Feedforward-Cutset-Free Pipelined Multiply–Accumulate Unit for the Machine Learning Accelerator

Sungju Ryu, *Student Member, IEEE*, Naebeom Park, and Jae-Joon Kim, *Member, IEEE*

***Abstract*— Multiply–accumulate (MAC) computations account for a large part of machine learning accelerator operations. The pipelined structure is usually adopted to improve the performance by reducing the length of critical paths. An increase in the number of flip-flops due to pipelining, however, generally results in significant area and power increase. A large number of flip-flops are often required to meet the feedforward-cutset rule. Based on the observation that this rule can be relaxed in machine learning applications, we propose a pipelining method that eliminates some of the flip-flops selectively. The simulation results show that the proposed MAC unit achieved a 20% energy saving and a 20% area reduction compared with the conventional pipelined MAC.**

***Index Terms*— Hardware accelerator, machine learning, multiply–accumulate (MAC) unit, pipelining.**

I. Introduction

RECENTLY, the deep neural network (DNN) emerged as a powerful tool for various applications including image classification [1]–[3] and speech recognition [4], [5]. Since an enormous amount of vector-matrix multiplication computations are required in a typical DNN application, a variety of dedicated hardware for machine learning have been proposed to accelerate the computations [6], [7]. In a machine learning accelerator, a large number of multiply–accumulate (MAC) units are included for parallel computations, and timing-critical paths of the system are often found in the unit.

A multiplier typically consists of several computational parts including a partial product generation, a column addition, and a final addition. An accumulator consists of the carry-propagation adder. Long critical paths through these stages lead to the performance degradation of the overall system. To minimize this problem, various methods have been studied. The Wallace [8] and Dadda [9] multipliers are well-known examples for the achievement of a fast column addition, and the carry-lookahead (CLA) adder is often used to reduce the critical path in the accumulator or the final addition stage of the multiplier.

Meanwhile, a MAC operation is performed in the machine learning algorithm to compute a partial sum that is the accumulation of the input multiplied by the weight. In a MAC unit, the multiply and accumulate operations are usually merged to reduce the number of carry-propagation steps from two to one [10]. Such a structure, however, still comprises a long critical path delay that is approximately equal to the critical path delay of a multiplier.

It is well known that pipelining is one of the most popular approaches for increasing the operation clock frequency. Although pipelining is an efficient way to reduce the critical path delays, it results in an increase in the area and the power consumption due to the insertion of many flip-flops. In particular, the number of flip-flops tends to be large because the flip-flops must be inserted in the feedforward-cutset to ensure functional equality before and after the pipelining. The problem worsens as the number of pipeline stages is increased.

The main idea of this paper is the ability to relax the feedforward-cutset rule in the MAC design for machine learning applications, because only the final value is used out of the large number of multiply–accumulations. In other words, different from the usage of the conventional MAC unit, intermediate accumulation values are not used here, and hence, they do not need to be correct as long as the final value is correct. Under such a condition, the final value can become correct if each binary input of the adders inside the MAC participates in the calculation once and only once, irrespective of the cycle. Therefore, it is not necessary to set an accurate pipeline boundary.

Based on the previously explained idea, this paper proposes a feedforward-cutset-free (FCF) pipelined MAC architecture that is specialized for a high-performance machine learning accelerator. The proposed design method reduces the area and the power consumption by decreasing the number of inserted flip-flops for the pipelining.

The remainder of this paper is organized as follows. Section II reviews the feedforward-cutset rule for general pipelining. In Section III, the FCF pipelining method is introduced along with the process regarding the way it is applied to the accumulator design. In Section IV, the proposed

Manuscript received May 25, 2018; revised August 18, 2018; accepted September 22, 2018. This work was supported in part by the Samsung Research Funding Center of Samsung Electronics under Project SRFC-TC1603-04 and in part by the Ministry of Science and ICT, South Korea, through the ICT Consilience Creative Program under Grant IITP-2018-2011-1-00783 supervised by the Institute for Information and Communications Technology Promotion. *(Corresponding author: Jae-Joon Kim.)*

The authors are with the Department of Creative IT Engineering, Pohang University of Science and Technology, Pohang 37673, South Korea (e-mail: sungju.ryu@postech.ac.kr; naebeom.park@postech.ac.kr; jaejoon@postech.ac.kr).

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

Digital Object Identifier 10.1109/TVLSI.2018.2873716

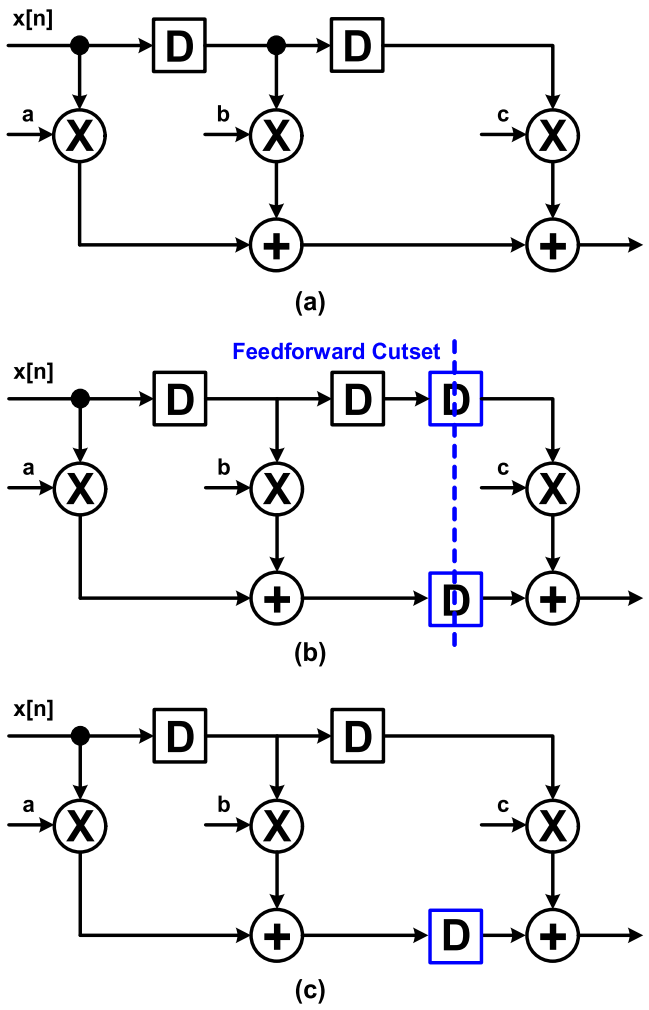


Fig. 1. (a) Block diagrams of the three-tap FIR filter [11]. (b) Valid pipelining. (c) Invalid pipelining. "D" indicates a flip-flop.

scheme is applied to the MAC architecture. An evaluation of the proposed pipelining method is described in Section V. Finally, we conclude this paper in Section VI.

