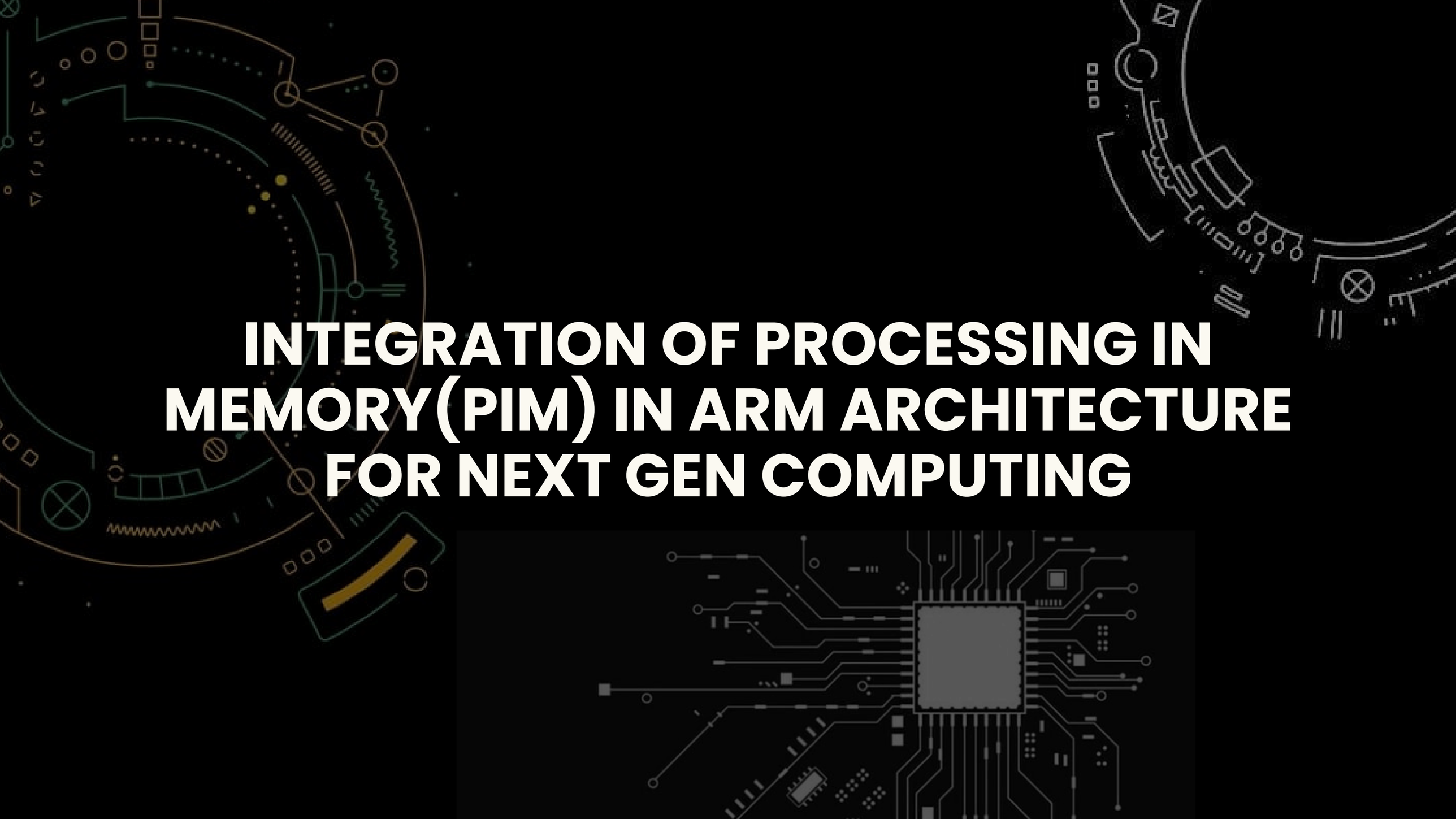
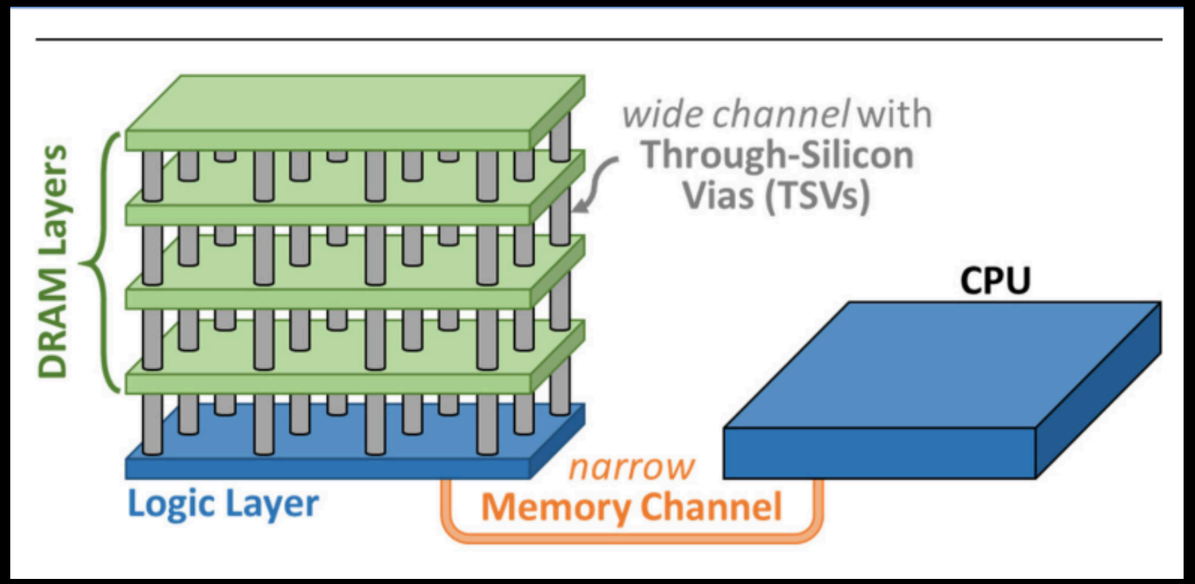
INTEGRATION OF PROCESSING IN MEMORY(PIM) IN ARM ARCHITECTURE FOR NEXT GEN COMPUTING



