Newton: A DRAM-maker's Accelerator-in-Memory (AiM) Architecture for Machine Learning

MICRO'20

Mingxuan He
Electrical and Computer Engineering
Purdue University
West Lafayette, IN, U.S.A.
he238@purdue.edu

Choungki Song
DRAM Design
SK Hynix
Icheon, South Korea
choungki.song@sk.com

Ilkon Kim
DRAM Design
SK Hynix
Icheon, South Korea
ilkon.kim@sk.com

Chunseok Jeong
DRAM Design
SK Hynix
Icheon, South Korea
chunseok.jeong@sk.com

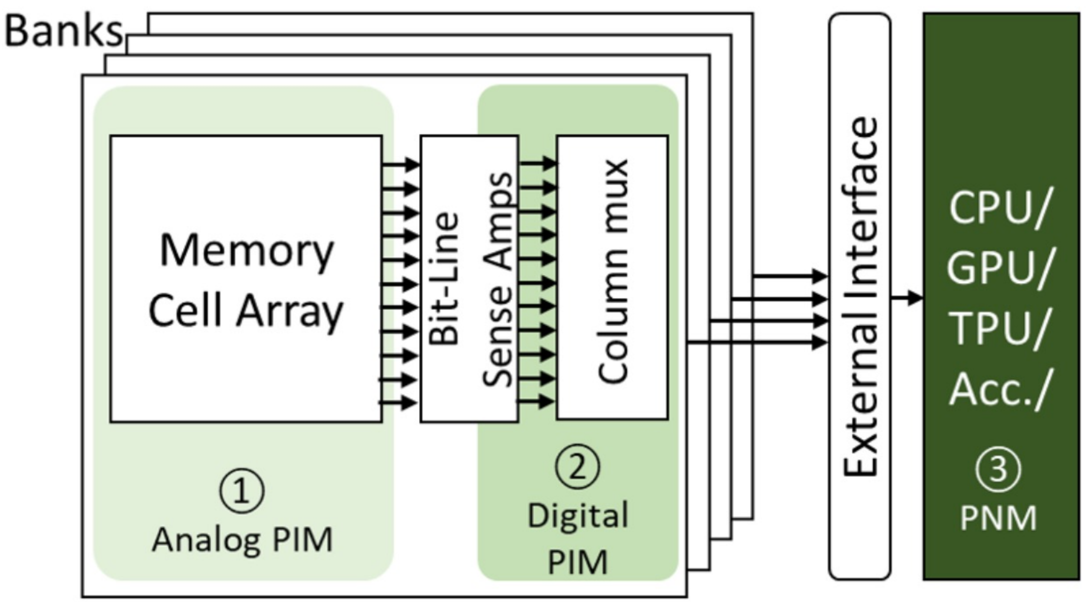
Seho Kim
DRAM Design
SK Hynix
Icheon, South Korea
seho5.kim@sk.com

Il Park
DRAM Design
SK Hynix
Icheon, South Korea
il.park@sk.com

Mithuna Thottethodi
Electrical and Computer Engineering
Purdue University
West Lafayette, IN, U.S.A.
mithuna@purdue.edu

T. N. Vijaykumar
Electrical and Computer Engineering
Purdue University
West Lafayette, IN, U.S.A.
vijay@ecn.purdue.edu

Processing in/near Memory



Analog PIM
- Low power, Fast
- **Accuracy loss**

Digital PIM
- Low power, Fast
- **Limited PE**

PNM
- Many computation units
- **Low bandwidth**

When is digital PIM beneficial?

- Compute-bound operation
  - Requires large # of PEs and high data reuse
  - Not suitable for PIM
- Memory-bound operation

	small (high temporal reuse)	large (low temporal reuse)
small (high temporal reuse)	e.g. Tiled Conv	Matrix-vector multiplication
large (low temporal reuse)	Matrix-vector multiplication	Element-wise Operation (e.g. Add)

