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

TEAM:

ManojKumar. A
Mohammed kareemullah. PS
Ragul .R
3RD YEAR ECE, SVCE

MENTOR:

Mrs.B.Sarala
Asst. Professor ECE,
SVCE

Agenda:

- Introduction
- Problem statement
- Existing Approaches
- Proposed Solutions
- Literature Survey
- Methodologies / Technology used
- Architecture
- Results
- Future Scope
- Conclusion
- Milestones

Introduction :

In today’s rapidly evolving technological landscape, Artificial Intelligence and Machine Learning (AI/ML) have become the backbone of innovation across sectors like healthcare, autonomous systems, financial analytics, and smart infrastructure. These applications rely heavily on data-intensive computations, especially during the training phase of large models. However, traditional computing architectures based on the von Neumann model suffer from a critical bottleneck—the constant data movement between memory and processor.

Processing-in-Memory (PIM) offers a promising alternative by performing computations directly within the memory subsystem, minimizing data transfer and enhancing efficiency. When combined with the power-efficient and widely used ARM architecture, PIM integration becomes highly practical for real-world edge and embedded AI applications. ARM's flexible memory hierarchy and low-power design make it an excellent fit for implementing PIM-based accelerators. This research explores the integration of a PIM logic layer within an ARM processor to accelerate a Convolutional Neural Network (CNN) model.

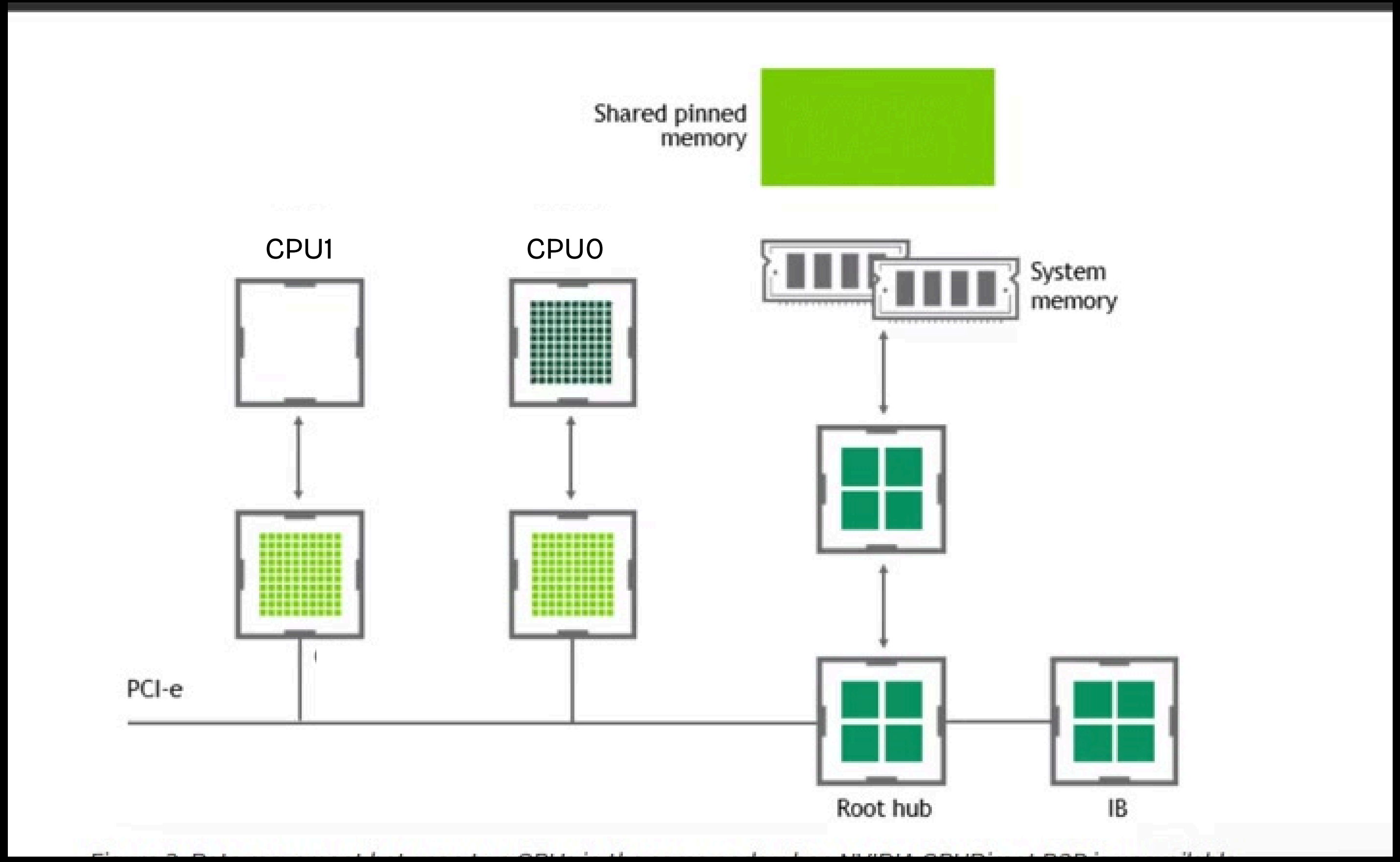
Problem Statement :

DATA MOVEMENT BOTTLENECK

In Traditional Architectures, especially during AI/ML workloads, the CPU (Processor Core) performs computations, while the DRAM (main memory) stores the data. However, moving data back and forth between these two components introduces a performance bottleneck.

