Quant-PIM an Energy-Efficient Processing-in-Memory Accelerator for Neural Networks.

Debarchana Jena

College of Engineering Bhubaneswar

Abstract—**Layerwise quantized neural networks (QNNs) have become a potential method for embedded systems since they adopt varying precisions for weights or activations in a layerwise way. Compared to conventional QNNs, layerwise QNNs use less computing energy since they only use the number of data bits needed for computations (such as the convolution of weights and activations). However, because memory accesses are not optimized for the necessary precision of each layer, layerwise QNNs still consume a significant amount of energy in conventional memory systems. In order to tackle this issue, we suggest Quant-PIM, an energy-efficient PIM accelerator for layerwise QNNs. Quant-PIM uses customized I/O gating logics in a 3-D stacked memory to selectively read just the necessary data bits inside a data word, depending on the accuracy. Quant-PIM thereby drastically lowers the energy used for memory accesses. Furthermore, layerwise QNN performance is enhanced by Quant-PIM. Quant-PIM uses the saved memory bandwidth from the selective memory access to read two data blocks in a single read operation, resulting in improved compute-throughput when the needed accuracy is half of the weight (or activation) size or less. Our simulation results demonstrate that, without sacrificing accuracy, Quant-PIM reduces system energy by 39.1% to 50.4% when compared to the PIM system with 16-bit quantized precision.**

I. Introduction

RECENTLY, deep neural networks (DNNs) have beenwidely adopted in various applications, such as image classification and speech recognition. In general, DNNapplications cause high energy consumption, since it requires millions of multiply-accumulate (MAC) operations and high memory bandwidth, both. This high energy consumption makes it difficult to run the DNN applications on energy- constrained embedded systems. To address this problem,a quantized neural network (QNN) has emerged as a viable solution for embedded systems [2]. The QNN reduces com- putation energy by replacing floating-point MAC operations with fixed-point MAC operations. In addition, since the QNN exploits relatively low-precision weight (or activation) instead of high-precision one, it has lower memory bandwidth requirement than the DNN.

However, as the volume of the input data and the number of layers increase, QNNs still cause high energy consumption. Several studies have reduced energy consumption by adopting different precisions for weights or activations in a layerwise manner [3], [10], since the precision requirement varies across layers within a network. The layerwise QNNs deploy only required number of data bits for the computation (e.g., convolution of weights and activations), which in turn reduces computation energy compared to the conventional QNNs. Though many researchers have focused on the computation energy reduction by adopting layerwise QNNs [3], [10], they did not consider the optimization of memory accesses for layerwise QNNs. Even with low precision in the layerwise QNNs, each single memory access still transfers a full data word (e.g., 64-bit or 32-bit); note, the conventional memory systems do not allow transferring data of which size is less than the data word size in a single memory access. Thus, the layerwise QNNs cause a large amount of energy due to the nonoptimized memory accesses, which is same as the conventional QNNs.

In this letter, we propose *Quant-PIM*, an energy-efficient processing-in-memory (PIM) accelerator for layerwise QNNs. We adopt Quant-PIM to a high bandwidth memory (HBM), which is widely deployed for neural network (NN) hardware accelerators [7], [8]; note HBM has recently been adopted to an embedded NN system [7]. Quant-PIM consists of two parts: 1) I/O gating logics and 2) processing units for the gated I/O. In Quant-PIM, a single memory access selectively loads/stores only required data bits within a data word depending on the precision. To enable such selective memory accesses, we modify the I/O gating logics of the HBM. Quant-PIM controls the I/O gating logics depending on the required precision of a layer. In addition, to guarantee the MAC operations with any precision, we allocate multiple binary MAC units and accumulators into the base die of the HBM. Thus, Quant-PIM significantly reduces energy consumption for memory accesses depending on the required precision of a layer, while ensuring normal computation

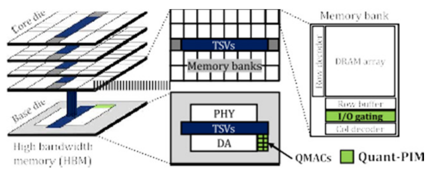

Fig. 1. Overview of the proposed Quant-PIM.

with any precision. Furthermore, Quant-PIM improves the performance of layerwise QNNs. When the required precision is half of the weight (or activation) size or less, Quant-PIM reads two data blocks[1] in a single read operation by exploiting the saved memory bandwidth from the selective memory access, achieving higher compute-throughput.

