuits wait for the enable signal to perform computation. The clock power accounts for about 70% of the total power of both chips in this mode, and clock buffers in the memory consume most of the clock power. Therefore, in the standby mode, the tested power consists of static power and clock power for both chips. In the working mode, the tested power of the synchronous accelerator power is composed of static power and dynamic power, including the clock power and dynamic power of the combinational logic in CA. In addition, the dynamic power of the Click element should be considered when calculating the dynamic power of the asynchronous accelerator. According to the tested static power, standby power, and working power, the power of CA can be calculated roughly based on the proportion of the area of CA to the total area of the chip and the proportion of the number of clock buffers in CA to the total number of clock buffers.

The comparison of power consumption between asynchronous and synchronous CAs is illustrated in Fig. 14. Figure 14 shows that the asynchronous CA saves 3.49 mW clock power at the cost of consuming 1.6 mW dynamic power of Click elements and that the

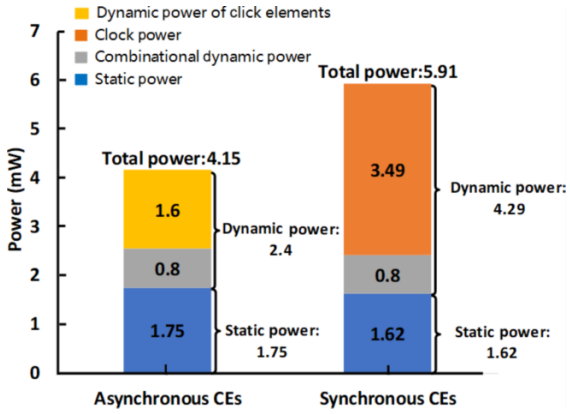

Fig. 14 Comparison of asynchronous and synchronous computing array. (Data in Fig. 14 is calculated based on the test results of the chips.)

dynamic power of combinational logic (both are 0.8 mW) is considerably less than the clock power of 3.49 mW. The static power of the asynchronous CA (1.75 mW) is only 0.13 mW greater than that of the synchronous one (1.62 mW). As a result, the asynchronous CA saves 30% power compared with the synchronous CA. Therefore, when the clock power accounts for a high percentage of the total power and the control circuits are complex with clock gating in some applications, asynchronous circuits are a promising alternative to design low-power circuits with a significant reduction in dynamic power consumption.

6 Conclusion and Future Work

We implement a reconfigurable asynchronous CNN accelerator with a TSMC 65 nm CMOS process that consumes 11% less total power than that of the synchronous one and an energy efficiency reaching 1.538 TOPS/W. The performance of the asynchronous accelerator can achieve 60.9 GOPS, which is 1.25 times than that of the synchronous one. In the computing cores, asynchronous circuits consume 30% less power than synchronous ones. We also propose a design flow to adopt a commercial EDA tool for asynchronous circuit design. This work proves that asynchronous circuits are an alternative to design low-power and high-energy-efficiency circuits. Our proposed design flow for asynchronous circuits is also verified.

Future works will include ultra-low-power-asynchronous circuits for the SNN accelerator, which is a biologically plausible neuronal model that uses sparse spikes to transmit information. The sparsity of spikes and the event-driven characteristics of asynchronous circuits allow the realization of ultra-low-power and high-energy-efficiency SNN accelerators.

Acknowledgment

This work was supported by National Science and Technology Major Project from Minister of Science and Technology, China (No. 2018AAA0103100) and the National Natural Science Foundation of China (No. 61674090), partly supported by Beijing National Research Center for Information Science and Technology (No. 042003266), and Beijing Engineering Research Center (No. BG0149).

References

[1] S. X. Zheng, P. Ouyang, D. D. Song, L. D. Liu, S. J. Wei and S. Y. Yin, An ultra-Low power binarized convolutional neural network-based speech recognition processor with on-chip self-learning, *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 12, pp. 4648–4661, 2019.

[2] S. Schneider, A. Baevski, R. Collobert, and M. Auli, wav2vec: Unsupervised pre-training for speech recognition, arXiv preprint arXiv: 1904.05862, 2019.

[3] A. Howard, M. Sandler, G. Chu, L. C. Chen, B. Chen, M. X. Tan, W. J. Wang, Y. K. Zhu, R. M. Pang, V. Vasudevan, et al., Searching for MobileNetV3, arXiv preprint arXiv: 1905.02244, 2019.

[4] Y. H. Chen, T. J. Yang, J. Emer, and V. Sze, Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices, *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 2, pp. 292–308, 2019.

