HieIM: Highly Flexible In-Memory Computing Using STT MRAM

Deliang Fan
Assistant Professor
dfan@ucf.edu
http://www.eecs.ucf.edu/~dfan/
Department of Electrical and Computer Engineering,
University of Central Florida, Orlando, FL
OUTLINE

- Motivation
- Post-CMOS Spintronic Devices
- In-Memory Processing Platform based on STT-MRAM
- Performance Evaluation
- Case Study I: In-memory Bulk Bitwise Vector Operation
- Case Study II: In-memory Data Encryption Engine
Energy efficient and high performance computing hardware development is beginning to stall fundamentally due to limitations in both devices and architectures.

First, the current computing platforms primarily depend on Complementary Metal Oxide Semiconductor (CMOS) technology, which is reaching its power wall.
MOTIVATION (ARCHITECTURE)

Von-Neumann architecture

Controller

Memory

Logic

- Energy hungry data transfer
- Long memory access latency
- Limited memory bandwidth

VS.

In-Memory Computing Cluster

Controller

In-Memory Computing Unit

Memory & Logic

Memory & Logic

- Parallel, local data processing
- Short memory access latency
- Ultra-low energy
- Programmable, Low cost/area

There is an urgent need to investigate fundamentally different devices and architectures for information processing and data storage with the ability to continuously deliver energy efficient and high performance computing solutions.

Ambit: DRAM-based

- Operand locality issue
- Original data overwritten
- Multi-Cycle operations
- Low area overhead
- Hardware-friendly
- Exploiting the full internal DRAM bandwidth

Pinatubo: NVM-based

- Operand locality issue
- Large area overhead
- Fast MG-based computation
- Ultra-low power

RIMPA: DWM-based

- Operand locality issue
- Modified SA
- Medium area overhead
- Support one-step multi-row operations
- General platform

HielM: MRAM-based

- Operand locality issue
- Large area overhead
- Fast MG-based computation
- Ultra-low power
OUTLINE

- Motivation
- Post-CMOS Spintronic Devices
- In-Memory Processing Platform based on STT-MRAM
- Performance Evaluation
- Case Study I: In-memory Bulk Bitwise Vector Operation
- Case Study II: In-memory Data Encryption Engine
POST-CMOS SPINTRONIC DEVICES

STT-MRAM

Domain Wall Motion Device

Possible wall structures in a magnetic nanowire

<table>
<thead>
<tr>
<th>Operations</th>
<th>Write ‘1’(‘0’)</th>
<th>Read</th>
</tr>
</thead>
<tbody>
<tr>
<td>WL</td>
<td>$V_{DD}$</td>
<td>$V_{DD}$</td>
</tr>
<tr>
<td>BL</td>
<td>GND ($V_{DD}$)</td>
<td>$I_{sense}$</td>
</tr>
<tr>
<td>SL</td>
<td>$V_{DD}$ (GND)</td>
<td>GND</td>
</tr>
</tbody>
</table>
OUTLINE

- Motivation
- Post-CMOS Spintronic Devices
- In-Memory Processing Platform based on STT-MRAM
- Performance Evaluation
- Case Study I: In-memory Bulk Bitwise Vector Operation
- Case Study II: In-memory Data Encryption Engine
IN-MEMORY PROCESSING PLATFORM

- **Dual mode architecture** that perform both memory read-write and in-memory logic (AND/NAND, OR/NOR, XOR/XNOR).

- **Memory Write**: To write data in a memory cell, the corresponding WL is activated using the row decoder. Then appropriate voltage difference is applied to the corresponding BL and SL using the voltage drivers.

- **Memory Read**: The corresponding WL is activated using the row decoder and the corresponding BL is connected to the sense amplifier (SA) using the column decoder.

- **Computing Mode**: We propose a sensing circuit design using 5T DWM device [1], as an extension to SA of memory array, to implement complete Boolean logic functions between any two cells in the memory array.

---

For a complete Boolean operation, the SA extension needs 3 subsequent stages: Reset, Compute and Sense.

