
Innocenzo Mungiello

Tutor: Alessandro Cilardo

XXXI Cycle - III year presentation

# *Improving Multibank Memory Access Parallelism on SIMT Architectures*


# Background

- Master of Science:
  - Cum laude in “Ingegneria Informatica” at University of Naples “Federico II”
- DIETI Group:
  - Seclab
- Type of Fellowship:
  - No Grant
- Collaboration:
  - With CeRICT in the context of the european project MANGO

# Training Activities

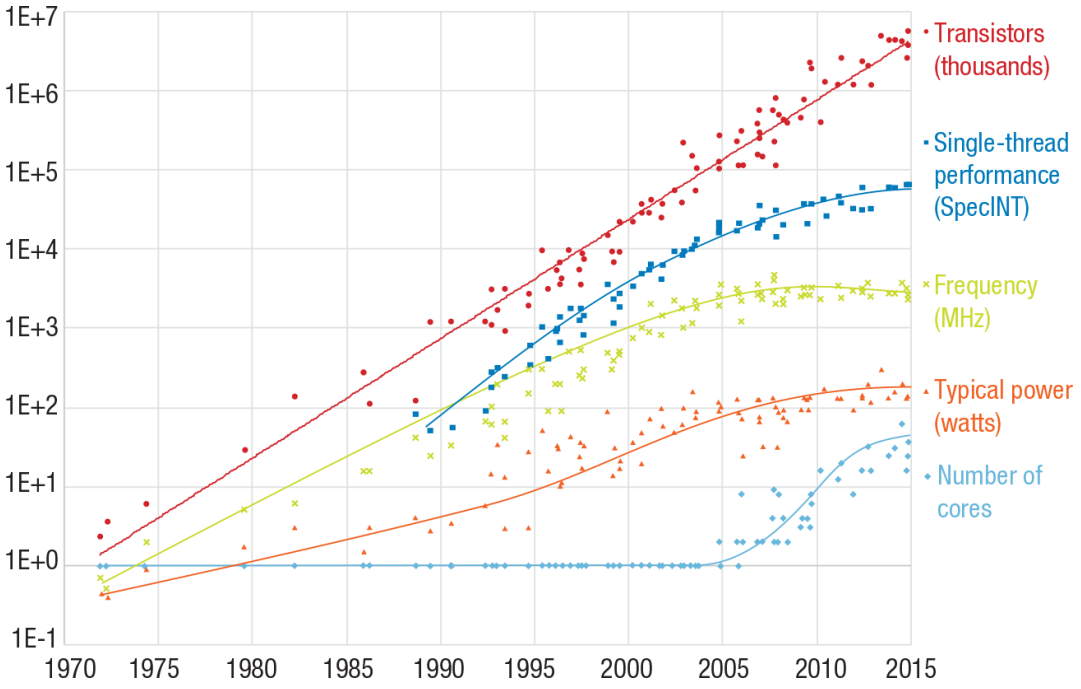
**Student: Innocenzo Mungiello** **Tutor: Alessandro Cilardo** **Cycle XXXI**

innocenzo.mungiello@unina.it acilardo@unina.it

| | Credits year 1 | | | | | | | | Credits year 2 | | | | | | | | Credits year 3 | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 | | | 1 | 2 | 3 | 4 | 5 | 6 | | | 1 | 2 | 3 | 4 | 5 | 6 | | | |
| | Estimated | bimonth | bimonth | bimonth | bimonth | bimonth | bimonth | Summary | Estimated | bimonth | bimonth | bimonth | bimonth | bimonth | bimonth | Summary | Estimated | bimonth | bimonth | bimonth | bimonth | bimonth | bimonth | Summary | Total | Check |
| **Modules** | 20 | 0 | 6 | 3 | 5 | 0 | 9 | **23** | 10 | 9 | 3 | 0 | 0 | 0 | 6 | **18** | | 0 | 2 | 0 | 0 | 0 | 0 | **2** | **43** | **30-70** |
| **Seminars** | 5 | 1,8 | 0,4 | 0,7 | 1 | 0 | 1,3 | **5,2** | 5 | 2,9 | 1,3 | 1,4 | 0 | 0 | 0 | **5,6** | | 0 | 0 | 0 | 0 | 0 | 0 | **0** | **11** | **10-30** |
| **Research** | 35 | 7 | 4 | 6 | 4 | 7 | 4 | **32** | 45 | 1 | 5 | 7 | 10 | 10 | 3 | **36** | 60 | 10 | 8 | 10 | 10 | 10 | 10 | **58** | **126** | **80-140** |
| | 60 | 8,8 | 10,4 | 9,7 | 10,0 | 7,0 | 14,3 | **60,2** | 60 | 13 | 9,3 | 8,4 | 10 | 10 | 9 | **60** | 60 | 10 | 10 | 10 | 10 | 10 | 10 | **60** | **180** | **180** |

# The End of Dennard Scaling

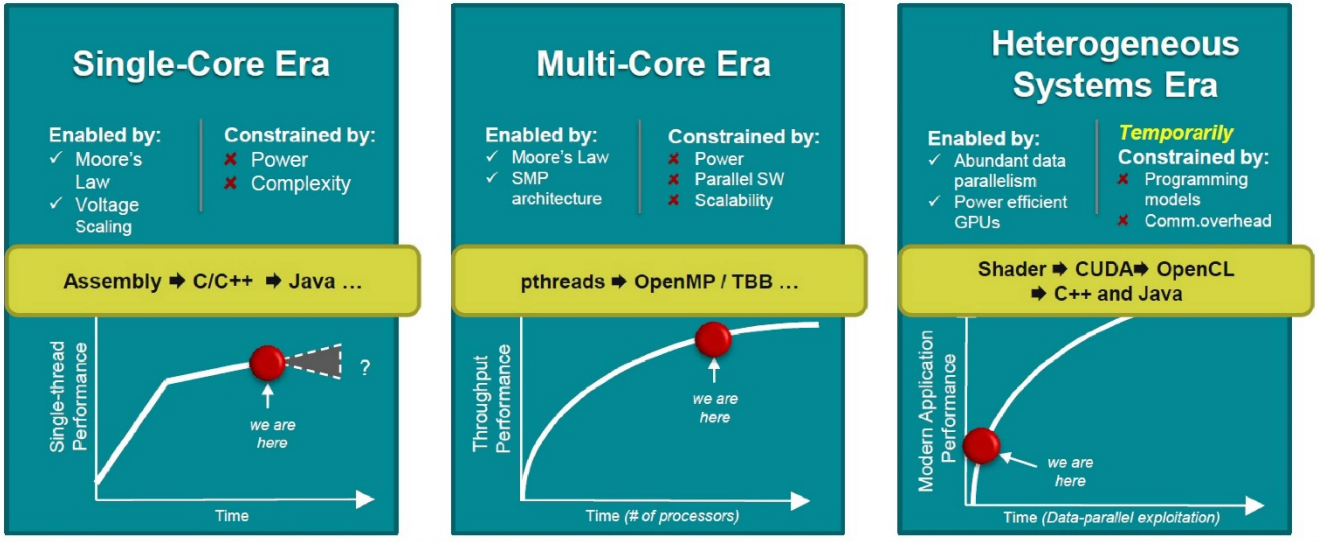
- Processors aren't getting faster, just wider
  - Future gains in performance are from parallelism
- Process matters less
  - One generation is 1.2x, not 2.8x
  - *Dark silicon* problem
- Future systems are energy limited
  - Efficiency ***IS*** Performance