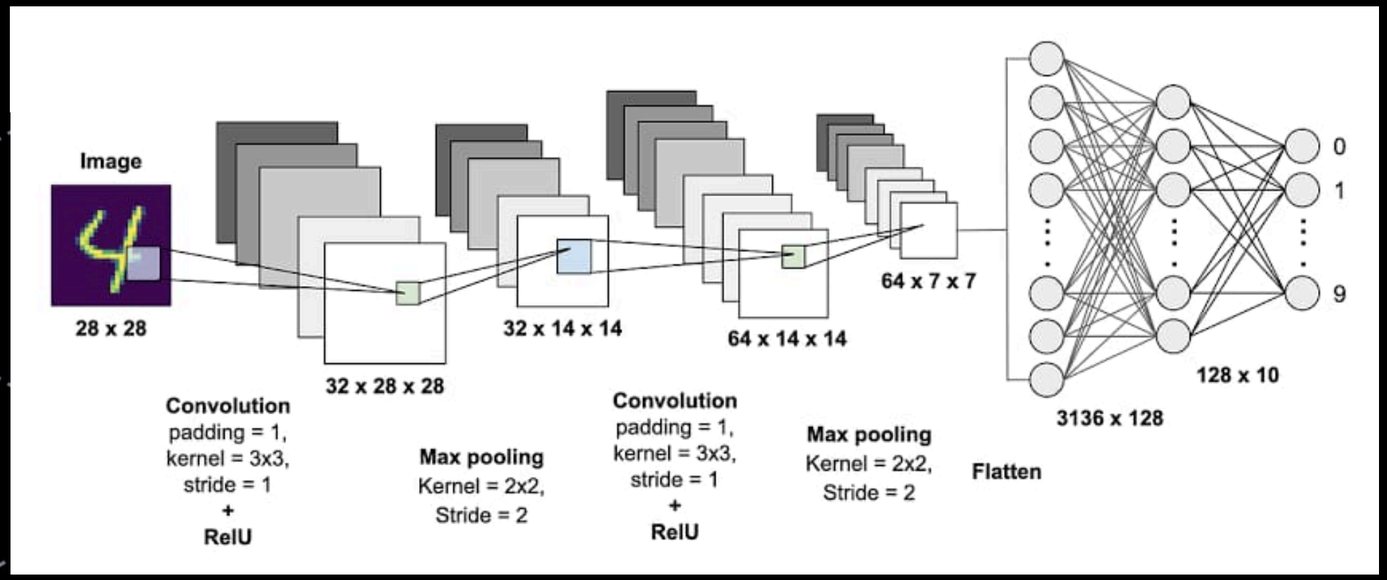
The Data Movement Bottleneck:

- **Definition:** A data movement bottleneck is when system performance is limited by slow or inefficient data transfer between components (e.g., memory to processor, across networks).
- **Cause:** Continuous data transfer via a narrow memory channel (e.g, 64-bit DDR)
- **Effects:**
  - Long latency: Data shuttles between memory and CPU.
  - High energy use: Dominates computation cost.
  - Reduced efficiency: Slows execution



Abstract:

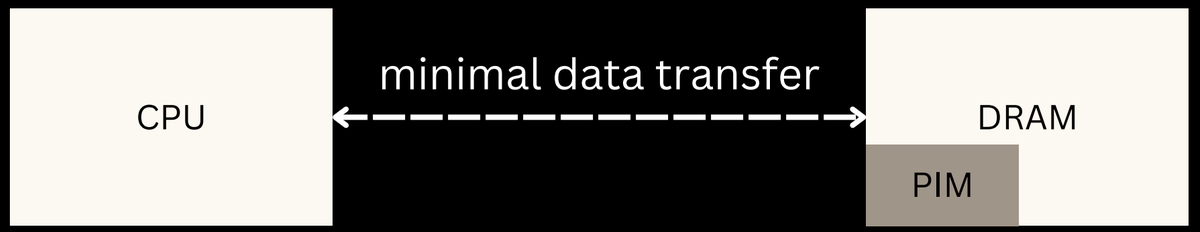
- Objective: Address data movement bottlenecks in data-intensive tasks like AI/ML (e.g., CNNs).
- Methodology:
  - Integrate PIM technology into ARM-based processors.
  - Process data directly in memory to reduce CPU-memory transfers.
- Evaluation: Train a CNN model on the PIM-enabled ARM system and measure efficiency gains.



Proposed solution:

- Solution: Processing-in-Memory (PIM) technology.
- Core Idea: Embed a PIM layer in memory to process data locally.
- How It Helps:
  - Eliminates unnecessary data movement.
  - Reduces power consumption.
  - Improves latency and efficiency.
- Application:
  - Enhances performance in data-intensive tasks (e.g., CNN training).

without PIM



with PIM

Types of PIM:

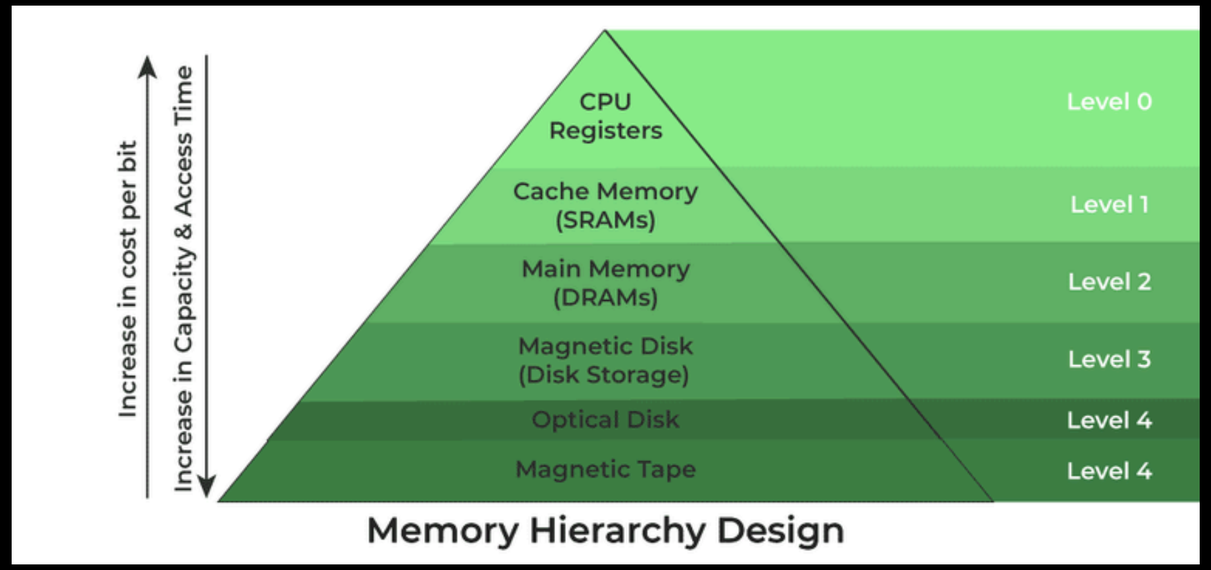
Type	Description	Example	Benefit
Processing-Near-Memory	Adds logic (e.g., cores, accelerators) near memory.	3D-stacked DRAM with logic layer.	High bandwidth, low latency.
Processing-Using-Memory	Uses memory cells for computation.	Logic operations in DRAM or NVM (e.g., PCM).	Energy-efficient, minimal data movement.