II. Related Works

There have been many studies on the accelerators for layerwise QNNs [3], [10], [13]. Judd *et al.* introduced an accelerator which provides the bit-serial execution of the MAC operation with any precision [3]. The accelerator serially executes a bit operation per clock cycle with high parallelism, which significantly reduces computation energy. Umuroglu *et al.* also presented a vectorized bit-serial matrix multiplication technique with high parallelism [13]. In addition, Sharma *et al.* proposed an accelerator with a bit-decomposition technique [10]. They divided an MAC operation into multiple sub-MAC operations to support variable precision, which reduces the amount of the required resources for the MAC operation. However, all the above studies focused on reducing computation energy for MAC operations, assuming that all the weights and activations are already prepared in the on-chip buffers. They did not consider the optimization of memory accesses for layerwise QNNs, thus causing a large amount of energy reading the data from the main memory. Different from the previous studies, our proposed technique selectively accesses only required data bits within the data word depending on the required precision of a layer, which significantly reduces energy consumption for memory accesses. To the best of our knowledge, this letter is the first study to optimize the memory accesses with any precision in the layerwise QNNs.

III. Quant-PIM

We propose a PIM accelerator for layerwise QNNs called Quant-PIM. Quant-PIM significantly reduces system energy, since: 1) it reduces memory power consumption by selectively accessing only required data bits at a bit-level granularity and 2) it improves performance by reading two data blocks in a single read operation when the required precision is half of the weight (or activation) size or less.

A. Overall Architecture

As shown in Fig. 1, we adopt Quant-PIM to the HBM. The HBM has two different parts: 1) base die and 2) core dies. Since the base die is a logic die (not a memory die), it has been widely deployed for implementing small accelerators [5], [11]. We implement processing units of Quant-PIM into the base die of the HBM, which is called quantized MAC units (QMACs); we allocate eight QMACs

[1]In this letter, a data block indicates the data accessed from the main memory with a single read operation, whose size is typically determined by the product of data bus width and burst length.

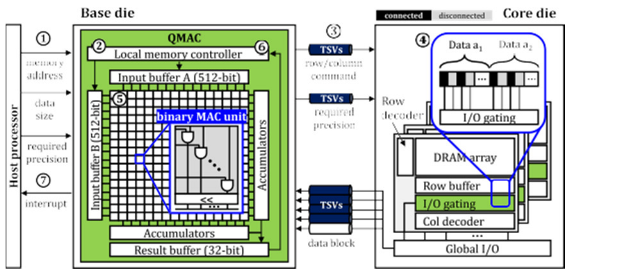

Fig. 2. Detailed design of Quant-PIM and procedures for Quant-PIM in a single memory channel.

(i.e., one QMAC per memory channel), since each memory channel is independently accessed. More importantly, we modify the I/O gating logics in the memory banks of the core dies. In the original HBM, the I/O gating logic accesses the columns in the row buffer at a word-level granularity, depending on the result of the column decoder. Accordingly, a single memory access in the original HBM transfers the full data word even with low precision. On the other hand, Quant-PIM allows memory accesses at a finer granularity than the original HBM. Depending on the required precision of a layer, Quant-PIM controls the I/O gating logics to access the columns in the row buffer at a bit-level granularity.

B. Processing Flow

Fig. 2 describes the detailed design of Quant-PIM and the procedures for Quant-PIM in a single memory channel. The procedures for Quant-PIM are as follows (① to ⑦ in Fig. 2).