# Heterogeneous Computing

- Today in a system we have:
  - manycore CPU
  - GPU
  - MIC
  - FPGA

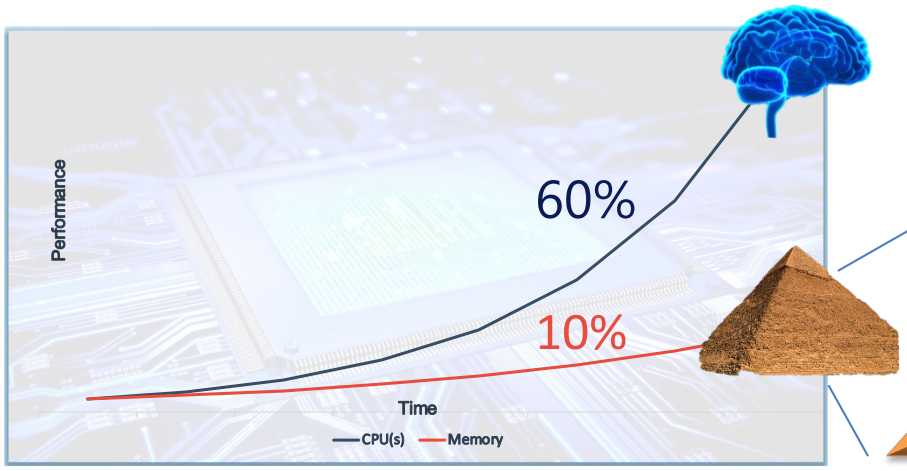
- The goal of the parallel revolution:
  - programs for parallel easy as programs for sequential

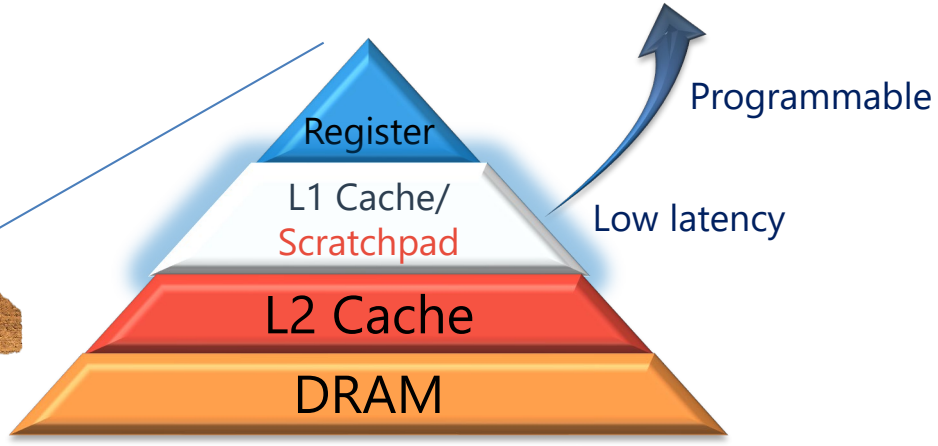


# Problem: The Unbroken Memory Wall

- Memory wall remains a fundamental limit to system performance
- Also in terms of performance per watt
  - Only the 15% of energy consumption is used for useful computation

- Convolution
- Transpose
- LU decomposition
- ...and more

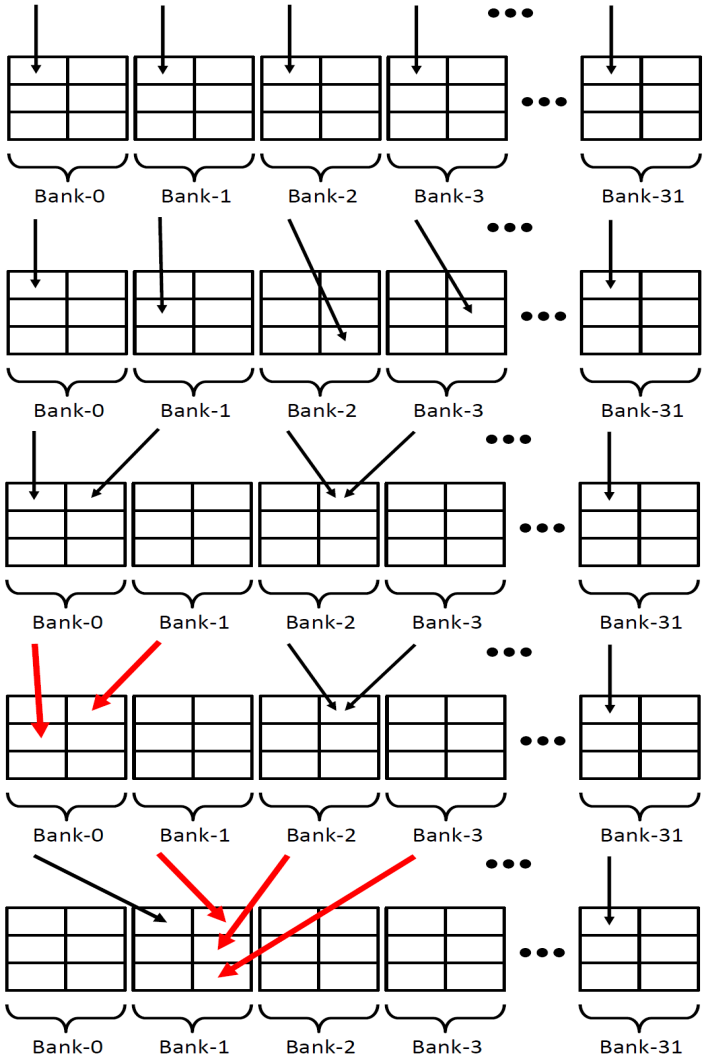



All memory levels must be Optimized!

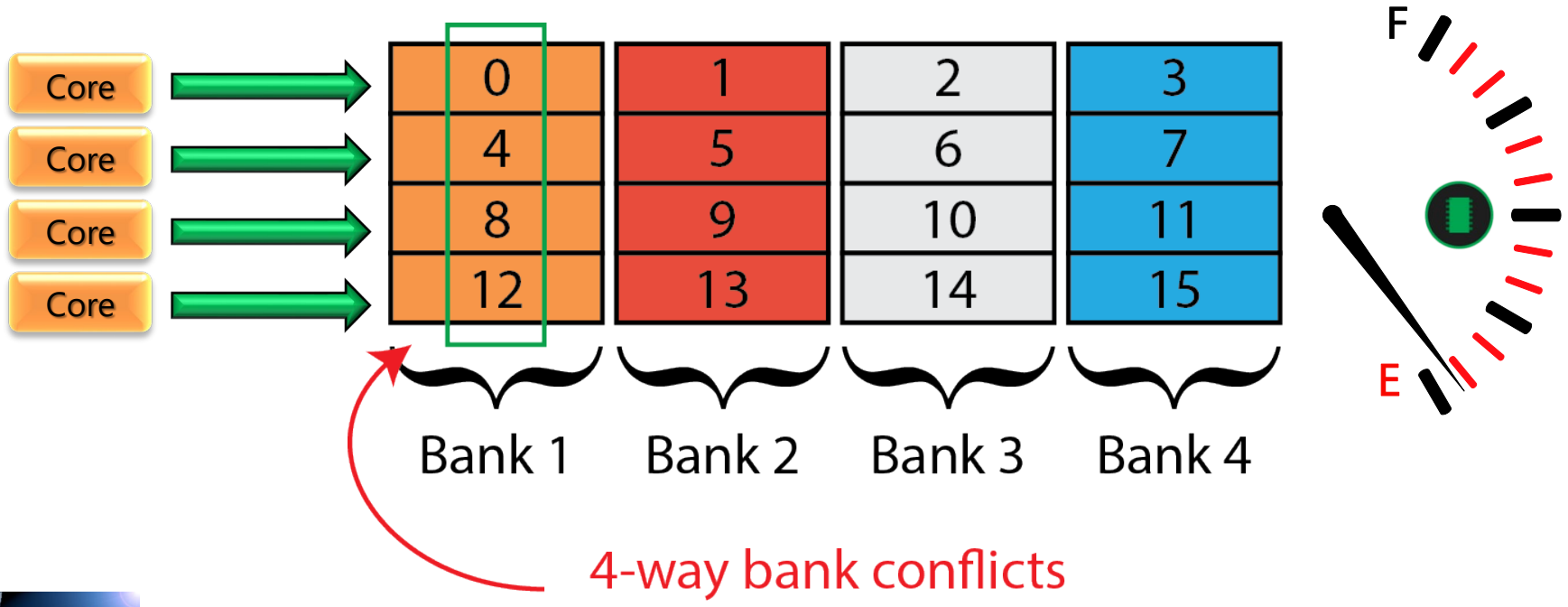
# A sub-problem: Bank Conflicts

- **A bank conflict occurs when:**
  - two or more threads *in a warp* access different words in the same bank
    - Think: two or more threads access different "rows" in the same bank
  - N-way bank conflict: N threads in a warp conflict
  - Instruction gets issued N times: increases latency
- **There is no bank conflict if:**
  - Several threads access the same word
  - Several threads access different bytes of the same word



# A sub-problem: Conflicts

- Shared memory performance may suffer bank conflicts
- Waste of memory ↔ conflict resolution Trade off
- Shared memory can become a limiting factor



# Related Works (1/2)

- Several techniques have been presented to solve bank conflicts and reduce memory access latency:
  - **NVIDIA Padding**: wastes shared memory to solve conflicts
  - **Mrugesh Gajjar and Ismayil Guracar**[1] do not modify the data layout of input and output arrays but they exploit the computational structure of the convolution filtering operation to modify the algorithm and avoid bank conflicts.
  - **Akihiko Kasagi, Koji Nakano, and Yasuaki Ito**[2] used a Graph-Coloring technique in order to implement a conflict-free permutation algorithm on these models. They must use extra data structures in the kernel code, in order to implement the technique.


