VLSI Implementation of Area-Efficient Parallelized Neural Network Accelerator Using Hashing Trick

Tae Koan Yoo, Jong Kang Park and Jong Tae Kim
Department of Electrical and Computer Engineering
Sungkyunkwan University
Suwon, Republic of Korea
{tk1star2, jkpark1, jtkim}@skku.edu

Abstract— **Recently, neural network accelerator has adopted model compression for embedded devices which have limited constraint such as area and power dissipation. However, area efficiency has not been seriously considered for those performances. Among model compression techniques, hashing trick requires the least weight data. Thus, using hashing trick, this paper proposes neural network accelerator whose performance is comparable to others, but its circuit complexity is reduced for an embedded device. With design exploration, we considered various design models, and determined target design and parallel factors for it. For 32nm cell library, our design operated at 200Mhz could be estimated in 32.06mm^2 chip area.**

Keywords; quantization; hashing trick; neural network accelerator

I. INTRODUCTION

The-state-of-art neural network accelerators (NNA), which are vector processors optimized for accelerating Neural Network (NN), are recently focused on optimization for embedded devices which have constrained resources. In NNA designs, to accelerate fully connected layer, numerous parameters should be reduced for less memory size. Previous works have exploited various model compression techniques such as pruning or quantization and utilized the compressed parameters for memory reduction. For example, EIE [1] used sparsity of matrix and shared weights made by K-means clustering so as to put all parameters in on-chip SRAM and eliminate memory access.

However, it said up to 93% of the chip area and 59% of power is dominated by SRAM in basic computing units called Processing Elements (PEs). The chip area highly increases according to the number of PEs when the speed-up of NNA is required; the area-performance efficiency was not considered. Similarly, this paper puts all parameters but employs different weight sharing technique called hashing trick [2] for minimizing required information to reduce the chip area. Furthermore, area-optimized architecture must not have performance degradation compared to other accelerators, so we explored design space with parallelism and sparsity utilization. Parallelism factors are determined by models which are implemented with 32nm cell library and have different factors. For guaranteeing the accuracy, we tried to follow the original hashing trick algorithm and used blocked hashing [3], which is suggested for solving the potential problem of hardware using hashing trick.

The EDA tool was supported by the IC Design Education Center (IDEC), Korea.

II. DESIGN METHODOLOGY

NNA requires a large amount of information such as weights and activations in bulky on-chip SRAM. Thus, to minimize chip size, it is often required to exploit a model compression technique. This paper employed hashing trick which is utilized for sharing weights with minimum information of weights, because they shared weights only through hashing without index. Also, for comparable performance, we exploited sparsity and Single Input Multiple Data (SIMD).

For sparse matrix-vector multiplication (SPMV), it is necessary to store relevant information in SRAM, so it is not appropriate to reduce size by employing all sparsity of activation and weight. Also, parallelism is hindered by a large number of lookup operations in weight sharing. This can be overcome with post-multiplication, which means the accumulation of activations followed by multiplication with centroids rather than the multiplication of each activation. However, this can be employed depending on the number of outputs. That is, unless a PE computes for only one output, the required logic circuit for activation accumulation can be highly increased. Considering previous constraints, there are the limited number of design models as illustrated in Table I. In the table, PARALLEL decides whose data will be parallelized, and SKIP decides whose sparsity will be used. As shown in Table I, the direction of parallelism (horizontally or vertically) or the format of the sparse matrix can be determined. Each of horizontal and vertical direction mean the way of the input and output vector, respectively.

Since Design 1, 2, and 5 exploit sparsity of weight, NNA must store information of the sparse matrix resulting in increasing hardware complexity of NNA. Furthermore, since Design 2 and 3 have more than one output, there is no way to parallelize all Look Up Tables (LUT) (i.e., post-multiplication is not possible). Design 4 does not need the information of sparse matrix and has one output for employing post-multiplication. As a result, we propose Design 4 can minimize NNA without performance degradation.

III. ARCHITEUCTURE DETAILS

The architecture of Design 4 is illustrated in Fig.1 and dotted line in PE means each pipeline stage. Central Control Unit (CCU) controls all PEs and Input Buffer depending on two

TABLE I. DESIGN EXPLORATION

	PARALLEL(A+W)		SKIP(A+W)		Post-multiply
	A	***W***	***A***	***W***	
D1	Serial	Serial	skip	skip(CSC)	Impossible
D2	Serial	Parallel(↕)	skip	skip(CSC)	Impossible
D3			skip	X	Impossible
D4	Parallel(↔)	Parallel(↔)	skip	X	Possible
D5			X	skip(CSR)	Possible

A : Activation, W : Weight ; CSC : Compressed Sparse Column, CSR : Compressed Sparse Row

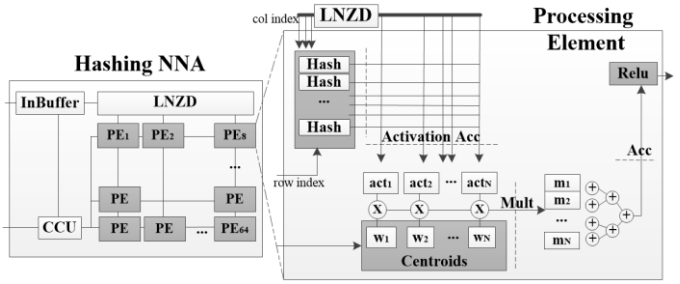

Figure 1. Architecture of Design 4

modes; I/O mode and computing mode. I/O mode means loading activations and weights, and computing mode is for computing loaded data. Leading Non-Zero Detection (LNZD) node detects non-zero edges for the sparsity of activation.