① The host processor offloads the MAC operations for layerwise QNNs by sending the memory address, data size, and required precision to the QMAC.

② The local memory controller of the QMAC generates row and column commands based on the memory address and data size.

③ The QMAC sends the row/column command and required precision to the target memory bank depending on the memory address.

④ Quant-PIM controls the I/O gating logics in the memory bank depending on the required precision, so that it selectively reads only required data bits within a data word.

⑤ The QMAC executes the MAC operations based on the data block loaded from the memory bank. Since one MAC operation with n-bit precision is replaced with n^2 1-bit MAC operations [13], the QMAC has multiple binary MAC units and accumulators. Considering the worst-case precision (i.e., 16-bit) of the weights or activations in the conventional QNNs, we allocate 256 ($=16^2$) binary MAC units and accumulators in the QMAC. Thus, the QMAC guarantees the MAC operations with 16-bit or less precision. The accumulated result of the MAC operations is stored in the result buffer.

⑥ The local memory controller of the QMAC stores the accumulated result to the memory bank. Similar to the read operation in step ④, Quant-PIM selectively stores the required data bits based on the precision.

⑦ Quant-PIM repeats the step ② to ⑥ until all the offloaded MAC operations are completed. Then, Quant-PIM notifies the host processor that all the offloaded MAC operations are completed through an interrupt signal.

Based on the procedures for Quant-PIM, Quant-PIM reduces I/O power when the required precision is lower than 16-bit, since it selectively accesses only required data bits within a data word. Thus, Quant-PIM reduces the total HBM power consumption, which results in the system energy reduction; I/O power occupies up to 70% of the total HBM power [12].

Furthermore, Quant-PIM improves the performance of layerwise QNNs, when the required precision is half of the weight (or activation) size or less (i.e., 8 bit or less). In step 4 , Quant-PIM reads two data blocks (data a_1 and a_2 in Fig. 2) ina single memory operation by exploiting the saved memory bandwidth from the selective memory access. Note Quant- PIM coalesces memory requests to two data block addresses. However, Quant-PIM is different from a memory coalescing technique which is widely adopted in GPU; the memory coa- lescing technique only coalesces memory requests to the same data block address. In step 5 , the QMAC executes the MAC operations for both data blocks a_1 and a_2 at the same time, by deploying the binary MAC units and accumulators; since the 8-bit MAC operations for a single data block requires only 64 binary MAC units among 256 binary MAC units, Quant-PIM is able to simultaneously execute the 8-bit MAC operations for two data blocks. Thus, Quant-PIM provides higher compute-throughput, resulting in the system energy reduction.

IV. Evaluation

A. Methodology

Table I shows the required precision per layer with relative accuracy compared to the 16-bit quantized precision for three representative neural networks [3]. For quantization, a uniform quantization method is adopted, which replaces 32-bit floating-point data with 16-bit integer data by deploying the following equation (which is a general quantization method [8])

$$\text{quantized data} = \text{real data} * \text{scale}. \qquad (1)$$

The required precision is obtained by repeatedly removing the least significant bit (LSB) of the 16-bit quantized weights and activations until the relative accuracy decreases; note removing the LSBs of the already quantized weights or activations is also the uniform quantization method. Based on Table I, we evaluate the execution time, power consumption, peak on-chip temperature, and system energy of Quant-PIM across NNs. We consider a 16-bit PIM system as our baseline; the 16-bit PIM system reads 16-bit quantized weights and activations from the HBM and then execute the MAC operations with 16-bit precision. We first implement the QMAC (in the base die) in Fig. 2 with Verilog HDL using Design Compiler and IC Compiler based on Samsung System LSI 28-nm process technology. We set the clock frequency of the QMAC to 1 GHz, operating with the HBM synchronously. According to the implementation result, the QMAC is able to operate at 1 GHz even in the worst-case precision (i.e., 16-bit). In addition, we obtain the dynamic power and leakage power of the QMAC by 1.0~12.7 mW and 36.2 μW, respectively; we extract dynamic power depending on the required precision. For the additional circuits in I/O gating logics (in the core die), we conservatively evaluate the power consumption based on logic process technology.[2] According to our implementation,