In Reset stage (Reset=1), the reset transistor is turned on for 1ns. A current of 48uA flows from W- to W+ terminals, which sets the DW back to its initial position at W- side.

In Compute stage, two operands stored in the memory array are read in two consecutive cycles using the SA and applied to DWM device.

In Sense stage, a small sensing current is injected through DWM device from R+ to R1- or from R+ to R2- terminals based on required logic implementation.
OUTLINE

- Motivation
- Post-CMOS Spintronic Devices
- In-Memory Processing Platform based on STT-MRAM
- Performance Evaluation
- Case Study I: In-memory Bulk Bitwise Vector Operation
- Case Study II: In-memory Data Encryption Engine
PERFORMANCE EVALUATION

Device to System Level Simulations:

Device Level:
Verilog-A model of 5T DWM device was developed to co-simulate with the interface CMOS circuits in SPICE to validate the functionality and evaluate performance of the proposed design. The STT-MRAM is simulated by solving LLG equation to model dynamics of MTJ free layer.

Circuit Level:
45nm North Carolina State University (NCSU) Product Development Kit (PDK) [1] library is used in SPICE to verify the proposed design and evaluate the performance.

System Level:
We employ the modified self-consistent NVSim [2] along with an in-house developed C++ code to verify the performance of memory.

[1] www.eda.ncsu.edu/wiki/FreePDK45
PERFORMANCE EVALUATION

Memory Mode:

- The proposed STT-MRAM memory model shows the least write dynamic energy in comparison to other designs.
- It reduces the total leakage power compared to SRAM.
- It shows longer average latency compared to SRAM due to the longer write latency of magnetic memory storage.
- Its area overhead is 29.1% more than DRAM but still 37.51% less than SRAM design.

Computing Mode:

- The in-memory AND operation shows 65.3% and 81.32% lower energy consumption than Domain-Wall (DW) Racetrack based and MTJ based in-memory non-volatile AND gate implementations.
- Our design requires longer latency to compute the logic result than other designs.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Energy (fJ)</td>
<td>23.5</td>
<td>67.72</td>
<td>125.85</td>
<td>504.36</td>
<td>6.69</td>
</tr>
<tr>
<td>Speed (ns)</td>
<td>4</td>
<td>1.12</td>
<td>1.18</td>
<td>2.14</td>
<td>0.062</td>
</tr>
</tbody>
</table>

SRAM, DRAM AND PROPOSED STT-MRAM MEMORY MODEL VALIDATION AND COMPARISON FOR A SAMPLE 4MB MEMORY

<table>
<thead>
<tr>
<th>Metrics</th>
<th>SRAM</th>
<th>DRAM</th>
<th>STT-MRAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average Latency (ns)</td>
<td>Write</td>
<td>Read</td>
<td>Write</td>
</tr>
<tr>
<td>Dynamic Energy (pJ)</td>
<td>1213</td>
<td>1483</td>
<td>917.8</td>
</tr>
<tr>
<td>Leakage Power (mW)</td>
<td>5316</td>
<td>185.5</td>
<td>830.847</td>
</tr>
<tr>
<td>Area (mm²)</td>
<td>10.613</td>
<td>4.702</td>
<td>6.632</td>
</tr>
</tbody>
</table>

PERFORMANCE EVALUATION OF FA CELLS

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Average Power (µW)</td>
<td>91.93</td>
<td>1354</td>
<td>85</td>
<td>15.6</td>
<td>49.4</td>
</tr>
<tr>
<td>Delay (ps)</td>
<td>26,000</td>
<td>269</td>
<td>877</td>
<td>10,000</td>
<td>1000</td>
</tr>
</tbody>
</table>

Ambit: DRAM-based
- Operand locality issue
- Original data overwritten
- Multi-Cycle operations
- Low area overhead
- Hardware-friendly
- exploiting the full internal DRAM bandwidth

Pinatubo: NVM-based
- Operand locality issue
- Modified SA
- Medium area overhead
- support one-step multi-row operations
- General platform