To decide inter-PE parallelism factor (i.e., the number of PEs) we explored various numbers of PEs and compared the normalized ratio of performance and area as shown in Fig.2. Performances were evaluated with the number of cycles assuming that the sparsity of activation is 0.7 as typical. The area-performance ratio is improved when the number of PEs increases. Furthermore, Intra-PE parallelism factor decides how many activations one PE can process in one cycle. For comparable performance to the accelerator which considers both sparsity of activation and weight, we choose 16 as the optimal parallelism factor in PE. With 784x1000 matrix, our design has 30% fewer cycles than accelerator considering all sparsity with on-chip SRAM, when the factor is 16.

PE has 5 pipeline stages; 2 stages for hashing, activation accumulation, multiplication with centroids, and accumulation followed by Relu operation. In the hashing step, the hash unit is implemented with XXhash logic with the same seed [2]. Even though it is not optimized for hardware because of multiplication, this paper followed the original algorithm to guarantee accuracy. In the activation accumulation step, all activations are accumulated according to the hashed result. With more centroids N, the chip area increases. Original hashing trick requires a lot more centroids than typically shared weights, but the authors in [3] showed a hashing trick could be implemented with fewer centroids through blocked hashing. They segmented centroids with a constant block size (64) to be mapped to each local region. We can exploit this technique by using different centroids in different PEs.

IV. EXPERIMENTS

We implement the register transfer level of the hashed accelerator in Verilog. It was synthesized with Design Compiler (DC) under 32nm CMOS technology and the circuit area of them is estimated with the Synopsys IC Compiler (ICC) at floor planning stage.

Table II summarizes details about the circuit implementation. At the number of centroids N = 32, the size of the circuit area is not dominated by SRAM. When it has 64PEs, the logic circuit increases more than the expected area (23.8mm^2), because our design uses 16 activations for input, so interconnection wires take a considerable area for 64PE. Also, with N = 64 like the block size of BHNN, it takes 1.86mm^2 more area for doubled centroids.

TABLE II. IMPLEMENTATION RESULTS

			N = 32	N = 64
Ciritical path delay			4.85 ns	4.87 ns
SRAM size			16 KB	16 KB
SRAM area			0.76 mm^2	0.76 mm^2
1PE	Logic gate count		388,955	414,109
	Area	Logic	0.46 mm^2	0.49 mm^2
		Total	1.12 mm^2	1.25 mm^2
64PE	Logic gate count		25,671,236	27,595,669
	Area	Logic	29.5 mm^2	31.3 mm^2
		Total	30.2 mm^2	32.06 mm^2

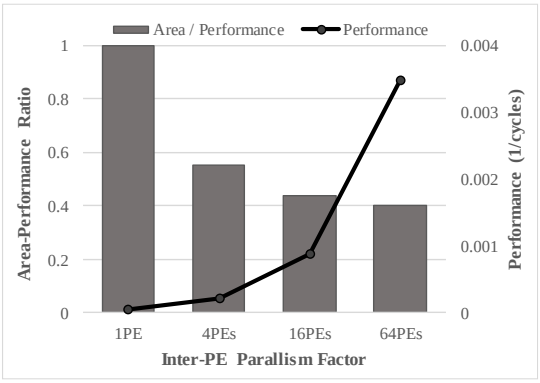

Figure 2. The Ratio of Area and Performance

V. CONCLUSION

Considering the cost-effective embedded device, it is desirable that NNA has moderate performance and a small area. This paper presents accelerator of which size is minimized using weight sharing and compensate performance with SIMD and sparsity of activations. Consequently, our design with 64PEs and 16 intra-PE parallelism reduced circuit area without performance degradation.

REFERENCES

[1] Han, S., Liu, X., Mao, H., Pu, J., Pedram, A., Horowitz, M. A., & Dally, W. J, "EIE: efficient inference engine on compressed deep neural network." In *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)* (pp. 243-254). IEEE.

[2] Chen, W., Wilson, J., Tyree, S., Weinberger, K., & Chen, Y. "Compressing neural networks with the hashing trick." In *2015 International Conference on Machine Learning* (pp. 2285-2294).

[3] Zhu, J., Qian, Z., & Tsui, C. Y. "BHNN: A memory-efficient accelerator for compressing deep neural networks with blocked hashing techniques." In *2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)* (pp. 690-695). IEEE.