1. Mrugesh Gajjar and Ismayil Guracar, Efficient rate conversion filtering on GPUs with shared memory access pattern scrambling, Signal Processing Systems (SiPS), 2016 IEEE International Workshop on, IEEE, 2016.
2. Akihiko Kasagi, Koji Nakano, and Yasuaki Ito, An implementation of conflict-free offline permutation on the GPU, Networking and Computing (ICNC), 2012 Third International Conference on, IEEE, 2012.

# Related Works (2/2)

- Several techniques have been presented to solve bank conflicts and reduce memory access latency:
  - **Dymaxion**[3] is an automatic tool that analyses GPU kernels code and provides a data layout transformation in order to improve memory coalescing accesses.
  - **Random Address Shift**[4] uses a vector of random numbers in order to perform a shift of the elements in shared memory and avoid conflicts. The technique must use extra scratch-pad memory in order to store data structures

- All these techniques:
  - **Waste** shared memory;
  - Are not focused on finding a **relationship** between the **memory access pattern**, the shared memory **bank conflict** and the **power consumption**.

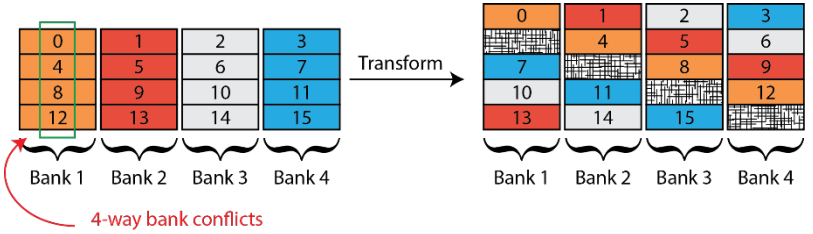
3. Shuai Che, Jeremy W Sheaer, and Kevin Skadron, Dymaxion: Optimizing memory access patterns for heterogeneous systems, Proceedings of 2011 international conference for high performance computing, networking, storage and analysis, ACM, 2011.
4. Koji Nakano, Susumu Matsumae, and Yasuaki Ito, The random address shift to reduce the memory access congestion on the discrete memory machine, Computing and Networking (CANDAR), 2013 First International Symposium on, IEEE, 2013.

# Idea: Use Matematical Models to solve Bank Conflicts

- Polyhedral Model Transformation
  - Useful in some conditions
  - can achieve a 30% increase in performance per watt[5]
  - Find the right ***T*** matrix is an hardest work
  - Wastes shared memory

$$T = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

$$Banks(x, y) = [1 \quad N] \cdot \begin{bmatrix} a & b \\ c & d \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} mod\ banks$$




- Linear Programming Model
  - Useful in any condition
  - Does not waste shared memory
  - Too many solutions

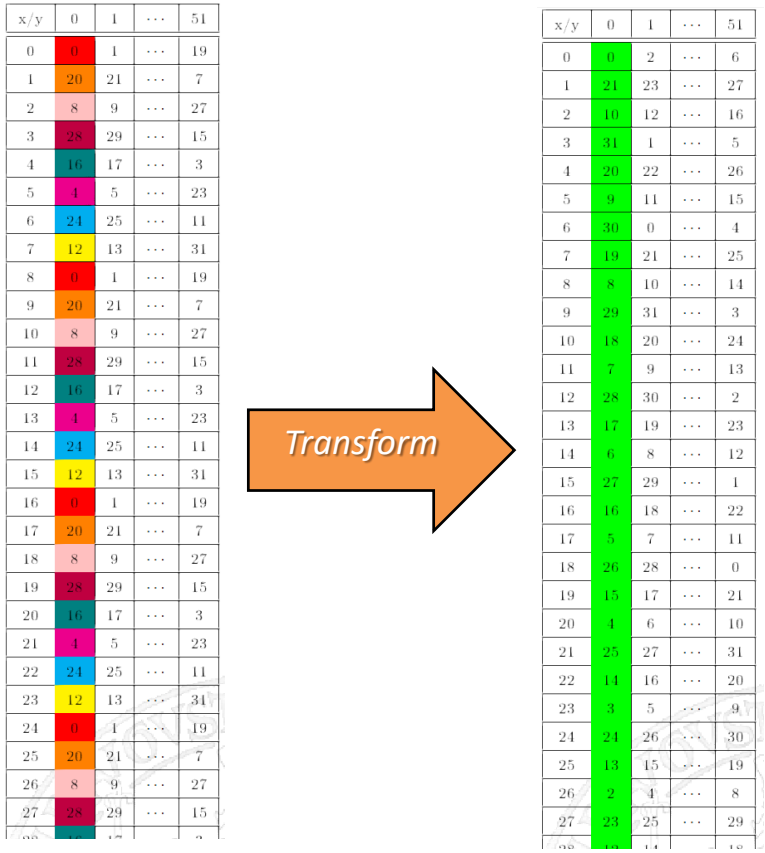
5. A. Cilardo, I. Mungiello, "Experimental evaluation of memory optimizations on an embedded GPU platform", *10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing* (3PGCIC), 2015

# Polyhedral Model Transformation

- **Matrix Multiplication**
  - In a warp of 32x1 threads: 4-way conflict
  - Find a transformation of the allocation function that solves this conflict
  - Use the formal methodology proposed by A. Darte[6]

$$\boldsymbol{T} = \begin{bmatrix} \mathbf{2} & \mathbf{1} \\ \mathbf{0} & \mathbf{1} \end{bmatrix}$$

$$\boldsymbol{Banks}(\boldsymbol{x},\boldsymbol{y}) = [\mathbf{1}\ \boldsymbol{N}] \cdot \begin{bmatrix} \mathbf{2} & \mathbf{1} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix} \boldsymbol{mod} \begin{bmatrix} \mathbf{1} \\ \mathbf{32} \end{bmatrix}$$




6. Darte, Alain, Michele Dion, and Yves Robert. "A characterization of one-to-one modular mappings." *Parallel Processing Letters 6.01 (1996): 145-157.*

# Polyhedral Model Transformation: Implementation

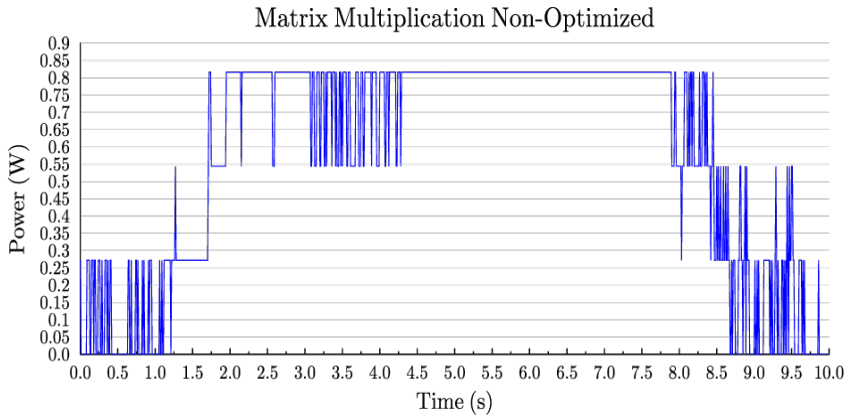
- **Using NVIDIA Jetson TK1 for**
  - Running two Kernels:
    - One Non-Optimized
    - One Optimized

