Digital Neuron: A Hardware Inference Accelerator for Convolutional Deep Neural Networks

Hyunbin Park, Dohyun Kim, and Shiho Kim

***Abstract*— We propose a Digital Neuron, a hardware inference accelerator for convolutional deep neural networks with integer inputs and integer weights for embedded systems. The main idea to reduce circuit area and power consumption is manipulating dot products between input feature and weight vectors by Barrel shifters and parallel adders. The reduced area allows the more computational engines to be mounted on an inference accelerator, resulting in high throughput compared to prior HW accelerators. We verified that the multiplication of integer numbers with 3-partial sub-integers does not cause significant loss of inference accuracy compared to 32-bit floating point calculation. The proposed digital neuron can perform 800 MAC operations in one clock for computation for convolution as well as full-connection. This paper provides a scheme that reuses input, weight, and output of all layers to reduce DRAM access. In addition, this paper proposes a configurable architecture that can provide inference of adaptable feature of convolutional neural networks. The throughput in terms of Watt of the digital neuron is achieved 754.7 GMACs/W.**

I. Introduction

DEEP learning have revolutionized the computer industry over the last decades, pushing image recognition beyond human accuracy [1]. However, the computational effort of current neural networks depends on power-hungry parallel floating point processors or GP-GPUs. Recent developments in hardware accelerators have reduced energy consumption significantly, however, devices for embedded systems such as drones, smart robots, autonomous vehicles, and etc. are still facing hard limitations caused by power budget for ultralow power applications.

A number of studies have adopted a strategy to perform the training using a desktop or a server and an inference using hardware accelerator on embedded devices [2]-[4]. To utilize the hardware inference accelerator in the embedded system, the neural weights with single- or half-precision Floating Point format (FP32 or FP16) should be converted to integer numbers.

The major issues in the hardware accelerators for embedded systems are accuracy loss, power consumption, circuit area, and throughput. To improve throughput and power consumption, a number of studies [5]-[8] has reduced bit-width of input and weight than conventional FP32. The reduction in bit-width allows circuit area of computational engine, which allows more computational engines to be mounted on a limited area. Therefore, it can raise throughput. P. Gysel [5] suggested to quantize inputs and weights into 8-bit in forward propagation, and it results in accuracy degradation of less than 1 % in LeNet-5 [9], AlexNet [10], and SqueezeNet [11]. BinaryConnect [12] reduced the digit of weights even further to 1-bit. However, limiting bit-width of weights to 1-bit degrades ImageNet Top-1 inference accuracy by 21.2 % in AlexNet [10]. Therefore, it is important to design the hardware accelerators with optimum

TABLE I
Simulated inference accuracy of MNIST handwritten number in LeNet-5 [9] according to resolution of weights, where pre-trained weights in the network with FP32 is quantized to integer weights.

	Inference accuracy
32-bit floating-point representation	99.10 %
8-bit integer weights	99.10 %
5-bit integer weights	98.95 %
Proposed 5-bit integer weights with two partial sub-integers	98.92 %
Proposed 8-bit integer weights with three partial sub-integers	99.10 %

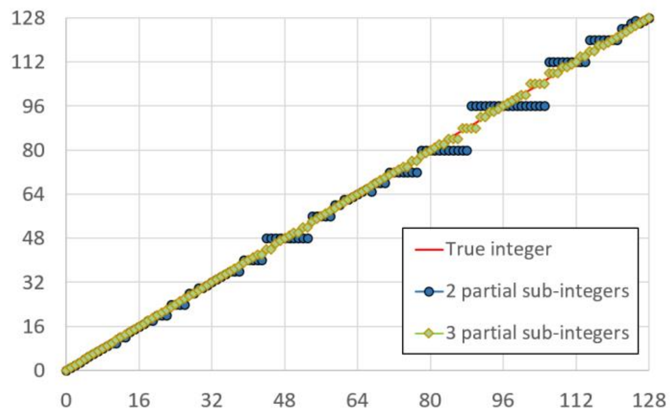


Fig. 1 Representation of integers from 0 to 128 partitioned into 2 or 3 partial sub-integers.

"This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the ICT Consilience Creative program(IITP-2018-2017-0-01015) supervised by the IITP(Institute for Information & communications Technology Promotion)"

The authors are with the School of Integrated Technology, Yonsei University, Seoul, Korea (e-mail: bin9000@yonsei.ac.kr; kimdh5032@naver.com; shiho@yonsei.ac.kr).

bit-width for accuracy to be within the target range.

We simulated inference accuracy of MNIST handwritten number in LeNet-5 [9] according to bit-width of weights in Table I, where pre-trained weights in the network with FP32 is quantized to integer weights. Quantization into 8-bit integer weights does not degrade inference accuracy compared to the FP32. In addition, accuracy degradation in the case of the network even with 5-bit integer weights is negligible.

The artificial neuron in Integrate and Fire model [13] performs a convolution by calculating inner product of input and weight vectors

$$\mathrm{Out}[x][y] = \mathbf{F}\left(b + \sum_{i=0}^{R-1}\sum_{j=0}^{S-1} \mathbf{X}[x+i][y+j]\cdot \mathbf{w}[x+i][y+j]\right)$$

$$= \mathbf{F}(\vec{\mathbf{X}} \bullet \vec{\mathbf{w}}^{T} + b) \qquad (1)$$

where **X**, **w**, b, **F** and Out are an input vector, a weight vector, a bias, an activation function, and an output, respectively. If multiplication of each component in the inner product is implemented by arithmetic shift and addition, a multiplier circuit can be designed more simply than the booth multiplier. The booth multiplication with an 8-bit weight produces five partial products including the addition of 1 for the 2's complement [14]. However, neural networks do not require exact algebraic calculations. Therefore, we quantize the weight into sum of a limited number of 2^ns, which we name 'partial sub integer', for example, $7 \cdot X = 8 \cdot X - X$ with two sub-partial integers. Fig. 1 shows representation of 8-bit integers partitioned into 2 or 3 partial sub-integers, which is expressed by

$$w \cdot X = X \cdot \sum_{i=1}^{N}(2^{n_i} \cdot w_i) \qquad (w_i \in -1,0,1) \qquad (2)$$

where N can be 2 or 3. It allows a multiplier to be implemented by only two- or three-barrel shift circuits. The maximum errors caused by the quantization with limiting partial sub-integers into two and three are approximately 9 % and 2 %. Despite the calculation error in multiplication, degradation caused by quantization into two and three partial sub-integers is negligible, as shown in the simulation in Table I. This paper employs the strategy limiting partial sub-integers to increase throughput and power consumption.