Impact on Performance :

- Significant energy consumption due to frequent data transfers.
- Slows down execution for data-intensive applications.



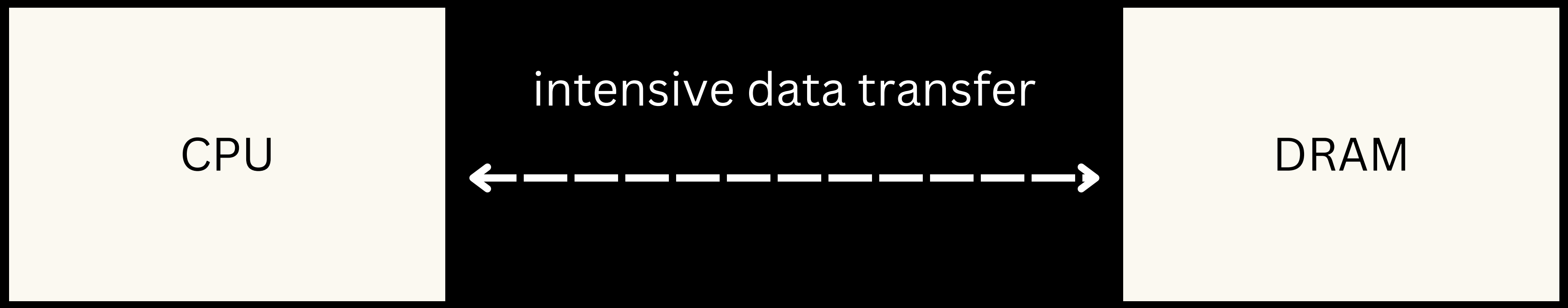
Existing Approaches:

- Caching:
  - Stores frequently accessed data closer to the CPU to reduce access time.
  - Limitation: Limited cache size leads to frequent data swaps, still requiring memory access.
- Prefetching:
  - Predicts and loads data into cache before it’s needed by the CPU.
  - Limitation: Inaccurate predictions waste energy and bandwidth, increasing overhead.
- Specialized Hardware (e.g., GPUs, TPUs):
  - Accelerates data-intensive tasks like CNN training with parallel processing.
  - Limitation: High power consumption, not optimized for energy-efficient systems like ARM.

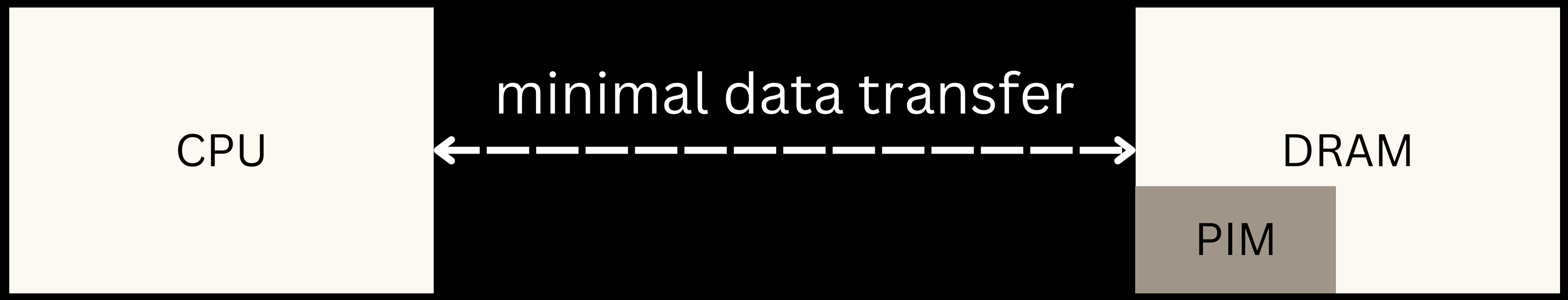
Proposed Solution :

Integrating a Dedicated Logic Layer with the memory Architecture of the ARM processor to avoid unwanted data movement between CPU and memory and hence reducing power consumptions and increasing efficiency. Instead of sending data to the CPU for computation, certain AI/ML operations are executed within or near the memory itself.

- ✅ Reduced Data Movement: Minimizes frequent transfers between CPU and DRAM.
- ✅ Lower Power Consumption: Less data movement = reduced energy overhead.
- ✅ Increased Efficiency: AI/ML tasks like matrix multiplication and convolution are handled in-place.