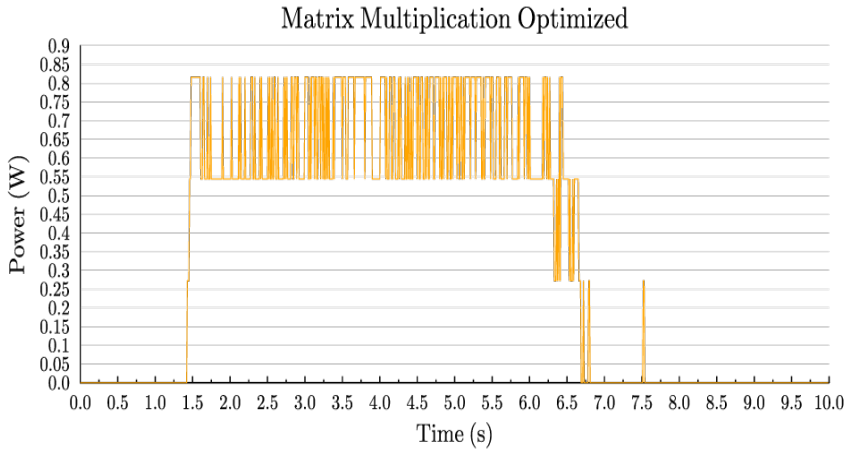
- **With the Digilent Analog Discovery,**
  - measure the power consumption through Resistor ***R5C11***

# Polyhedral Model Transformation: Results

- **Matrix Multiplication, Non-Optimized**
  - Duration: **6,5** s
  - Energy consumption: **5,2** J
  - **10,816** MFLOPS/watt

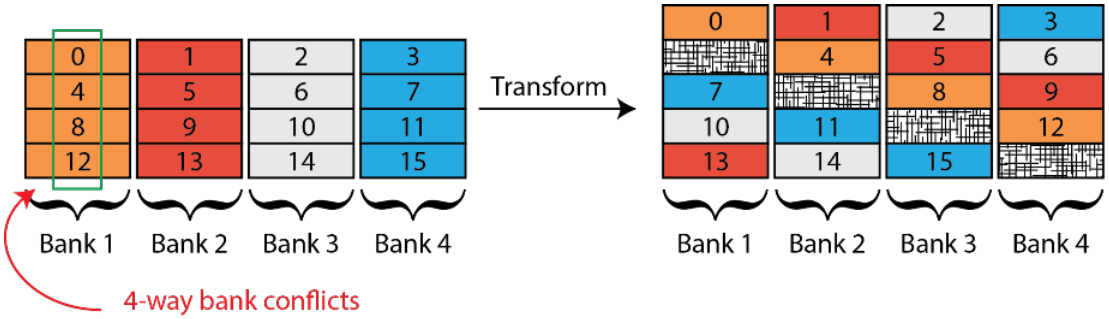
- **Matrix Multiplication, Optimized**
  - Duration: **5** s
  - Energy consumption: **4** J
  - **14,0608** MFLOPS/watt, **30%** more
  - Use **3,73%** more memory for the same kernel
  - Perform **3,84%** more instructions for the same kernel

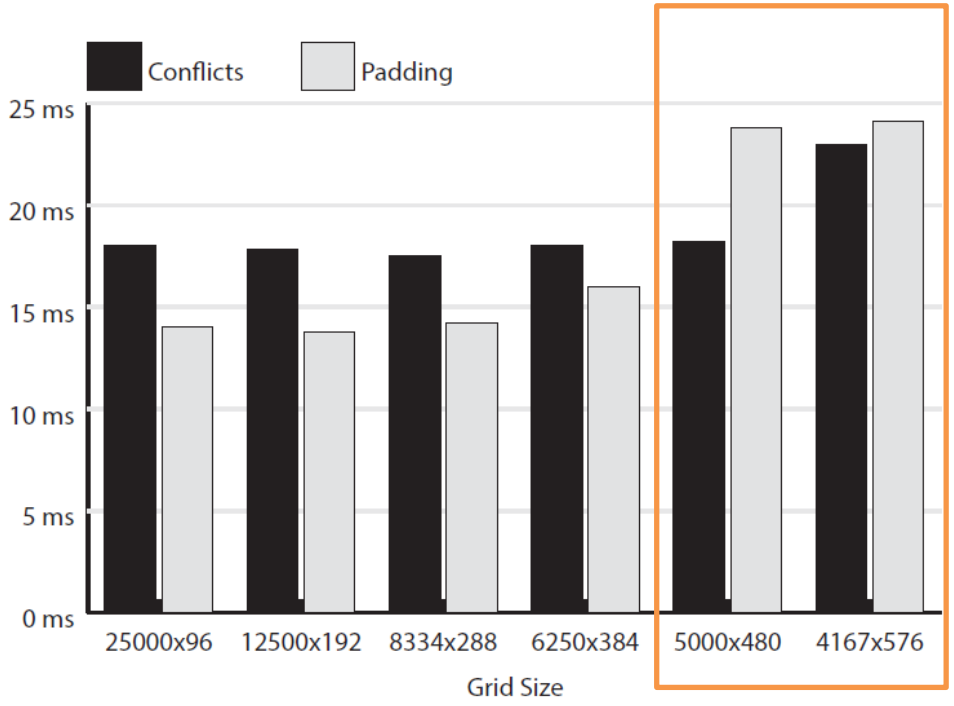


# Polyhedral Model Transformation: Trade off

- Shared memory performance may suffer bank conflicts
- Waste of memory / conflict resolution Trade off
- Shared memory can become a limiting factor



# Idea: Use Matematical Models to solve Bank Conflicts

- Polyhedral Model Transformation
  - Useful in some conditions
  - can achieve a 30% increase in performance per watt[5]
  - Find the right ***T*** matrix is an hardest work
  - Wastes shared memory
- Linear Programming Model[7]
  - Useful in any condition
  - Does not waste shared memory
  - Too many solutions

$$\text{minimize} \left( \sum_{t=1}^{N_{TH}} \sum_{b=1}^{N_{BK}} \sum_{i=1}^{N_{IT}} x_{t,b,i} \right)$$

subject to :

$$\sum_{i=1}^{N_{IT}} x_{t,b,i} = 1 \quad \forall\, t \in [1, N_{TH}] \quad , \quad \forall\, b \in [1, N_{BK}]$$

$$\sum_{b=1}^{N_{BK}} x_{t,b,i} = 1 \quad \forall\, t \in [1, N_{TH}] \quad , \quad \forall\, i \in [1, N_{IT}]$$

$$\sum_{t=1}^{N_{TH}} x_{t,b,i} = 1 \quad \forall\, b \in [1, N_{BK}] \quad , \quad \forall\, i \in [1, N_{IT}]$$

$$x_{t,1,i} = 1 \qquad \forall t \in [1, N_{TH}]\ \forall i \in [1, N_{IT}]\ \wedge\ t = i$$

$$x_{t,b,i} \quad \text{is binary}$$

7. A. Cilardo and I. Mungiello, "Zero-conflict Memory Mapping for Transpose-like Kernels in SIMT Architectures", *Journal of Parallel and Distributed Computing, 2018*, submitted.