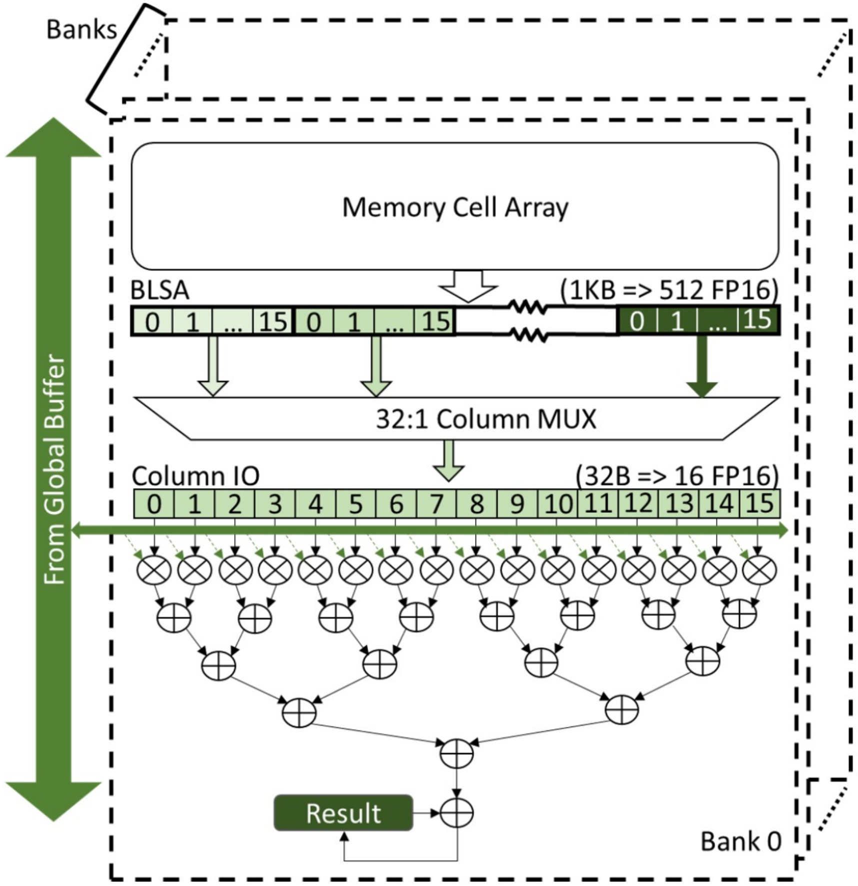
Matrix-Vector Multiplication

- LSTM, RNN, FC
- MAC operations

Contributions

- Place a minimal compute of **only MAC units and buffers**
  - Previous works use an large number of PEs
    - e.g. superscalar, vector/SIMD
    - But PIM is subject to severe are and power constraints
- **DRAM command-like interface** for the host to issue commands
- Prevent PIM-host interface from becoming bottleneck
  - ***gang*** multiple compute operations
  - ***complex*** compute commands
  - targeted reduction of timing overhead (e.g. t_{FAW})
- Reduce output vector write traffic
  - unusually-wide **interleaved layout for the filter matrix**

Newton Datapath



All banks perform compute simultaneously (*gang*)

Reduction tree for reducing # MAC units

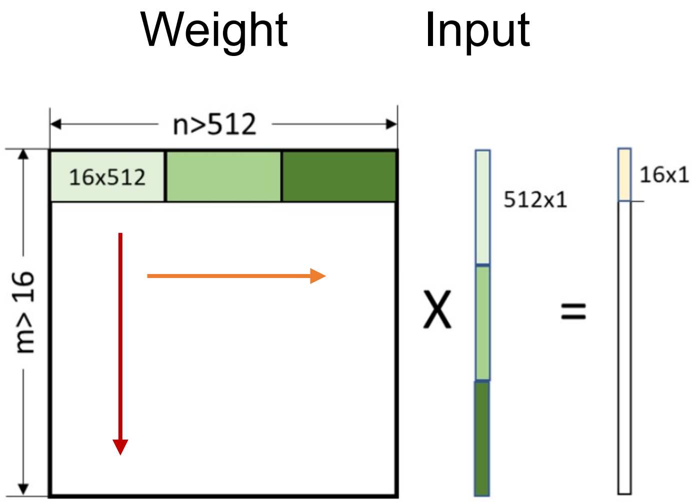
Interleaved accumulation for reducing output traffic

