An area-efficient memory-based multiplier powering eight parallel multiplications for convolutional neural network processors

Seongrim Choi, Suhwan Cho, and Byeong-Gyu Nam
Department of Computer Science and Engineering, Chungnam National University, Daejeon, Korea
Email: bgnam@cnu.ac.kr

Convolutional neural network (CNN) is widely used for various deep learning applications because of its best-in-class classification performance. However, CNN needs several multiply-accumulate (MAC) operations to realize human-level cognition capabilities. In this regard, an area-efficient multiplier is essential to integrate a large number of MAC units in a CNN processor. In this letter, we present an area-efficient memory-based multiplier targeting CNN processing. The proposed architecture adopts a 32-port memory shared across eight multiplications. Simulation results show that area is reduced by 18.4% compared with the state-of-the-art memory-based multiplier.

Introduction: Convolutional neural network (CNN) is widely adopted in a variety of deep learning applications due to its best-in-class classification performance [1]. Recent advances in CNNs have achieved human-level classification accuracy through billions of synaptic multiply-accumulate (MAC) operations. However, integrating a large number of MAC units on a single CNN processor is still demanding due to the stringent constraint on the area and cost of a chip. To deal with this problem, an area-efficient MAC unit is essential, and its multiplier part is very critical due to its significant contribution to the area of the MAC unit.

There have been memory-based approaches to the multipliers for improved power and area compared with the conventional booth multipliers [2–4]. They were commonly used in digital signal processors (DSPs) with fixed coefficients [2], but recently studied for CNN processors to exploit the high reusability of parameters like weights and activations. A lookup table (LUT) was used to reduce the power consumption in multiplications by just retrieving pre-computed multiplication results stored in the LUT [3]. In addition, a custom quad-port SRAM was studied to improve the area efficiency of the LUT by incorporating a special quad-port bitcell [4]. In this letter, we present a novel memory-based multiplier with memory sharing architecture to further improve the area efficiency of the memory-based multiplier. In the proposed architecture, eight multipliers share a single 32-port LUT to improve the area efficiency. As a result, the proposed work consumes a smaller area compared with the state-of-the-art memory-based multiplier [4].

Memory-Sharing Multiplier: A single weight parameter is shared across multiple activations when convolving an input feature map as illustrated in Figure 1. Therefore, a memory-sharing parallel multiplier can be exploited in a convolution to benefit the shared weight parameters producing an area-efficient design. We investigate the number of multipliers sharing a single memory block to maximize the area-efficiency. As shown in Figure 2, the effective area per multiplier rapidly drops as the number of multipliers sharing a memory becomes larger, while the benefit diminishes at around the number of eight. Therefore, we chose eight multipliers to share a single memory block for the maximal area efficiency. For CNN applications, 16-bit fixed-point format is widely accepted for both activation and weight for its minimal impact on accuracy compared with the single-precision floating-point [5], hence in this letter, we discuss the design of a memory-based multiplier for 16b × 16b operations. We use the activation rather than the weight for the address to the shared memory block of the multiplier to read out a pre-computed multiplication result of weight and activation because of the higher reusability of the weight parameter compared with that of the activation. In the area-efficient approach, it is also important to limit the word length of the activation since the number of memory entries is exponential to the activation word length. Therefore, we slice the activation into 4-bit partial activations for an optimal number of 16 memory entries and hence get four partial products, as shown in the graph of Figure 3.