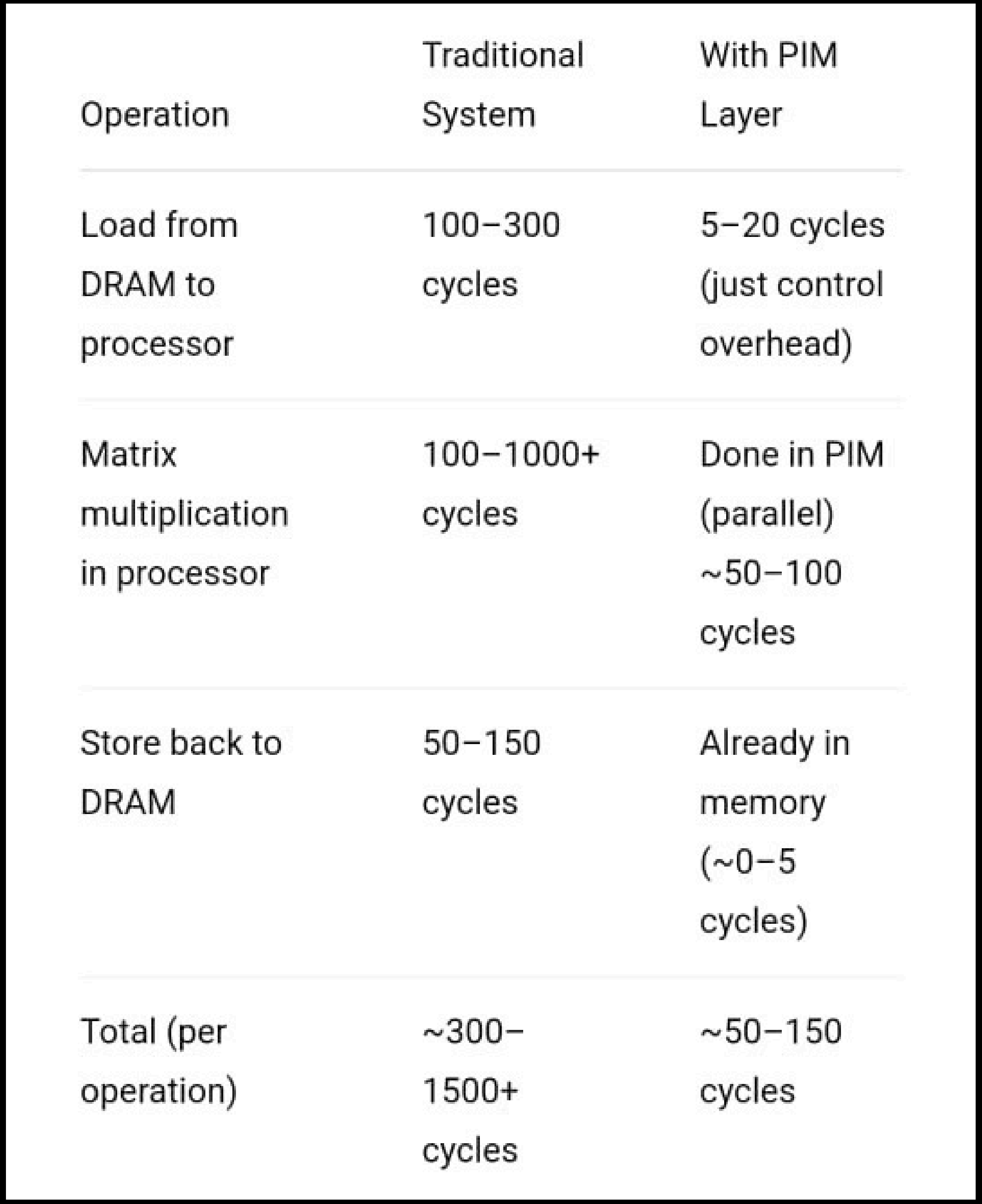
Without PIM



With PIM

Proposed Solution :

Comparison of Data movement in computer architecture with and without PIM:



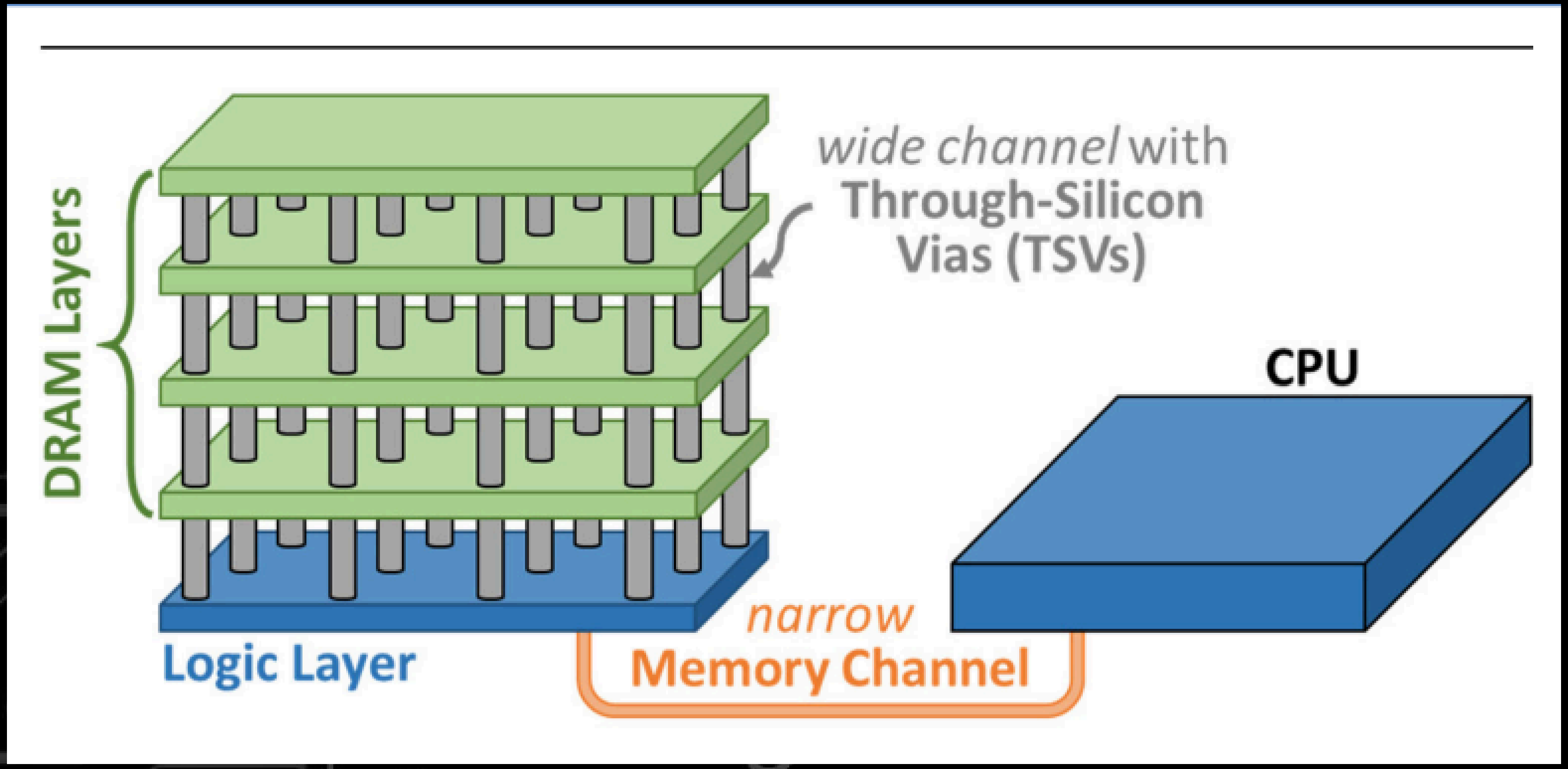
Operation	Traditional System	With PIM Layer
Load from DRAM to processor	100–300 cycles	5–20 cycles (just control overhead)
Matrix multiplication in processor	100–1000+ cycles	Done in PIM (parallel) ~50–100 cycles
Store back to DRAM	50–150 cycles	Already in memory (~0–5 cycles)
Total (per operation)	~300–1500+ cycles	~50–150 cycles