Efficiency of PIM:

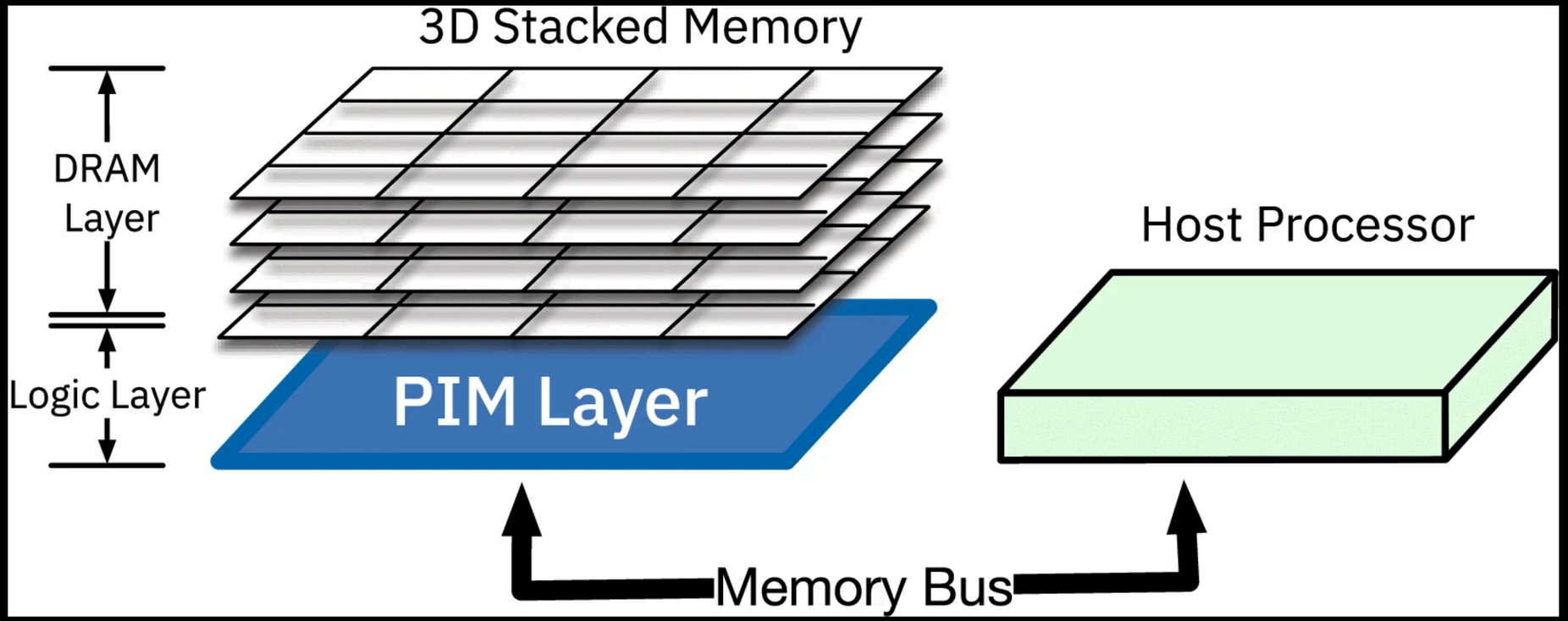
- Performance Gain:
  - Reduces latency by eliminating data transfers.
  - Example: TensorFlow Lite packing/unpacking improved by 31% in execution time.
- Energy Savings:
  - Cuts power consumption significantly.
  - Data movement accounts for 82.1% of packing energy—PIM offloads this.
  - Example: 40% energy reduction in TensorFlow Lite operations.
- Case Study:
  - Quantization in ResNet-v2: 73.5% reduction in data movement.