The study of [15] states that DRAM access consumes 200 times more power than register access. The inference accelerators with spatial architecture reported in [16]-[18] allows a computational engine to deliver input, weight, and Partial sum of output (Psum) to an adjacent computational engine. Such spatial architectures reuses input, weight, and Psum, therefore helping to reduce DRAM access. However, computation of Fully-Connected (FC) layer does not require input and weight reuse, and thus spatial architecture has structural disadvantage in the FC computation. Also, works in [16]-[18] does not reuse outputs of layers. It requires additional DRAM access for load and store outputs of layers.

This paper proposes a digital neuron, a configurable inference accelerator for embedded systems. We improve throughput with reducing circuit area of computational engine. The proposed digital neuron also fully utilizes computational engines with massive parallel structure. In addition, reusing scheme of outputs of layers for reducing DRAM access is presented.

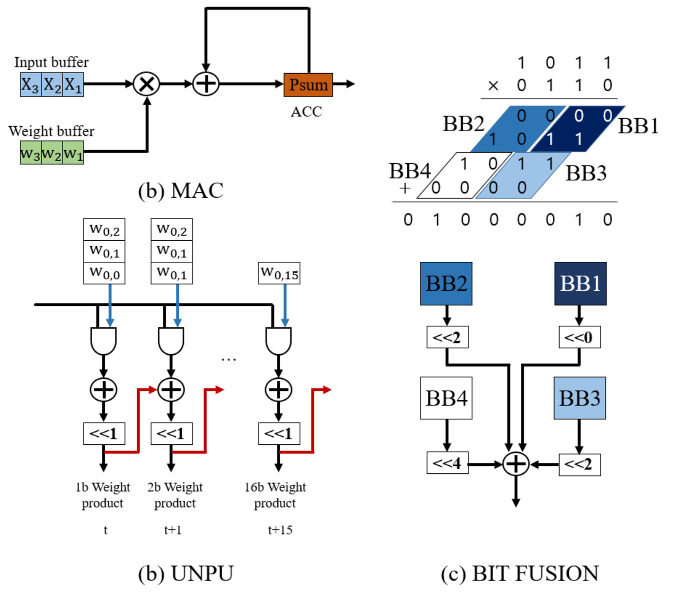


Fig. 2. Computation engine in previous works with Multiply and Accumulation (MAC) and shift-based multiplier. (a) MAC [19]-[21], (b) UNPU [23], (c) BIT FUSION [24].

II. Related Works

This section reviews previous works that performs multiplication with a shift circuit to reduce complexity of computation engine. Also it reviews prior-arts of inference accelerators with spatial architecture, and how they reuse data and perform FC computation.

A. *Previous works with shift-based multiplier*

A number of previous hardware accelerators have adopted Multiply and Accumulation (MAC) structure for computing dot product [19]-[21]. The MAC structure multiplies input and weight with a binary multiplier (e.g. booth multiplier [14]) and accumulates the multiplied value with a binary adder (e.g. Carry Look Ahead adder [22]), as shown in Fig. 2 (a).

To reduce the circuit area of a multiplier, researches in [23], [24] have adopted shift circuit for arithmetic multiplication. UNPU [23] produces 16 partial products with 16 sets of AND gate, binary adder, and 1-bit shift circuit, as shown in Fig. 2 (b). The UNPU can support multiplication with weight of all bit-width from 1-bit to 16-bit. BIT FUSION scheme [24] decomposes, for example, 4-bit multiplication into four 2-bit multiplications, as shown in Fig. 2 (c). With adjusting combination of the decomposed multiplications, it also can support flexible bit-width of weight.

The proposed digital neuron, to reduce circuit area and power consumption, performs 5-bit or 8-bit multiplication with 2- or 3-barrel shift circuit, respectively, as described in introduction section in this paper. In addition, this paper proposes massive parallel multi-operand adder that can improve circuit area and power consumption.

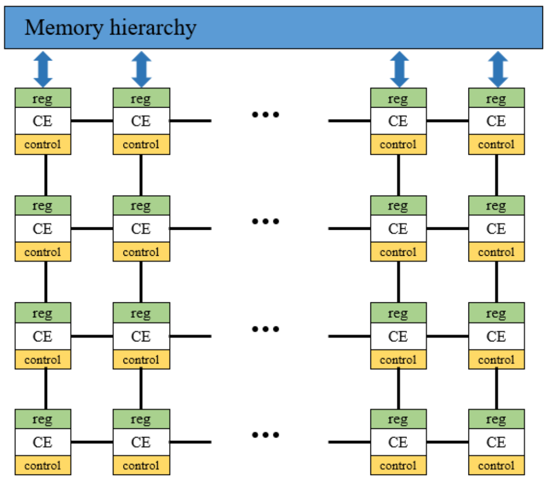


Fig. 3. Computational engine array in previous works in hardware accelerators with spatial architecture [16]-[18]

B. *Previous works in hardware accelerators with spatial architecture*

Previous hardware accelerators, Eyeriss [16], DSIP [17], and COSY [18] have proposed spatial architecture. Computation engine of the spatial architecture delivers input, weight, and Psum to an adjacent, as shown in Fig. 3. The data delivery between computational engines allows to reuse input, weight, and Psum. Therefore, in case of computation of convolution with filter sweep, the data reuse helps to reduce DRAM access. However, in case of computation of FC layer, data reuse of weight is not required, since there is no sweep. Therefore, only one column of the engine array is utilized for computation of FC layer. The massive parallel structure of the digital neuron allows full utilization not only for convolutional computation but also fully-connected computation. The studies in [16]-[18] reuse Psum, but does not reuse output of each layer. In this paper, an architecture for reusing output of layers is presented for DRAM access reduction.