# Linear Programming Model: Implementation

$$\text{minimize} \left( \sum_{t=1}^{N_{TH}} \sum_{b=1}^{N_{BK}} \sum_{i=1}^{N_{IT}} x_{t,b,i} \right)$$

subject to :

$$\sum_{i=1}^{N_{IT}} x_{t,b,i} = 1 \quad \forall\, t \in [1, N_{TH}] \quad , \quad \forall\, b \in [1, N_{BK}]$$

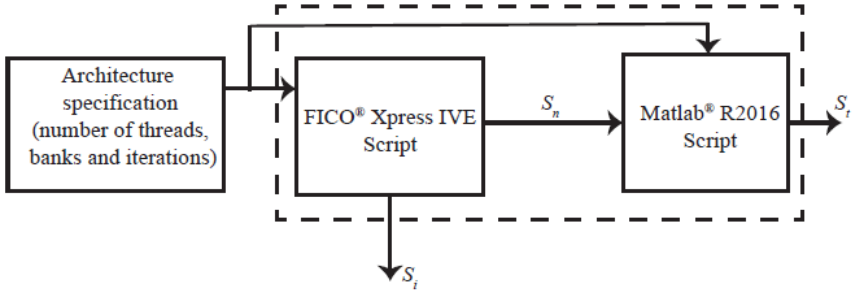
$$\sum_{b=1}^{N_{BK}} x_{t,b,i} = 1 \quad \forall\, t \in [1, N_{TH}] \quad , \quad \forall\, i \in [1, N_{IT}]$$

$$\sum_{t=1}^{N_{TH}} x_{t,b,i} = 1 \quad \forall\, b \in [1, N_{BK}] \quad , \quad \forall\, i \in [1, N_{IT}]$$

$$x_{t,1,i} = 1 \qquad \forall t \in [1, N_{TH}]\ \forall i \in [1, N_{IT}]\ \wedge\ t = i$$

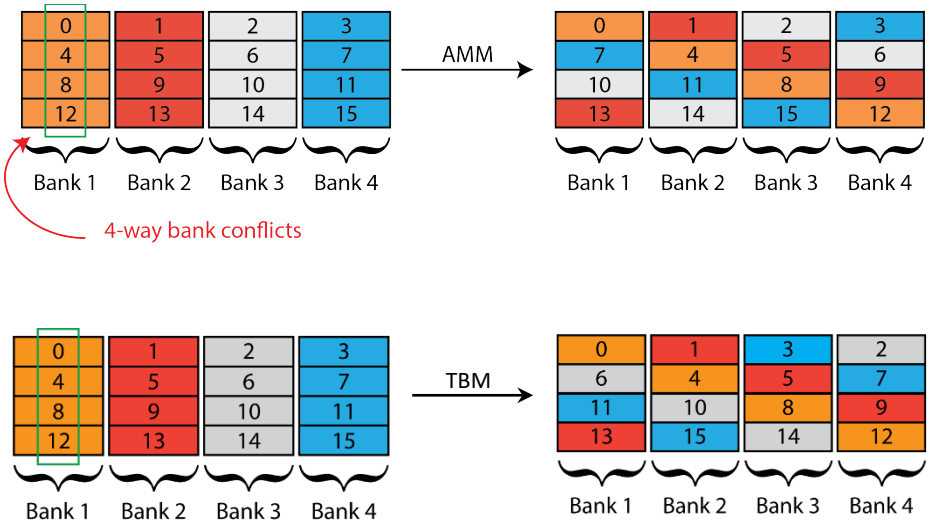
$x_{t,b,i}$ is binary

- **ILP Model**
  - Generalizes on existing conflict-avoiding techniques
  - Does not waste shared memory
  - supporting a systematic exploration of feasible mapping schemes
  - Not all solutions are useful for a SIMT architecture
- **Filtering Tool-chain**
  - check on the solutions generated by the ILP model, filtering SIMT feasible solutions



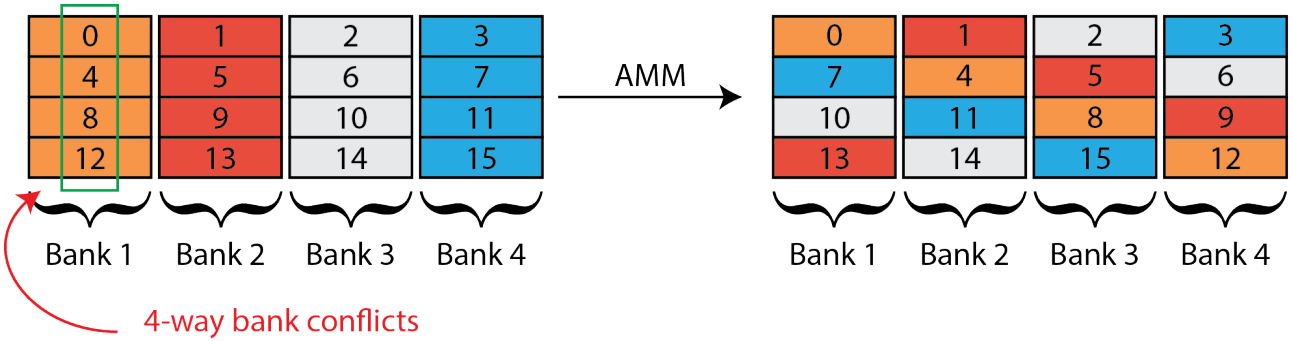
# Linear Programming Model: Results

- Discovered some useful mapping schemes:
  - Adaptive Modular Mapping[8]
  - Triangular Based Mapping
  - Inverse Adaptive Modular Mapping
  - Inverse Triangular Based Mapping
  - And more…




8. A. Cilardo, I. Mungiello and F. De Rosa, "Adaptive Modular Mapping to Reduce Shared Memory Bank Conflicts on GPUs", *11th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing* (3PGCIC), 2016

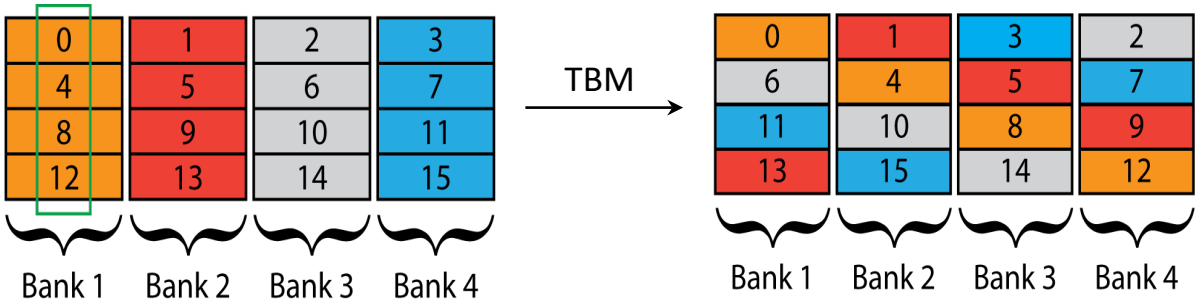
# Linear Programming Model: Adaptive Modular Mapping



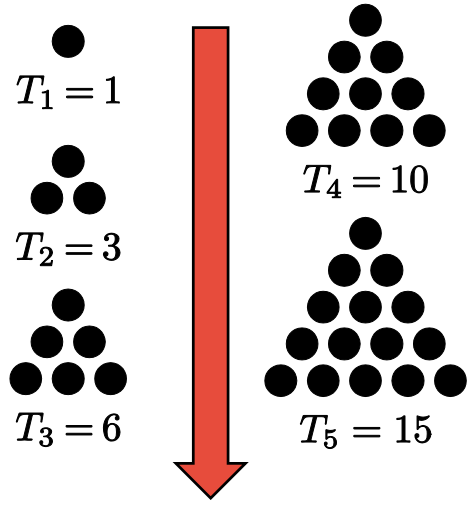

```
matrix_f[index][col] = p;
```

```
matrix_f[index][(index+col)%WIDTH)] = p;
```