ARM Memory architecture:

- Overview:
  - Hierarchical memory system: Cache (L1, L2), DRAM.
  - Narrow memory channel (e.g., 64-bit DDR).
- Limitation:
  - Data-intensive tasks (e.g., AI/ML) overload the memory controller.
  - High latency and energy use due to frequent data transfers.
- Opportunity:
  - ARM’s efficiency in embedded systems makes it ideal for PIM integration.
  - Enables low-power, high-performance computing.



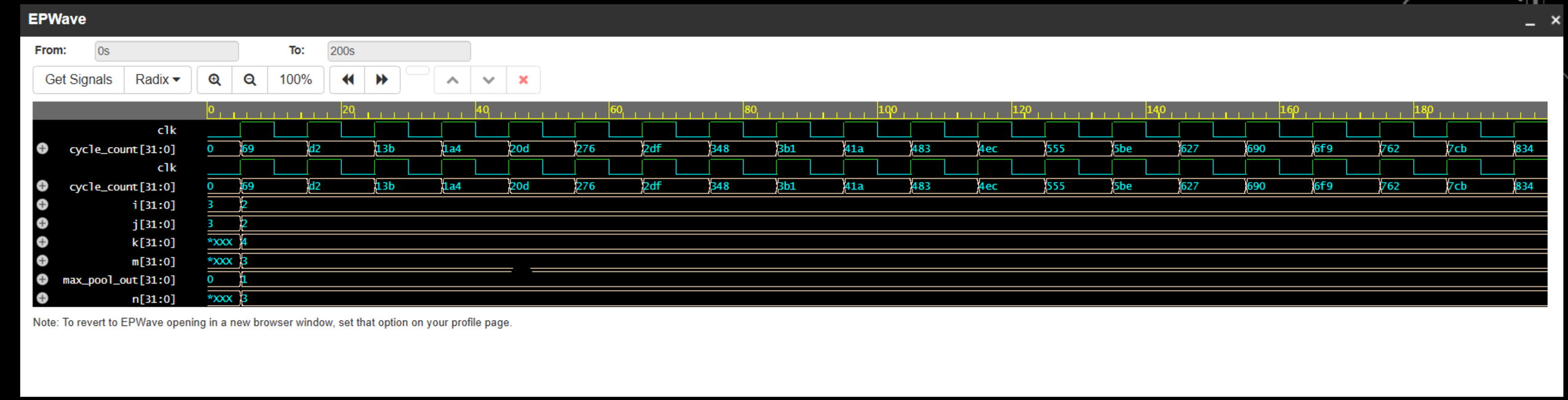
Integration of PIM in ARM architecture:



Comparing CNN models with and without PIM

CNN model without PIM:

```
[2025-03-27 05:17:45 UTC] iverilog '-Wall' '-g2012' design.sv testbench.sv  && unbuffer vvp a.out
VCD info: dumpfile dump.vcd opened for output.
========== CNN Without PIM ==========
Matrix Multiplication Cycles:       1280
Max-Pooling Cycles:           80
Convolution Cycles:          720
Total Cycles:         2100
=====================================
testbench.sv:36: $finish called at 200 (1s)
Finding VCD file...
./dump.vcd
[2025-03-27 05:17:46 UTC] Opening EPWave...
Done
```



Note: To revert to EPWave opening in a new browser window, set that option on your profile page.

