FePIM: Processing-in-Memory Based on Ferroelectric Field-Effect Transistors

**ABSTRACT**

The memory wall bottleneck has caused a large portion of the energy to be consumed by data transfer between processors and memories which dealing with data-intensive workloads. Processing-in-Memory (PIM) is considered as a promising technique to relieve the memory wall bottleneck by giving certain processing abilities to memories. In this work, we propose a PIM computer by employing ferroelectric field-effect transistors (FeFETs). The proposed design, named FePIM, is able to perform in-memory bit-wise logic and add operations between two selected rows or between one selected row and an immediate operand. We further propose novel solutions to eliminate simultaneous-read-and-write contentions such that stalls are eliminated. Experimental results show that Processing-in-Memory architecture can reduce memory access latency and energy, compared with FeFET-based PIM design which cannot handle read and write contentions.

**1 BACKGROUND**

The performance gap between processors and memories, which is known as the memory wall bottleneck, has been continuously widened in the past decades[1]. A number of emerging big data applications have exacerbated the gap. Processing-in-Memory is considered as a promising technique to relieve the memory wall bottleneck. Processing-in-Memory was processed in early 1970[2]. The idea did not take off in the last century due to the technical limitation. The emergence of big-data applications has required for more efficient data access, which has brought an opportunity for the development of Processing-in-Memory. 3D stacking technology and new non-volatile memory devices are two main types of Processing-in-Memory implementations. The first implementation employs the 3D stacking technology, to place the processing unit at the bottom of the 3D stacking memories[3,4,5]. The second implementation is realized by adopting the non-volatile memory and modifying the design of the peripheral circuit of memory to carry out the logic operation in memories. In recent years, people have proposed the second type of PIM computer architectures based on spin-transfer torque random-access memories (STT-RAMs)[6], resistive random-access memories (RRAMs)[7,8], phase-change memories (PCMs)[9] , and ferroelectric field-effect transistors (FeFETs)[10].

Although the aforementioned PIM designs based on non-volatile device have shown high capacity to achieving high energy efficiency or performance, there are several practical and important issues that have rarely been considered.

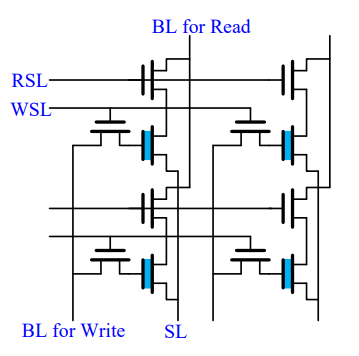
Most existing PIM designs provide only in-memory bit-wise logic operations, without consideration for write back. How to control write back in an efficient way is an important problem.. Write back and read operations may have contention in cases where the PIM operation has completed and the result needs to be written back while we need read out some data for the subsequent PIM operations. This problem can cause PIM operations stall and slow down calculation.

In this work, we employ FeFETs to build a contention-free PIM architecture. We intend to eliminate simultaneous-read-and-write(SRAW) data contentions in an FeFET-based PIM architecture named FePIM. We designed a forwarding mechanism that supports SRAW situations. As far as we know, this is the first time that SRAW contentions has been considered and solved in Processing-in-Memory.

**2 METHOD**

In this work, we employed FeFETs to design a contention-free Processing-in-Memory architecture. We simulated FePIM with HSPICE using FeFET model proposed in [11]. We used the 45nm predictive technology model [12] as the basic MOSFET model.

An FeFET is made by integrating a ferroelectric material layer in the gate stack of a metal-oxide-semiconductor field-effect transistor (MOSFET). The behavior of an FeFET depends on the thickness of the ferroelectric material layer. We used a SPICE-compatible FeFET simulation model proposed in [11]. The FeFET simulation model shows good nonvolatile read and write characteristics. Fig. 1 shows the smallest memory unit of FePIM. The blue transistor stands for the ferroelectric material layer of FeFETs. The memory unit can perform read, write and process operations.