III. Architecture of The Proposed Digital Neuron

In this section, we present architecture of the proposed digital neuron in case of employing 8-bit integer weights which is produced by the addition of three partial sub-integers as indicated in Eq. (2). In addition, we propose a configurable system architecture that can adjust filter size.

A. *Architecture of the computational engine*

The artificial neuron of the proposed digital neuron is a computational engine that computes the dot product of two vectors digitally. This section shows an architecture of the digital neuron adopting unsigned 8-bit integer inputs and signed 8-bit integer weights with three partial sub-integers. Fig. 4 shows block diagram of the proposed digital neuron. The digital neuron performs the convolution with a 3-dimensional filter, in which we denote the number of the input channels by N_{ch} and the number of the filtered inputs in one channel by N.

The Neural element block in Fig. 4 calculates 2-D convolution of one channel in input feature map with N inputs and N weights of each channel. The neural element consists of

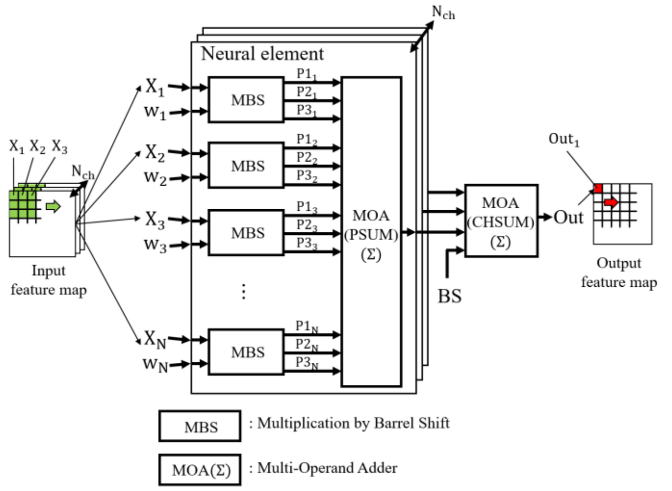


Fig. 4. Block diagram of computational engine of the proposed digital neuron. The digital neuron calculates convolution with a 3-dimensional filter. The digital neuron contains M neural elements, and the neural element performs convolution with 2-deminsional filter of one input channel. Each MBS block contains two barrel shift circuits which produce two partial products. The MOA blocks perform arithmetic sum of input numbers.

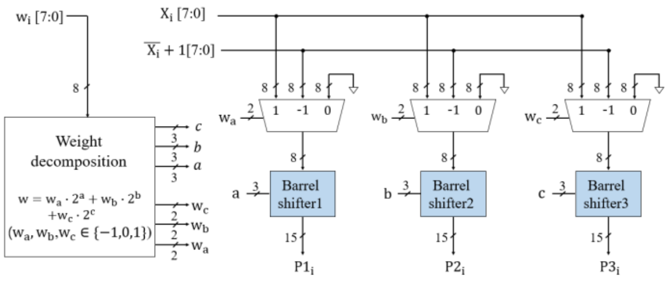


Fig. 5. Block diagram of the MBS block of the proposed digital neuron. The MBS block performs arithmetic multiplication. Three barrel shifters produce three partial sub-integers.

Multiplication by Barrel Shift (MBS) block and Multi-Operand Adder (MOA) block. The MBS block performs arithmetic multiplication, and the MOA block aggregates the outputs of the MBS blocks.

The MBS block calculates multiplication expressed in Eq. (2). Each MBS block contains three barrel shift circuits, which produce three partial products P1, P2, and P3, which are $w_a 2^a$, $w_b 2^b$, and $w_c 2^c$ in Eq. (2), respectively. The MBS block receives a 8-bit weight, and produce six signals, a, b, c, w_a, w_b, and w_c for controlling barrel-shift and mux circuits. The MBS block also receives an input, a negatized input, and 8-bit zeros. Mux circuits select one of inputs based on w_i signal. The barrel shift circuit multiplies the output of the mux circuit by 2^n.

The MOA(PSUM) block collects 3N sub-partial integers (i.e. outputs of the MBS blocks), and aggregates them in massive-parallel. Fig. 6 shows schematic view of the proposed MOA(PSUM) block, where N is 25. The operational principle of the MOA block is similar with the Wallace tree adder [25]. The MOA block groups three bits in the same column of P1, P2 and P3 vectors along the vertical direction into one group. After that, full adder arrays in each stage reduce the number of input vectors by two-thirds, and simultaneously expands bit-width of them by one bit. At the output of the stage 10, the number of P1 and P2 vectors is reduced to two. Finally, the two vectors are

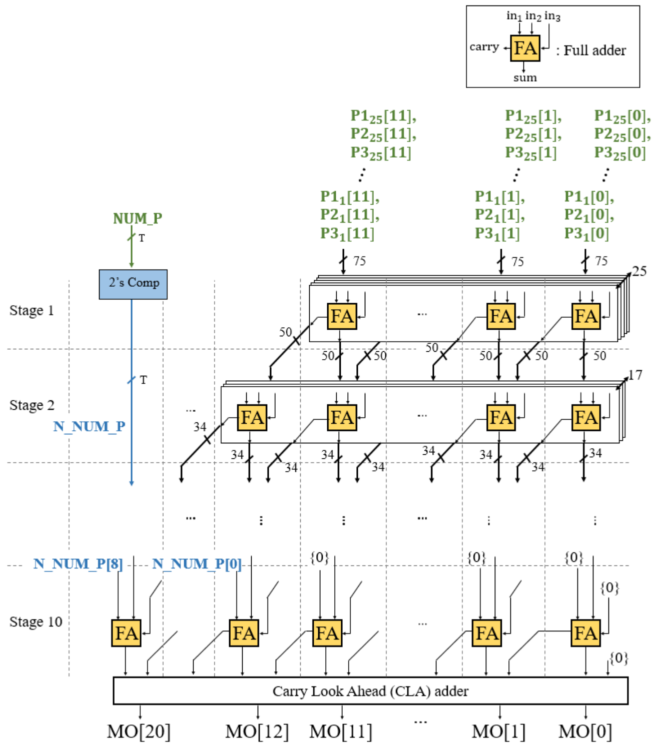


Fig. 6. Schematic view of the proposed MOA(PSUM) block, where N is 25. Full adder arrays in each stage reduces the number of partial elements by two-thirds, and simultaneously expands bit-width of them by one bit. N_NUM_P, two's complemented-NUM_P, enables calculation of the sum of individual signed inputs. NUM_P and N_NUM_P are added in the last stage, since they are delayed than P1, and P2.