Assumption: BFloat16 / FP16
Justification: High accuracy for NLP task

Newton's Tiled MV computation

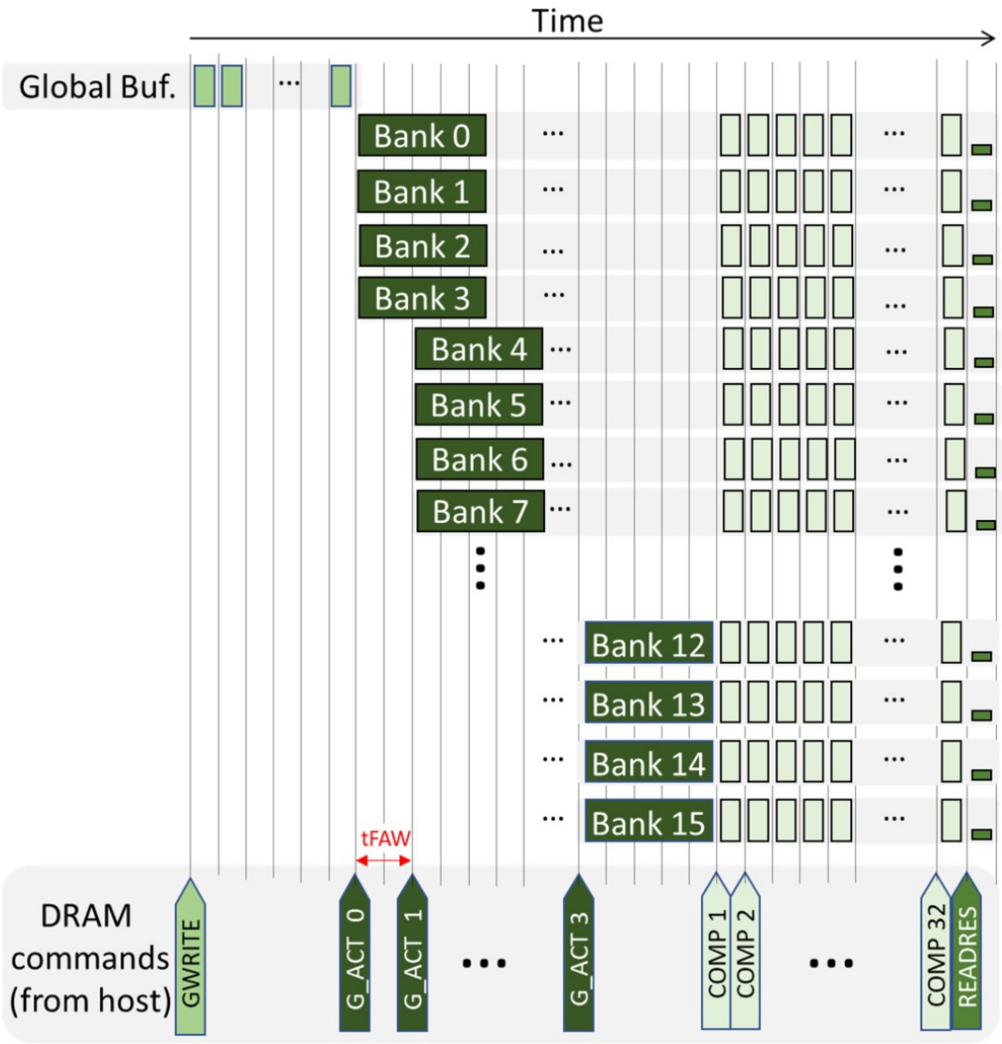
Algorithm 1 Newton's Tiled MV computation

```
function MVPRODUCT(InputVector V, Matrix M, m, n)
    numChunks = n/512                                  ▷ Number of chunks
    C[1..numChunks] ← split(V)                         ▷ Split vector to chunks
    for i ∈ 1..numChunks do                            ▷ Outermost loop.
        GlobalBuffer ← C[i]
        r = m/16                                       ▷ Number of vertical tile positions
        for j ∈ 1..r do
    ▷ Compute 1 DRAM row x Global Buffer per bank
            for all b ∈ 1..numBanks do
                Results[b] ← ComputeTile(Tile j, Row b)
            end for
            TileResult ← ReadResultsFromAllBanks()
                ▷ Tile result sent for accumulation at host
        end for
    end for
end function
```



Interleaved (row-first) to reduce output buffer traffic

Newton computation



Command	Operation
COMP#	Ganged multiply of sub-chunk# in all banks
READRES	Read the Result latches of all banks
GWRITE#	WRITE sub-chunk# to the Global Buffer
G_ACT#	Ganged activation of 4-bank cluster#

