A Dynamic Timing Enhanced DNN Accelerator With Compute-Adaptive Elastic Clock Chain Technique

Tianyu Jia, *Member, IEEE*, Yuhao Ju, *Student Member, IEEE*, and Jie Gu, *Senior Member, IEEE*

***Abstract*— This article presents a deep neural network (DNN) accelerator using an adaptive clocking technique (i.e., elastic clock chain) to exploit the dynamic timing margin for the 2-D processing element (PE) array-based DNN accelerator. To address two major challenges on exploiting dynamic timing margin for modern deep learning accelerators (i.e., diminishing dynamic timing margin on a large array and strong timing dependence on runtime operands), in this work, we proposed an elastic clock chain scheme to provide a flexible multi-domain clock management scheme for *in situ* compute adaptability. More specifically, a total of 16 clock domains have been created for the 2-D PE array with the clock periods dynamically adjusted based on both runtime instructions and operands. The multi-domain clock sources are generated from a multi-phase delay-locked loop (DLL) and delivered by a global clock bus. The clock offsets between neighboring domains are deliberately managed to maintain the synchronization among clock domains. A 16 × 8 PE array that supports different DNN dataflows and bit-precisions was fabricated using a 65-nm CMOS process. The measurement results on MNIST and CIFAR-10 data sets showed that the effective operating frequency was improved by up to 19% for a single instruction multiple data (SIMD) data flow by enabling the operation of the proposed elastic clock chain. The performance improvement was converted into up to 34% energy saving. Compared with SIMD data flow, the systolic dataflow shows reduced performance improvement of up to 11% due to the consideration of all in-flight operand values.**

***Index Terms*— Adaptive clocking, deep neural network (DNN) accelerator, dynamic timing margin, multiple clock domains, processing element (PE), systolic array.**

I. Introduction

IN RECENT years, machine learning-related computing tasks, especially those using deep neural networks (DNNs), have attracted huge interests from both software and hardware communities due to the algorithm's groundbreaking accuracy achievement on many applications, such as image or voice recognition, natural language processing, etc. The ubiquitous application of DNN and the intensive computation requirement strongly stimulate the design of domain-specific application-specified integrated circuit (ASIC) DNN accelerators [1]–[4]. To improve the energy efficiency of DNN accelerators, many design approaches have been explored at architecture or micro-architecture level, such as flexible dataflows [1], bit-precision [2], [3], quantization [4], etc. However, there have been very few explorations of adaptive techniques for specialized DNN accelerators yet. In this work, an adaptive clocking scheme with an elastic clock chain is developed to exploit the dynamic timing margin during runtime and improve the performance for a 2-D array-based DNN accelerator architecture.

Previously, adaptive techniques have been widely studied and adopted for microprocessors to improve the energy efficiency of the digital system [5]–[8]. In the conventional microprocessor design, each microprocessor core requires significant frequency or voltage guardband margin to secure the chip's operations in the presence of various sources of variations such as process, voltage, and temperature (PVT), aging, and jitter, etc. However, these guardbands are determined based on the worst-case variations leading to degradation of processors' performance and energy efficiency in normal operations. As a result, many adaptive techniques have been developed for processors to reduce this guardband. For example, adaptive clocking schemes have been developed to dynamically adjust the clock period to mitigate voltage droop impact [5], [6]. Fully integrated voltage regulators have been leveraged to adaptively scale the supply voltage levels to reduce the voltage guardband [7], [8]. In these adaptive techniques, the voltage droop events are monitored to guide the regulations of clocks or voltages.

Recently, a few adaptive techniques have been explored on DNN accelerators. In [9], the globally asynchronous locally synchronous (GALS) adaptive clocking has been implemented on a scalable processing element (PE) array-based DNN accelerator [10]. Each PE partition contains its clock domain provided by a local adaptive clock generator. As a result, the droop protection guardband of each PE can be reduced leading to promising performance improvement. However, this work only focuses on mitigating the impact of voltage droop and does not exploit the finer-grained dynamic timing slack (DTS). In [11], the error resiliency feature of DNN has been exploited on a 1-D single instruction multiple data (SIMD) DNN accelerator by utilizing Razor flip-flops. An adaptive

Manuscript received April 25, 2020; revised July 8, 2020 and September 2, 2020; accepted September 27, 2020. Date of publication October 13, 2020; date of current version December 24, 2020. This article was approved by Guest Editor Ping-Hsuan Hsieh. This work was supported in part by the National Science Foundation under Grant CCF-1618065. *(Corresponding author: Tianyu Jia.)*

Tianyu Jia was with the Department of Electrical and Computer Engineering, Northwestern University, Evanston, IL 60208 USA. He is now with the School of Engineering and Applied Sciences, Harvard University, Cambridge, MA 02138 USA (e-mail: tjia@seas.harvard.edu).

Yuhao Ju and Jie Gu are with the Department of Electrical and Computer Engineering, Northwestern University, Evanston, IL 60208 USA.

Color versions of one or more of the figures in this article are available online at https://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSSC.2020.3027953

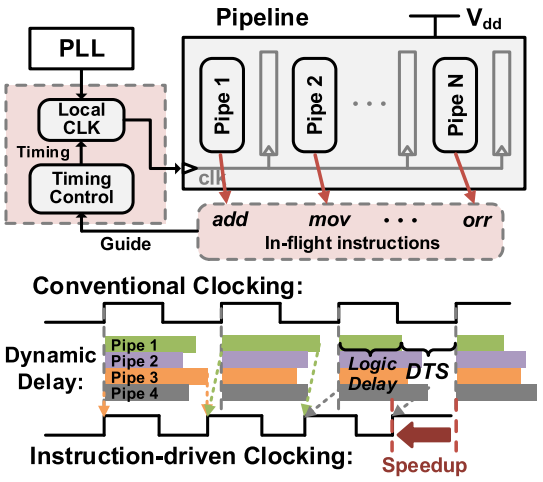


Fig. 1. Instruction-driven adaptive clocking scheme for a pipelined microprocessor [16]–[18].

clocking scheme is further developed to manage the clock period when the timing errors are detected to improve energy efficiency [12]. However, the above schemes are only implemented for a group of 1-D vector MAC units and will be challenging to be applied for a more commonly used 2-D PE array-based architecture.

Exploiting runtime dynamic timing margin brings additional performance and energy gains beyond the droop-based adaptive schemes. In typical digital circuits, the critical paths with the longest logic delay are not always excited during the program's runtime. The timing margin existed between the actual cycle-by-cycle path delay and the worst-case clock period is referred to as "DTS" [13]. In the past, the DTS margin has been exploited by instruction-driven adaptive clocking schemes for simple in-order CPUs [14], [15] or a more complicated GPGPU architecture [16]. As the example illustrated in Fig. 1, all the in-flight instructions in the pipeline stages were monitored during runtime to guide the clock period adjustment. For each clock cycle, the dynamic delays from all pipeline stages were analyzed in advance. The longest dynamic delay among all pipeline stages was identified as the target clock period to guide the dynamic clock period adjustment. As the clock period can be dynamically scaled cycle-by-cycle, a notable performance speedup was obtained on the benchmark programs [14]–[16].