[5] J. Song, Y. Cho, J. S. Park, J. W. Jang, S. Lee, J. H. Song, J. G. Lee, I. Kang, An 11.5 TOPS/W 1024-MAC butterfly structure dual-core sparsity-aware neural processing unit in 8 nm flagship mobile SoC, in *Proc. IEEE Int. Solid-State Circuits Conf.*, San Francisco, CA, USA, 2019, pp. 130–132.

[6] J. Lee, J. Lee, D. Han, J. Lee, G. Park, and H. J. Yoo, LNPU: A 25.3 TFLOPS/W sparse deep-neural-network learning processor with fine-grained mixed precision of FP8-FP16, in *Proc. IEEE Int. Solid-State Circuits Conf.*, San Francisco, CA, USA, 2019, pp. 142–144.

[7] J. Sparsø and S. Furber, *Principles of Asynchronous Circuit Design: A Systems Perspective*. Boston, MA, USA: Springer, 2001, pp. 3–11.

[8] P. A. Beerel, R. O. Ozdag, and M. Ferretti, *A Designer's Guide to Asynchronous VLSI*. Cambridge, UK: Cambridge University Press, 2010, pp. 7–9.

[9] H. van Gageldonk, K. van Berkel, A. Peeters, D. Baumann, D. Gloor, and G. Stegmann, An asynchronous low-power 80C51 microcontroller, in *Proc. 4th Int. Symp. Advanced Research in Asynchronous Circuits and Systems*, San Diego, CA, USA, 1998, pp. 96–107.

[10] P. A. Beerel and M. E. Roncken, Low power and energy efficient asynchronous design, *Journal of Low Power Electronics*, vol. 3, no. 3, pp. 234–253, 2007.

[11] I. E. Sutherland, Micropipelines, *Communications of the ACM*, vol. 32, no. 6, pp. 720–738, 1989.

[12] A. Steininger, V. S. Veeravalli, D. Alexandrescu, E. Costenaro, and L. Anghel, Exploring the state dependent SET sensitivity of asynchronous logic – The muller-pipeline example, in *Proc. 32nd Int. Conf. Computer Design (ICCD)*, Seoul, South Korea, 2014, pp. 61–67.

[13] F. Akopyan, J. Sawada, A. Cassidy, R. Alvarez-Icaza, J. Arthur, P. Merolla, N. Imam, Y. Nakamura, P. Datta, G. J. Nam, et al., TrueNorth: Design and tool flow of a 65mW 1 million neuron programmable neurosynaptic chip, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 10, pp. 1537–1557, 2015.

[14] M. Davies, N. Srinivasa, T. H. Lin, G. Chinya, Y. Q. 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.

[15] W. J. Chen, H. Wu, S. J. Wei, A. P. He, and H. Chen, An asynchronous energy-efficient CNN accelerator with reconfigurable architecture, in *Proc. IEEE Asian Solid-State Circuits Conf. (A-SSCC)*, Tainan, China, 2018, pp. 51–54.

[16] A. Peeters, F. te Beest, M. de Wit, and W. Mallon, Click elements: An implementation style for data-driven compilation, in *Proc. IEEE Symp. Asynchronous Circuits and Systems*, Grenoble, France, 2010, pp. 3–14.

[17] D. E. Muller, Theory of asynchronous circuits, http://hdl.handle.net/2027/uiuo.ark:/13960/t7pp0n320.

Jilin Zhang received the BS degree from Lanzhou University, China in 2019. He is pursuing the master degree in microelectronics at Tsinghua University, China. His research interests include asynchronous circuit design and spiking neural network.

Hui Wu received the MS degree from Tsinghua University, Beijing, China in 2020. His current research interests include asynchronous low power circuit design and electronic design automation methodology.

Weijia Chen received the BS degree from Chongqing University, China in 2016, and the MS degree from Tsinghua University in 2019. His research interests include asynchronous circuit design and neural network processor.

Shaojun Wei received the PhD degree from the Faculte Polytechnique de Mons, Mons, Belgium, in 1991. He is a professor at the Institute of Microelectronics, Tsinghua University, Beijing. His current research interests include the VLSI SoC design, EDA methodology, and communication ASIC design. He is a fellow of the IEEE.

Hong Chen received the PhD degree from the Department of Electronic Engineering from Tsinghua University in 2005. From 2005 to 2007, she worked at the Institute of Microelectronics in Tsinghua University (IMETU) as a post-doctoral fellow. Since 2007, she has been working with IMETU, and currently she is an associate professor. Her research interests include monitoring-system design for TKR/THR surgery, low-power digital integrated-circuit design, asynchronous circuit design, PZT power electronics, and low-power mixed-signal SoC design.