# Linear Programming Model: Triangular Based Mapping




$$T(k) = k + T(k-1)$$

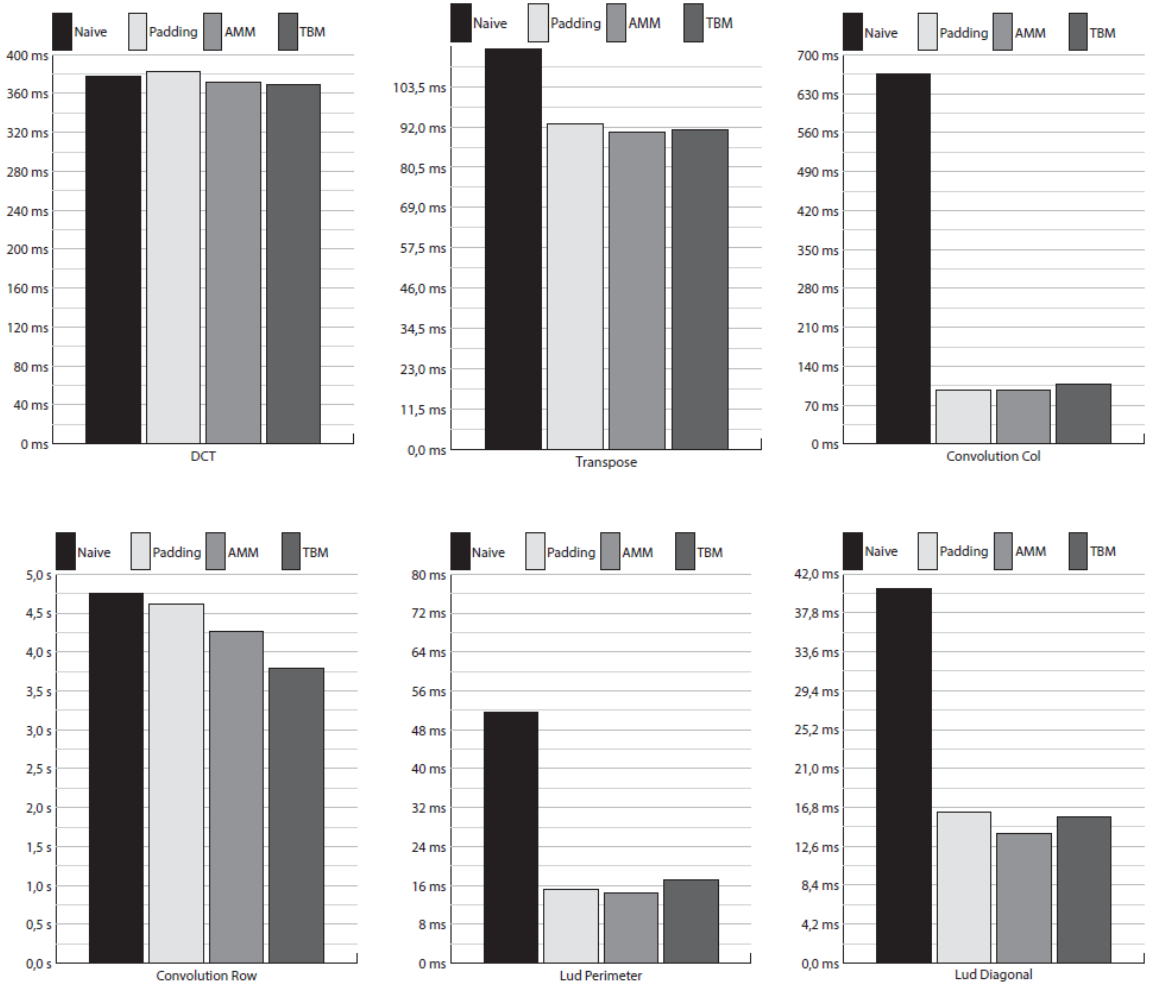



```
matrix_f[index][col] = p;
```

$$matrix[index][K] \Longrightarrow \left(index + \sum_{i=1}^{K} i\right) Mod\, W$$

```
matrix_f[index][(index+(col*(col+1)/2))%WIDTH] = p;
```

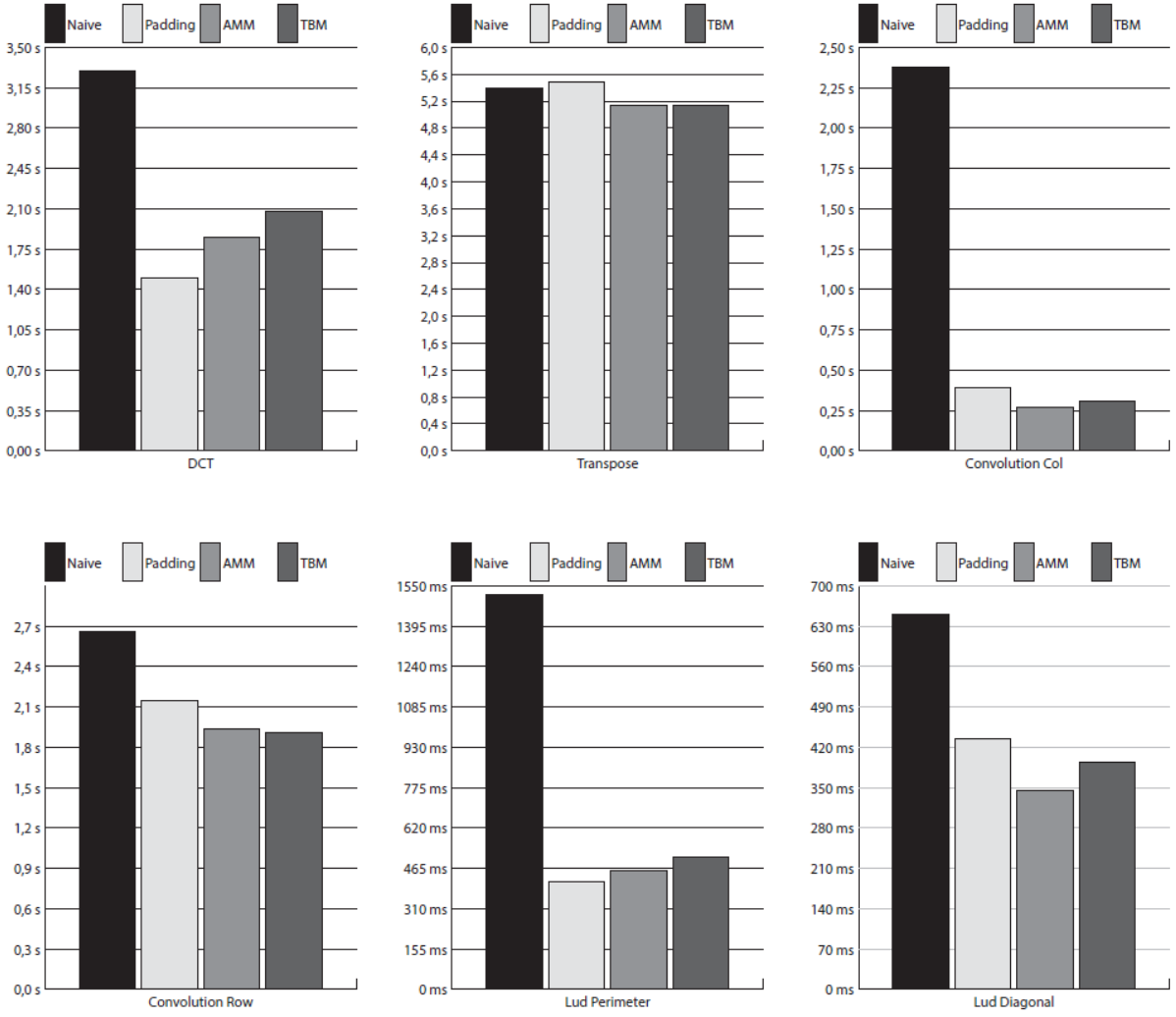
# Linear Programming Model: Experimental Results (TK1)



# Linear Programming Model: Experimental Results (TK1)

| Parameters | Mapping | DCT | Transpose | Convolution Col | Convolution Row | Lud Perime-ter | Lud Diago-nal |
|---|---|---|---|---|---|---|---|
| Dataset Size (MB) | All | 664.06 | 256 | 256 | 256 | 1024 | 1024 |
| CUDA Block Size | All | 128 | 256 | 128 | 64 | 64 | 64 |
| Resident CUDA Blocks | Native | 12 | 3 | 6 | 6 | 4 | 3 |
| | Padding | **11** | **2** | **5** | **5** | **3** | **2** |
| | AMM | 12 | 3 | 6 | 6 | 4 | 3 |
| | TBM | 12 | 3 | 6 | 6 | 4 | 3 |
| Registers | Native | 19 | 23 | 33 | 32 | 36 | 34 |
| | Padding | 19 | 23 | 33 | 32 | 36 | 34 |
| | AMM | 32 | 30 | 33 | 41 | 36 | 35 |
| | TBM | 32 | 28 | 41 | 49 | 36 | 35 |
| Shared Memory (KB) | Native | 4 | 16 | 8 | 8 | 12 | 16 |
| | Padding | 4.125 | 16.25 | 8.125 | 8.125 | 12.125 | 16.25 |
| | AMM | 4 | 16 | 8 | 8 | 12 | 16 |
| | TBM | 4 | 16 | 8 | 8 | 12 | 16 |
| Shared Memory Efficiency | Native | 25% | 3% | 1.56% | 3.12% | 8.54% | 4.52% |
| | Padding | 50% | 50% | 50% | 50% | 50% | 50% |
| | AMM | 50% | 50% | 50% | 50% | 50% | 50% |
| | TBM | 50% | 50% | 50% | 50% | 50% | 50% |
| *N*-way Conflicts | Native | *4*-way | *32*-way | *32*-way | *16*-way | *16*-way | *20*-way |
| | Padding | None | None | None | None | None | None |
| | AMM | None | None | None | None | None | None |
| | TBM | None | None | None | None | None | None |
| GPU Occupancy | Native | 73.04% | 36% | 36% | 18% | 12% | 9% |
| | Padding | 67.03% | 24% | 24% | 14% | 9% | 6% |
| | AMM | 73.04% | 36% | 36% | 18% | 12% | 9% |
| | TBM | 73.04% | 36% | 36% | 18% | 12% | 9% |