Although some promising results have been reported to exploit DTS for microprocessors, it is still very challenging to apply similar adaptive clocking technique for DNN accelerator, especially the 2-D array-based accelerator architecture. There are two major challenges to exploit DTS inside the accelerator PE array. First, the DTS benefit becomes very small after considering the worst-case dynamic timing within the entire 2-D PE array. Second, there is a lack of runtime instructions for the accelerator to guide the cycle-by-cycle clock management. In this work, we developed a compute-adaptive elastic clock chain technique, which utilizes a multi-domain clocking scheme to exploit DTS for a PE array-based DNN accelerator. In addition to the runtime instructions, the real-time operand values are also utilized to guide the clock

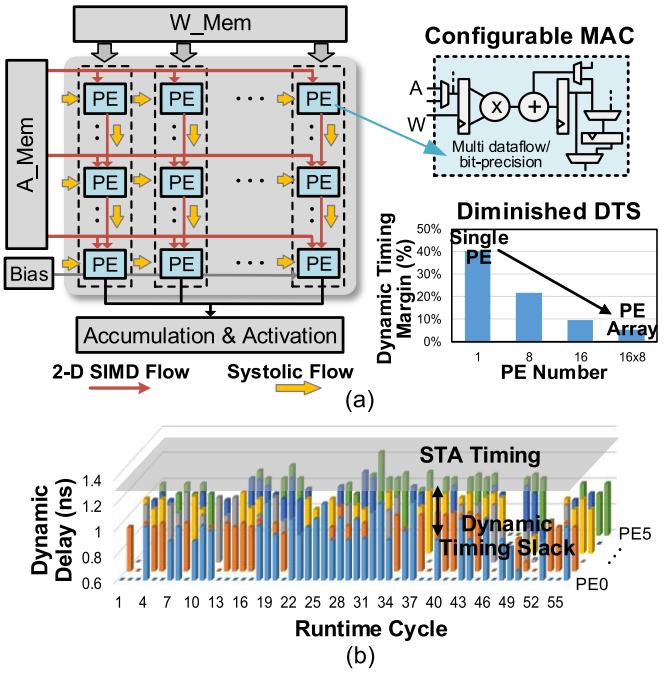


Fig. 2. (a) 2-D PE array architecture used in this work with diminished DTS margin. (b) Dynamic timings of six PEs during MNIST inference runtime.

management. The experiments on a 65-nm CMOS silicon test chip showed that up to 19% performance gain or 34% energy saving can be obtained for DNN inference using MNIST and CIFAR-10 data set [18].

The rest of this article is organized as follows. The DTS margin for a PE array is analyzed in Section II, which also illustrates the design challenges. The overview of the developed multi-domain clocking scheme for PE array architecture is presented in Section III. Section IV introduces the compute-adaptive timing analysis and the clock management strategy. The synchronization policy for the elastic clock chain is presented in Section V. The implementation and measurement results obtained from the test chip are shown in Section VI, followed by conclusions in Section VII.

II. Dynamic Timing Study in PE Array

To evaluate the dynamic timing behaviors inside the PE array during DNN operation, a 2-D PE array with 16 rows and 8 columns is built, as shown in Fig. 2(a). Each PE is a reconfigurable MAC unit, which can support various precision including INT1/4/8. The PE array has two operation configurations for different dataflow mappings. First, each column can independently process one output channel value of the neural network in an output stationary fashion, which is the same as the 2-D vector SIMD architecture in [4]. Second, the data path of the PE array can also be configured as a systolic array with the input activation (ACT) values shared across each row as in [17]. These two representative dataflows are implemented in the same PE array with very small reconfiguration overhead. The DTS exploitation benefit for both configurations will be demonstrated later in Section VI.

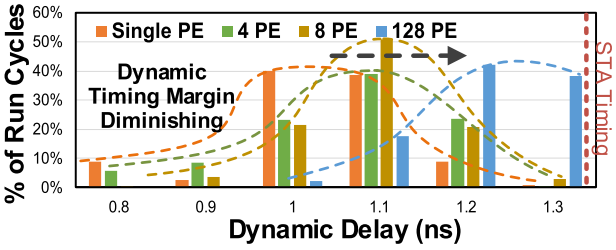


Fig. 3. Histograms of the worst-case dynamic timing considering different numbers of PEs.

The gate-level netlist of the PE unit is generated after synthesis and place and route using industry-standard EDA tools. The speed of the PE unit is closed at 700 MHz. To simulate the dynamic timing behavior for every PE, all 128 PE units in the array are using gate-level netlist with back-annotated timing from *RC* extraction. Fig. 2(b) shows the simulated runtime dynamic timings of six individual PE units under 2-D SIMD INT8 operation mode for the MNIST data set inference. During the simulation, the data toggling time of every flip-flop inside the PE unit is monitored for every clock cycle, with the last data transition time recorded as the dynamic timing of the PE. It is observed that there is a large variation of the dynamic timing for a single PE unit cycle-by-cycle. As analyzed later in Section IV, this dynamic timing variation is highly related to the operand value changes and different configuration modes. Besides, the longest critical paths are rarely exercised during runtime leading to a significant DTS margin between the dynamic timings and the worst-case static timing analysis (STA) timing. However, every PE unit shows different runtime dynamic timing patterns due to the difference of real-time operands (i.e., weights and input values). To apply prior cycle-level adaptive clocking to PE array [15], [16], the worst-case dynamic timing among all PE units needs to be considered to guide the clock adjustment every cycle.

Fig. 3 shows the distribution histogram of dynamic timing considering the worst-case dynamic timing among different number of PE units for the MNIST inference task. Within a single PE, the dynamic timings of a majority of run cycles are only around 1 ns, which is much shorter than the STA reported clock period of 1.43 ns. Considering the longest dynamic timing among four or eight PEs every cycle, the dynamic timing cluster shifts toward STA timing while still centralized around 1.1 ns. However, if considering the worst-case dynamic timing among the entire PE array, the majority of run cycles have dynamic timings very close to STA timing, e.g., around 1.3 ns. As every PE is processing different operand values, each PE has a different dynamic timing at any given cycle. The diminishing DTS margin indicates there is at least one PE unit within the array exercising its critical path and dominating the DTS margins for other PE units. Fig. 2(a) shows the diminished DTS margin. It is observed that the DTS margin diminishes with the size of PE array increasing (i.e., reducing from 40% to only 4% when the number of PEs increases from 1 to 128).

As DTS almost disappears when considering the worst-case dynamic timing for the entire 2-D PE array, the previous

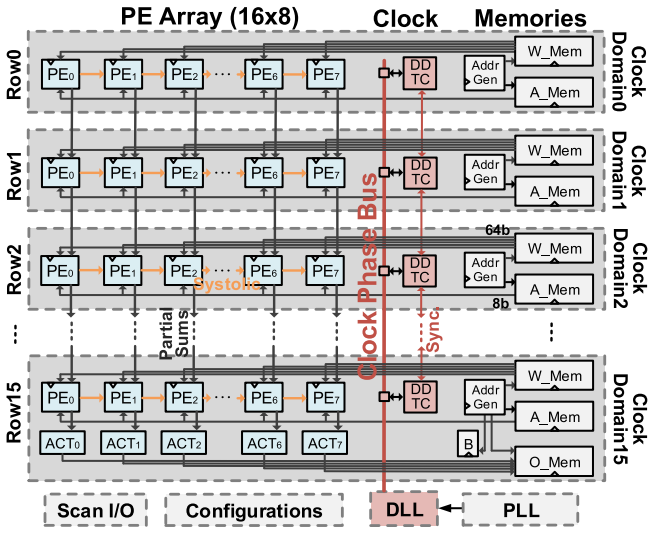


Fig. 4. Overview of the multi-domain clocking design on the 2-D PE array.

centralized DTS adaptive techniques [14]–[16] cannot exploit the dynamic timing margin effectively for a large 2-D PE array. For the pipelined microprocessor with a few pipeline stages [14]–[16], it is feasible to capture all the in-flight instructions. The dynamic timing for each instruction can be analyzed using STA in advance to guide the clock adjustment. For the PE array-based accelerator, it is more challenging to perform case timing analysis as the diversity of input activity conditions. As illustrated in Fig. 3, the maximum DTS margin can only be obtained by assigning a different clock domain for each PE. However, this will require large design overhead and complicated synchronization scheme. Therefore, we chose a compromised design choice (i.e., each PE row has one clock domain). Each row of eight PEs contains their clock domains with a local manager module to guide clock period adjustment. For the PE array with different sizes, the proposed elastic clock chain scheme is scalable with both the row number and PE number in a row. Although the row numbers can scale freely in our scheme, the DTS benefits will degrade with the increase of PE numbers in a clock domain, especially for the systolic array dataflow. Therefore, we suggest to limit the PE numbers within a clock domain to be 8–16 in practice. To fit the reality of fewer instructions in accelerator design, the clock management is guided by both runtime instructions and real-time operand values to exploit the cycle-level DTS margin.