added by a Carry Look Ahead (CLA) adder. Finally, the outputs of all neural elements and a bias BS are collected and summed by MOA(CHSUM) block.

To permit calculation of the sum of numbers with individual signs, a sign-extension of inputs of the MOA to 18-bits would be required, where the 18-bit is the bit-width of the output of the MOA. However, the sign-extension increases circuit area by approximately 21 %. Instead of sign-extending, we negatize NUM_P, and add it to from 12-bit to 18-bit in Stage 10, where NUM_P is the number of negative partial sub-integers. It requires only 2's complement circuit, and allows the calculation of the sum of numbers with individual signs. The principle of the simplified sign-extension is described in Appendix with Fig. A1.

The proposed MOA circuit (N=25) reduces critical path and total gates by 42 % and 36 % compared to the summation circuit with binary-adder-based-tree-structure with 75 inputs.

B. *System architecture of the proposed inference accelerator*

The proposed inference accelerator employs a computational engine composed of 8 5×5 Neural elements, and we allow it to accumulate according to the ACC_SEL signal, as shown in Fig. 7. The computational engine is named Neural Tile (NT).

Fig. 8 shows a system architecture of the proposed inference accelerator. Four NTs, which are NT1, NT2, NT3, and NT4, compute 3-D convolution of convolutional layer, and dot product of two vectors for fully connected layer. NTs compute

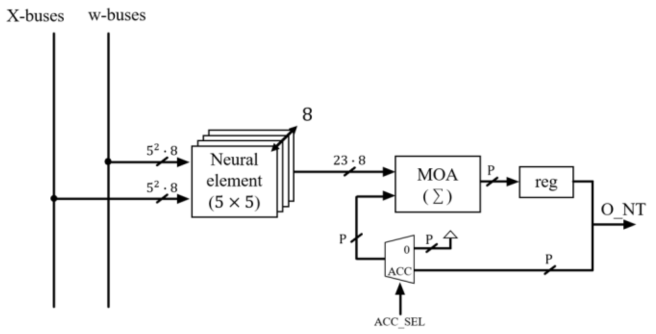


Fig. 7. Block diagram of the system architecture of the proposed inference accelerator.

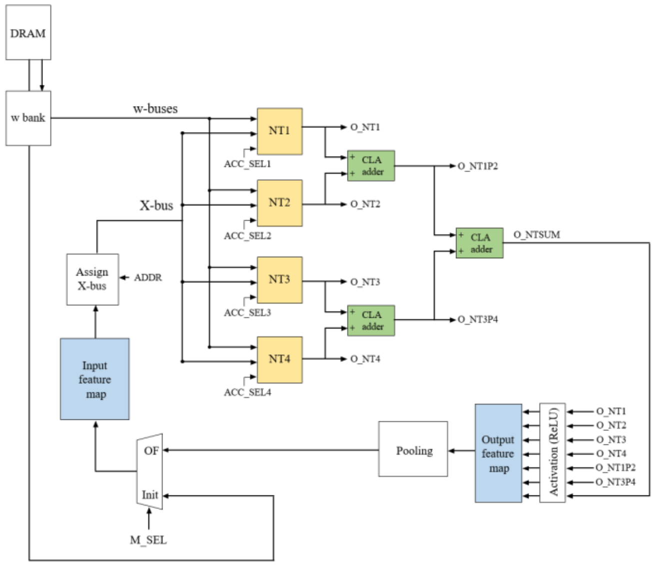


Fig. 8. Block diagram of the system architecture of the proposed inference accelerator.

3-D convolution and full-connection in one clock.

Weights of all layers are loaded from DRAM to w bank registers and are reused throughout every inference. Inputs is loaded from DRAM to Input feature map registers for one inference. Four outputs of the NTs, which are O_NT1, O_NT2, O_NT3, and O_NT4, are added by two CLA adders. The outputs of the two CLA adders, O_NT1P2 and O_NT3P4, are added by the other CLA adder, and produce the O_NTSUM. We will describe detail utilization of these signals in the sub-section C. These calculation results of computational engines are stored in Output feature map registers after applying activation in Activation (ReLU) block. Pooling operation is performed in outputs of the Output feature map register. The pooled outputs are stored in Input feature map.

The outputs of the Input feature map are assigned to the inputs of from NT1 to NT4 block. Fig. 9 shows data transfer from Input feature map to X-bus and behavior of w bank. Elements of filtered area that requires dot product computation is assigned to X-bus from the Input feature map. If one clock is

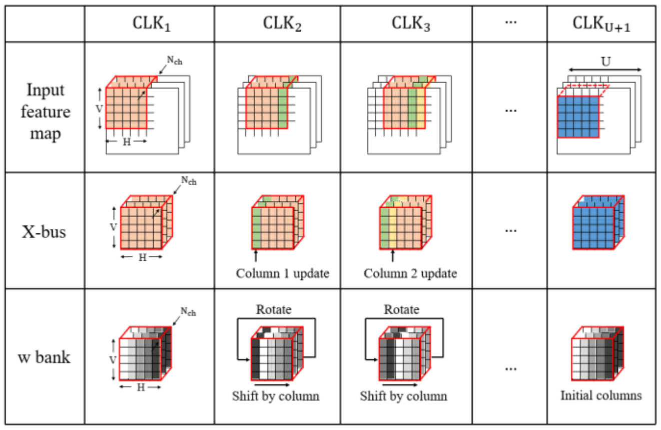


Fig. 9. Data transfer from Input feature map to X-bus, and column rotation of w bank. The Assign X-bus block assigns elements to X-bus with controlling address signal(ADDR).