Complex command ➔ low PIM-host BW consumption

Methodology

- Benchmarks

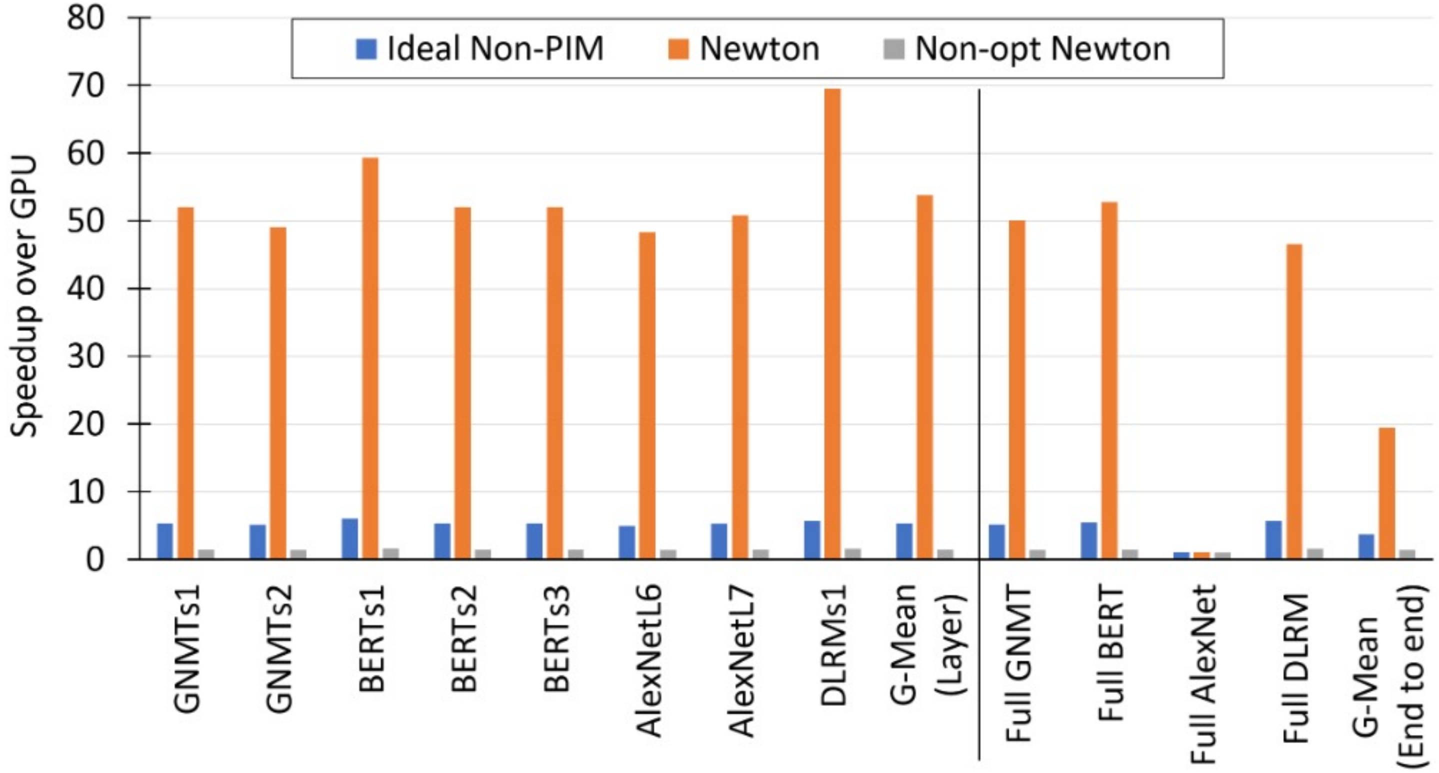
Workload	Matrix	Vector
GNMT LSTMs1 [45]	4096×1024	1024×1
GNMT LSTMs2 [45]	4096×2048	2048×1
BERTs1 [10]	1024×1024	1024×1
BERTs2 [10]	1024×4096	4096×1
BERTs3 [10]	4096×1024	1024×1
AlexNetL6 [26]	21632×2048	2048×1
AlexNetL7 [26]	2048×2048	2048×1
DLRMs1 [31]	512×256	256×1

- DRAM configuration

Num of Ranks	1
Num of Banks	16
Num of Rows in each bank	32768
Num of Column I/Os per row	32
Column I/O bit width	256b (16 `bfloat16`)
Num of Multipliers per bank	16
Timing Parameters (in nanoseconds)	
t_{AA} = 22–29 ns; t_{RP} = 14 ns; t_{RCD} = 14 ns; t_{RAS} = 33 ns	

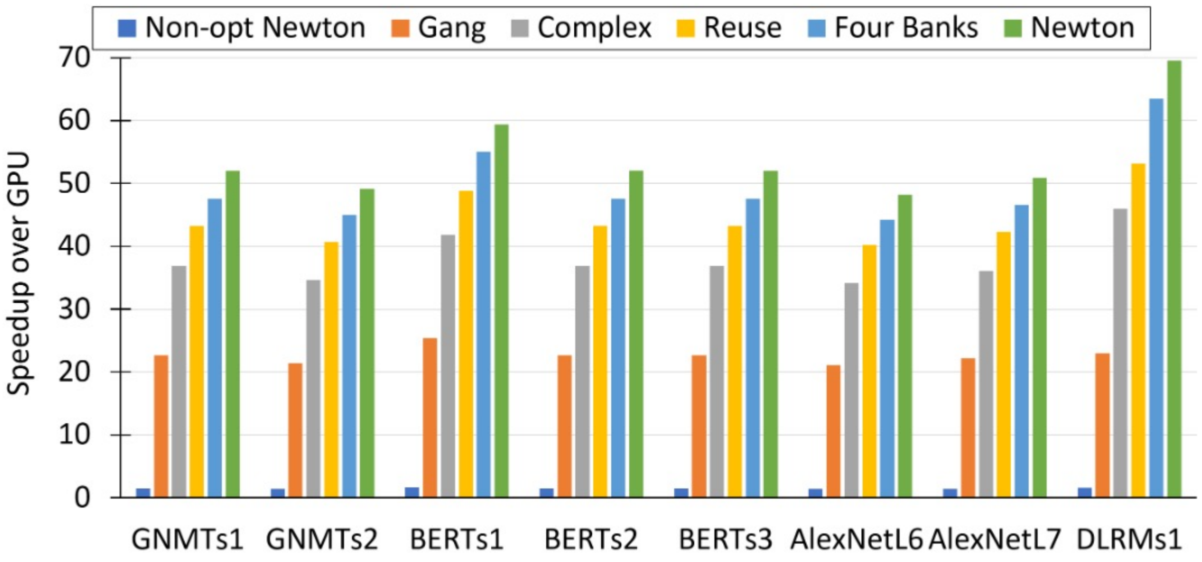
Only show partial parameters (proprietary)
No power parameters (proprietary)