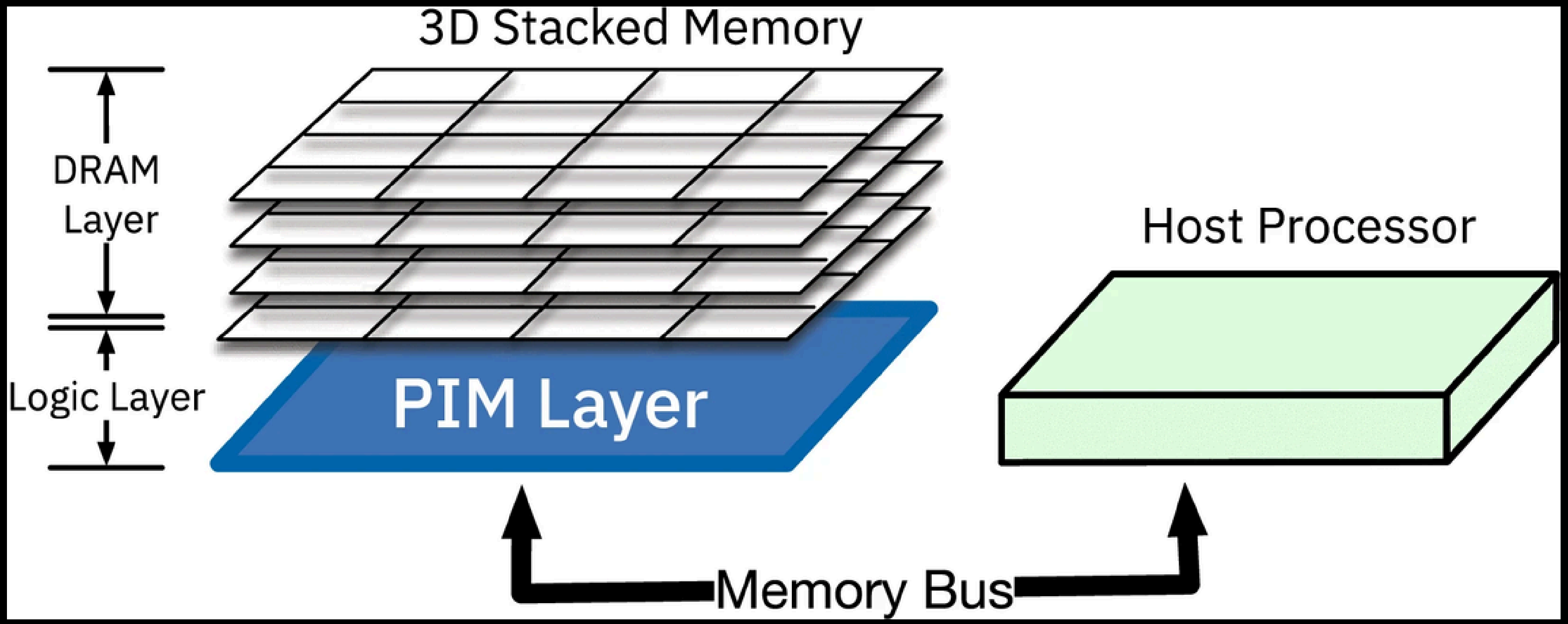
Literature Survey :

- PIM: A Workload-Driven Perspective (2019)
  - Explores PIM for various workloads (e.g., AI, big data).
  - Highlights challenges: data mapping, logic integration in memory.
  - [Source: Base paper, likely "Processing-in-Memory: A Workload-Driven Perspective"]
- Active Memory Cube for Exascale Systems (2015)
  - Proposes PIM architecture for exascale computing.
  - Demonstrates energy efficiency and scalability benefits.
  - [Source: Nair et al., IBM Journal of R&D]
- PIM for Neural Networks (2021)
  - Focuses on PIM for accelerating neural network training.
  - Shows significant speedup and energy savings for CNNs.
  - [Source: Recent studies on PIM in AI]
- Relevance to Our Work
  - Builds on PIM's proven benefits for data-intensive tasks.
  - Extends research by integrating PIM with ARM for energy-efficient systems.