increased, the filtered area in the input feature map must be shifted one column to the right. This is implemented by updating one column in X-bus; deleting the leftmost column in the previous clock in the input feature map and, in its place, loading the rightmost column of the next clock. It can be performed by increasing the address (ADDR) of the column of the Input feature map by 5. Columns that are not updated do not consume dynamic power. Therefore, it helps to reduce power consumption. The w bank rotates to the right one by one according to the sweep of the filter. It is implemented with shift registers, connecting them in the horizontal direction, and the rightmost columns to the leftmost columns. At the beginning of the next row of the filter sweep, the whole filtered area of the input feature map itself is assigned to the X-bus, and w-bus is updated with initial columns of CLK1.

C. *Configurable scheme adjusting filter size*

Fig. 10 shows example of applicable different filter sizes in the proposed inference accelerator depending on input area of NTs and the combination of adders. The case 1 shows an example of applying a $\mathbf{5 \times 5 \times 32}$ filter, in which each of the four NTs receives a quarter of the filter divided in the depth direction, and calculates dot product of the quarter. Then, the outputs of the four NTs are summed by the three CLA adders. In this case, the Output feature map collects four O_NTSUMs and delivers them to the Pooling block.

Case 2 shows an example of applying a $\mathbf{7 \times 7 \times 16}$ filter, where NT1 and NT2 receive the half and the remainder of the $\mathbf{7 \times 7}$ filter, that are $[\mathbf{0{:}\,24}] \times [\mathbf{0{:}\,7}]$ and $[\mathbf{0{:}\,48}] \times [\mathbf{0{:}\,7}]$. Likewise, NT3 and NT4 receive the half and the remainder of the $\mathbf{7 \times 7}$ filter, that are $[\mathbf{0{:}\,24}] \times [\mathbf{8{:}\,15}]$ and $[\mathbf{0{:}\,48}] \times [\mathbf{8{:}\,15}]$. The four outputs of the four NTs are summed by the three CLA adders. In this case, also, the output feature map collects four O_NTSUMs and delivers them to the Pooling block.

Case 3 shows an example of applying a $\mathbf{9 \times 9 \times 8}$ filter, in which NT1, NT2, NT3, and NT4 receive four fraction of the filter, that are $[\mathbf{0{:}\,24}] \times [\mathbf{0{:}\,7}]$, $[\mathbf{25{:}\,49}] \times [\mathbf{0{:}\,7}]$, $[\mathbf{50{:}\,74}] \times [\mathbf{0{:}\,7}]$, and $[\mathbf{75{:}\,80}] \times [\mathbf{0{:}\,7}]$ of the filter. In this case, also, the output feature map collects four O_NTSUMs and delivers them to the Pooling block.

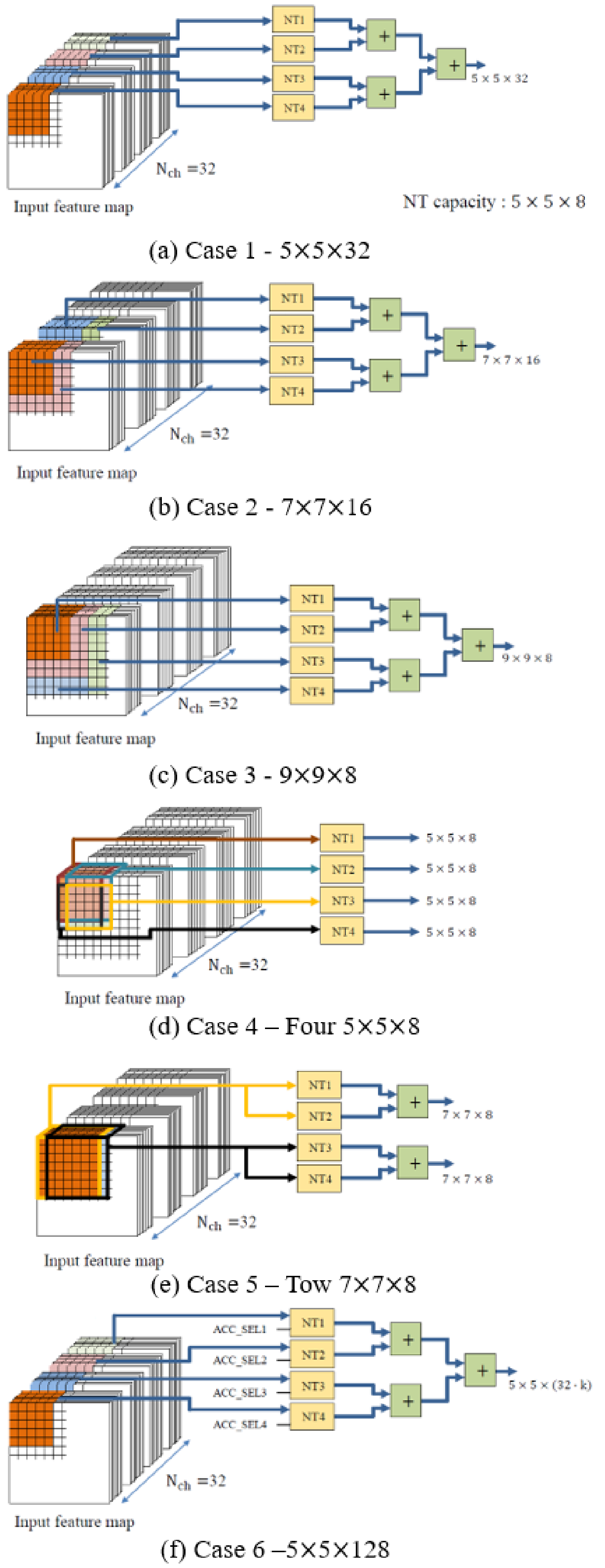
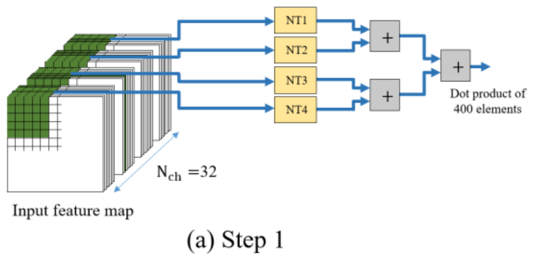


Fig. 10. Example of applicable different filter sizes in the proposed inference accelerator.