II. Preliminary: Feedforward-Cutset Rule for Pipelining

It is well known that pipelining is one of the most effective ways to reduce the critical path delay, thereby increasing the clock frequency. This reduction is achieved through the insertion of flip-flops into the datapath.

Fig. 1(a) shows the block diagram of a three-tap finite-impulse response (FIR) filter [11]

$$y[n] = ax[n] + bx[n-1] + cx[n-2]. \tag{1}$$

Fig. 1(b) and (c) shows pipelining examples regarding the FIR filter. In addition to reducing critical path delays through pipelining, it is also important to satisfy functional equality before and after pipelining. The point at which the flip-flops are inserted to ensure functional equality is called the feedforward-cutset. The definitions of cutset and feedforward-cutset are as follows [11]:

Cutset: A set of the edges of a graph such that if these edges are removed from the graph, and the graph becomes disjointed.

Feedforward-cutset: A cutset where the data move in the forward direction on all of the cutset edges.

Fig. 1(b) shows an example of valid pipelining. The two-stage pipelined FIR filter is constructed by inserting two flip-flops along feedforward-cutset. In contrast, Fig. 1(c) shows an example of invalid pipelining. Functional equality is not guaranteed in this case because the flip-flops are not inserted correctly along the feedforward-cutset.

III. Pipelined Accumulator

While the conventional pipelining method is advantageous because it effectively reduces the critical path delays, it leads mostly to an increase in the area and the power consumption due to the insertion of a large number of flip-flops. Moreover, the constraint that requires the insertion of the flip-flops according to the feedforward-cutset rule tends to significantly increase the overhead. This section proposes a method for the selective elimination of the flip-flops from the conventional pipeline boundary by exploiting the unique characteristics of the machine learning algorithm.

A. Proposed FCF Pipelining

Fig. 2 shows examples of the two-stage 32-bit pipelined accumulator (PA) that is based on the ripple carry adder (RCA). $A[31:0]$ represents data that move from the outside to the input buffer register. $A_{\mathrm{Reg}}[31:0]$ represents the data that are stored in the input buffer. $S[31:0]$ represents the data that are stored in the output buffer register as a result of the accumulation. In the conventional PA structure [Fig. 2(a)], the flip-flops must be inserted along the feedforward-cutset to ensure functional equality. Since the accumulator in Fig. 2(a) comprises two pipeline stages, the number of additional flip-flops for the pipelining is 33 (gray-colored flip-flops). If the accumulator is pipelined to the n-stage, the number of inserted flip-flops becomes $33(n-1)$, which confirms that the number of flip-flops for the pipelining increases significantly as the number of pipeline stages is increased.

Fig. 2(b) shows the proposed FCF-PA. For the FCF-PA, only one flip-flop is inserted for the two-stage pipelining. Therefore, the number of additional flip-flops for the n-stage pipeline is $n-1$ only.

In the conventional PA, the correct accumulation values of all the inputs up to the corresponding clock cycle are produced in each clock cycle as shown in the timing diagram of Fig. 2(a). A two-cycle difference exists between the input and the corresponding output due to the two-stage pipeline. On the other hand, in the proposed architecture, only the final accumulation result is valid as shown in the timing diagram of Fig. 2(b).

Fig. 3 shows examples of the ways that the conventional PA and the proposed method (FCF-PA) work. In the conventional two-stage PA, the accumulation output (S) is produced two-clock-cycle after the corresponding input is stored in the

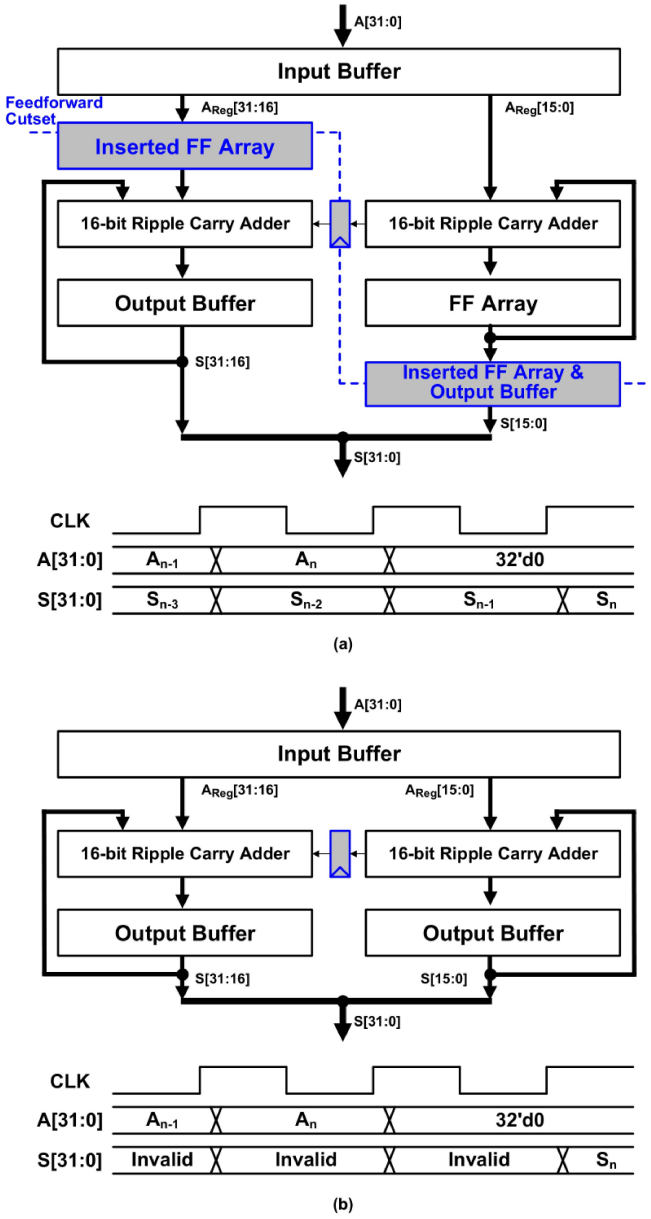


Fig. 2. Schematics and timing diagrams of two-stage 32-bit accumulators. (a) Conventional PA. (b) Proposed FCF-PA.