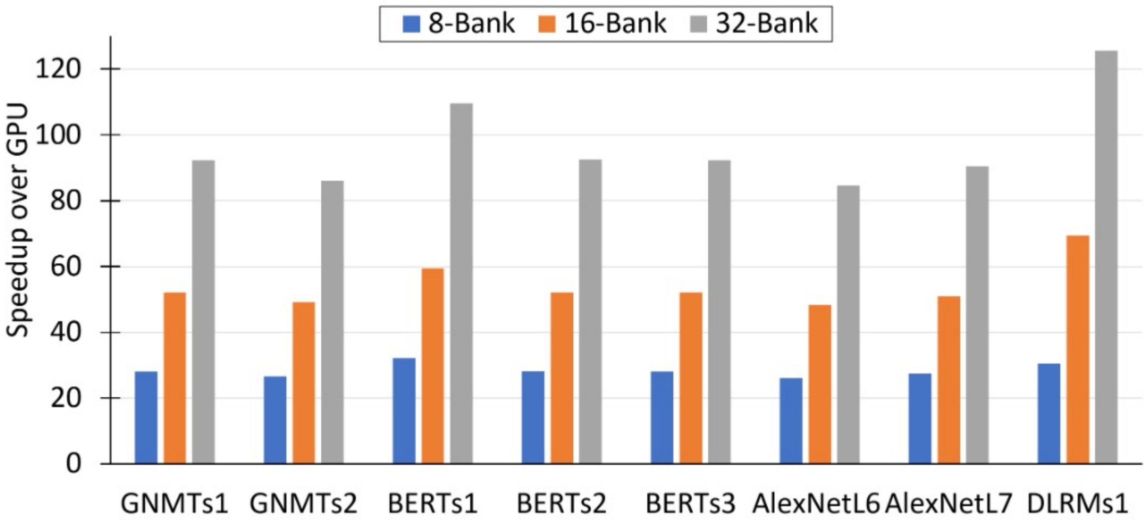
Evaluation



- Newton's speedup comes from **BW reduction**, not from computation capability (vs Ideal Non-PIM)
- DLRMs1 shows better performance since there's no DRAM refresh (short computation)

Sensitivity Study

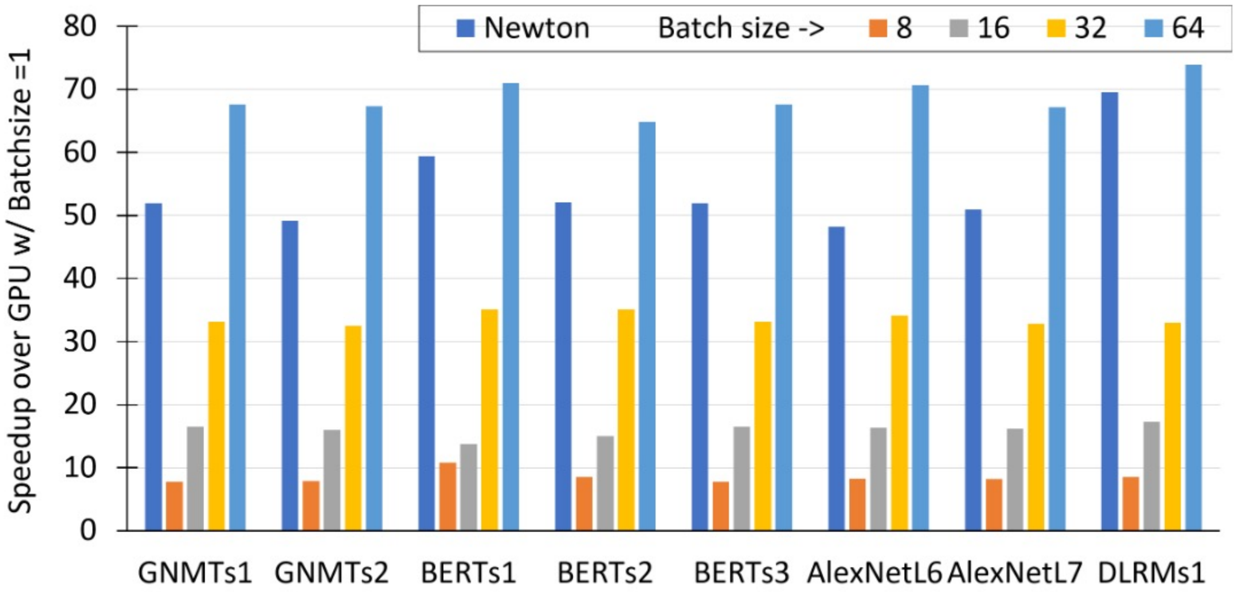




- Gang: all-bank ganged compute commands
- Complex: multi-step compute commands
- Reuse: reuse via tiling and interleaved layout for the filter matrix
- Four Banks: four-bank ganged activations
- Newton: aggressive t_{FAW}