TABLE I
REQUIRED PRECISION PER LAYER WITH RELATIVE ACCURACY

Network	100% relative accuracy	99% relative accuracy
GoogLeNet	10-8-10-9-8-10-9-8-9-10-7	10-8-9-8-8-9-10-8-9-10-8
AlexNet	9-8-5-5-7	9-7-4-5-7
NiN	8-8-8-9-7-8-8-9-9-8-8-8	8-8-7-9-7-8-8-9-9-8-7-8

we extract the dynamic power and leakage power of the additional circuits in I/O gating logics by 22.9 mW and 14.7 μW per memory channel, respectively.

We evaluate the execution time and power consumption of Quant-PIM across NNs using gem5-aladdin [9] with a cycle-accurate DRAM simulator [6]. We reflect the clock frequency and power consumption for both the implemented QMAC and additional circuits in I/O gating logics to the simulator. We also reflect the performance and power of the HBM2 referring to the timing/current parameters [5], [6], [11]. Based on the power consumption, we evaluate the peak on-chip temperature of Quant-PIM using HotSpot 6.0 [14]. Since peak on-chip temperature is strongly affected by heat dissipated from power consuming units such as GPU, we assume that the HBM (including Quant-PIM) is integrated with a high-end GPU for a gaming console [15]. We also evaluate the system energy and area overhead of Quant-PIM.

B. Results

1) Execution Time: Fig. 3 (left) shows the execution time of Quant-PIM across NNs. Quant-PIM reduces execution time by 19.9%~42.1% (27.4% ~42.1%) compared to the 16-bit PIM system, while maintaining 100% (99%) relative accuracy. We break down the execution time for GoogLeNet[3] into each layer, as shown in Fig. 3 (right). Quant-PIM (100%) significantly reduces execution time at the convolution layer 2, 5, 8, and 11 compared to 16-bit PIM. When 1% relative accuracy loss is tolerable, Quant-PIM (99%) additionally reduces execution time at the convolution layer 4. As explained in Section III, when the required precision is 8-bit or less, Quant-PIM simultaneously reads two data blocks and executes MAC operations for the two data blocks. Thus, Quant-PIM offers high compute-throughput, resulting in the short execution time for layerwise QNNs.

2) Power Consumption: Fig. 4 (left) shows the power consumption of Quant-PIM across NNs. Quant-PIM reduces power consumption by 15.4%~ 18.2% (14.2%~19.6%) compared to the 16-bit PIM system, while maintaining 100% (99%) relative accuracy. We break down the power consumption for GoogLeNet into each layer, as shown in Fig. 4 (right). As explained in Section III, Quant-PIM reduces I/O power when the required precision is lower than 16-bit, since it selectively accesses only required data bits. However, Quant-PIM (100%) does not reduce the power (not energy) consumption at the convolution layer 2, 5, and 8, since two weights or activations (16 bit, in total) in different data blocks are transferred together; the required precision is 8 bit. With 1% relative accuracy loss, Quant-PIM (99%) does not reduce power consumption at the convolution layer 4 and 11 as well; the required precision is 8-bit. Though Quant-PIM causes additional power from the additional circuits in I/O gating logics, this power overhead is only 2.2% of the total power consumption, which is much smaller than the I/O power reduction (up to 19.6%).

[2]The logic implemented with memory process technology is more energy-efficient than that implemented with logic process technology [4].

[3]We present the layerwise results only for GoogLeNet due to the page limit.

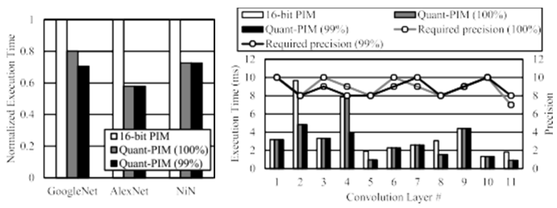

Fig. 3. Execution time across NNs (left) and layerwise execution time for GoogLeNet depending on the required precision (right).