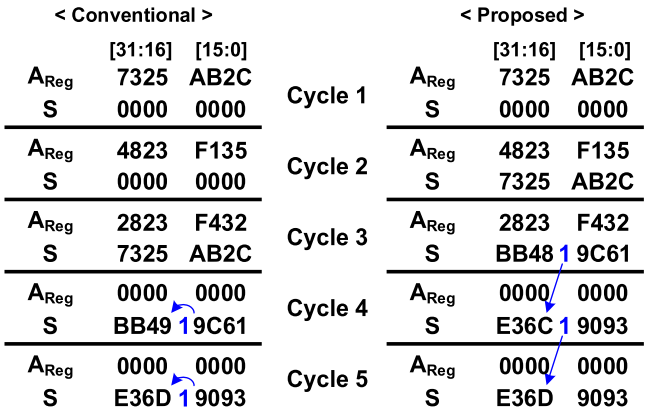


Fig. 3. Two-stage 32-bit pipelined-accumulation examples with the conventional pipelining (left) and proposed FCF-PA (right). Binary number "1" between the two 16-bit hexadecimal numbers is a carry from the lower half.

input buffer. On the other hand, regarding the proposed structure, the output is generated one clock cycle after the input arrives. Moreover, for the proposed scheme, the generated carry from the lower half of the 32-bit adder is involved in the accumulation one clock cycle later than the case of the conventional pipelining. For example, in the conventional case, the generated carry from the lower half and the corresponding inputs are fed into the upper half adder in the same clock cycle as shown in the cycles 4 and 5 of Fig. 3 (left). On the other hand, in the proposed FCF-PA, the carry from the lower half is fed into the upper half one cycle later than the corresponding input for the upper half, as depicted in the clock cycles 3-5 of Fig. 3 (right). This characteristic makes the intermediate result that is stored in the output buffer of the proposed accumulator different from the result of the conventional pipelining case. The proposed accumulator, however, shows the same final output (cycle 5) as that of the conventional one. In addition, regarding the two architectures, the number of cycles from the initial input to the final output is the same. The characteristic of the proposed FCF pipelining method can be summarized as follows: In the case where adders are used to process data in an accumulator, the final accumulation result is the same even if binary inputs are fed to the adders in an arbitrary clock cycle as far as they are fed once and only once.

In the machine learning algorithm, only the final result of the weighted sum of the multiplication between the input feature map and the filters is used for the subsequent operation, so the proposed accumulator would produce the same results as the conventional accumulator.

Meanwhile, the CLA adder has been mostly used to reduce the critical path delay of the accumulator. The carry prediction logic in the CLA, however, causes a significant increase in the area and the power consumption. For the same critical path delay, the FCF-PA can be implemented with less area and lower power consumption compared with the accumulator that is based on the CLA.

B. Modified FCF-PA for Further Power Reductions

Although the proposed FCF-PA can reduce the area and the power consumption by replacing the CLA, there are certain input conditions in which the undesired data transition in the output buffer occurs, thereby reducing the power efficiency when 2's complement numbers are used. Fig. 4 shows an example of the undesired data transition. The inputs are 4-bit 2's complement binary numbers. $A_{\mathrm{Reg}}[7:4]$ is the sign extension of $A_{\mathrm{Reg}}[3]$, which is the sign bit of $A_{\mathrm{Reg}}[3:0]$. In the conventional pipelining [Fig. 4 (left)], the accumulation result (S) in cycle 3 and the data stored in the input buffer (A_{Reg}) in cycle 2 are added and stored in the output buffer (S) in cycle 4. In this case, the "1" in $A_{\mathrm{Reg}}[2]$ in cycle 2 and the "1" in $S[2]$ in cycle 3 are added, thereby generating a carry. The carry is transmitted to the upper half of the S, and hence, $S[7{:}4]$ remains as "0000" in cycle 4.

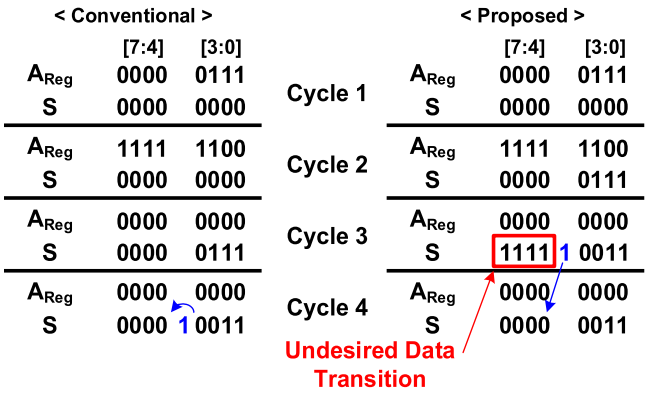


Fig. 4. Example of an undesired data transition in the two-stage 8-bit PAs with 4-bit 2's complement input numbers. Binary number "1" between the two 4-bit hexadecimal numbers is a carry from the lower half.