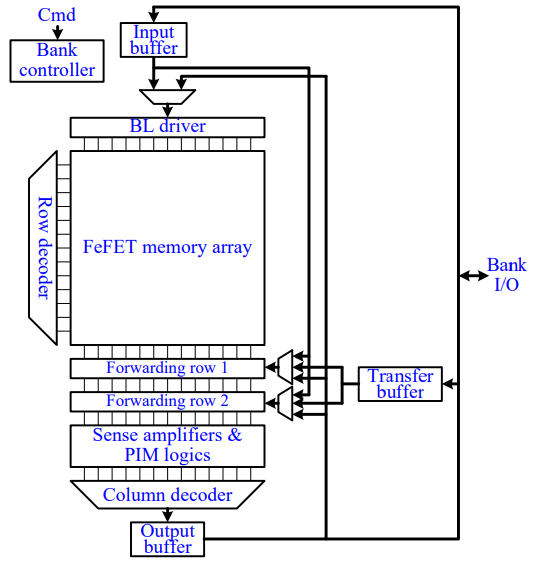
**Figure 1 the smallest memory unit of FePIM.**

The voltage settings of the lines for memory operations are shown in Table 1. To read the polarization of a memory unit, we need to supply a zero voltage to the gate terminal by setting the selected WSL as -VDD and RSL as VREAD.To write the the polarization of a FeFET, we need to supply a positive or negative VWRITE pulse to the gate terminal by setting the selected WSL as -VDD and RSL as 0.

**Table 1 Voltage settings for memory operations**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Read |  | Write |  |
|  | selected row | unselected row | selected row | unselected row |
| RSL | VREAD | 0 | 0 | X |
| WSL | -VDD | -VDD | VDD | -VDD |
| BL | X | | ±VWRITE | |
| SL | 0 | | | |

We designed the FePIM by . Fig 2 shows the top-level of FePIM architecture. The primary components of the FePIM are a FeFET memory array made of FeFET memory unit, sense amplifiers (SAs) and PIM logic calculation unit, a controller, two forwarding rows , and other peripheral circuitry. The controller receives commands from the upper level and generates all the control signals as well as the addresses to schedule PIM operations with simultaneous-read-and-write data contentions. The SA design and PIM logic calculation unit realize PIM operations. FeFET memory array is the operation object for reading and writing and processing of the FePIM.



**Figure 2 the top-level of FePIM architecture**

In our work, we use the following voltage settings: VDD = 1V,VREAD = 0.1V and VWRITE = 1V. The baseline for comparison is an enhanced version of a state-of-the-art FeFET-based PIM design [13].

We use Design Compiler to calculate the memory energy consumption. We use the HSPICE simulation model to get the memory latency. We perform an apple-to-apple comparison with [13] on the energy consumption of basic operations. We also evaluate a 1MB memory array with 32-bit word size (same as [13]). We perform evaluations on the entire FePIM architecture, where the analog components are evaluated by HSPICE and the controllers are implemented by Verilog and evaluated by Design Compiler. The FePIM memory array size is 1024×1024. The clock frequency is 500MHz. The baseline has the same memory size for a fair comparison. We use 10 benchmarks which are implemented by C to conduct a system-level evaluation. The 10 benchmarks are from different areas, including matrix add (MA), histogram (HIST), quick sort (QSORT), radix sort (RSORT), XOR encryption (XORENC), advanced encryption standard (AES), Knuth-Morris-Pratt string matching algorithm (KMP), 0-1 knapsack problem (KNAPSACK), Dijkstra shortest path algorithm (DIJKSTRA), and Floyd shortest path algorithm (FLOYD). The computation-to-communication ratio of the 10 benchmarks is low so they are suitable for PIM architectures.

Although our FeFET considered and solve the SRAW contention problem, we did not have a chip packaging for the system structure for the lack of copyright protection of FeFETs. We did not consider the job partition of processing between memories and processors, which is inevitable for an efficient Processing-in-Memory system.

**3 RESULT**

We proposed FeFET architecture to alleviate the memory wall bottleneck. We intended to solve SRAW contention.

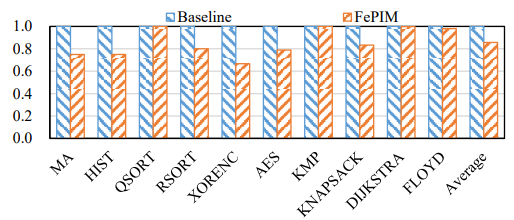
Our FePIM support SRAW operations on both different rows and the same row. We eliminate all stalls caused by SRAW contentions. Compared with an enhanced version of a state-of-the-art FeFET-based PIM design that cannot handle SRAW contentions, the proposed FePIM architecture reduces the memory latency by 15% and the memory energy consumption by 44% on average.