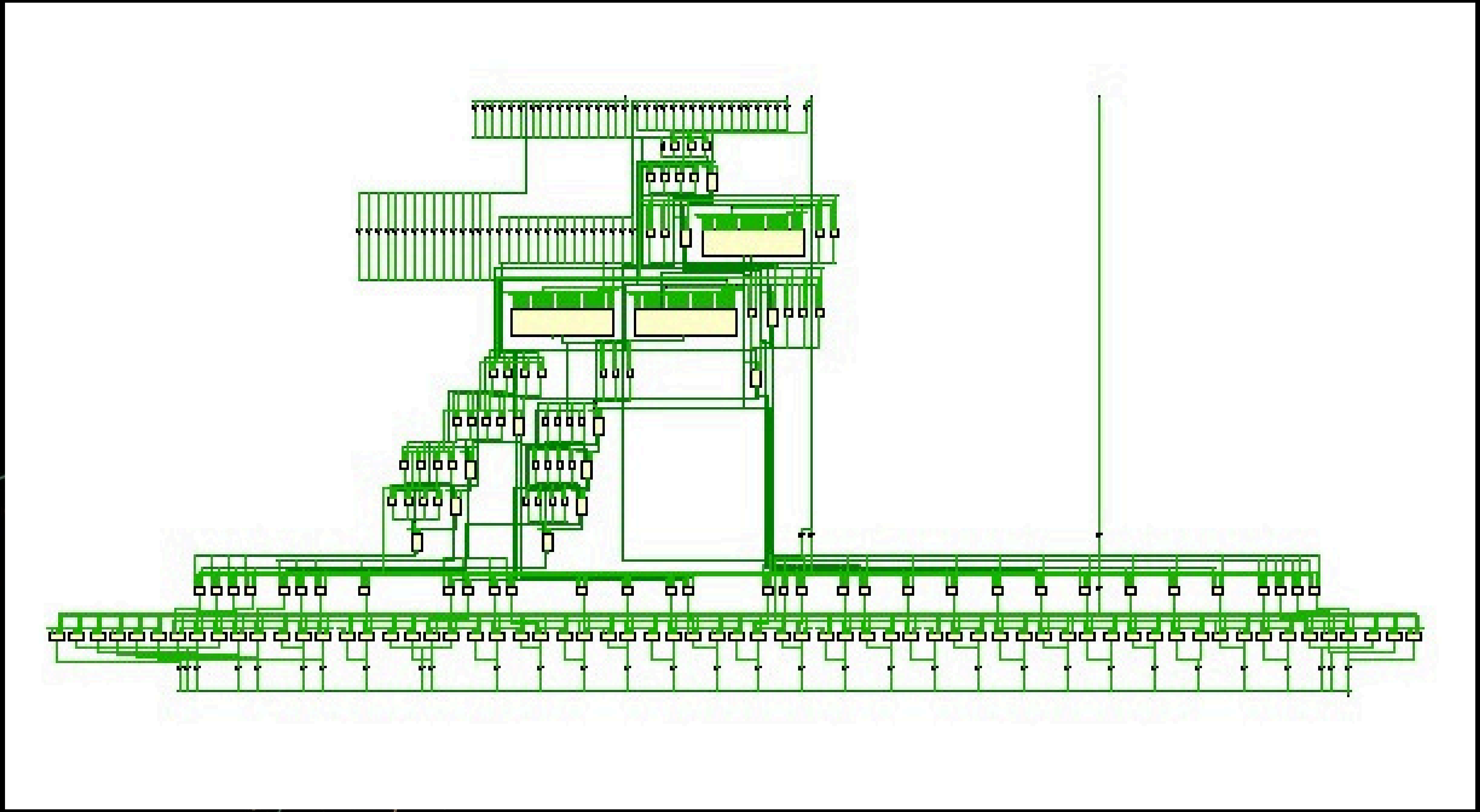
Methodologies / Technologies used :

- PIM Logic Layer: generated on Verilog with the RTL Schematic to be integrated with the 3D Stacked Memory. (Vivado and Xilinx)
- Simulation-Based Evaluation: Used architecture simulators to model PIM integration with ARM. (Gem5 Simulator)
- Performance Metrics: Number of Clock cycles for Matrix Multiplication, Maxpooling, and Convolution.