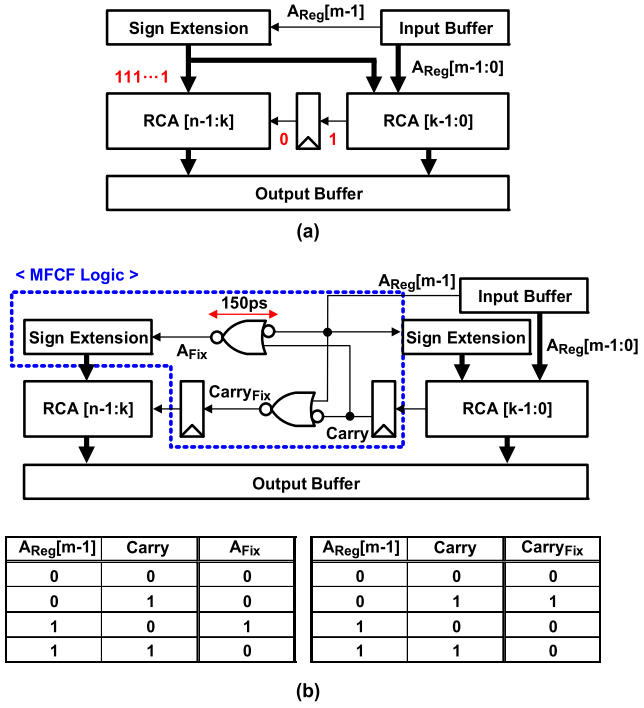


Fig. 5. Proposed (a) FCF-PA and (b) MFCF-PA for the improvement of the power efficiency.

On the other hand, in the FCF-PA [Fig. 4 (right)], $A_{\mathrm{Reg}}[2]$ and $S[2]$ are added in cycle 2, thereby generating a carry. In cycle 3, the generated carry from the lower half is stored in the flip-flop. The carry is no longer propagated toward the upper half in this clock cycle. In the next clock cycle (cycle 4), the carry that is stored in the flip-flop is transferred to the carry input of the upper half. During the calculation, it can be observed that $S[7:4]$ changes to "1111" in cycle 3 and returns to "0000" in cycle 4. Although this undesired data transition does not affect the accuracy of accumulation results, it reduces the power efficiency of the FCF-PA. Fig. 5(a) shows the structure of the FCF-PA for the 2's complement numbers. The binary numbers in the diagram indicate the sign extension of $A_{\mathrm{Reg}}[m-1]$ and the carry bits for the undesired data transition case.

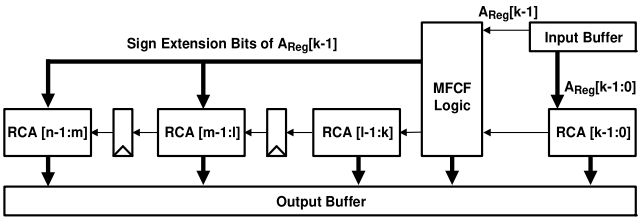


Fig. 6. Block diagram of MFCF-PA.

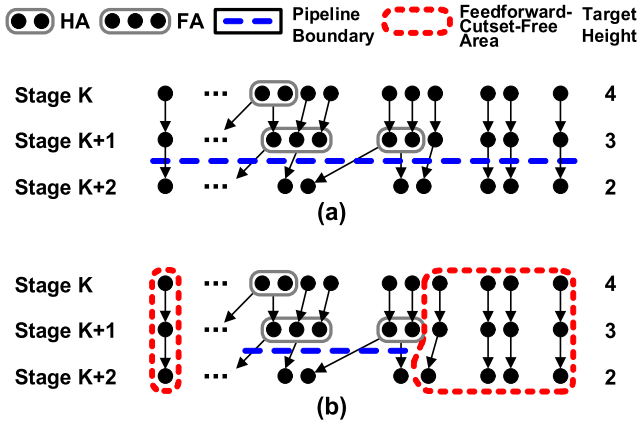


Fig. 7. Pipelined column addition structure with the Dadda multiplier. (a) Conventional pipelining. (b) Proposed FCF pipelining. HA: half-adder. FA: full adder.

To prevent the undesired data transitions, the sign extended the input number to RCA$[n-1:k]$ and the carry out of RCA$[k-1:0]$ must be modified to "0" if both of $A_{\mathrm{Reg}}[m-1]$ and the carry out from RCA$[k-1:0]$ are "1"; therefore, the undesired data transition can be prevented by detecting such a condition. Since the critical delay becomes too long if the upper half-addition needs to wait until the decision regarding the lower half carry-out condition detection, the modified version of the FCF-PA (MFCF-PA) is proposed here as shown in Fig. 5(b). Accordingly, an additional flip-flop was added between the two RCAs to prevent the formation of a long critical path. RCA$[n-1:k]$ receives both A_{Fix} with the sign extension and the $Carry_{Fix}$ signals as modified input values. A_{Fix} generates "0" when $A_{\mathrm{Reg}}[m-1]$ and $Carry$ are both "1." Otherwise, $A_{\mathrm{Reg}}[m-1]$ is buffered in A_{Fix} as it is. Similarly, $Carry_{Fix}$ generates "0" when $A_{\mathrm{Reg}}[m-1]$ and $Carry$ are both "1." Otherwise, $Carry$ is buffered in $Carry_{Fix}$ as it is. Although an additional NORI(NOR + INV) gate causes an additional delay (150 ps at SS corner, 10% supply voltage drop, 125 °C temperature in our analysis) in the critical path, the overhead is negligible considering that target clock period ranges from 1.1 to 2.7 ns (in Section V). In the event that the accumulator is pipelined to multiple stages, the insertion of the additional logic into all of the pipeline stages may increase the area overhead. To reduce the overhead, the modified structure [Fig. 5(b)] is inserted into only one pipeline stage. For the rest of the pipeline stages, only the FCF-PA [Fig. 5(a)] is used. Fig. 6 shows a block diagram of the MFCF-PA. A good power efficiency is still shown, because the probability of the sign extension bit becoming "1" is reduced,