III. Multi-Domain Adaptive Clocking Scheme

A. Design Overview

In this work, we developed an adaptive clocking scheme with multiple clock domains to exploit the runtime DTS in PE array-based DNN accelerator. Fig. 4 shows the overview of the DNN accelerator architecture and clocking scheme designed for it. As introduced before, the data path of the PE array can be configured into two operation modes. For the 2-D SIMD configuration, the input ACT value is provided from the A_Mem with 8-bit width and shared among all PEs in the same row. The weight memory W_Mem has a wider

64-bit width to support simultaneous eight weights fetched from different filter channels. The partial sum results inside each PE are propagated vertically along with columns and eventually accumulated by the accumulator on the bottom of the array, which is the same as design [4]. In this operation mode, each PE column is independently processing different output channel values, with no data transfer across columns. The PE array is also able to be configured as a systolic array, in which the input values are reused horizontally from PE_0 to PE_7 in each row. The multiplication values are accumulated immediately by the PE unit one row below, which is the same as [17]. There is an address generator module in each row which manages SRAM address based on the operation configurations.

To exploit the DTS benefit, each row of eight PE units with their input/weight SRAMs are clocked by a different clock domain. Therefore, there is a total of 16 clock domains in this 16 × 8 array. The SRAMs are only accessed by the local PEs within the same clock domain. To accommodate the adaptive clocking technique, the interface of SRAM is timing closed based on the fastest adaptive clock frequency. The PE array generates the accumulation results at the bottom row and stores the results to output memory O_Mem. The accumulation and ACT modules on the bottom of the array and the output memory are clocked by clock domain 15. An all-digital PLL is designed to feed the root clock to a delay-locked loop (DLL), which generates a clock phase bus sent to all the clock domains. Inside each clock domain, a data detection and timing controller (DDTC) module is designed to dynamically select clock phases as a local clock based on the runtime compute operands. Therefore, each clock domain could adjust the clock period cycle-by-cycle to exploit the DTS margin. Besides, the DDTC module also manages the synchronizations through sync data paths across clock domains. The maximum clock phase offset between neighbor domains is constrained to guarantee the timing correctness of data paths crossing domains, such as partial sum propagations.

B. Clock Phase Bus Design

Fig. 5 shows the design details of the clock phase bus design. The architecture of DLL design in this work is similar to previous work [19], in which a simple up/down counter is used to tune the loading capacitance at every delay stage. DLL is locked at the same frequency as PLL. The DLL delay line is designed to provide 28 equally delayed phases of clock edges. The identical amount of load capacitance is inserted at each delay line stage, with careful layout optimization to minimize the delay mismatch. Each delay stage generates one clock phase, with a phase step about 50 ps. These 28 clock phases are sent to all 16 clock domains through a global clock phase bus, which travels a total distance of 1.5 mm. To minimize the clock skew between different domains, the DLL is placed in the center location of the floorplan, with eight PE rows at both top and bottom sides.

The 28 clock phases (i.e., p_0–p_{27}) travel from DLL delay line to phase selection multiplexers in all clock domains using high-level metal layers. As shown in Fig. 5(b), the routing

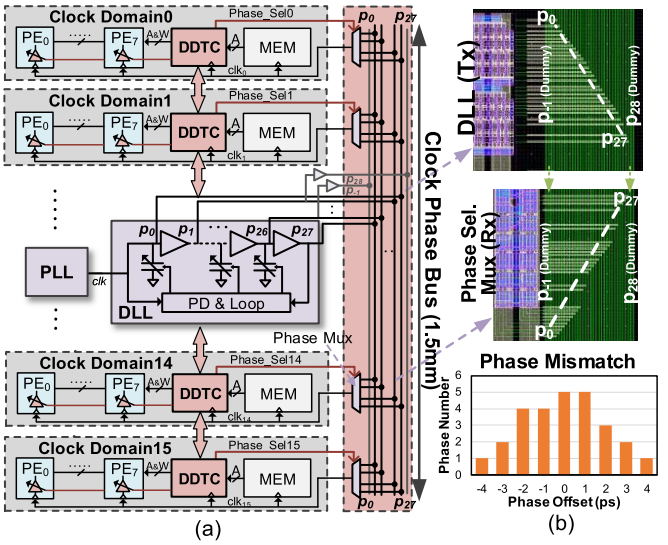


Fig. 5. (a) Design details of the multi-phase clock bus design. (b) Layout optimizations for the clock phase routings.

layout for each clock phase has been carefully optimized with similar lengths and loadings to reduce phase skew. In addition, there are two additional clock phases p_{-1} and p_{28}, which is one phase step earlier and later than phase p_0 and p_{27}, respectively, intentionally inserted at the clock bus boundaries. These two dummy phases are also generated from the DLL delay line and aim to provide a uniform and clean routing environment for the clock bus. With these layout optimizations, the skew mismatch between neighbor clock phases has been limited within 4 ps. The long clock routing distance leads an 18-ps static mismatch from end to end domains, e.g., between clock domain 0 and clock domain 7. However, this static mismatch across the long clock trace is not critical because each clock domain only needs to synchronize with its neighboring domains.

The DDTC module is the clock manager inside each clock domain. Based on runtime operand values and the instruction configurations, DDTC will update its clock phase selection every cycle. One clock phase is chosen from the clock bus through a glitch-free phase-selection mux as the local clocking. As shown in Fig. 6, all 28 clock phases are connected to the phase selection mux in every clock domain. The key feature of this phase selection mux is to support fast clock phase adjustment within one cycle. In our design, only one NMOS switch is turned on at a time to pass through the selected clock phase. The pull-up PMOS is designed using a small size transistor, with NMOS 4× wider. To avoid the clock glitch during the phase selection, each clock phase needs to pass through a negative edge-triggered clock gating cell. The duration of the high phase of the clock remains the same in every cycle for the simplicity of implementation, while the low phase is reduced by the dynamic phase selection. As all the logics and SRAMs in our design are triggered by the positive edge, the unbalanced clock duty cycle will not impact the normal operations. Other alternative solution for phase selection mux design (e.g., stacking PMOSs) may lead

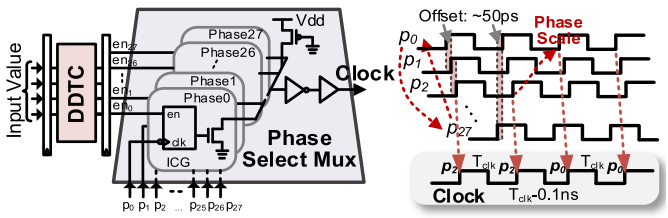


Fig. 6. Glitch-free phase selection mux and the clock adjustment waveform.