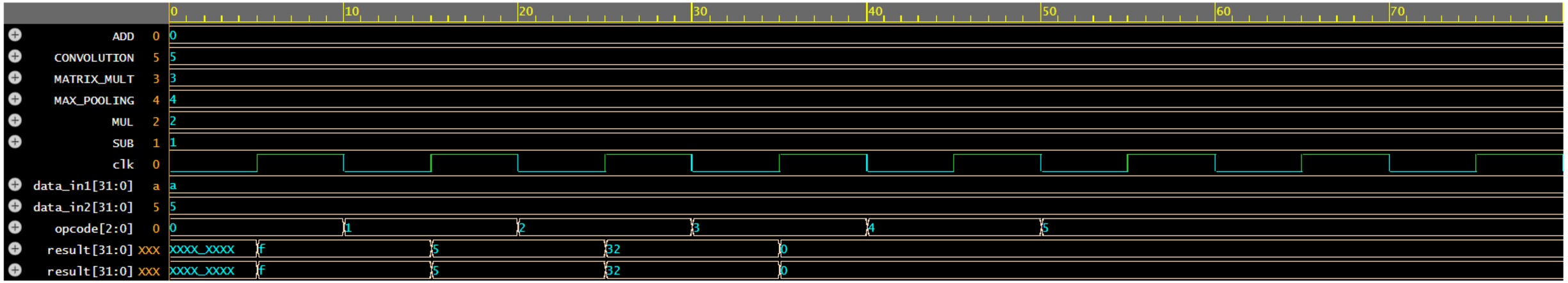
CNN with PIM:

```
[2025-03-27 04:52:33 UTC] iverilog '-Wall' '-g2012' design.sv testbench.sv  && unbuffer vvp a.out
VCD info: dumpfile dump.vcd opened for output.
VCD warning: ignoring signals in previously scanned scope testbench.uut.
ADD Result:          15
SUB Result:           5
MUL Result:          50
Matrix Multiplication Output (Cycle Count:          4):
          0           1           2           3
          1           2           3           4
          2           3           4           5
          3           4           5           6
Max-Pooling Result:           0 (Cycle Count:          5)
Convolution Output (Cycle Count:          6):
          9          18
         18          36
testbench.sv:69: $finish called at 80 (1s)
Finding VCD file...
./dump.vcd
[2025-03-27 04:52:33 UTC] Opening EPWave...
Done
```