RIMPA: DWM-based
- Operand locality issue
- Large area overhead
- Fast MG-based computation
- Ultra-low power

HielM: MRAM-based
- Long Latency
- Modified SA
- Medium area overhead
- Ultra-low power
- No operand locality issue
OUTLINE

- Motivation
- Post-CMOS Spintronic Devices
- In-Memory Processing Platform based on STT-MRAM
- Performance Evaluation
- Case Study I: In-memory Bulk Bitwise Vector Operation
- Case Study II: In-memory Data Encryption Engine
CASE STUDY I: IN-MEMORY BULK BITWISE VECTOR OPERATION

- Four different vector datasets [1] have been used. Here, a dataset ‘19-16-1s’ refers to a vector dataset with vector length=2^19, number of vectors=2^16, and AND/OR operation is done between 2^1 rows.

- Each compute (AND/OR) operation has been carried out using 4 consecutive clock cycles (1ns each).

- HieIM offers ~8× energy saving and ~5× speed up compared to that using Ambit-DRAM based in-memory computing platform [2].

![Data mapping for performing vector operation between two 32 bit vectors using an 8*8 STT-MRAM array](image)


OUTLINE

- Motivation
- Post-CMOS Spintronic Devices
- In-Memory Processing Platform based on STT-MRAM
- Performance Evaluation
- Case Study I: In-memory Bulk Bitwise Vector Operation
- Case Study II: In-memory Data Encryption Engine
CASE STUDY II: IN-MEMORY DATA ENCRYPTION ENGINE

- Advanced Encryption Standard (AES) has been used to employ in-memory data encryption engine using HieIM.
- HieIM can achieve 51.5% and 68.9% lower energy consumption compared to CMOS-ASIC and CMOL based implementations, respectively.
- HieIM occupies ~3.5× less area compared to baseline DW-AES.
- Note that, Baseline DW AES [36] requires lower number of cycles due to intrinsic shift operation and multi-bit data storage of DWM racetrack devices.

AES PERFORMANCE

<table>
<thead>
<tr>
<th>Platforms</th>
<th>Energy (nJ)</th>
<th>Cycles</th>
<th>Area (μm²)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPP [37]</td>
<td>460</td>
<td>2309</td>
<td>2.5e+6</td>
</tr>
<tr>
<td>ASIC [41]</td>
<td>6.6</td>
<td>336</td>
<td>4400</td>
</tr>
<tr>
<td>CMOL [42]</td>
<td>10.3</td>
<td>470</td>
<td>320</td>
</tr>
<tr>
<td>Baseline DW [36]</td>
<td>2.4</td>
<td>1022</td>
<td>78</td>
</tr>
<tr>
<td>Pipelined DW [36]</td>
<td>2.3</td>
<td>2652</td>
<td>83</td>
</tr>
<tr>
<td>Multi-issue DW [36]</td>
<td>2.7</td>
<td>1320</td>
<td>155</td>
</tr>
<tr>
<td>HieIM</td>
<td>3.2</td>
<td>1620</td>
<td>21.8</td>
</tr>
</tbody>
</table>

CONCLUSION

- In this work, we develop a new in-memory processing architecture based on STT-MRAM called HeiIM, which could be used as both non-volatile memory and reconfigurable in-memory logic.

- HeiIM offers several significant features as non-volatility, in-memory logic, operation with high data mapping flexibility, low dynamic power consumption and high packing density.

- The in-memory AND operation itself shows 65.3% and 81.32% lower energy consumption than Domain-Wall (DW) Racetrack based and MTJ based in-memory non-volatile AND implementations.

- In-memory bulk bitwise Boolean vector logic (AND/OR) operation for different vector datasets ~8× energy saving and ~5× speed up compared to that using DRAM based in-memory computing platform.

- We further have employed in-memory data encryption engine using AES algorithm, which shows 51.5% and 68.9% lower energy consumption compared to CMOS-ASIC and CMOL-based implementations, respectively.
THANKS