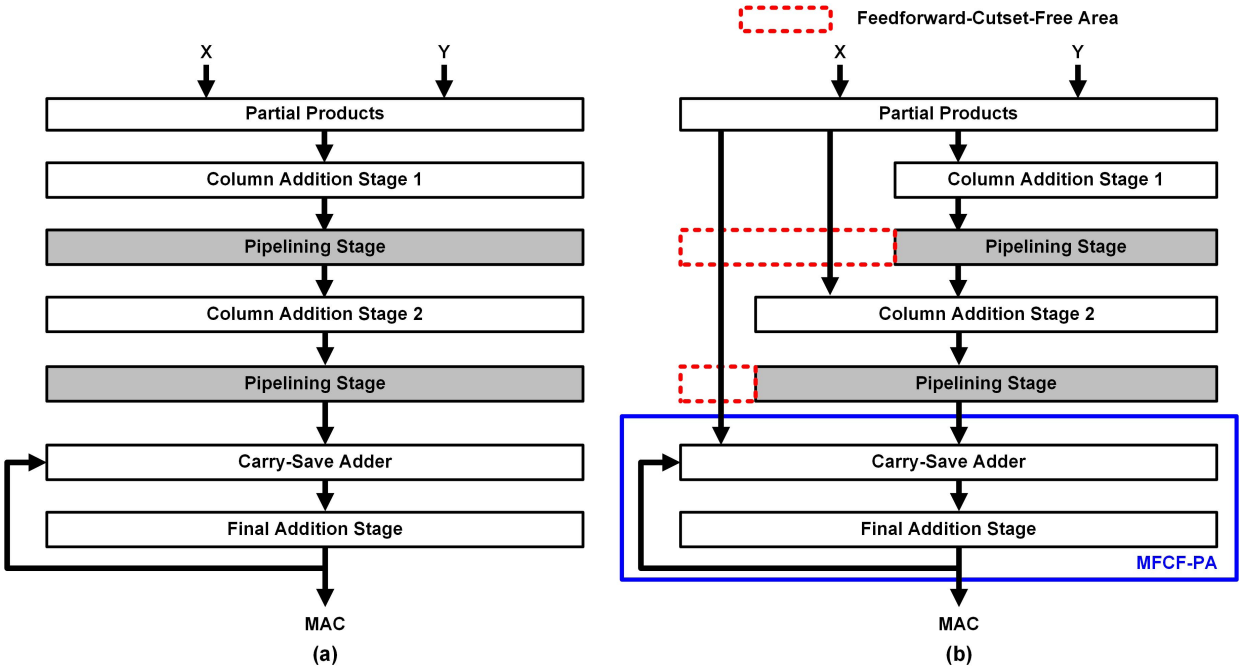


Fig. 8. Block diagrams of pipelined MAC architectures. (a) State-of-the-art merged MAC [12] with pipelining. (b) Proposed FCF-MAC with MFCF-PA. Dotted box: FCF area where the flip-flops are removed from the conventional pipelined MAC.

and therefore, the chance of the undesired data transitions in the other RCAs is reduced. Please note that the accuracy of computation is not affected even if the rare case of undesired data transition occurs.

IV. Pipelined MAC Unit

The column addition in the MAC operation is for the calculation of binary numbers in each addition stage using the half-adders and/or full adders and then for the passing of the results to the next addition stage. Since MAC computations are based on such additions, the proposed pipelining method can also be applied to the machine learning-specific MAC structure. In this section, the proposed pipelining method is applied to the MAC architecture by using the unique characteristic of Dadda multiplier. The Dadda multiplier performs the column addition in a similar fashion to the Wallace multiplier which is widely used, and it has less area and shorter critical path delay than the Wallace multiplier [13].

Fig. 7 shows the pipelined column addition structures in the Dadda multiplier. The Dadda multiplier performs the column addition to reduce the height of each stage. If a particular column already satisfies the target height for the next column addition stage, then no operation is performed during the stage [13]. Using this property, the proposed pipelining method can be applied to the MAC structure as well. Fig. 7(a) is an example of pipelining where the conventional method is used. All of the edges in the feedforward-cutset are subject to pipelining. On the other hand, in the proposed FCF pipelining case [Fig. 7(b)], if a node in the column addition stage does not need to participate in the height reduction, it can be excluded from the pipelining [the group in the dotted box of Fig. 7(b)]. In other words, in the conventional pipelining method, all the edges in the feedforward-cutset must be pipelined to ensure functional equality regardless of a timing slack of each edge [Fig. 7(a)]. However, in the FCF pipelining method, some edges in the cutset do not necessarily have to be pipelined if the edges have enough timing slacks [Fig. 7(b)]. As a result, a smaller number of flip-flops are required compared with the conventional pipelining case. On the other hand, in the Wallace multiplier, as many partial products as possible are involved in the calculation for each column addition stage. Because the partial products do not have enough timing slack to be excluded from pipelining, the effectiveness of the proposed FCF pipelining method is smaller in the Wallace multiplier case than in the Dadda multiplier case.

Fig. 8 shows the block diagrams of pipelined MAC architectures. The proposed MAC architecture [Fig. 8(b)] combines the FCF-MAC (MAC with the proposed FCF pipelining) for the column addition and the MFCF-PA for the accumulation. Instead of pipelining all of the final nodes in the column addition stage as in Fig. 8(a), the proposed FCF-MAC architecture is used to selectively choose the nodes for the pipelining. For the proposed architecture, the merged multiply–accumulation style is adopted [12]. The final adder is placed in the last stage of the MAC operation. In general, the final adder is designed using the CLA to achieve a short critical path delay. In contrast, the proposed design uses the MFCF-PA style in the accumulation stage in consideration of the greater power and the area efficiency of the MFCF-PA, as described in Section III.

V. Results

A. Experimental Setup

We evaluate the proposed FCF pipelining method in this section. First, for application of the accumulator-only case,

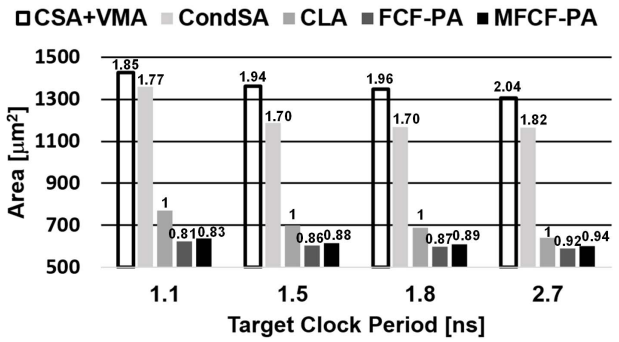


Fig. 9. Comparison of area among CSA + VMA, CondSA, CLA-based accumulators, proposed FCF-PA, and MFCF-PA. Numbers above bars in the figures: normalized area to the CLA-based accumulator.