Architecture :

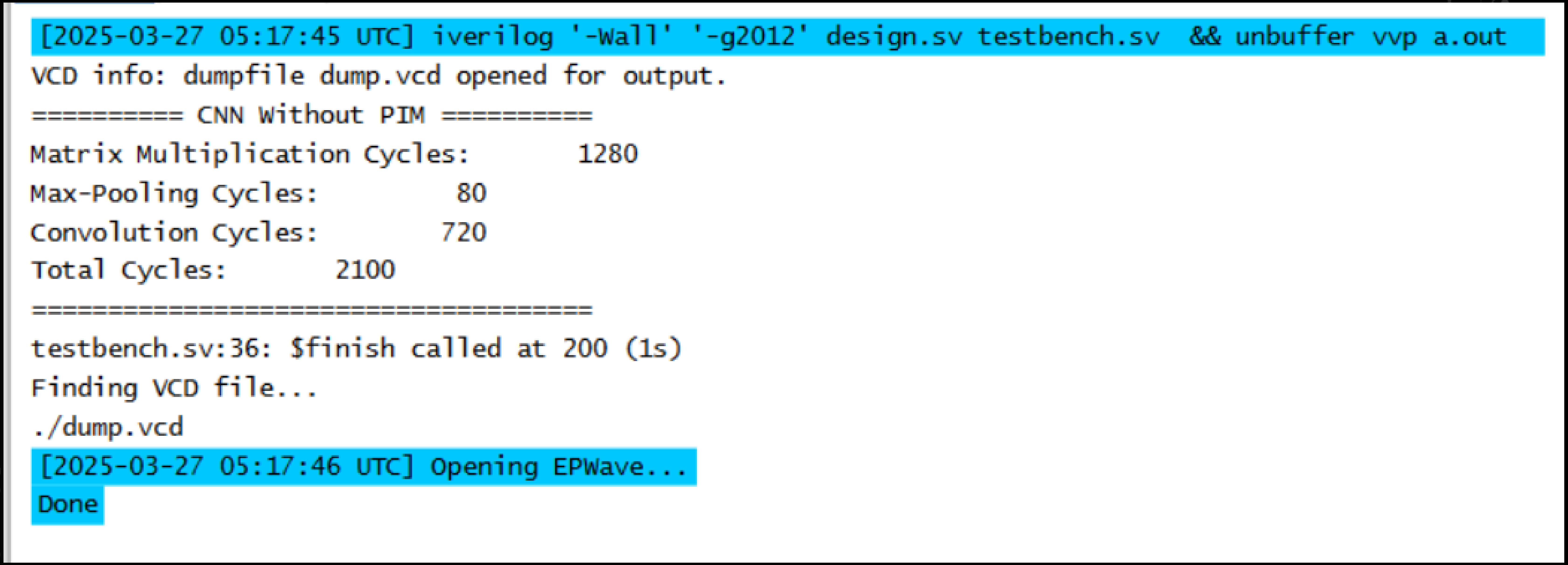


Operations and Opcode:

- Matrix Multiplication (011)
- Convolution (101)
- Max Pooling (100)
- Addition (000)
- Subtraction (001)
- Multiplication (010)

RESULT :

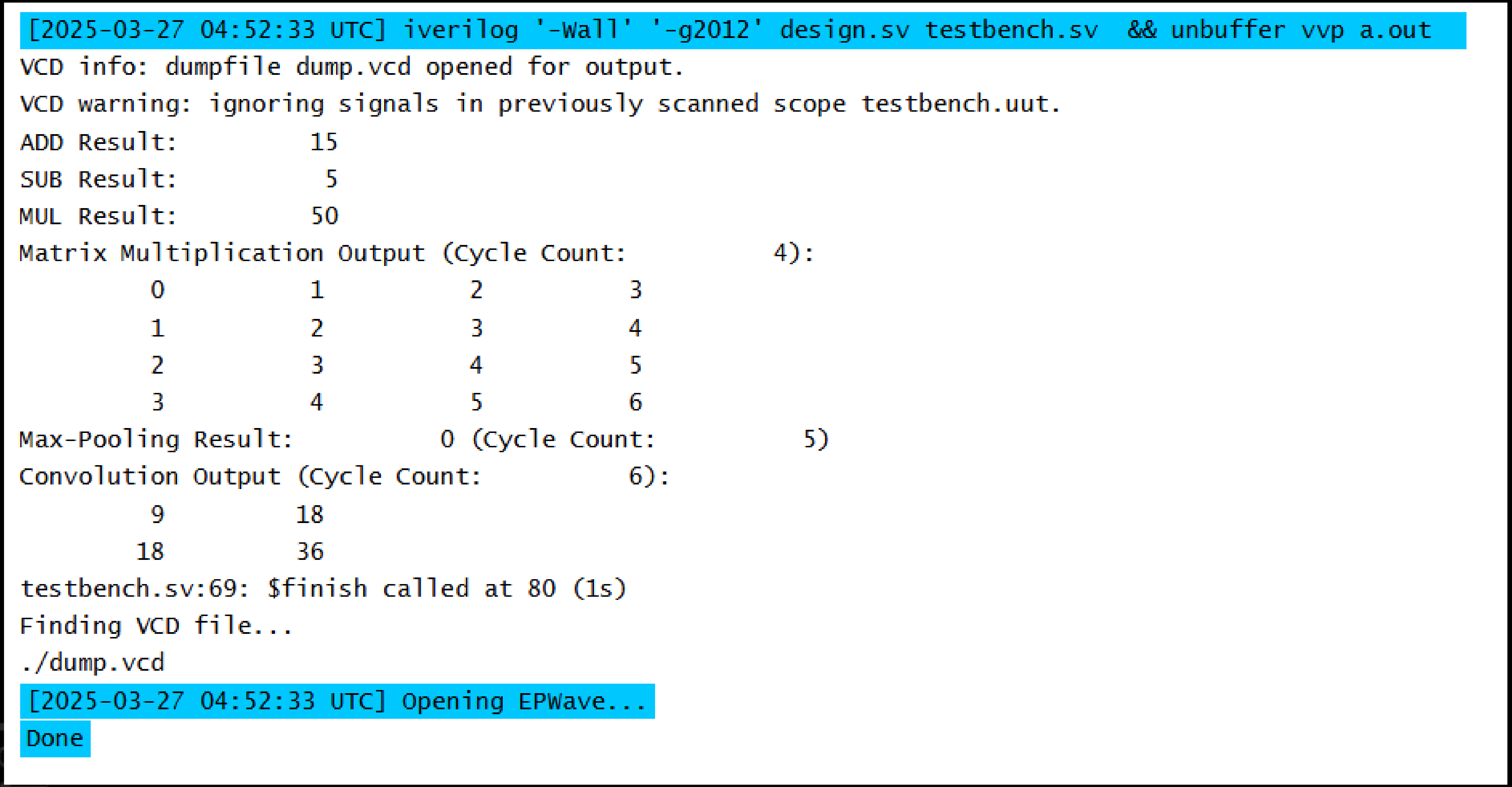
ARM Processor without PIM :



```
[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
```