We compared the the energy consumption of basic operations on [13] and our FePIM.. Table 2 shows the energy consumption of read, write, PIM and SRAW operations. For the simple operation (a read, write or PIM operation), our FePIM consumes a little higher energy than the design of [13]. For the SRAW case, our FePIM consumes lower energy than the design of [13].

**Table 2 Comparison on energy consumption (in pJ) of basic operations of a 1MB array with 32-bit word siz**e

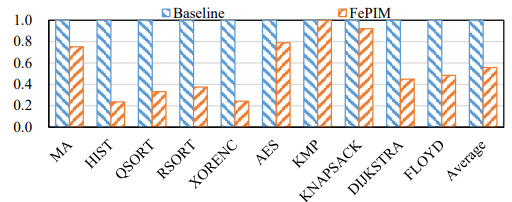
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Read | Write | PIM operation | SRAW |
| [13] | 45.65 | 45.02 | 75.72 | 90.07 |
| Our FePIM | 59.63 | 63.57 | 79.35 | 65.01 |

Fig. 3 shows the system-level normalized memory access latency comparison. Compared with the baseline, FePIM reduces 15% of the memory access latency on average . The elimination of stalls caused by SRAW contentions contributes to the reduction of memory access latency. The reduction of memory access latency of benchmarks QSORT, KMP, DIJKSTRA and FLOYD is negligible, due to their small or even zero contending reading rates.



**Figure 3 Normalized memory access latency comparison**

Fig. 4 shows the system-level memory access energy comparison. On average, Our FePIM reduces the memory access energy consumption by 44%. The elimination of the static power consumed by stalling cycles caused by SRAW contentions reduces the memory access energy.



**Figure 4 Normalized memory access energy comparison**

**4 DISCUSSION**

Processing-in-Memory is a promising technique to alleviate the memory wall bottleneck. Existing nonvolatile device based PIM designs rarely handle the SRAW contention problem. In this work, we employed FeFETs to build a contention-free PIM architecture named FePIM. We utilized the non-volatile feature of FeFETs and designed separated read and write paths to build a FeFET memory array that can support SRAW operations on different rows. Furthermore, a forwarding mechanism is proposed to realize SRAW operations on the same row. Therefore, all possible stalls caused by SRAW contentions can be solved. Compared with an enhanced version of a state-of-the-art FeFET-based PIM design that cannot handle SRAW contentions, our FePIM reduces the memory latency by 15% and the memory energy consumption by 44% on average.

The FePIM architecture did not consider the calculation job partition of the processing unit and memory. This work is expected to have a chip packaging to test the performance in reality.

[1] D. Patterson et al. 1997. A case for intelligent RAM. IEEE Micro 17, 2 (March

1997), 34–44

[2]WILLIAM H. KAUTZ,1969 Cellular logic-In-Memory Arrays, IEEE Trans. Computers, vol. pp.719-727

[3]J. Ahn et al. 2015. A scalable Processing-in-Memory accelerator for parallel graph

processing. In ISCA. 105–117.

[4]M. Gao et al. 2017. TETRIS: Scalable and Efficient Neural Network Acceleration

with 3D Memory. In ASPLOS (ASPLOS ’17). 751–764

[5] S. H. Pugsley et al. 2014. NDC: Analyzing the impact of 3D-stacked memory+logic devices on MapReduce workloads. In ISPASS. 190–200

[6] S. Jain et al. 2018. Computing in Memory With Spin-Transfer Torque Magnetic

RAM. IEEE TVLSI 26, 3 (March 2018), 470–483.

1. P. Gaillardon et al. 2016. The Programmable Logic-in-Memory (PLiM) computer. In DATE. 427–432.
2. M. Imani et al. 2017. MPIM: Multi-purpose in-memory processing using configurable resistive memory. In ASP-DAC. 757–763
3. S. Li et al. 2016. Pinatubo: A Processing-in-Memory architecture for bulk bitwise operations in emerging non-volatile memories. In DAC. 1–6.

[10]D. Reis et al. 2018. Computing in Memory with FeFETs. In ISLPED (ISLPED ’18).24:1–24:6.

[11]A. Aziz et al. 2016. Physics-Based Circuit-Compatible SPICE Model for Ferroelectric Transistors. IEEE EDL 37, 6 (June 2016), 805–808.

[12]ASU. 2011. Predictive Technology Model. <http://ptm.asu.edu/>

[13]D. Reis et al. 2018. Computing in Memory with FeFETs. In ISLPED (ISLPED ’18).24:1–24:6.