The overall architecture of the proposed memory-based multiplier is depicted in Figure 4a. It consists of a memory block accessed to get 32 partial products and eight shift-add trees to sum up the four partial products in each tree producing eight multiplication results. The memory block incorporates 32 read address decoders and supports 32 read-out ports accordingly to provide the eight parallel multiplications that take four partial activations for each. The memory array has 16 entries of 20-bit word-length to accommodate the 20-bit pre-computed partial products of 16-bit weight and 4-bit partial activation. As shown in Figure 4b,
the proposed multiplier operates in update and read-out phases of partial products. During the update phase, the pre-computed partial products are updated sequentially every clock cycle. Once the pre-computed results are updated, they remain constants for a while until the filter weights change, which occurs rarely and thus making the latency impact from the updates minimal in the entire operation cycles. In the read-out phase, 32 partial products are read out simultaneously to get the eight parallel multiplication results that require four partial products for each multiplication due to the activations sliced into four pieces. Eight shift-add trees take the 32 partial products, that is four partial products for each tree, and produces the eight multiplication results by summing up in each tree the four partial products properly shifted for position alignments.

**Memory Array:** The memory array in the multiplier consists of $20 \times 16$ matrix of 32-port bitcells, as described in Figure 5. There are 16 rows of 32 read wordlines (RWLs) and 20 columns of 32 read bitlines (RBLs) in the array, where each RWL has 20 bitcells and each RBL has 16 bitcells. The proposed 32-port bitcell is shown in Figure 6a. It consists of a data storage, one write port and 32 read ports selected via the 32 R WLs and the 32 RBLs. It has a two-stage FO4 buffering structure to drive a large capacitance of the 32 read ports loaded on a bitcell. Domino buffers are used for fast switching and lightweight driving of output ports. Polarity for the read ports from the QB node is adjusted in the final sensing stage. In the write operation, the storage node Q is written to '0' or '1', according to the value on the write bitline (WBL), as illustrated in Figure 6b. In the read operation, the R WL is enabled after all the RBLs are precharged, and then the RBL selected by the R WL is evaluated depending on the value of the storage node Q in the bitcell.

Figure 7a shows a layout diagram of the 32-port bitcell in 65 nm technology. Dimension of the cell is 7.18 $\mu$m $\times$ 6.75 $\mu$m, but a metal-only area is created since the area for floor planning the 32 R WLs and 32 RBLs is much larger than the device area. The 32 RPs are floor planned into the left and right wings of 16 RPs, which are placed in a staggered manner for a back-to-back placement of bitcells as shown in Figure 7c to save the metal-only area. The array adopts domino sensing without the column multiplexing due to the small capacitance of 16 bitcells loaded on an RBL.

**Implementation Results:** The proposed memory-based multiplier is designed using 65 nm CMOS technology. The layout figure is depicted in Figure 8. The total area is $150 \times 224 \mu$m$^2$. Table 1 shows the comparison with previous works, which are redesigned in 65 nm technology for a comparison purpose. Power is simulated for a convolution of a $256 \times 256$ image with a $3 \times 3$ filter. Results show that the proposed memory-based multiplier attains 18.4% and 29.4% reductions in the area and the 32 RBLs. It has a two-stage FO4 buffering structure to drive a large capacitance of the 32 read ports loaded on a bitcell. Domino buffers are used for fast switching and lightweight driving of output ports. Polarity for the read ports from the QB node is adjusted in the final sensing stage. In the write operation, the storage node Q is written to '0' or '1', according to the value on the write bitline (WBL), as illustrated in Figure 6b. In the read operation, the R WL is enabled after all the RBLs are precharged, and then the RBL selected by the R WL is evaluated depending on the value of the storage node Q in the bitcell.

Figure 7a shows a layout diagram of the 32-port bitcell in 65 nm technology. Dimension of the cell is 7.18 $\mu$m $\times$ 6.75 $\mu$m, but a metal-only area is created since the area for floor planning the 32 RWLs and 32 RBLs is much larger than the device area. The 32 RPs are floor planned into the left and right wings of 16 RPs, which are placed in a staggered manner for a back-to-back placement of bitcells as shown in Figure 7c to save the metal-only area. The array adopts domino sensing without the column multiplexing due to the small capacitance of 16 bitcells loaded on an RBL.