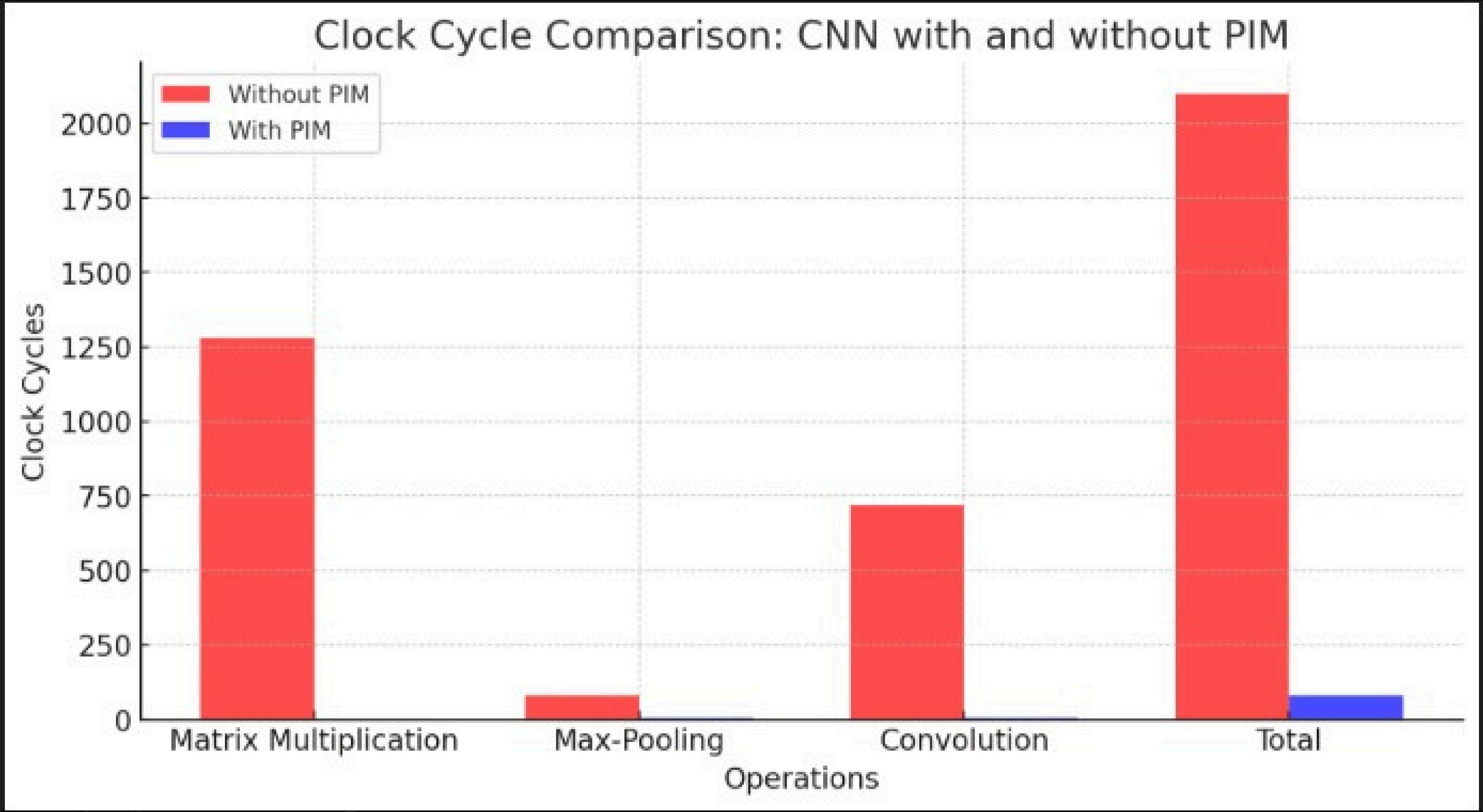
RESULT :