binary-weight-networks [14], [15] are considered. For the input of the accumulator, the 16-bit 2's complement number is used considering that many state-of-the-art DNN hardware accelerators use the 16-bit as their bit precision [6], [16], [17]. For the accumulator to construct an output feature in AlexNet [1], 11 extra bits are required to fully accumulate the input features. Hence, we determine the number of bits in the accumulator to be [16 (InputFeature) +11 (Accumulation) =] 27-bit. For the MAC case, the 16-bit 2's complement number is used for both the input feature and weight. In that case, the number of bits in the final adder is determined to be [16 × 2 (Multiplication) +11 (Accumulation) =] 43-bit. The design is synthesized with the gate-level cells in a 65-nm CMOS technology using Synopsys Design Compiler. For some custom designs, "set_dont_touch" command of Design Compiler is used after the direct instantiation of the cells in the standard library, rather than the synthesis of cells using the register-transfer level description. Synopsys PrimeTime is used to analyze the area and the power consumption. For the evaluation of the power consumption, we run the time-based analysis with a value charge dump file in the PrimeTime PX. Both the actual input features (ImageNet data set) and random vectors generated by the pseudorandom number generator (PRNG) are fed as input data. We design the CLA by simply describing it as "A + B" in Verilog Hardware Description Language and by synthesizing/optimizing using Design Compiler. All simulation results for the area and the power consumption include input and output buffers. The numbers above bars in the figures indicate normalized area/power to the CLA-based accumulator (accumulator-only case) or "MAC + CLA" configuration (MAC case).

B. Power and Area Analysis

Fig. 9 shows the area comparison between the conventional and proposed accumulators. The areas of the accumulators are compared with each other using the designs that are synthesized for the same target clock period. When synthesized for the 2.7-ns target, the area of the FCF-PA is 8% smaller than the area of the CLA-based accumulator. For the 1.1-ns designs, the area reduction increases to 19%, because the carry prediction logic of CLA becomes more complicated for lower target clock period. Meanwhile, a carry-save adder

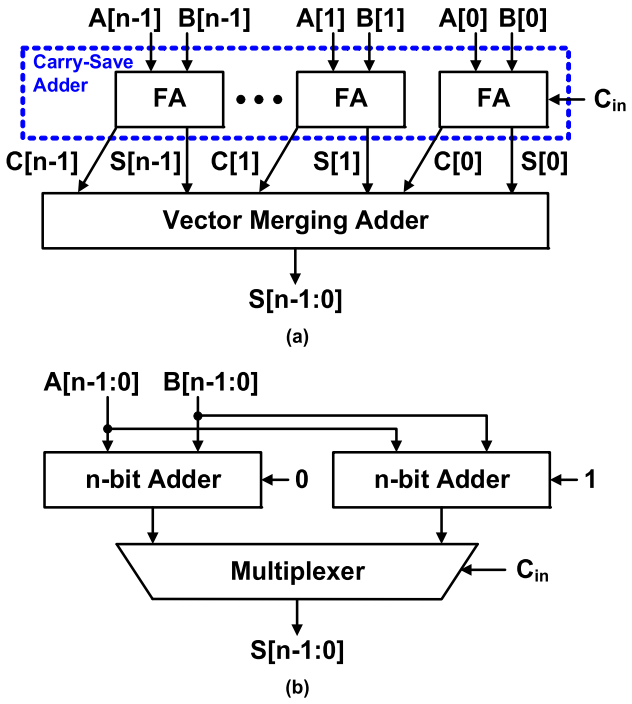


Fig. 10. Block diagrams of (a) CSA + VMA and (b) CondSA.

(CSA)-based accumulator [Fig. 10(a)] itself does not experience carry propagation because each digit of the addition result is composed of 2-bit, thereby achieving both the shorter latency and lower area/power. However, carry propagation in vector merging adder (VMA) is necessary to convert the 2-bit digit into the 1-bit digit, before the final accumulation result is transferred to another functional block. The area of the CSA + VMA-based accumulator is 85%–104% larger than the CLA-based accumulator. The conditional-sum adder (CondSA)-based accumulator [Fig. 10(b)] requires a 70%–82% larger area than the CLA-based accumulator, because the CondSA requires extra adders to precalculate the expected sum values for both "0" and "1" of carry-in values and requires extra muxes to select one of the precalculated sum values.

Fig. 11 shows that the proposed accumulators consume less power than the conventional accumulators. When the target clock period is 2.7 ns, the power consumption of the FCF-PA is 4% smaller that of the CLA-based accumulator, and 13% smaller with the 1.1-ns target. Moreover, MFCF-PA saves more power than FCF-PA as expected. Compared with the CLA-based accumulator, MFCF-PA consumes 12% less power at the 2.7-ns clock period and 19% less power at the 1.1-ns period, respectively. The area overhead for the conversion of the FCF-PA to the MFCF-PA is approximately 2%. In addition, CSA + VMA- and CondSA-based accumulators experience an increase in power consumption by 60%–78% and 18%–78%, respectively, compared with the CLA-based accumulator due to the extra adders and multiplexers. The power dissipation of the accumulator is similar in both the cases when the random values generated from the PRNG are fed and when the actual ImageNet data set is fed.