**Implementation Results:** The proposed memory-based multiplier is designed using 65 nm CMOS technology. The layout figure is depicted in Figure 8. The total area is $150 \times 224 \mu$m$^2$. Table 1 shows the comparison with previous works, which are redesigned in 65 nm technology for a comparison purpose. Power is simulated for a convolution of a $256 \times 256$ image with a $3 \times 3$ filter. Results show that the proposed memory-based multiplier attains 18.4% and 29.4% reductions in the area and the 32 RBLs. It has a two-stage FO4 buffering structure to drive a large capacitance of the 32 read ports loaded on a bitcell. Domino buffers are used for fast switching and lightweight driving of output ports. Polarity for the read ports from the QB node is adjusted in the final sensing stage. In the write operation, the storage node Q is written to '0' or '1', according to the value on the write bitline (WBL), as illustrated in Figure 6b. In the read operation, the R WL is enabled after all the RBLs are precharged, and then the RBL selected by the R WL is evaluated depending on the value of the storage node Q in the bitcell.

**Table 1. Comparison with previous works**

|                  | Booth multiplier | LUT multiplier [3] | Quad-port SRAM multiplier [4] | This work |
|------------------|------------------|--------------------|-------------------------------|-----------|
| Technology (nm)  | 65               | 65                 | 65                            | 65        |
| Area ($\mu$m$^2$)| 53.630           | 113.678            | 41.181                        | 33.600    |
| Supply voltage (V)| 1.2              | 1.2                | 1.2                           | 1.2       |
| Input frequency (MHz)| 200              | 200                | 200                           | 200       |
| Power (mW)       | 2.63             | 1.63               | 1.43                          | 1.01      |
| Propagation delay (ns)| 2.11             | 2.33               | 1.94                          | 1.88      |

*a* [3] and [4] are redesigned using 65 nm technology node for comparison purpose.
power, respectively, compared with the state-of-the-art memory-based multiplier [4].

**Conclusion:** We propose an area-efficient memory sharing eight-parallel multiplier optimized for CNN processing. A single 32-port memory is shared across eight multipliers to improve the area-efficiency. The proposed multiplier achieves an 18.4% area reduction compared with the state-of-the-art and thereby can readily be adopted in area-efficient CNN processors.

**Acknowledgments:** This work was supported by research fund of Chungnam National University. The authors would like to thank IDEC for CAD support.

© 2021 The Authors. *Electronics Letters* published by John Wiley & Sons Ltd on behalf of The Institution of Engineering and Technology

This is an open access article under the terms of the Creative Commons Attribution License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited.

Received: 12 March 2021 Accepted: 10 April 2021 doi: 10.1049/ell2.12206

**References**

1 Karpathy, A. et al.: Large-scale video classification with convolutional neural networks. In: Proceeding IEEE Conference Computing Vision Pattern Recognition, Columbus, OH, USA 23–28 June 2014, pp. 1725–1732, https://doi.org/10.1109/CVPR.2014.223

2 Meher, P.K. et al.: New approach to LUT implementation and accumulation for memory-based multiplication. In: IEEE International Symposium on Circuits and Systems, Taipei, Taiwan 24–27 May 2009, pp. 453–456, https://doi.org/10.1109/ISCAS.2009.5117783

3 Shin, D. et al.: DNPU: An 8.1TOPS/W reconfigurable CNN-RNN processor for general-purpose deep neural networks. In: 2017 IEEE International Solid-State Circuits Conference, California, USA 5-9 Feb. 2017, pp. 240–242, https://doi.org/10.1109/ISSCC.2017.7870350

4 Gong, Y. et al.: ARA: Cross-layer approximate computing framework based reconfigurable architecture for CNNs. *Microelectron. J.* 87, 33–44 (2019) https://doi.org/10.1016/j.mejo.2019.03.011

5 Tianshi, C. et al.: Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning. *ACM SIGPLAN Not.* 49(4), 249–284 (2014) https://doi.org/10.1145/2541940.2541967