ARM Processor 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
```

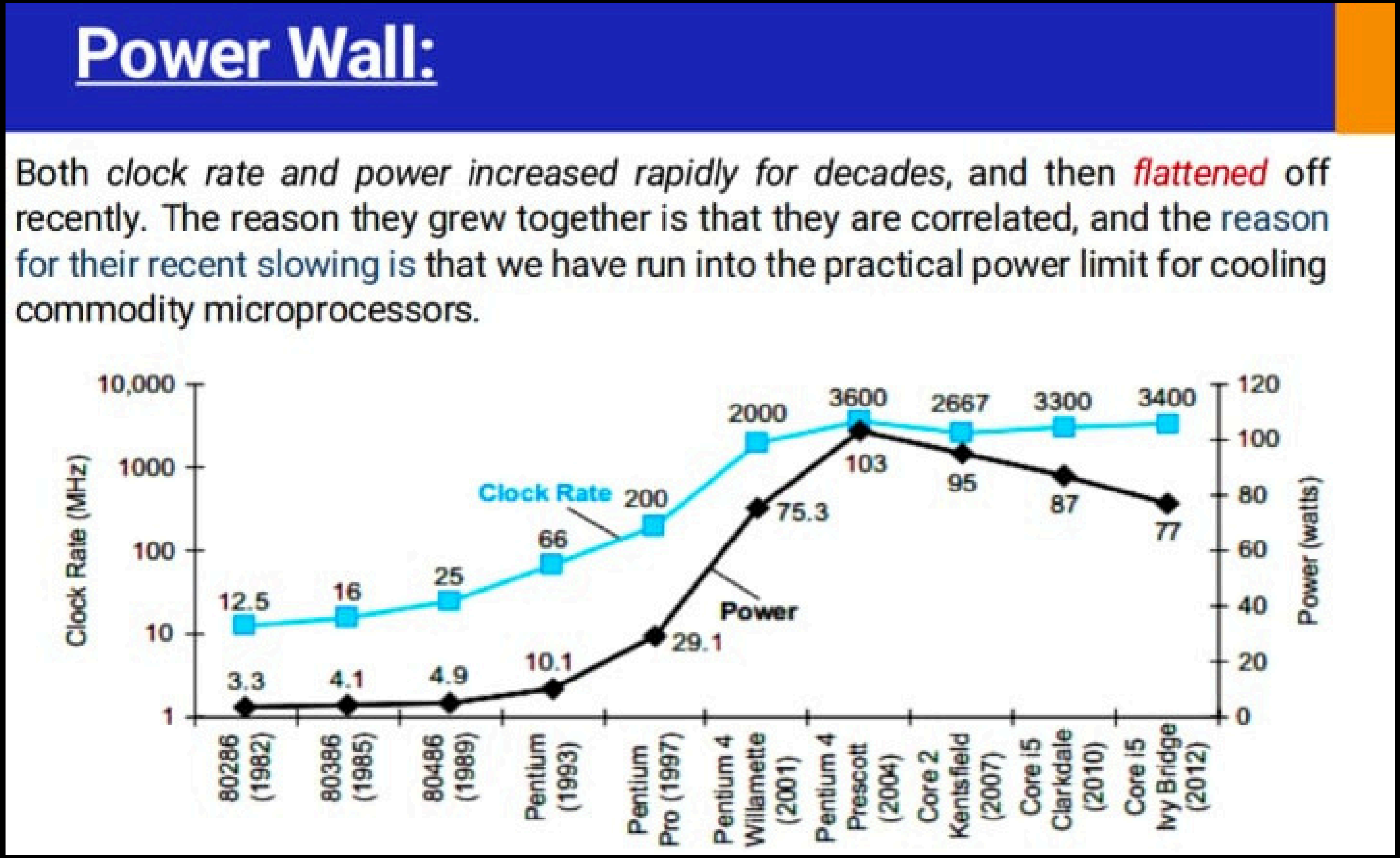
RESULT :


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

FUTURE SCOPE:

Traditional CPU architectures have long relied on increasing clock speeds to improve performance. However, as clock rates increased, power consumption and heat dissipation also surged, leading to the Power Wall problem

Overcoming the Power Wall: By minimizing memory-CPU data transfers, PIM can break the power efficiency stagnation.



Power Wall:

Both *clock rate and power increased rapidly for decades*, and then *flattened* off recently. The reason they grew together is that they are correlated, and the reason for their recent slowing is that we have run into the practical power limit for cooling commodity microprocessors.

Conclusion :

- Key Findings
  - Successfully integrated Processing-in-Memory (PIM) with ARM architecture.
  - Demonstrated significant improvements in latency and energy efficiency.
- Impact on Data-Intensive Tasks
  - Enhanced performance for CNN training on large datasets.
  - Reduced data movement, addressing the memory bottleneck.
- Implications for Next-Gen Computing
  - Enables faster, more energy-efficient systems for AI and IoT applications.
  - Supports scalability for future computing demands (e.g., exascale systems

Milestones :

- Literature Review (Week 1-2): Conducted a comprehensive survey of PIM research.
- Simulation Setup (Week 3-4): Configured Gem5 simulator for ARMv8-A with PIM.
- Implementation & Testing (Week 5-6): Integrated PIM and evaluated CNN training performance.
- Analysis & Documentation (Week 7): Analyzed results and prepared presentation/paper.

Future Milestones

- Hardware Prototype : Develop a PIM-integrated ARM prototype.
- Extended Workloads : Test PIM on other tasks (e.g., graph analytics).