# Linear Programming Model: Experimental Results (TX2)



# Linear Programming Model: Experimental Results (TX2)

| Parameters | Mapping | DCT | Transpose | Convolution Col | Convolution Row | Lud Perime-ter | Lud Diago-nal |
|---|---|---|---|---|---|---|---|
| Dataset Size (MB) | All | 1638 | 1024 | 1024 | 1024 | 1024 | 1024 |
| CUDA Block Size | All | 128 | 256 | 512 | 256 | 64 | 64 |
| Resident CUDA Blocks | Native | 16 | 4 | 2 | 2 | 5 | 4 |
| | Padding | **15** | **3** | **1** | **1** | **5** | **3** |
| | AMM | **12** | 4 | 2 | 2 | 5 | 4 |
| | TBM | **12** | 4 | 2 | 2 | 5 | 4 |
| Registers | Native | 25 | 32 | 32 | 32 | 32 | 32 |
| | Padding | 24 | 32 | 32 | 32 | 32 | 32 |
| | AMM | 37 | 32 | 32 | 40 | 32 | 32 |
| | TBM | 37 | 32 | 48 | 48 | 32 | 32 |
| Shared Memory (KB) | Native | 4 | 16 | 32 | 32 | 12 | 16 |
| | Padding | 4.125 | 16.25 | 32.5 | 33 | 12.125 | 16.25 |
| | AMM | 4 | 16 | 32 | 32 | 12 | 16 |
| | TBM | 4 | 16 | 32 | 32 | 12 | 16 |
| Shared Memory Efficiency | Native | 30% | 6.1% | 3.1% | 3.12% | 6.2% | 6% |
| | Padding | 100% | 100% | 100% | 100% | 100% | 100% |
| | AMM | 100% | 100% | 100% | 100% | 100% | 100% |
| | TBM | 100% | 100% | 100% | 100% | 100% | 100% |
| *N*-way Conflicts | Native | *4*-way | *32*-way | *32*-way | *16*-way | *16*-way | *20*-way |
| | Padding | None | None | None | None | None | None |
| | AMM | None | None | None | None | None | None |
| | TBM | None | None | None | None | None | None |
| GPU Occupancy | Native | 96.83% | 47% | 49.7% | 23.2% | 15.5% | 12.4% |
| | Padding | 90.50% | 35.5% | 41.7% | 21.4% | 15.4% | 9.1% |
| | AMM | 71.90% | 46.6% | 48% | 24.3% | 15.4% | 12.5% |
| | TBM | 72.3% | 46.3% | 48% | 23.7% | 15.4% | 12.5% |

# Conclusions

- **Polyhedral Model Transformation**
  - A methodology based on Z-modules, Left Hermite Normal Form and Smith Normal Form has been proved to be effective to provide improvements on SIMT architecture performance also in terms of power consumption.
  - prove that there exists a strong relationship between the access pattern to the memory, the shared memory bank conflicts and the power consumption
- **Linear Programming Model**
  - Approach that explores the solution space in order to find memory mapping schemes that avoid bank conflicts and memory waste.
  - The results pointed out a significant impact of the specific mapping choice adopted as a result of this analysis.

# Future Works

- **Polyhedral Model Transformation**
  - Automate the processing of the transformation
- **Linear Programming Model**
  - Automate the entire process, from the discovery of the mapping scheme to the source code transformation
    - E.g. implementing a parser which applies the scheme chosen by the user before the compilation process
    - insert an ad-hoc hardware component that routes the memory requests to the corresponding banks by computing the mapping dynamically
- **Explore the other factors that affect the performance and the power consumption**
  - E.g  Half-Precision Floating Point Arithmetic in AI problems

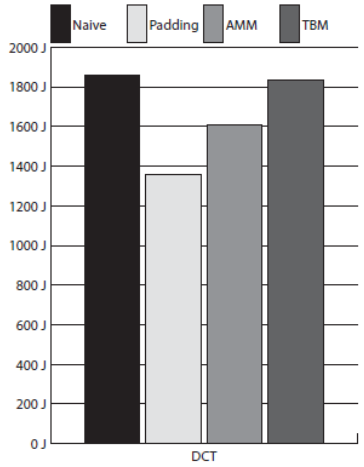
# Publications

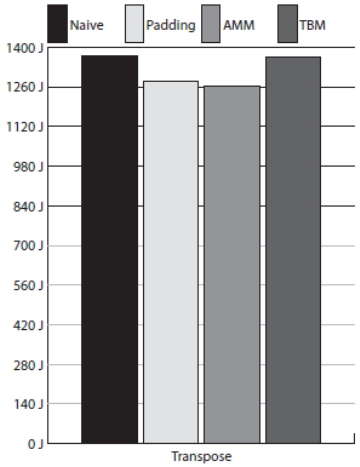
- A. Cilardo, I. Mungiello, "**Experimental evaluation of memory optimizations on an embedded GPU platform**", *10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing* (3PGCIC), 2015
- A. Cilardo, I. Mungiello and F. De Rosa, "**Adaptive Modular Mapping to Reduce Shared Memory Bank Conflicts on GPUs**", *11th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing* (3PGCIC), 2016
- A. Cilardo and I. Mungiello, "**Zero-conflict Memory Mapping for Transpose-like Kernels in SIMT Architectures**", *Journal of Parallel and Distributed Computing*, 2018, submitted.

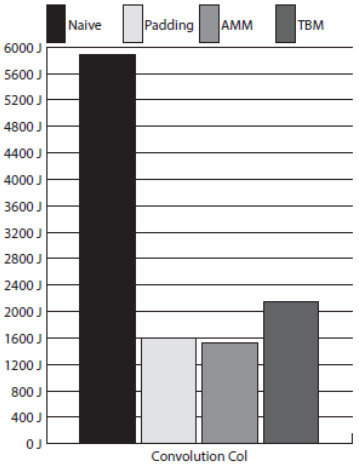
# Thanks!

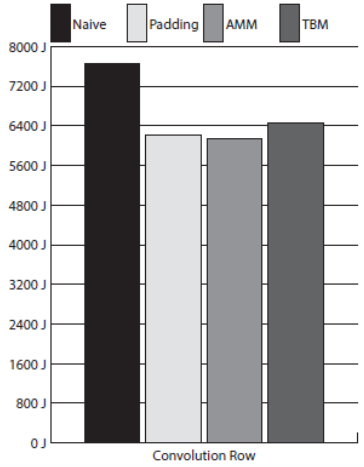
innocenzo.mungiello@unina.it
wpage.unina.it/innocenzo.mungiello

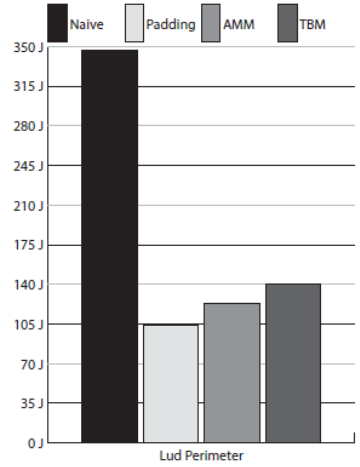
# Linear Programming Model: Experimental Results (TX2 - Energy)