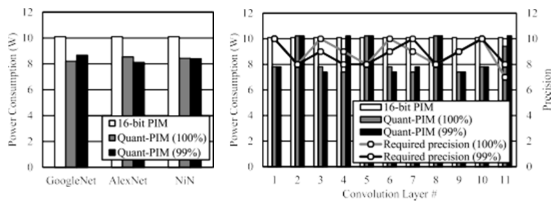

Fig. 4. Power consumption across NNs (left) and layerwise power consumption for GoogLeNet depending on the required precision (right).

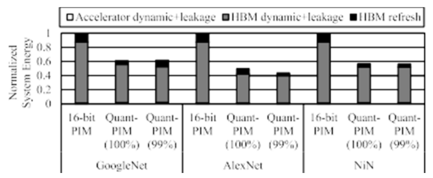

Fig. 5. System energy breakdown across NNs.

3) Peak On-Chip Temperature: Based on the power consumption for NNs, we analyze the peak on-chip temperature, considering heat dissipated from the GPU. The peak on-chip temperature of the 16-bit PIM system is 88.2 ˚C for all the NNs. On the other hand, the peak on-chip temperature of Quant-PIM is 84.8 ˚C, 85.4 ˚C, and 84.9 ˚C for GoogLeNet, AlexNet, and NiN, respectively. In all the NNs, Quant-PIM has lower peak on-chip temperature than the 16-bit PIM system, since it reduces I/O power consumption.

4) System Energy: Fig. 5 shows the system energy of Quant-PIM across NNs. Quant-PIM reduces system energy by 39.1%-50.4% (38.3%-56.4%) compared to the 16-bit PIM system, while maintaining 100% (99%) relative accuracy. Since the accelerator itself causes negligible energy (<1% of the total system energy) due to its extremely small power, the HBM (memory) energy accounts for most of the total system energy. Quant-PIM (both 100% and 99%) significantly reduces the dynamic and leakage energies of the HBM compared to the 16-bit PIM system due to the following reasons: 1) Quant-PIM reduces power consumption when the required precision is lower than 16-bit and 2) Quant-PIM provides higher compute-throughput when the required precision is 8-bit or less, which results in the short execution time for layerwise QNNs. In addition, Quant-PIM (both 100% and 99%) has lower HBM refresh energy than the 16-bit PIM system. When peak on-chip temperature exceeds 85 ˚C, the HBM requires frequent refresh operations to retain the data in the memory cells, which in turn increases refresh energy [1]. As explained earlier, Quant-PIM has peak on-chip temperature lower than 85 ˚C in most cases, which leads to lower refresh energy. Though the peak on-chip temperature of Quant-PIM exceeds 85 ˚C, HBM refresh energy is reduced due to the short execution time.

5) Area Overhead: According to the implementation results, the area is only 0.16 mm^2 and 0.02 mm^2 for all the QMACs and additional circuits in I/O gating logics, respectively. Since the area of the HBM base die is 96 mm^2 [11], Quant-PIM causes negligible area overhead (<0.2% of the HBM base die area).

V. Conclusion

We introduce Quant-PIM, a PIM accelerator for layerwise QNNs. Because it 1) selectively accesses only the necessary data bits at a bit-level granularity to reduce memory power consumption and 2) enhances performance by reading two data blocks in a single read operation when the required precision is half of the weight (or activation) size or less, Quant-PIM significantly lowers system energy consumption. Comparing Quant-PIM to the 16-bit PIM system, our simulation~results demonstrate that Quant-PIM decreases system energy by 39.1% to 50.4% without sacrificing accuracy. Quant-PIM could further increase energy efficiency when accepting moderate accuracy loss, even though we only take into account the layerwise QNNs keeping 100% (99%) relative accuracy. For instance, compared to the 16-bit PIM system, Quant-PIM lowers system energy by 67.6% with 89.4% relative accuracy in the case of GoogLeNet with 4-bit quantization. It is anticipated that Quant-PIM will work in concert with the most modern accelerators to create layerwise QNNs that are energy-efficient.