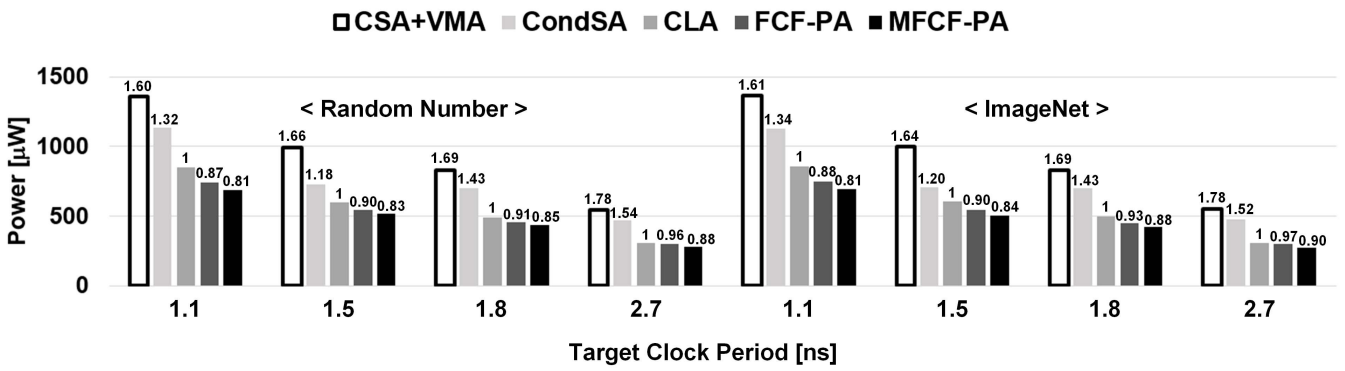


Fig. 11. Comparison of power dissipation among CSA + VMA, CondSA, CLA-based accumulators, proposed FCF-PA, and MFCF-PA. Numbers above bars in the figures: normalized power to the CLA-based accumulator.

TABLE I

EXTRA CLOCK CYCLES FOR THE PROPOSED ACCUMULATORS. BOTH CONVOLUTIONAL AND FULLY CONNECTED LAYERS OF ALEXNET ARE CONSIDERED. BASELINE IS CLA-BASED ACCUMULATOR

Target Clock Period	Extra Clock Cycles (% of Accumulation Cycles)	
	FCF-PA	MFCF-PA
1.1	5 (0.07%)	6 (0.08%)
1.5	3 (0.04%)	4 (0.05%)
1.8	2 (0.03%)	3 (0.04%)
2.7	1 (0.01%)	2 (0.03%)

TABLE II

EXTRA CLOCK CYCLES FOR THE PROPOSED MAC UNITS. BOTH CONVOLUTIONAL AND FULLY CONNECTED LAYERS OF ALEXNET ARE CONSIDERED. BASELINE IS MAC + CLA

Pipeline Stages in Baseline MAC	Extra Clock Cycles (% of Accumulation Cycles)	
	FCF-MAC + CLA	FCF-MAC + MFCF-PA
2	0	5 (0.07%)
3	0	8 (0.11%)
4	0	13 (0.18%)

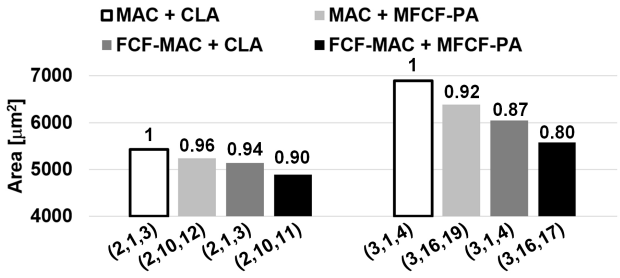


Fig. 12. Comparison of MAC area between conventional and proposed pipelining methods. (a,b,c) along the x-axis implies that the MAC unit consists of an "a"-stage column addition and a "b"-stage final addition. The MAC unit is composed of the "c"-stages in total. Numbers above bars in the figures: normalized area to the "MAC + CLA" configuration.

Table I shows additional clock cycles required to compute the final result when the CLA-based accumulator is changed to the proposed PAs. The FCF-PA requires additional 1–5 clock cycles depending on the target clock periods. The MFCF-PA requires one additional clock cycle compared with the FCF-PA (Section III-B). Considering the average number of accumulations [363 (ConvLayer1) +1200 (ConvLayer2) +2304 (ConvLayer3) +2 × 1728 (ConvLayer4,5) +43 264 (FCLayer1) +2 × 4096 (FCLayer2,3)/8 = 7347] for the computation of convolutional and fully connected layers of AlexNet, the extra clock cycle overhead is 0.01%–0.08%, which is negligible.

Figs. 12 and 13 show the area and power consumption of the MAC units with the conventional and proposed pipelining methods. The baseline is the MAC architecture with the conventional pipelining and the CLA for the final adder (MAC + CLA). When the accumulator is changed to MFCF-PA in the four-stage pipelined MAC configuration (MAC + MFCF-PA), the area and the power consumption are reduced by 8% and 3%, respectively. Applying the FCF pipelining method to the column addition architecture in the MAC, and using the CLA as the final adder (FCF-MAC + CLA), the area and the power consumption decreased by 13% and 17%, respectively. When the proposed scheme is applied to both the column addition and accumulation stages (FCF-MAC + MFCF-PA), the area and the power consumption are reduced by 20% and 19%, respectively, compared with the baseline design. FCF pipelining method is very helpful for power saving, because the power consumed by the clock network and the flip-flops occupies a large portion of the total power consumption. For example, in the four-stage "MAC + CLA" configuration, the power consumed by the clock network and the flip-flops is 70% of the total power, and the proposed scheme removes 28% of the total flip-flops. As with the accumulator-only case, the power dissipation of the MAC is similar in both the cases when the random values generated from the PRNG are fed and when the actual ImageNet data set is fed.

Table II shows additional clock cycles required for the final result when the conventional MAC architecture (MAC + CLA) is converted to the proposed MAC architectures. When the MAC architecture is configured as "FCF-MAC + CLA," the same number of clock cycles is required to compute the final MAC result compared with the baseline design. When the MFCF-PA is used as an accumulator (FCF-MAC + MFCF-PA), it requires 5–13 more clock cycles compared with the baseline. The number of additional clock cycles, however, is only 0.07%–0.18% of the average total clock cycles for the MAC operation in the convolutional and fully connected layers of AlexNet.