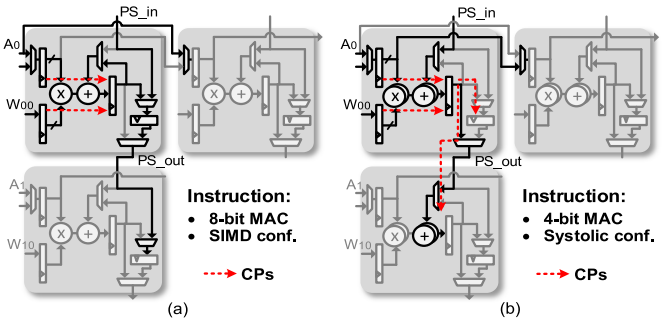


Fig. 7. PE unit design and data path connections, and the PE critical paths under (a) 2-D SIMD INT8 and (b) systolic INT4 modes.

to significant delay and is not adopted here. The slew rates of rising and falling edge inside mux are balanced after passing through the following clock buffers. Compared with the total power, the phase selection mux power is negligible, but it supports fast dynamic clock phase selection for our special requirement.

The phase selection follows a rotary manner (i.e., selecting an earlier phase to shrink the output clock period). DDTC takes one clock cycle response time for data detection and timing decision, which is equivalently one pipeline stage inserted in the data path. Therefore, the DDTC only introduces one clock cycle latency for the entire matrix multiplication operation, which is negligible. In addition, the DDTC will not impact the throughput of the PE array.

IV. Compute-Adaptive Clocking Management

A. PE Critical Path Study

Fig. 7 shows the design details of the PE unit and the data path connections between neighbor PEs. In our design, each PE unit can be configured to perform one 8-bit MAC, two 4-bit MACs, or eight simultaneous 1-bit MACs every cycle. There are three additional multiplexers added at the input values port A, the partial sum input port PS_in and output port PS_out to support both 2-D SIMD and systolic array configuration. During 2-D SIMD configuration, each PE accumulates partial sums and propagate the results to the bottom accumulator. The input ACT value is shared among all PEs in the same row. In the systolic configuration, each PE unit accumulates the MAC results from the PE unit at one row above. The input values are propagated horizontally from PE_0 to PE_7 with eight different input values in-flight simultaneously in each row. The design overhead for supporting these two data path

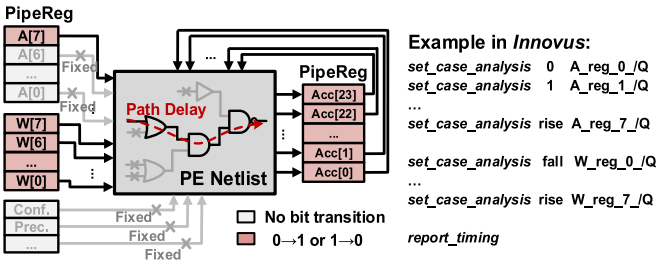


Fig. 8. Case STA method to find out the worst-case path delay under certain input conditions.

configurations is only three multiplexers, which is very small and will not impact the design timing closure.

To understand the dynamic timing characteristics inside the PE unit, we first utilize the conventional STA method to find out the critical paths with long delay under different configurations. As the dash paths shown in Fig. 7, the critical paths vary based on instructions (configurations) being issued. During the 8-bit precision mode, the longest paths are dominated by MAC operations in the first pipeline stage. However, when switching to 4- or 1-bit precision, both the MAC and data paths at the second pipeline stage become critical paths. In addition, the partial sum data paths crossing PE unit are also identified to have long delays after considering the speed offset between neighbor clock domains, which will be further explained in Section V. Therefore, it is observed that critical paths vary at different PE locations under different instruction configurations. To exploit this instruction based dynamic timing variation, a timing borrowing scheme is designed inside PE to dynamically rebalance the pipeline timing.

B. Case Static Timing Analysis

To discover the dynamic timing dependence on the runtime operands, a case-based STA method from the commercial EDA tool is leveraged. As shown in Fig. 8, we first define the transition conditions for all input registers inside PE final gate-level netlist. Based on bit transition conditions during operand update, the register activities of input value A, weight value W, and accumulated value Acc are constrained to have either certain fixed values or rising/falling transitions. In addition, to define operation configurations, the configuration registers are also set accordingly for proper dataflow and bit precision. The register transition activities can be constrained using the "set_case_analysis" command in commercial EDA tools. For example, we can use "set_case_analysis rise A_reg_0_/Q" to define bit $A[0]$ to have a rising transition.

In the conventional STA timing closure, the longest data path is closed timing with assuming both input $A[7:0]$ and $W[7:0]$ have transition activities (e.g., transit from $0 \times$ FF to 0×00, or vice versa). After defining input register activities, the worst-case path delay is reported under the designated transitioning conditions using the STA timing method. It is worth mentioning that we only focus on the dynamic timing dependence study for the input value A, assuming all the bits of weight value W and accumulation value Acc can have transitions. This is because the input value is shared among

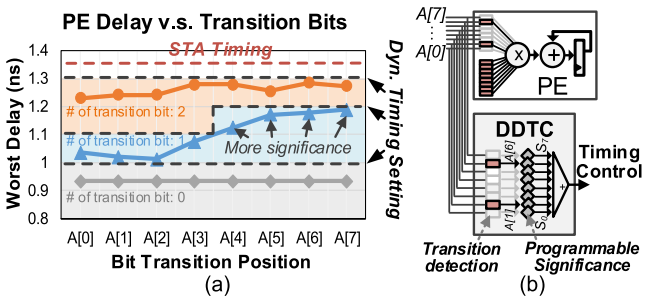


Fig. 9. (a) Dynamic timing results under different operand transition conditions. (b) Dynamic timing control mechanism.

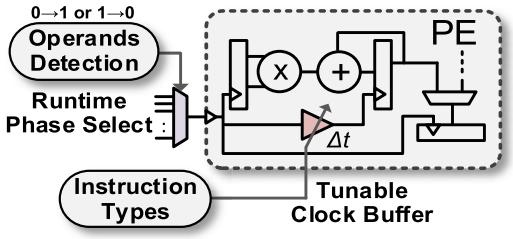


Fig. 10. Compute-adaptive clock management strategy for PEs.

all PE units in the same row in SIMD configuration, which is the same as other accelerator designs [4], [9]. It is hence easier for the adaptive clocking management to only detect A value. In the dynamic timing analysis, since we made no case timing constraint for W bits, our analysis always captures the worst-case W activity. For the systolic array configuration, the clock period is determined by the longest dynamic timing among eight in-flight A values in the row.

Fig. 9(a) shows the reported dynamic timings under different input A transition conditions. Significant dynamic timing dependencies on both the number of transitioning bits and the transitioning bits' positions are observed. As mentioned previously, the PE timing is closed at 1.43 ns by the STA method. If there is no bit flip of A value (i.e., A maintain the same value within two consecutive cycles), the worst-case path delay is only around 0.9 ns, which is much shorter than the STA timing. In fact, due to the sparse characteristics of the neural network, a significant amount of run cycles has no A bit transition as A value is consistent to be 0. If there is a single bit flip in A, the dynamic timing becomes longer to between 1 and 1.2 ns. In addition, the bit transition positions also contribute differently to the dynamic timing. In our PE design, the toggling of MSB bits will lead to longer path delay than LSB bits. When considering the cases with more than two bits' transition, the various combinations between different bit flip positions make the exploration of worst-case dynamic timing harder. Fig. 9(a) also shows the worst-case delay for two bits of A being transitioned, e.g., the worst-case delay with both $A[0]$ and $A[x]$ (x could be 1–7) transitioning. Under this condition, the worst-case dynamic timing becomes around 1.3 ns, which is close to the STA timing.

To exploit the dynamic timing dependence on the operand values, the transition activity of input A values is detected inside the DDTC module to guide the fine-grained clock management. Fig. 9 shows the dynamic timing control settings considering both the transition bit number and bit position. The difference between dynamic timing levels is set to be 0.1 ns, which is the delay of two DLL delay stages. We choose the 0.1-ns interval to reduce the complexity of dynamic timing selection and also add a protection margin for the cycle-to-cycle jitter. As there are only a few timing levels, we adopt the weighted sum approach to represent the dynamic timing levels. Each bit of transition has a programmable significance, which is a 3-bit value. During the elastic clock chain mode, the significance values for all transitioning bits are summed up to guide the dynamic clock management. Hence, both the transition bit number and bit position significance are leveraged to define proper clock adjustment levels.