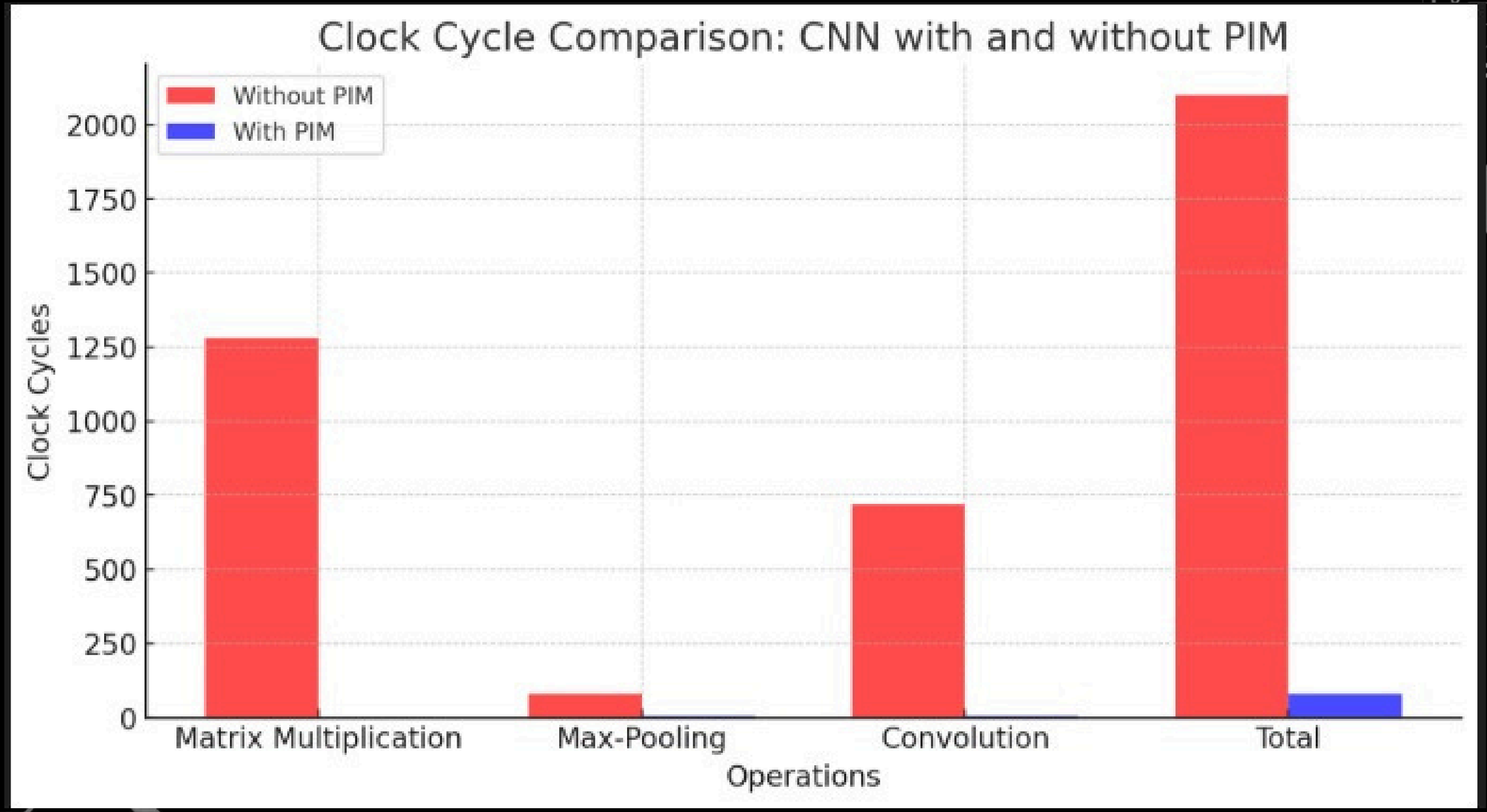
EPWave

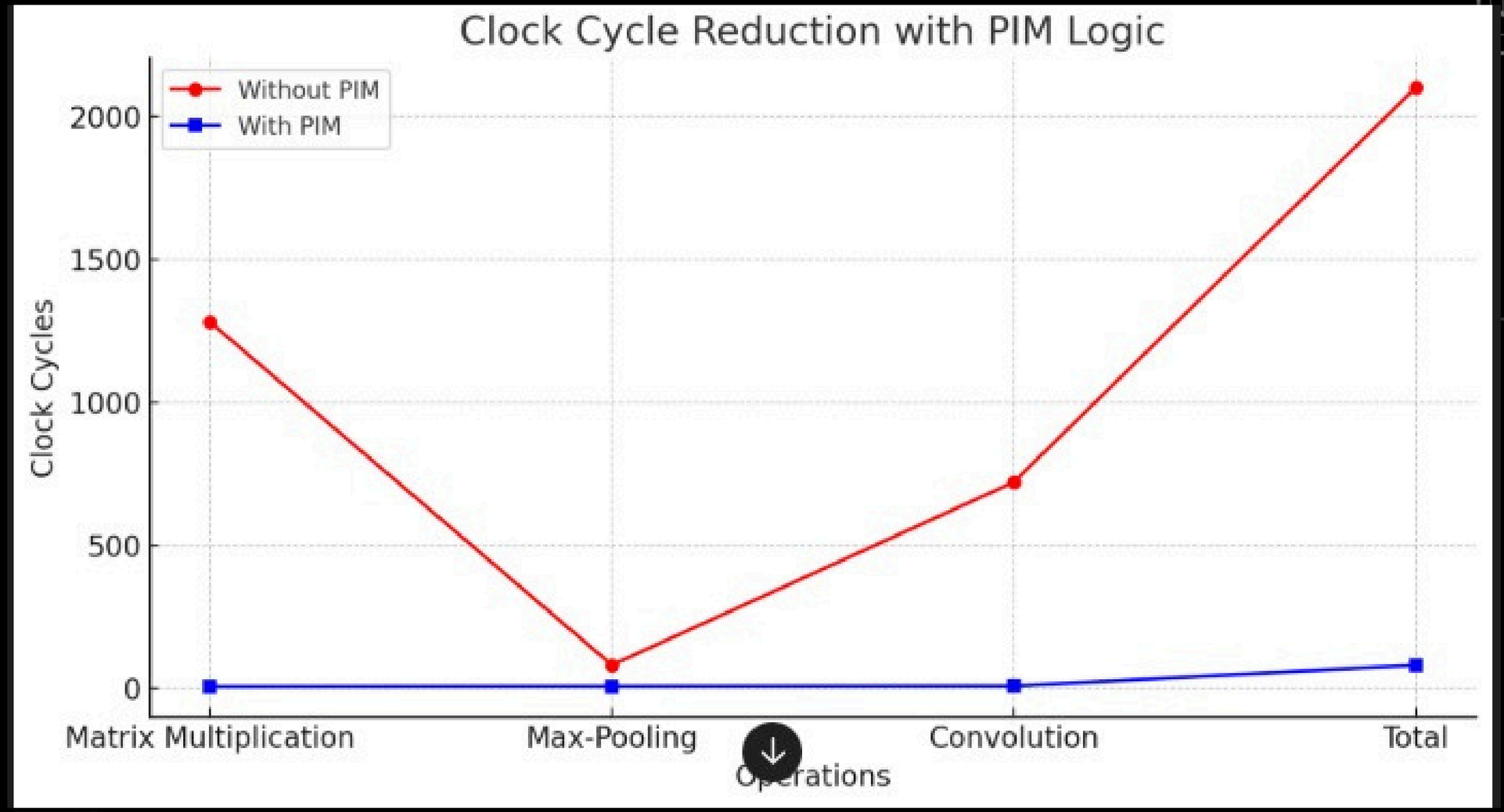
From: 0s To: 80s

Get Signals | Radix | 100% | 0s



Note: To revert to EPWave opening in a new browser window, set that option on your profile page.