Case 4 shows an example of applying a $\mathbf{5 \times 5 \times 8}$ filter, where each of the four NTs receives a separate $\mathbf{5 \times 5 \times 8}$ area in the input feature map. The four NTs calculate four $\mathbf{5 \times 5 \times 8}$ convolution in parallel, and four outputs of the NTs, O_NT1,

TABLE II
Performance of the implementation of the proposed inference accelerator in FPGA DE2-115 board for MNIST handwritten number recognition operating LeNet-5 [6].

Performance of the proposed system architecture	
Inference accuracy	98.92 %
Number of clocks required in computing dot product of NTs (25 MHz)	1 clock
Number of clocks/computation time (25 MHz) for inference of one image	2,384 clocks/ 95 µs
Frame rate (25 MHz)	19 k frame/s



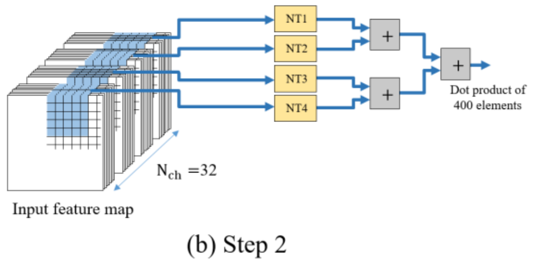


Fig. 11. Computation steps of fully connected layer. (a) step1 (b) step2.

O_NT2, O_NT3, and O_NT4 are stored in the output feature map. The output feature map collects four O_NTs and delivers them to the Pooling block.

Case 4 shows an example of applying a $\mathbf{7 \times 7 \times 8}$ filter, where two groups of two NTs receive separately two $\mathbf{7 \times 7 \times 8}$ area in the input feature map. The outputs of NT1 and NT2 are added by the CLA adder, and the other outputs of NT3 and NT4 are added by the CLA adder. These two outputs of the CLA adders, O_NT1P2 and O_NT3P4 are stored in the output feature map. The output feature map collects four O_NT1P2/O_NT3P4s and delivers them to the Pooling.

Case 6 shows an example of applying a $\mathbf{5 \times 5 \times 128}$ filter, where the depth of the filter exceeds the summed depth of four NTs. In this case, the ACC_SEL signal allows the outputs of NTs to be accumulated. The depth of the filter is 128. Therefore, it is accumulated four times in total. However, in this case, all signals of X-bus and w-bus should be updated. Thus, it permits the calculation, but increases power consumption.

D. Computation of fully connected layer

Fig. 11 shows the computation steps of fully connected layer. The 1-D vectors of the fully-connected layer is stored in $\mathbf{5 \times 5 \times 32}$ elements of the Input feature map. In step 1 in Fig. 11 (a), the digital neuron calculates dot products of 400 elements, and, in step 2 in Fig. 11 (b), calculates dot products of the next 400 elements. The proposed massive parallel architecture does

TABLE III
Comparison of performance with an artificial neuron in a prior art computing dot product of two vectors with 50 components.

	Adder tree with booth multiplier (N=25) [4]	YodaNN (N=25) [16]	Proposed digital neuron (N=25)	
Weight precision	5-bit (exact)	1-bit	5-bit integers (2 partial sub-integers)	8-bit integers (3 partial sub-integers)
Activation precision	8-bit	8-bit	8-bit	8-bit
Critical path	57	40	41	46
Total gates	15,688	4,715	8,201	12,502
Simulated power consumption	4.940 mW	1.452 mW	2.576 mW	3.953 mW

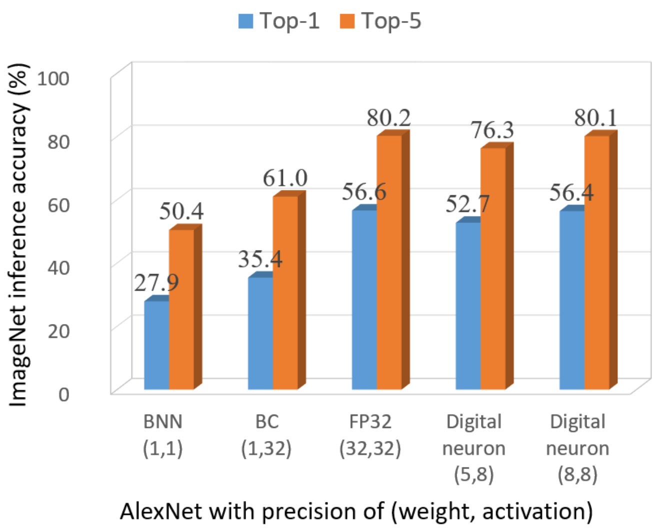


Fig. 12. Comparison of ImageNet inference accuracy of the proposed digital neuron with previous works, FP32, Binarized Neural Network (BNN) [26], and BinaryConnect (BC) [12]. In the proposed digital neuron, 5-bit weight comprises two partial sub-integers and 8-bit weight comprises three partial sub-integers.

not degrade throughput in computation of FC layer.

IV. Experimental Results

In this section, we implement of a CNN network, Lenet-5 [9] for MNIST handwritten number recognition using the proposed digital neuron in the FPGA DE2-115 board and show the performance of the implementation.