C. Compute-Adaptive Clocking Strategy

Based on the previous dynamic timing studies, it is observed that the PE dynamic timing is based on both instruction configurations and the runtime operand value transitioning. As shown in Fig. 10, to exploit the instruction-based timing variation, a tunable delay buffer is implemented in the PE to rebalance the pipeline timing under different instructions. The tunable delay is realized by a series delay buffers with very small area overhead, which is similar to [5]. The delay setting of the tunable clock buffer is controlled by configuration registers in each clock domain. Before the operation of each neural network layer, the delay of clock buffer will be updated according to the required bit precision. The buffer can be tuned with a step of 50 ps to borrow timing Δt from the second pipeline stage to the MAC stage based on different instructions. For example, the tunable delay Δt can be set as large as 200 ps during INT8 mode, in which the critical paths are dominated by the first pipeline stage MAC operations. For IN1 mode, the timing difference between pipeline stags becomes small and the timing borrowing Δt is also set to be small. It is worth to mention that it is quite common to use the same bit precision for one entire DNN model (e.g., 8-bit precision for all neural network layers). Therefore, the setting of the tunable clock buffer only needs to be updated before the entire DNN computation, which has negligible overhead.

To exploit the DTS margin existed within every cycle, the clock period needs to be adjusted dynamically cycle-by-cycle. In every clock domain, the DDTC module detects the transitioning of local runtime operand values and determines the clock period requirement for the next cycle. Based on DDTC's target clock period, one clock phase is selected from the clock phase bus to either shrink or maintain a constant clock period for the clock domain. Combining two management strategies above, our adaptive clocking scheme exploits both instruction-based and runtime operand-based dynamic timing variations.

V. Elastic Clock Chain and Its Synchronization

A. Clock Chain Synchronization

As mentioned earlier, there are a total of 16 clock domains in the PE array, each clock domain needs to synchronize its

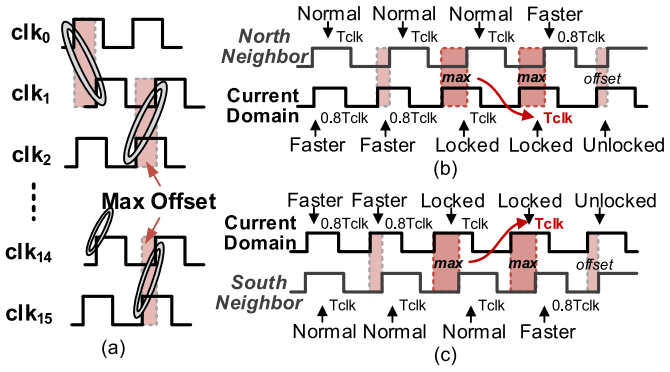


Fig. 11. (a) Clock synchronization inside the elastic clock chain, with each clock domain constrains the maximum phase offset between both (b) north neighbor domain and (c) south neighbor domain.

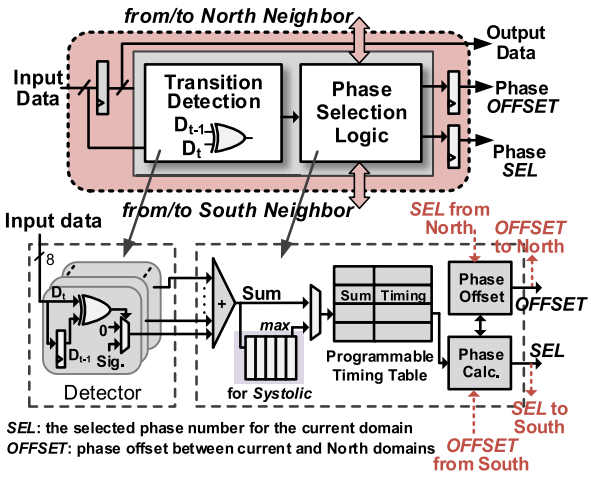


Fig. 12. Design details of DDTC module.

clock with its neighbor domains. As shown in Fig. 11(a), each clock domain is running at a different speed based on different local runtime operand values. To maintain synchronization between neighbor domains, we constrain the maximum phase offset between neighbor domains. When the phase offset reaches the max constraint, the clock domain that runs faster than its neighbor will be locked (i.e., not allow to speedup) by using the worst-case STA clock period T_{clk}. The synchronization policy adopts a simple unidirectional clock adjustment strategy (i.e., only forcing the faster clock domain to be locked until the neighbor domains catch up). Each clock domain needs to monitor the phase offset between itself and its neighbor domains. As the examples shown in Fig. 11(b) and (c), whenever the current clock domain becomes too fast and reaches the phase offset limit with either its North neighbor or South neighbor domain, it will be forced to use the worst-case clock period T_{clk}, until its neighbor clock domains speedup to reduce the phase offset. Because all the 16 clock domains run at different speeds while synchronizing neighbors within offset margin, we refer the proposed adaptive clocking scheme as "elastic clock chain," as all the domains are synchronized with some speed offsets forming a chain style clock propagation.

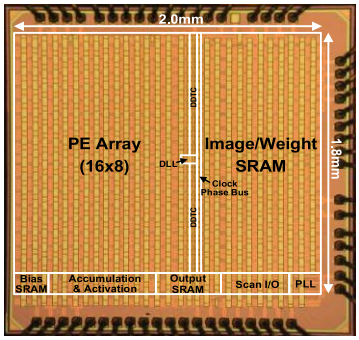
The main reason for maintaining synchronization between neighbor clock domains is to guarantee that data paths which cross time domains can be completed within the same amount of run cycles (i.e., no degradation of the throughput). In addition, asynchronous clock domain crossing logic can be avoided, which minimizes design costs. Inside the PE array, one representative data path that crosses clock domains is the partial sum propagation path which is connected vertically along the PE column. To maintain the timing margin for the partial sum data path, the maximum phase offset is confined to be within 300 ps. In addition, this phase offset margin is specially considered during the timing closure of the PE unit. The partial sum data paths are given careful timing closure (i.e., considering the maximum phase offset inside the timing constraints of external input–output delay) to satisfy both setup and hold timing. If the partial sum data path is critical with long delay initially, we have to limit a smaller maximum clock phase offset between clock domains.

B. Data Detection and Timing Controller

The DDTC module is implemented in every clock domain to manage the local clock period adjustment. Fig. 12 shows the design details of DDTC, which has two main functionalities. First, DDTC detects the bit transition conditions of input value. The input value A is fetched from SRAM and buffered in DDTC for the timing detection. The transition of each A bit is detected by XOR transition detectors. The significances of all the transitioning bits are summed up as the Sum value. A small lookup table is implemented inside DDTC to store dynamic timing settings for different transition significance summations Sum. After enabling the proposed elastic clock chain, a target clock period will be selected out based on the significance summation value every cycle. Second, DDTC determines the target clock period for the following cycle. Based on the summed significance Sum and the phase offsets with neighbor domains, the DDTC determine the dynamic clock period. There is a phase offset module that continuously monitors the phase offset between the current domain and its North neighbor. This phase offset is considered for phase calculation for the current domain and is also sent to its North neighbor domain.

The phase calculation module determines the clock phase selection based on three items: the target clock period selected from the lookup table, the North neighbor phase offset recorded in phase offset module, and the South neighbor phase offset sent from the South clock domain. If both North and South phase offsets are within the phase offset limit, the phase selection will allow the target clock period to speedup (i.e., select earlier clock phase to shrink the clock period). If one neighbor phase offset has reached the max offset, the clock phase selection is maintained the same as the worst-case clock period T_{clk}.