Clock Cycle Comparison: CNN with and without PIM
Without PIM
With PIM
Clock Cycles
2000
1750
1500
1250
1000
750
500
250
0
Matrix Multiplication
Max-Pooling
Convolution
Total
Operations


Clock Cycle Reduction with PIM Logic
Without PIM
With PIM
2000
1500
1000
500
0
Clock Cycles
Matrix Multiplication
Max-Pooling
Convolution
Total
Operations

Efficiency Outlook with PIM:

- Latency Reduction:
  - Minimized data transfers between CPU and memory.
  - Projection: Similar to TensorFlow’s 27.4% execution time reduction.
- Energy Savings:
  - Lower power consumption by reducing memory channel usage.
  - Expected: Significant energy cuts in ARM-based systems.
- Performance Impact:
  - Faster CNN training on ARM with PIM.
  - Measurable via benchmarks (e.g., execution time, power usage).

Challenges:

- Design Constraints:
  - Limited area: 50-60 mm² in 3D-stacked logic layer.
  - Thermal issues: Heat dissipation in dense memory stacks.
- Programming Complexity:
  - Identifying tasks suitable for PIM offloading.
  - Ensuring compatibility with ARM architecture.
- System Integration:
  - Cache coherence: Maintaining data consistency.
  - Address translation: Managing memory access overhead.

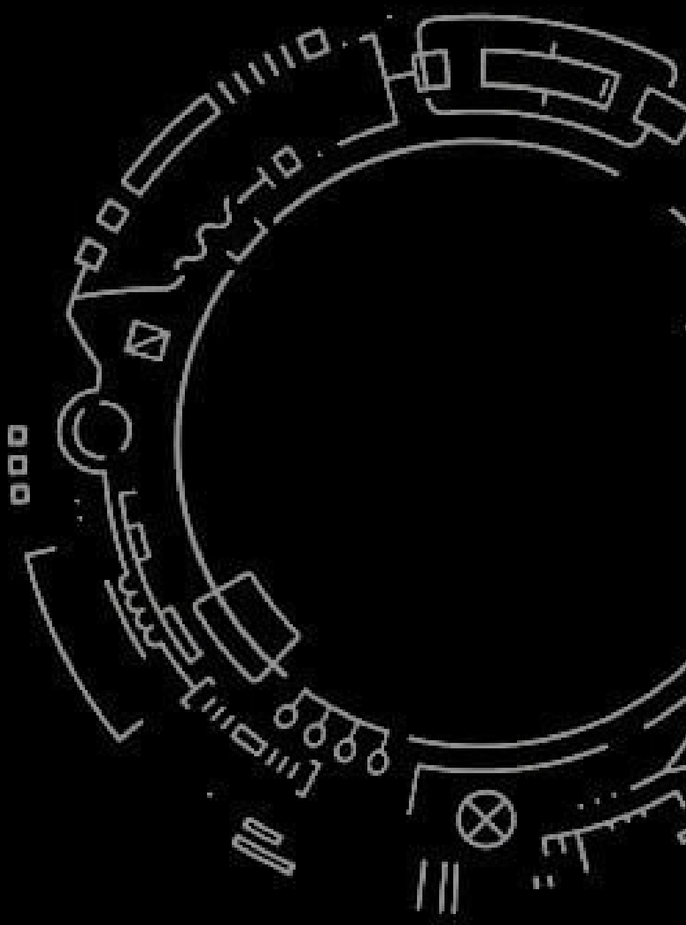
Overview:

Problem: Data Movement Bottleneck

Solution: PIM in ARM

Impact: Efficiency Gains

Scope: Embedded Systems, VLSI, ML



Conclusion:

- ✅ Summary:
  - PIM integration with ARM tackles data movement bottlenecks.
  - Improves CNN performance in embedded systems.
- 💡 Future Work:
  - Address challenges: area constraints, programming complexity.
  - Explore broader adoption in real-world applications.
- ⭐ Significance:
  - Enables efficient, low-power computing for next-gen AI/ML tasks.