References

[1] R. Hadidi, B. Asgari, B. A. Mudassar, S. Mukhopadhyay, S. Yalamanchili, and H. Kim, "Demystifying the characteristics of 3D-stacked memories: A case study for hybrid memory cube," in *Proc. IISWC*, Seattle, WA, USA, 2017, pp. 66–75.

[2] I. Hubara, M. Courbariaux, D. Soudry, Y. Bengio, and R. El-Yaniv, "Quantized neural networks: Training neural networks with low precision weights and activations," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 6869–6898, Jan. 2017.

[3] P. Judd, J. Albericio, T. Hetherington, T. M. Aamodt, and A. Moshovos, "Stripes: Bit-serial deep neural network computing," in *Proc. MICRO*, Taipei, Taiwan, 2016, pp. 1–13.

[4] Y.-B. Kim and T. W. Chen, "Assessing merged DRAM/logic technology," *Integration*, vol. 27, no. 2, pp. 179–194, 1999.

[5] Y. S. Lee, K. M. Kim, J. H. Lee, J. H. Choi, and S. W. Chung, "A high-performance processing-in-memory accelerator for inline data deduplication," in *Proc. ICCD*, Abu Dhabi, United Arab Emirates, 2019, pp. 515–523.

[6] S. Li, Z. Yang, D. Reddy, A. Srivastava, and B. Jacob, "DRAMsim3: A cycle-accurate, thermal-capable DRAM simulator," *IEEE Comput. Archit. Lett.*, vol. 19, no. 2, pp. 106–109, Jul.–Dec. 2020.

[7] H. Liao, J. Tu, J. Xia, and X. Zhou, "DaVinci: A scalable architecture for neural network computing," in *Proc. HCS*, Cupertino, CA, USA, 2019, pp. 1–44.

[8] T. Norrie *et al.*, "Google's training chips revealed: TPUv2 and TPUv3," in *Proc. HCS*, Palo Alto, CA, USA, 2020, pp. 1–70.

[9] Y. S. Shao, S. L. Xi, V. Srinivasan, G.-Y. Wei, and D. Brooks, "Co-designing accelerators and SoC interfaces using gem5-Aladdin," in *Proc. MICRO*, Taipei, Taiwan, 2016, pp. 1–12.

[10] H. Sharma *et al.*, "Bit fusion: Bit-level dynamically composable architecture for accelerating deep neural networks," in *Proc. ISCA*, Los Angeles, CA, USA, 2018, pp. 764–775.

[11] K. M. Sohn, "HBM DRAM for energy-efficient near-memory computing," presented at the Workshop Memory Centric Comput. Future Challenges, Seoul, South Korea, Apr. 2019.

[12] D. Stow, A. Farmahini-Farahani, S. Gurumurthi, Y. Xie, and M. Ignatowski, "Power profiling of modern die-stacked memory," *IEEE Comput. Archit. Lett.*, vol. 18, no. 2, pp. 132–135, Jul. 2019.

[13] Y. Umuroglu, L. Rasnayake, and M. Sjalander, "BISMO: A scalable bit-serial matrix multiplication overlay for reconfigurable computing,"in *Proc. FPL*, Dublin, Ireland, 2018, pp. 307–314.

[14] R. Zhang, M. R. Stan, and K. Skadron, "HotSpot 6.0: Validation, acceleration and extension," IBM T. J. Watson Res. Center, Univ. Virginia, Charlottesville, VA, USA, Rep. CS-2015-04, 2015.

[15] Microsoft. *Xbox One Series X.* [Online]. Available: https://www.xbox.com/en-US/consoles/xbox-series-x