For 2-D SIMD configuration, the target clock period is selected from the lookup table every cycle based on the operand detection. For systolic array configuration, an additional queue is designed to store the operand detection results for the past eight cycles, as the input value will be propagated



Technology	65nm CMOS
Active Area	3.6mm^2
Total Power	288mW
Base Freq.	700MHz
Boosted Freq.	820MHz
Supply Vdd	0.5-1V
SRAM	84KB
DLL Area	1.6%
Clk Bus Area	1.1%
DDTC Area	2.2%

Fig. 13. Die photograph and the implementation details of chip.

TABLE I
POWER BREAKDOWN OF THE CHIP (mW)

Total Power	288
16×8 PE array	165 (57.3%)
SRAM memories	95.6 (33.2%)
PLL	11.5 (4.0%)
DLL	5.1 (1.8%)
Clock Bus	3.6 (1.2%)
DDTC	7.2 (2.5%)

inside PE row and last for eight cycles. The largest significance summation values from these eight cycles will be used for the target clock period selection. The overall data buffering and DDTC only introduces a negligible one clock cycle of latency in the accelerator's execution. After DDTC updates the phase selection, the clock period will be dynamically adjusted within the following cycle.

VI. CHIP IMPLEMENTATION AND MEASUREMENT

A. Chip Implementation

A 2-D PE array-based DNN accelerator with the elastic clock chain techniques was fabricated using a 65-nm CMOS process. The chip die photo and implementation details are shown in Fig. 13. The active die area is 3.6 mm^2. The nominal supply voltage and operating frequency are 1 V and 700 MHz. The chip is tested with voltage scaled down to 0.5 V.

There is a total of 128 PE units in the 2-D array with 16 rows and 8 columns. The DLL is placed in the center position to reduce the clock skew between different domains. The clock phase bus and DDTC modules are placed between PE array and all SRAM memories. The input ACT values are fetched from image memory and sent to DDTC first. The DDTC detects runtime compute operands and determines clock phase selection cycle-by-cycle. The accumulation and ACT modules are located at the bottom of the PE array and clocked by clock domain 15. All the on-chip SRAM content can be preload and read out through a scan IO interface. FPGA board is utilized as the interface to communicate data between the laptop and the chip for scanning in or out the memory data for functional verification.

To realize the elastic clock chain design, the clock phase bus consumes 1.1% of the total accelerator area and the DDTC modules consume 2.2% of the area. Table I provides

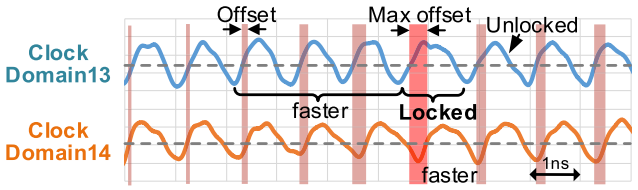


Fig. 14. Measured clock waveforms for two clock domains.

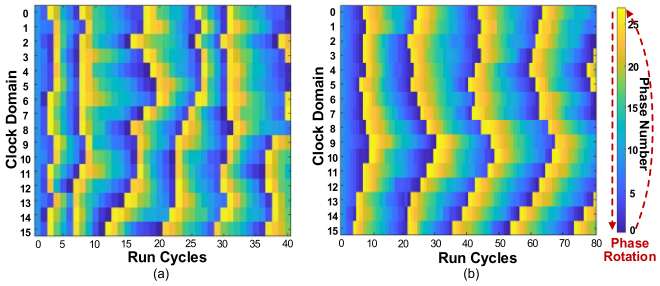


Fig. 15. Reconstructed clock propagation color map for all 16 clock domains under (a) 2-D SIMD and (b) systolic array configurations.

a detailed power breakdown of the chip at 1 V. The distribution of 28 clock phases and phase selection for all clock domains consumes about 3.6 mW at 1 V, which is 1.2% of the total active power. The customized clock phase bus has been carefully designed for the 1.5-mm-long-distance clock routing using high-level metals. After all the clock phases reach each PE row, the local clock is selected for each clock domain. The local clock tree is identical as conventional clock distribution, without any overhead. The DDTC modules cost 2.5% of total power. Overall, the proposed clocking schemes consume 5.5% of total power including DLL, clock phase bus, and DDTCs.

B. Elastic Clock Chain Measurement

Figs. 14 and 15 show the measured clock waveform and reconstructed color maps with the elastic clock chain technique. During testing, up to four real-time clock domains can be captured simultaneously using high-speed oscilloscopes. The phase offsets between measurement ports are precalibrated. As a measurement example shown in Fig. 14, if one domain is too fast (i.e., reaching the max phase offset with its neighbor), it will be locked using the longest clock period until the neighbors catch up, leading to a wavelike phase propagation.

In our measurement, the PLL frequency is locked at a constant frequency. To capture accurate clock propagation, we first measured the effective clock period by keeping rotating a fixed amount of clock phases (i.e., shrink the clock period constantly every cycle). During the benchmark test, the actual dynamic clock period for every cycle can be obtained based on the number of clock phase rotation. To measure the effective frequency of a DNN model with enabling the elastic clock chain, we run the entire model and verify the output results to be correct. With the correct DNN computation results, we can deterministically calculate the effective execution time based

TABLE II
DETAILS OF THE TEST DNN MODELS

Dataset	Model	Accuracy
MNIST	L1-L4 (FC): 784×256×256×256×10	97.22% (8b) 96.36% (4b) 95.88% (1b)
CIFAR-10	L1-L4 (Conv): (32,32,3) × (3,3,128) × (3,3,128) × (3,3,256) × (3,3,256) L5-L6 (FC): 1024×1024×10	86.53% (8b) 84.56% (4b) 78.67% (1b)

on the predefined dynamic timing control settings and the measured effective clock period.

The clocks across all clock domains were repetitively measured to reconstruct a clock propagation map, which represents the phase selection at each clock domain along with execution cycles. Fig. 15(a) shows the color map under the configuration of 2-D SIMD INT8 operation mode for MNIST data set inference. In our adaptive clock design, the clock period can either shrink or maintain constant by default. Depending on the transition bit number and position, the dynamic clock period is shrunk in different magnitudes. It is observed that each clock domain rotativity selects earlier clock phases from phase 27 (yellow) to phase 0 (blue) to shrink clock period and speedup the execution. Besides, every clock domain synchronizes with its neighbor clock domains within the phase offset limitation. The speedup in each clock domain is related to the runtime data pattern of the local operand values. The measured color map matches simulated wave propagation.

Fig. 15(b) shows the color map running the same data sequence under the systolic array configuration. For systolic dataflow, because the input value data travels horizontally among PEs, the worst-case target clock period is selected across eight previous clock cycles. Therefore, it is observed the clock propagation speed is slower than the 2-D SIMD configuration, which also leads to less performance improvement.

It is worth mentioning that data path delays vary at different voltage levels. Therefore, the post-silicon timing calibration is helpful to capture more accurate critical path delays on silicon and guide the dynamic timing settings with PVT variation. During our test, several customized weight and input value patterns with a certain amount of operand bit transitions are built to calibrate the timing for the dynamic timing settings. For the benchmark test, the timing table inside the DDTC is initially programmed with timing control settings based on case STA results. After the timings of individual transition condition have been calibrated, the calibrated timing control settings are then scanned into the timing table for the benchmark run again.

C. Benchmark Measurement Results

To evaluate the benefit of the elastic clock chain, we tested the inference of two DNN models for MNIST and CIFAR-10 data sets, as listed in Table II. A multilayer perceptron model (four fully connected layers) and a convolution neural network model (four convolution layers, two fully connected layers) are built for benefit evaluation. The proposed elastic clock chain scheme is also applicable to DNN models with different layer configurations and dimensions. These DNN models have been run layer by layer on the test chip. The average energy efficiency for one inference task is 310 GOPS/W using INT8 and 2.8 TOPS/W using INT1 at 1 V.