- Nearly perfect scaling

Sensitivity Study (Batch size)



- Newton is good for inference (batch size <= 8)
- More data-reuse in GPU → Less profitability for Newton

Hardware Architecture and Software Stack for PIM Based on Commercial DRAM Technology

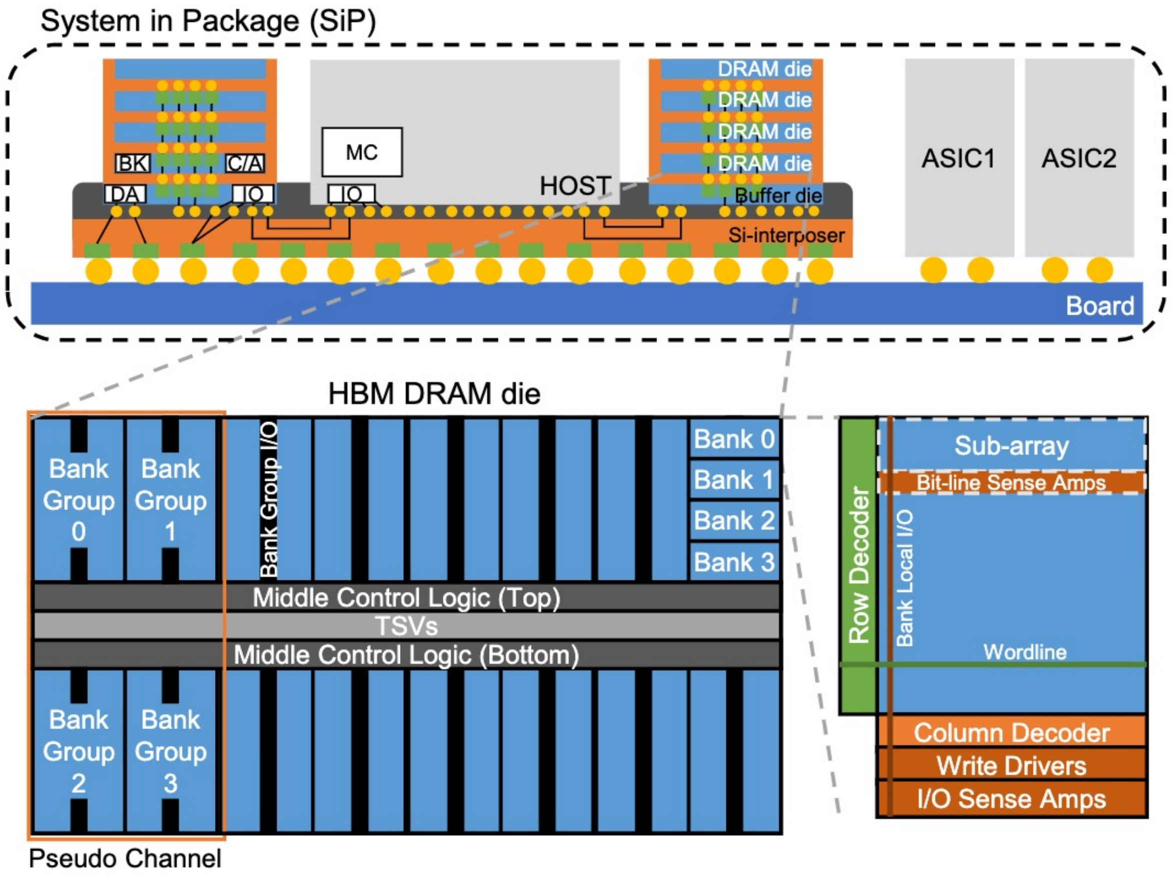
ISCA'21

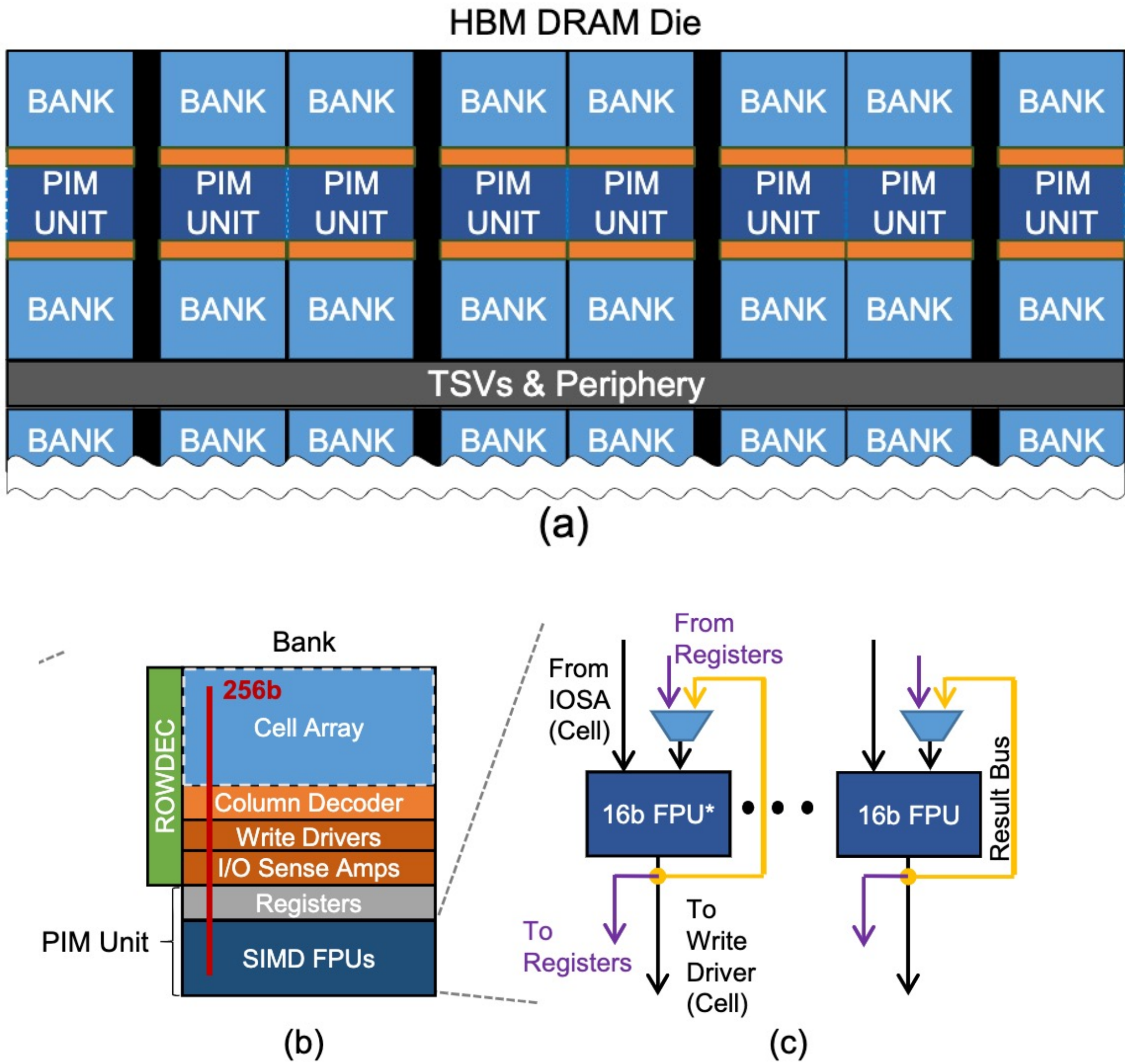
Industrial Product

Sukhan Lee[§1], Shin-haeng Kang[§1], Jaehoon Lee[1], Hyeonsu Kim[2], Eojin Lee[1], Seungwoo Seo[2], Hosang Yoon[2], Seungwon Lee[2], Kyounghwan Lim[1], Hyunsung Shin[1], Jinhyun Kim[1], Seongil O[1], Anand Iyer[3], David Wang[3], Kyomin Sohn[1] and Nam Sung Kim[§1]

[1]Memory Business Division, Samsung Electronics
[2]Samsung Advanced Institute of Technology, Samsung Electronics
[3]Device Solutions America, Samsung Electronics