We trained weights with LeNet-5 with FP32 precision. To integerize the pre-trained weights, we multiplied all weights by 16 and quantized the weights into 5-bit integer weights with two partial sub-integers. The quantized integer weights are delivered to the DRAM of the FPGA board. Since we multiplied the weights by 16 to integerize the weights, we truncated the least significant 4-bits of the output of the digital neuron to normalize them again.

Table II summarizes performance of the implemented hardware inference accelerator. It achieves 98.95 % of the inference accuracy when training the MNIST database. The number of clocks required in computing dot product of NTs is only one clock in 25 MHz. The number of clocks and

TABLE IV
Performance comparison of inference accelerators with previous works

	DSIP [16]	ConvNet [27]	Eyeriss [17]	This work (Simulation)
Weight precision	16-bit fixed point	16-bit fixed point	16-bit fixed point	INT5
Activation precision	16-bit fixed point	16-bit fixed point	16-bit fixed point	INT8
Number of MACs	168	256	64	800
Technology	65 nm	40 nm	65 nm	65 nm
Operating voltage	1.0 V	0.9 V	1.2 V	1.2 V
power consumption	278 mW	274 mW	88.6 mW	212 mW
Frequency	250 MHz	204 MHz	250 MHz	200 MHz
Throughput	23.1 GMACS	52.2 GMACS	30.1 GMACS	160 GMACS
GMACS/W	83.1 GMACS/W	190.6 GMACS/W	136.8 GMACS/W	754.7 GMACS/W

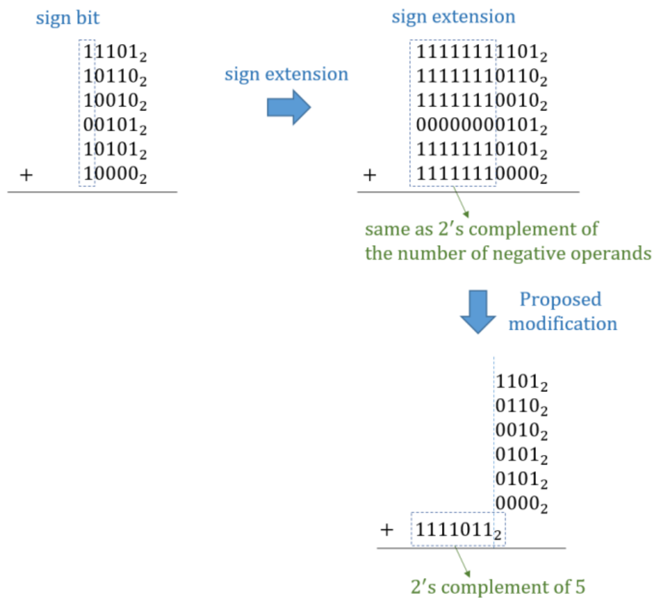


Fig. A1. An example of summation of six 5-bit binary numbers. This example describes summation of extended sign bits is same as 2's complement of the number of negative operands. Therefore, sign extension can be replaced to addition of one binary number.

computation time for inference of one image is 2,384 clocks in 25 MHz, that is 95 µs. The frame rate for inference is 19 k frame/s.

V. Discussion

In this section, we compare performance of the proposed computational engine and inference accuracy with prior works. In addition, we compare performance of the proposed inference accelerator with prior works.

For comparison of performance, we simulate three kinds of dot product-computing artificial neurons, adder tree with booth multiplier, YodaNN (1-bit weight), and the proposed neural element. Table III summarizes the comparison. The proposed neural element surpasses the adder tree with booth multiplier in gate delay and power consumption by ~1.4x and ~1.9x, respectively. Also, it can be estimated circuit area is reduced with comparison of total gates of the digital neuron and the adder tree scheme. However, the proposed neural element consumes ~1.8x more power than YodaNN. However, bit-width of the proposed digital neuron is five times more than YodaNN.

Quantization of weights into binary bit results in a significant reduction in the computational power and storage requirements. However, excessive reduction in bit-width of weights results in degradation of inference accuracy. Fig. 12 shows comparison of ImageNet inference accuracy of the proposed digital neuron with previous works, FP32, Binarized Neural Networks (BNN) [26], and BinaryConnect (BC) [12]. As shown in Fig. 12, quantization of weights into 1-bit causes inference accuracy to be decreased more than 20 % compared to FP32. However, AlexNet with the proposed digital neuron (5,8) gives additional error of 3.9%/3.9% in Top-1/Top-5 accuracy, compared to FP32. The additional error can be reduced by increasing precision. AlexNet with the proposed digital neuron (8,8) gives additional error of only 0.2%/0.1% in Top-1/Top-5 accuracy.

Table IV shows comparison of performance of the simulated inference accelerators with the implementations of previous works, simulation environment is HSPICE in 65 nm CMOS process technology at operation voltage of 1.2 V at 200 MHz frequency. The digital neuron achieved the simulated power consumption of 212 mW while operating convolutional layer 2 of AlexNet. The throughput of the digital neuron is achieved 160 Giga MACs (GMACS), and GMACS per Watt is achieved 754.4 GMACS/W. Those surpass previous inference accelerators in Tabe IV. Compared with ConvNet [27], throughput per Watt is improved by ~3.96x.

VI. Conclusion

We proposed and implemented a digital neuron, a configurable hardware inference accelerator with integer weights and inputs for embedded systems. This paper proposes a quantization scheme that partitions the integer weight into the limited number of partial sub-integers. Degradation of ImageNet inference accuracy is negligible with the weight partitioned into 3 partial sub-integers, which is ~0.2%. The proposed massive parallel architecture does not degrade throughput in computation of FC layer. With the proposed quantization scheme and massive parallel summation technique, we improved power consumption of the computational engine by ~1.9x compared to the conventional engine. We implemented the digital neuron in FPGA DE2-115 board to verify the operation, and simulated the digital neuron at operating frequency of 200 MHz to compare performance with prior arts. The simulated digital neuron improves throughput per Watt by ~3.96x compared to the previous work.