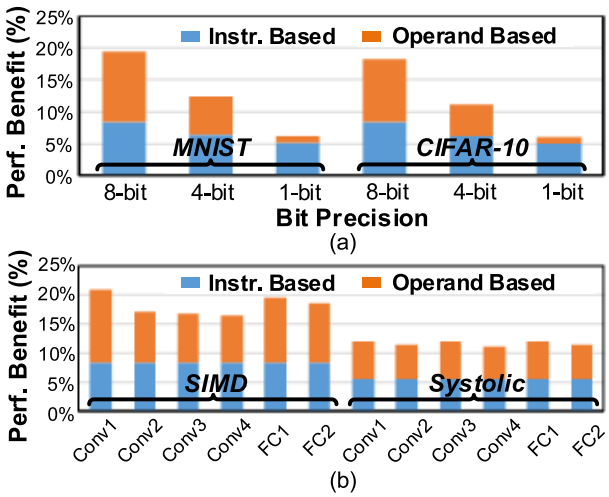


Fig. 16. (a) Performance improvement results under different bit-precisions. (b) Improvement breakdown at each neural network layer for CIFAR-10 under different configurations at INT8 mode.

Fig. 16 shows the performance improvement by exploiting both instruction-based and operand-based dynamic timing variations. During testing, each neural network layer has been broken into several small data pieces. After finishing the run of each data piece, the intermediate accumulation results have been read off-chip to verify the computation correctness. By this way, we guarantee all the matrix intermediate and final accumulation results are identical with the golden software model. For performance measurement, we first measured the maximum operating frequency (i.e., highest frequency with correct final results) with elastic clock chain disabled. After that, the effective frequency was measured again with enabling the elastic clock chain on the same test chip. Hence, the performance improvement can be measured in an apple-to-apple comparison with and without our proposed scheme.

Fig. 16 also shows the overall performance improvement results with different bit precision under 2-D SIMD configuration and the benefit breakdown at each layer at INT8 mode. Up to 19% performance improvement is obtained at INT8 mode, which effectively improves the operation frequency from 700 to more than 820 MHz. Within the benefit, about 9% is obtained by leveraging the instruction-based pipeline timing rebalance. The rest benefit comes from exploiting operand-based dynamic timing variation. When PE array is switched to INT1 mode, there are eight 1-bit MAC operations simultaneously processed in every PE. Therefore, eight operand numbers are fetched from the image memory of each PE, which significantly increases the register transition activities and reduces the operand-based dynamic timing benefit. Similar performance improvement is obtained from both MNIST and CIFAR-10 data set. It is also observed that the systolic array obtained about 8% less benefit for the same

TABLE III
COMPARISON WITH PRIOR ADAPTIVE TECHNIQUES

	[5] JSSC'16	[6] ISSCC'17	[16] ISSCC'19	[11] JSSC'18	[12] JSSC'19	This work
Process	16 nm	14 nm	65 nm	28 nm	16 nm	65nm
Architecture	MAC Unit	CPU cores	GPU cores	DNN Accelerator	DNN Accelerator	DNN Accelerator
Adaptive Technique	Droop Detection	Droop Detection	Instruction Based	Razor Detection	Droop Detection	Instruction and Operand Based
Datapath	64-bit	64-bit	32-bit	8/16-bit	8/16-bit	1/4/8-bit
PE Number	1	4	2	8	8	128
Frequency (MHz)	2500	4000	417	1200	1000	700
Voltage (V)	0.9	Not reported	1.0	0.9	0.8	1.0
Power (mW)	250	Not reported	468	63.5	59.5	288
ML Efficiency	N/A	N/A	N/A	1.209 TOPS/W (8b, 0.9V)	~500 GOPS/W (8b, 0.8V)	310 GOPS/W (8b) 2.8 TOPS/W (1b)
Perf. Improvement	13%	3.5%	18.2%	50%	11.7%	19%
Energy Saving	5%	8%	30.4%	30%	8.4%	34%

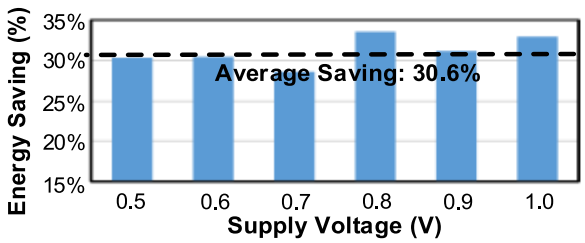


Fig. 17. Energy saving benefit with scaling down voltage while maintaining same total execution latency.

test run using the elastic clock chain. As explained before, this is because the systolic dataflow needs to consider the worst-case target clock period among the past eight cycles. The achieved benefits of performance improvement can be equivalently converted to energy savings by scaling down to a lower supply voltage while maintaining the same total execution time. As shown in Fig. 17, an average of 30.6% and up to 34% energy savings were achieved with the supply voltage down to 0.5 V.

Table III summarizes the design specifications of the developed elastic clock chain scheme and compares it with prior adaptive techniques. Previously, droop-based adaptive clocking schemes have been widely developed for processors to detect dynamic voltage droop and mitigate the voltage guardband by adaptively adjusting the operation frequency [5], [6]. A voltage variation monitor, e.g., tunable replica circuits, has been used to monitor the voltage droop. Compared with the droop-based adaptive clocking, the elastic clock chain is focusing on exploiting the slack of the runtime dynamic timing variation. There are a few adaptive clocking schemes exploiting DTS for some simple in-order CPU pipelines [14], [15] or more complicated GPGPU architecture [16], in which the clock adjustment settings are guided based on runtime instructions. In this work, multiple clock domains are supported with clock management guided by both runtime instructions and operand transitions for the DNN accelerator. Previously, resilient Razor flip-flops or adaptive droop detection schemes have been applied to a 1-D pipelined neural network accelerator to explore the error resiliency [11], [12]. However, this scheme will be very challenging to be applied to a 2-D PE array-based accelerator. The proposed elastic clock chain technique can exploit the dynamic timing variation within the PE unit. To fit the adaptive clocking properly into the 2-D PE array, there are a total of 16 clock domains implemented inside the array. In addition, the clock management is guided by both runtime instruction and operand values to realize fine-grained cycle-by-cycle clock adjustment.

It is worth mentioning that there are many schemes proposed to exploit the data sparsity at the architecture level [20], [21]. The zero values are skipped during DNN computation to improve the performance. However, complicated indexing mechanism is required to indicate the memory accessing addresses and the unstructured "0" pattern makes indexing challenging and heavy cost. Compared with those work, the proposed adaptive clock technique exploits the DTS margin from a clocking design angle, which does not require the complex indexing design. In addition, our solution can be applied to the sparsity exploitation PE-array orthogonally with special clocking considerations for the sparsity management modules.

VII. CONCLUSION

In this article, an adaptive clocking technique, elastic clock chain, is proposed to exploit the dynamic timing margin for a 2-D PE array-based DNN accelerator. The proposed scheme implemented a total of 16 clock domains through a multi-phase clock bus. The clocking for each domain is guided by both runtime instructions and local operands. Every clock domain synchronizes with its neighbors by constraining the maximum clock phase offset. Measurement results on a 65-nm test chip show that the effective frequency has been improved by up to 19% for MNIST and CIFAR-10 data sets

under 2-D SIMD configuration. The obtained performance benefit is equivalently converted to 34% energy saving. For the systolic dataflow, 8% less performance gain is obtained due to consideration of the worst-case dynamic timing among eight run cycles.

REFERENCES

[1] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE J. Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, Jan. 2017.