HBM structure and PIM units

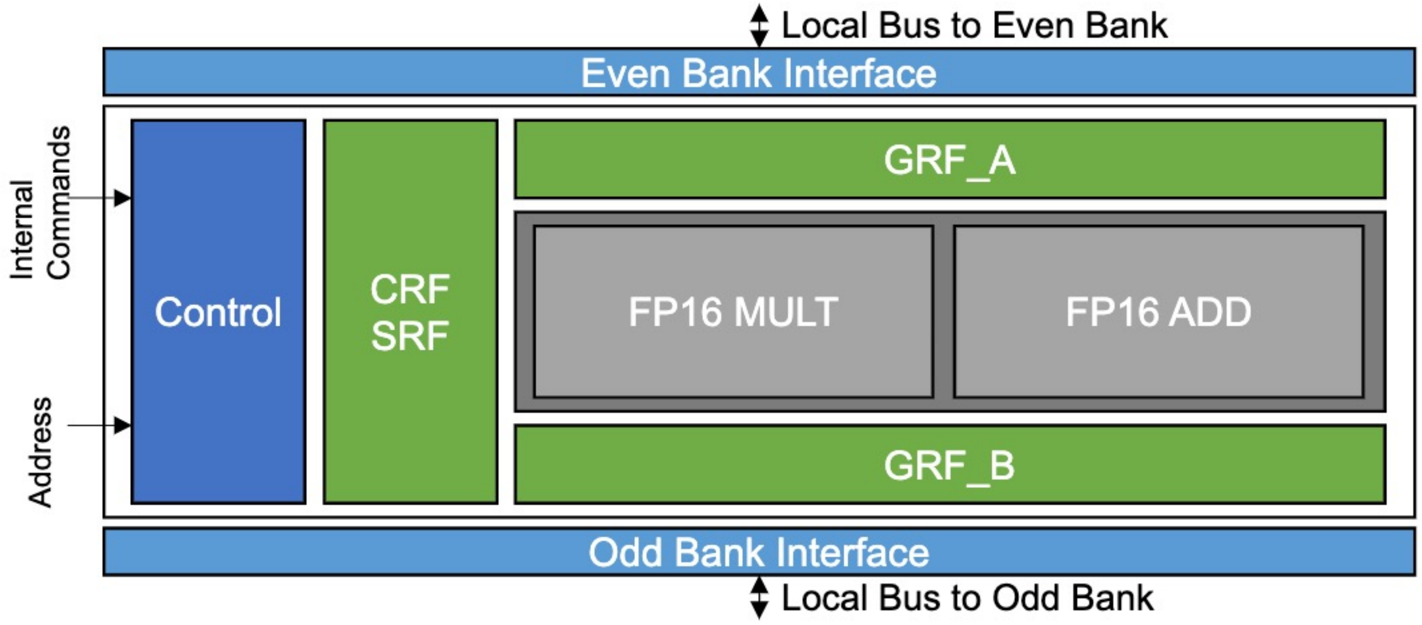




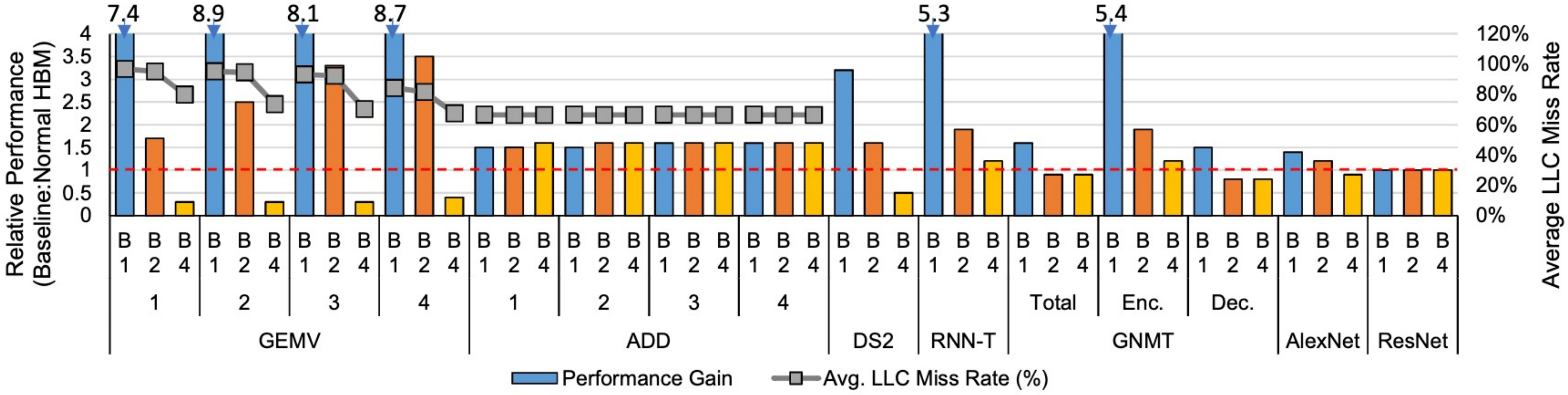
RISC-V-like instruction format

TABLE III: The PIM-HBM instruction format.

<table>
<tr><td></td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>Control</td><td colspan="4">OPCODE</td><td colspan="9">U</td><td colspan="8">IMM0</td><td colspan="11">IMM1</td></tr>
<tr><td>Data</td><td colspan="4">OPCODE</td><td colspan="3">DST</td><td colspan="3">SRC0</td><td colspan="9">U</td><td>R</td><td>U</td><td colspan="3">DST #</td><td>U</td><td colspan="3">SRC0 #</td><td>U</td><td colspan="3">SRC1 #</td></tr>
<tr><td>ALU</td><td colspan="4">OPCODE</td><td colspan="3">DST</td><td colspan="3">SRC0</td><td colspan="3">SRC1</td><td colspan="3">SRC2</td><td>A</td><td colspan="3">U</td><td>U</td><td colspan="3">DST #</td><td>U</td><td colspan="3">SRC0 #</td><td>U</td><td colspan="3">SRC1 #</td></tr>
</table>



Evaluation



- Much less speedup than Newton (IF/ID and control overhead, ¼ PEs)
- Support wide range of operations (e.g. Add, BN)

Drawbacks of digital PIM

- Tricky cache coherence (Both)
- Lack of software stack (Newton)
  - Samsung is building its own stack
- Limited operation support (Newton)
  - What about other DL models?