APPENDIX

In Appendix, the principle of the proposed sign extension of the MOA circuit is described. Fig. A1 shows an example of summation of six 5-bit binary numbers. In the conventional way, negative numbers should extend 1s and positive numbers should extend 0s, as shown in Fig. A1. In a closer look, binary number of the extended 1s is same as -1 and that of the extended 0s is same as 0. Therefore, the summation of the extended bits can be replaced to 2's complement of the number of negative numbers.

REFERENCES

[1] C. R. Qi *et al.*, "Volumetric and multi-view CNNs for object classification on 3D data," in *Proc. CVPR 2016,* Las Vegas, NV, USA, 2016, pp. 5648–5656.

[2] J. Kim, B. Grady, R. Lian, J. Brothers, and J. H. Anderson, "FPGA-based CNN inference accelerator synthesized from multi-threaded C software," in *Proc. 30th IEEE International System-on-Chip Conference (SOCC) 2017,* Munich, Germany, 2017, pp. 268–273.

[3] M. Peemen, A. A. Setio, B. Mesman, and H. Corporaal, "Memory-Centric accelerator design for convolutional neural networks," in *Proc. 2013 IEEE 31th International Conference on Computer Design (ICCD), Asheville,* NC, USA, 2013, pp. 13–19.

[4] C. Zhang *et al*., "Optimizing FPGA-based accelerator design for deep convolutional neural networks," in *Proc. 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays,* Monterey, CA, USA, 2015, pp. 161-170.

[5] P. M. Gysel, "Ristretto: Hardware-oriented approximation of onvolutional neural networks," M.S. thesis, Dept. Electrical and Computer Eng., Univ. of California, CA, USA, 2016.

[6] L. Lai, N. Suda, and V. Chandra, "Deep convolutional neural network inference with floating-point weights and fixed-point activations," in *Proc. 34rd International Conference on Machine Learning (ICML 2017)*, Sydney, Australia, 2017.

[7] P. Judd *et al.*, "Reduced-precision strategies for bounded memory in deep neural nets," in *Proc. 6rd International Conference on Learning Representations (ICLR 2016),* San Juan, Puerto Rico, 2016.

[8] K. Hwang, and W. Sung, "Fixed-point feedforward deep neural network design using weights +1, 0, and -1," in *Proc. 2014 IEEE Workshop on Signal Processing Systems (SiPS),* Belfast, UK, 2014.

[9] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proc. IEEE,* vol. 86, no. 11, pp. 2278-2324, Nov. 1998.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in Proc. 2012 *Advances in neural information processing systems (NIPS 2012),* Lake Tahoe, NV, USA, 2012, pp. 1097-1105.

[11] F. Iandola et al., "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size," arXiv:1602.07360.

[12] M. Courbariaux, Y. Bengio, and J.-P. David, "BinaryConnect: training deep neural networks with binary weights during propagations," in Proc. Advances in Neural Information Processing Systems (NIPS 2015), Montreal, Montreal, Canada, 2015.

[13] R. Jolivet, T. J. Lewis, and W. Gerstner, "Generalized integrate-and-fire models of neuronal activity approximate spike trains of a detailed model to a high degree of accuracy," J. Neurophysiology, vol.92, no. 2, pp. 959-976, Aug. 2004.

[14] R. Hussin *et al.*, "An efficient modified booth multiplier architecture," in *Proc. 2008 International Conference on Electronic Design*, Penang, Malaysia, 2008.

[15] V. Sze, Y. Chen, T. Yang, and J. Emer, "Efficient Processing of Deep Neural Networks: A Tutorial and Survey," arXiv:1845160.

[16] 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.

[17] J. Jo, S. Cha, D. Rho, and I. Park, "DSIP: a scalable inference accelerator for convolutional neural networks," *IEEE J. Solid-State Circuits*, vol. 53, no. 2, pp. 605–618, Feb. 2018.

[18] C. Xin et al., "COSY: an energy-efficient hardware architecture for deep convolutional neural networks based on systolic array," in Proc. *2017 23rd International conference on Parallel and Distributed Systems*, Shenzhen, China, 2017.

[19] T. Nowatzki, *et al*., "Pushing the Limits of Accelerator Efficiency While Retaining Programmability," in Proc. IEEE International Symposium on High Performance Computer Architecture (HPCA 2016), Barcelona, Spain, 2016.

[20] Y. Shen, M. Ferdman, and P. Milder, "Maximizing CNN accelerator efficiency through resource partitioning," in Proc. *2017 International Symposium on Computer Architecture (ISCA 2017),* Toronto, ON, Canada, 2017.

[21] N. Jouppi, *et al*., "In-datacenter performance analysis of a tensor processing unit," in *Proc. International Symposium on computer Architecture (ISCA 2017),* Toronto, ON, Canada 2017.

[22] D. Crawley, and G. Amaratunga, "Pipelined carry look-ahead adder," *Electronics Letters*, vol. 22, no. 12, pp. 661-662, 1986.

[23] J. Lee et al., "UNPU: an energy-efficient deep neural network accelerator with fully variable weight bit precision," *IEEE J. Solid-State Circuits*, early access, 2018.

[24] H. Sharma et al., "Bit fusion: bit-level dynamically composable architecture for accelerating deep neural networks," arXiv:1712.01507v2.

[25] A. Dandapat, S. Ghosal, P. Sarkar, and D. Mukhopadhyay, "A 1.2-ns 16x16-bit binary multiplier using high speed compressors," *J. Electrical and Electron. Eng.*, vol. 4, no. 3, pp. 234-239, 2010.

[26] M. Rastegari, V. Ordonez, J. Redmon, and Ali Farhadi, "XNOR-Net: ImageNet classification using binary convolutional neural networks," in *Proc. 2016 14th European Conference on Computer Vision (ECCV 2016),* Amsterdam, Netherlands, 2016, pp. 525-542.

[27] B. Moons and M. Verhelst, "An energy-efficient precision-scalable ConvNet processor in 40-nm CMOS," *IEEE J. Solid-State Circuits*, vol. 52, no. 4, pp. 903–914, Apr. 2017.