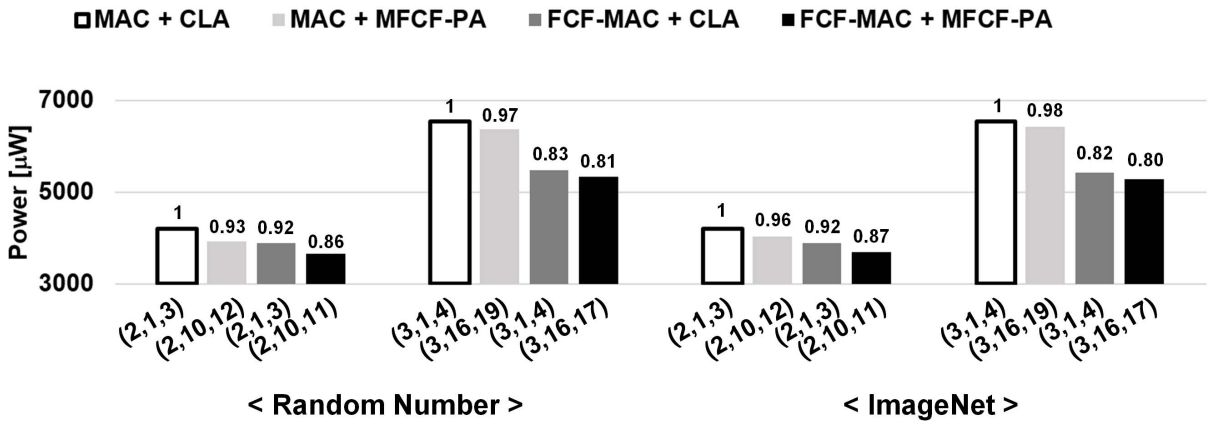


Fig. 13. Comparison of MAC power consumption between conventional and proposed pipelining methods. (a,b,c) along the x-axis implies that the MAC unit consists of an "a"-stage column addition and a "b"-stage final addition. The MAC unit is composed of the "c"-stages in total. Numbers above bars in the figures: normalized power to the "MAC + CLA" configuration.

VI. Conclusion

We introduced the FCF pipelining method in this paper. In the proposed scheme, the number of flip-flops in a pipeline can be reduced by relaxing the feedforward-cutset constraint, thanks to the unique characteristic of the machine learning algorithm. We applied the FCF pipelining method to the accumulator (FCF-PA) design, and then optimized the power dissipation of FCF-PA by reducing the chance of undesired data transitions (MFCF-PA). The proposed scheme was also expanded, and applied to the MAC unit (FCF-MAC). For the evaluation, the conventional and proposed MAC architectures were synthesized in a 65-nm CMOS technology. The proposed accumulator showed the reduction of area and the power consumption by 17% and 19%, respectively, compared with the accumulator with the conventional CLA adder-based design. In the case of the MAC architecture, the proposed scheme reduced both the area and power by 20%. We believe that the proposed idea to utilize the unique characteristic of machine learning computation for more efficient MAC design can be adopted in many neural network hardware accelerator designs in the future.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[2] K. Simonyan and A. Zisserman. (2014). "Very deep convolutional networks for large-scale image recognition." [Online]. Available: https://arxiv.org/abs/1409.1556

[3] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 1–9.

[4] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2013, pp. 6645–6649.

[5] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 577–585.

[6] 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.

[7] 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 *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC)*, Feb. 2017, pp. 246–247.

[8] C. S. Wallace, "A suggestion for a fast multiplier," *IEEE Trans. Electron. Comput.*, vol. EC-13, no. 1, pp. 14–17, Feb. 1964.

[9] L. Dadda, "Some schemes for parallel multipliers," *Alta Frequenza*, vol. 34, no. 5, pp. 349–356, Mar. 1965.

[10] P. F. Stelling and V. G. Oklobdzija, "Implementing multiply-accumulate operation in multiplication time," in *Proc. 13th IEEE Symp. Comput. Arithmetic*, Jul. 1997, pp. 99–106.

[11] K. K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*. New Delhi, India: Wiley, 1999.

[12] T. T. Hoang, M. Sjalander, and P. Larsson-Edefors, "A high-speed, energy-efficient two-cycle multiply-accumulate (MAC) architecture and its application to a double-throughput MAC unit," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, no. 12, pp. 3073–3081, Dec. 2010.

[13] W. J. Townsend, E. E. Swartzlander, and J. A. Abraham, "A comparison of Dadda and Wallace multiplier delays," *Proc. SPIE, Adv. Signal Process. Algorithms, Archit., Implement. XIII*, vol. 5205, pp. 552–560, Dec. 2003, doi: 10.1117/12.507012.

[14] M. Courbariaux, Y. Bengio, and J.-P. David, "BinaryConnect: Training deep neural networks with binary weights during propagations," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 3123–3131.

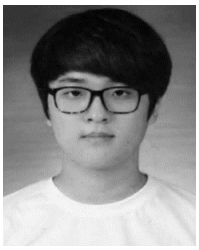
[15] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet classification using binary convolutional neural networks," in *Proc. Eur. Conf. Comput. Vis.* Springer, 2016, pp. 525–542.

[16] M. Gao, J. Pu, X. Yang, M. Horowitz, and C. Kozyrakis, "Tetris: Scalable and efficient neural network acceleration with 3d memory," in *Proc. 22nd Int. Conf. Archit. Support Program. Lang. Oper. Syst.*, 2017, pp. 751–764.

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

Sungju Ryu (S'16) received the B.S. degree in electrical engineering from Pusan National University, Busan, South Korea, in 2015. He is currently working toward the Ph.D. degree at the Department of Creative IT Engineering, Pohang University of Science and Technology, Pohang, South Korea.

His current research interests include low-power machine learning hardware accelerator, adaptive/resilient circuits, and near-threshold voltage circuit design.



Naebeom Park received the B.S degree in electrical engineering from the Pohang University of Science and Technology, Pohang, South Korea, in 2016, where he is currently working toward the Ph.D. degree in creative IT engineering.

His current research interests include neural network data flow and accelerator.

Jae-Joon Kim (M'04) received the B.S. and M.S. degrees from Seoul National University, Seoul, South Korea, in 1994 and 1998, respectively, and the Ph.D. degree from Purdue University, West Lafayette, IN, USA, in 2004.

From 2004 to 2013, he was a Research Staff Member with the IBM T. J. Watson Research Center, contributing to POWER6 and POWER7 microprocessor design. He is currently an Associate Professor at the Department of Creative IT Engineering and Electrical Engineering, Pohang University of Science and Technology, Pohang, South Korea, where he is involved in neuromorphic circuits, machine learning hardware accelerator, and flexible device/circuits. His research interests included 3-D VLSI integration, robust SRAM cache memory design, and low-voltage circuit designs.