[2] B. Moons, R. Uytterhoeven, W. Dehaene, and M. Verhelst, "Envision: A 0.26-to-10TOPS/W subword-parallel dynamic-voltage-accuracy-frequency-scalable convolutional neural network processor in 28nm FDSOI," in *IEEE ISSCC Dig. Tech. Papers*, Feb. 2017, pp. 246–247.

[3] J. Lee, C. Kim, S. Kang, D. Shin, S. Kim, and H.-J. Yoo, "UNPU: A 50.6TOPS/W unified deep neural network accelerator with 1b-to-16b fully-variable weight bit-precision," in *IEEE ISSCC Dig. Tech. Papers*, Feb. 2018, pp. 218–219.

[4] K. Ueyoshi *et al.*, "QUEST: A 7.49TOPS multi-purpose log-quantized DNN inference engine stacked on 96MB 3D SRAM using inductive-coupling technology in 40nm CMOS," in *IEEE ISSCC Dig. Tech. Papers*, Feb. 2018, pp. 160–161.

[5] K. A. Bowman *et al.*, "A 16 nm all-digital auto-calibrating adaptive clock distribution for supply voltage droop tolerance across a wide operating range," *IEEE J. Solid-State Circuits*, vol. 51, no. 1, pp. 8–17, Jan. 2016.

[6] M. Floyd *et al.*, "Adaptive clocking in the POWER9 processor for voltage droop protection," in *IEEE ISSCC Dig. Tech. Papers*, Feb. 2017, pp. 444–445.

[7] S. T. Kim *et al.*, "Enabling wide autonomous DVFS in a 22 nm graphics execution core using a digitally controlled fully integrated voltage regulator," *IEEE J. Solid-State Circuits*, vol. 51, no. 1, pp. 18–30, Jan. 2016.

[8] M. Cho *et al.*, "Postsilicon voltage guard-band reduction in a 22 nm graphics execution core using adaptive voltage scaling and dynamic power gating," *IEEE J. Solid-State Circuits*, vol. 52, no. 1, pp. 50–63, Jan. 2017.

[9] B. Zimmer *et al.*, "A 0.11 pJ/op, 0.32-128 TOPS, scalable Multi-Chip-Module-based deep neural network accelerator with ground-reference signaling in 16nm," in *Proc. Symp. VLSI Circuits*, Jun. 2019, pp. 300–301.

[10] M. Fojtik *et al.*, "A fine-grained GALS SoC with Pausible adaptive clocking in 16 nm FinFET," in *Proc. 25th IEEE Int. Symp. Asynchronous Circuits Syst. (ASYNC)*, May 2019, pp. 27–35.

[11] P. N. Whatmough, S. K. Lee, D. Brooks, and G.-Y. Wei, "DNN engine: A 28-nm timing-error tolerant sparse deep neural network processor for IoT applications," *IEEE J. Solid-State Circuits*, vol. 53, no. 9, pp. 2722–2731, Sep. 2018.

[12] S. K. Lee, P. N. Whatmough, D. Brooks, and G.-Y. Wei, "A 16-nm always-on DNN processor with adaptive clocking and multi-cycle banked SRAMs," *IEEE J. Solid-State Circuits*, vol. 54, no. 7, pp. 1982–1992, Jul. 2019.

[13] Y. Fan, S. Campanoni, and R. Joseph, "Time squeezing for tiny devices," in *Proc. 46th Int. Symp. Comput. Archit. (ISCA)*, Jun. 2019, pp. 657–670.

[14] J. Constantin, A. Bonetti, A. Teman, C. Muller, L. Schmid, and A. Burg, "DynOR: A 32-bit microprocessor in 28 nm FD-SOI with cycle-by-cycle dynamic clock adjustment," in *Proc. ESSCIRC Conf., 42nd Eur. Solid-State Circuits Conf.*, Sep. 2016, pp. 261–264.

[15] T. Jia, R. Joseph, and J. Gu, "An instruction-driven adaptive clock management through dynamic phase scaling and compiler assistance for a low power microprocessor," *IEEE J. Solid-State Circuits*, vol. 54, no. 8, pp. 2327–2338, Aug. 2019.

[16] T. Jia, R. Joseph, and J. Gu, "An adaptive clock management scheme exploiting instruction-based dynamic timing slack for a general-purpose graphics processor unit with deep pipeline and out-of-order execution," in *IEEE ISSCC Dig. Tech. Papers*, Feb. 2019, pp. 318–319.

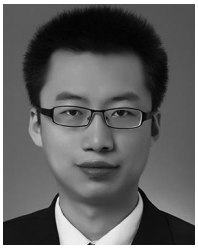
[17] N. Jouppi *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proc. 44th Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2017, pp. 1–12.

[18] T. Jia, Y. Ju, and J. Gu, "A compute-adaptive elastic clock-chain technique with dynamic timing enhancement for 2D PE-array-based accelerators," in *IEEE ISSCC Dig. Tech. Papers*, Feb. 2020, pp. 482–483.

[19] B. Mesgarzadeh and A. Alvandpour, "A low-power digital DLL-based clock generator in open-loop mode," *IEEE J. Solid-State Circuits*, vol. 44, no. 7, pp. 1907–1913, Jul. 2009.

[20] A. Parashar *et al.*, "SCNN: An accelerator for compressed-sparse convolutional neural networks," in *Proc. 44th Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2017, pp. 27–40.

[21] Z. Yuan *et al.*, "Sticker: A 0.41-62.1 TOPS/W 8Bit neural network processor with multi-sparsity compatible convolution arrays and online tuning acceleration for fully connected layers," in *Proc. IEEE Symp. VLSI Circuits*, Jun. 2018, pp. 33–34.



Tianyu Jia (Member, IEEE) received the B.S. and M.S. degrees from the Beijing University of Posts and Telecommunications, Beijing, China, in 2011 and 2014, respectively, and the M.S. and Ph.D. degrees in computer engineering from Northwestern University, Evanston, IL, USA, in 2018 and 2019, respectively.

He was interned at the IBM Thomas J. Watson Research Center, Yorktown Heights, NY, USA, in 2018, and Apple Inc., Cupertino, CA, USA, in 2019. He is currently a Post-Doctoral Fellow with Harvard University, Cambridge, MA, USA. His current research interests include the architecture explorations for domain-specific accelerator design, and efficient power and clock management circuits.

Dr. Jia was a recipient the IEEE SSCS Predoctoral Achievement Award from 2019 to 2020.

Yuhao Ju (Student Member, IEEE) received the B.S. degree from the University of Electronic Science and Technology of China, Chengdu, China, in 2017, and the M.S. degree from Northwestern University, Evanston, IL, USA, in 2019, where he is currently pursuing the Ph.D. degree in computer engineering.

His current research interests include computer architecture and machine learning accelerator design.

Jie Gu (Senior Member, IEEE) received the B.S. degree from Tsinghua University, Beijing, China, in 2001, the M.S. degree from Texas A&M University, College Station, TX, USA, in 2003, and the Ph.D. degree from the University of Minnesota, Minneapolis, MN, USA, in 2008.

He worked as an IC Design Engineer with Texas Instruments, Dallas, TX, USA, from 2008 to 2010, focusing on ultra-low-voltage mobile processor design and integrated power management techniques. He was a Senior Staff Engineer with Maxlinear, Inc., Carlsbad, CA, USA, from 2011 to 2014, focusing on low-power mixed-signal broadband system-on-chip (SoC) design. He is currently an Assistant Professor with Northwestern University, Evanston, IL, USA. His research interests include ultra-dynamic clock and power management for microprocessor and accelerators, emerging mixed-signal computing circuit, and the design of machine learning capable edge devices.

Dr. Gu was a recipient of the NSF CAREER Award. He has served as the Co-Chair of program committees and conference for numerous low-power design conferences and journals, such as ISPLED, DAC, ICCAD, and ICCD.