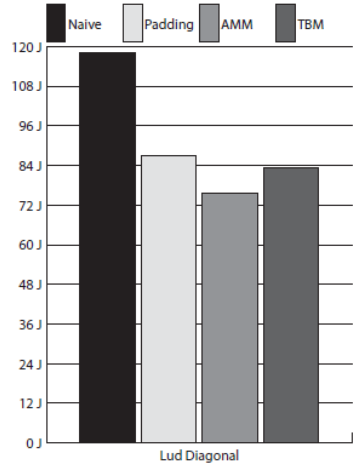


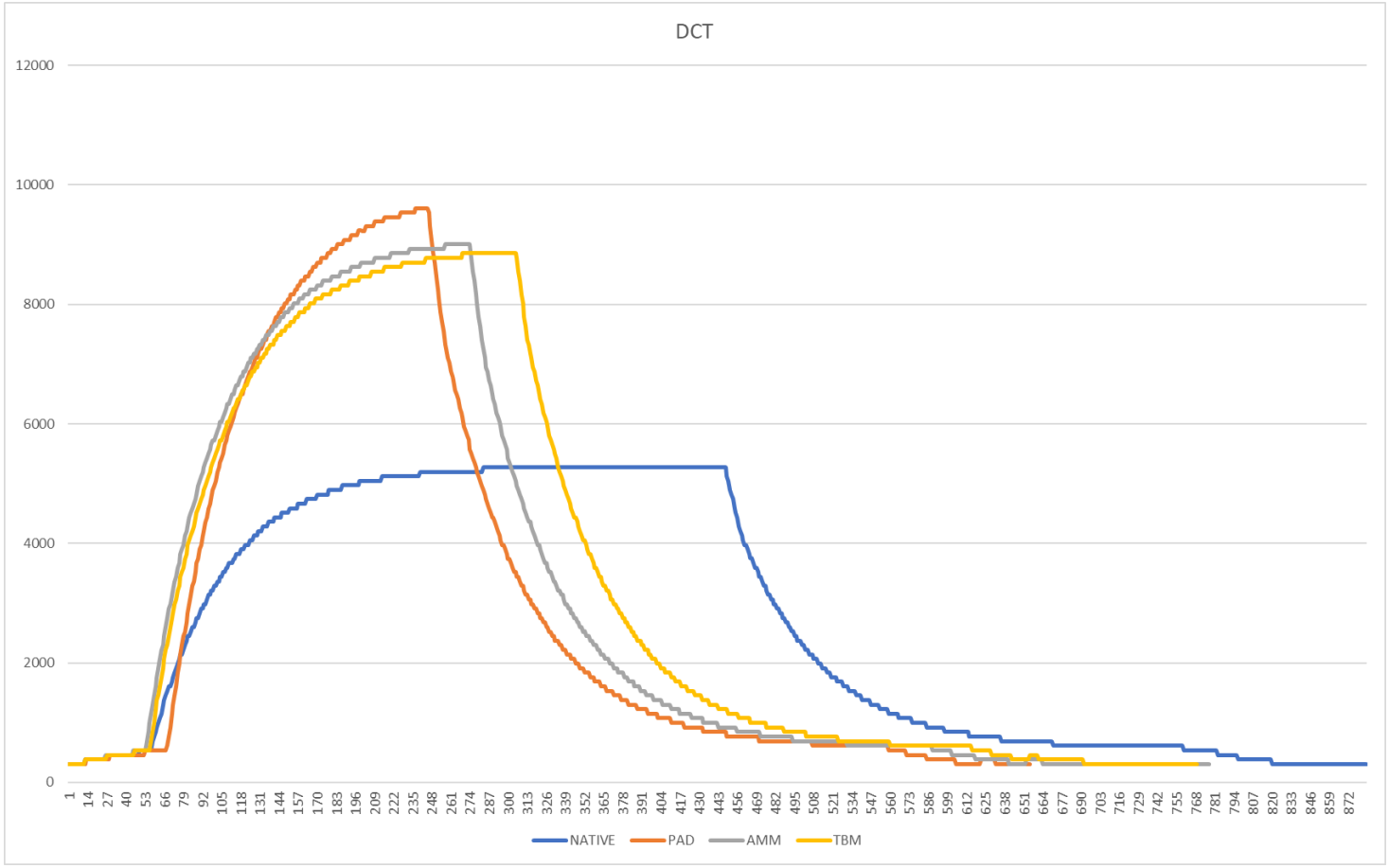



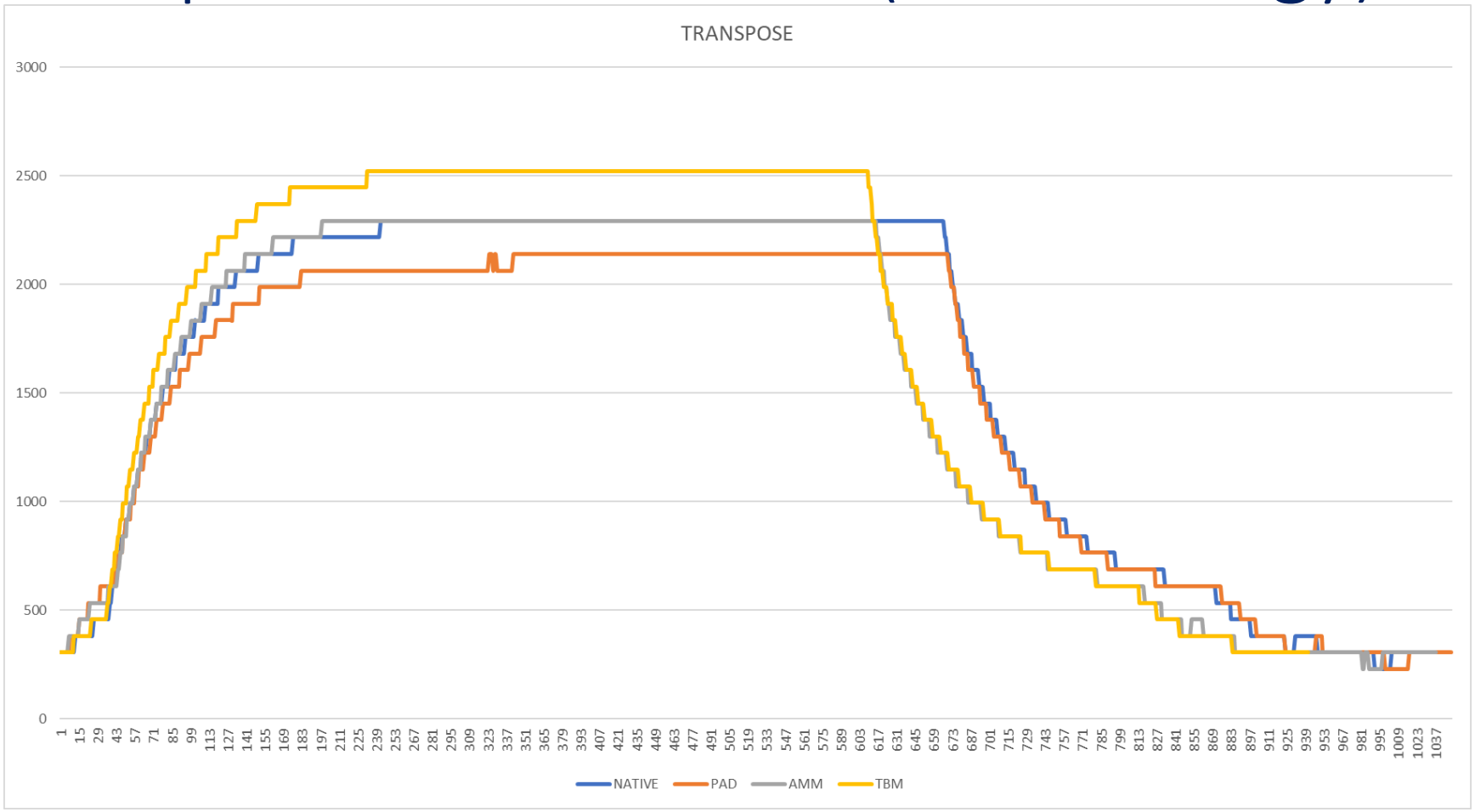



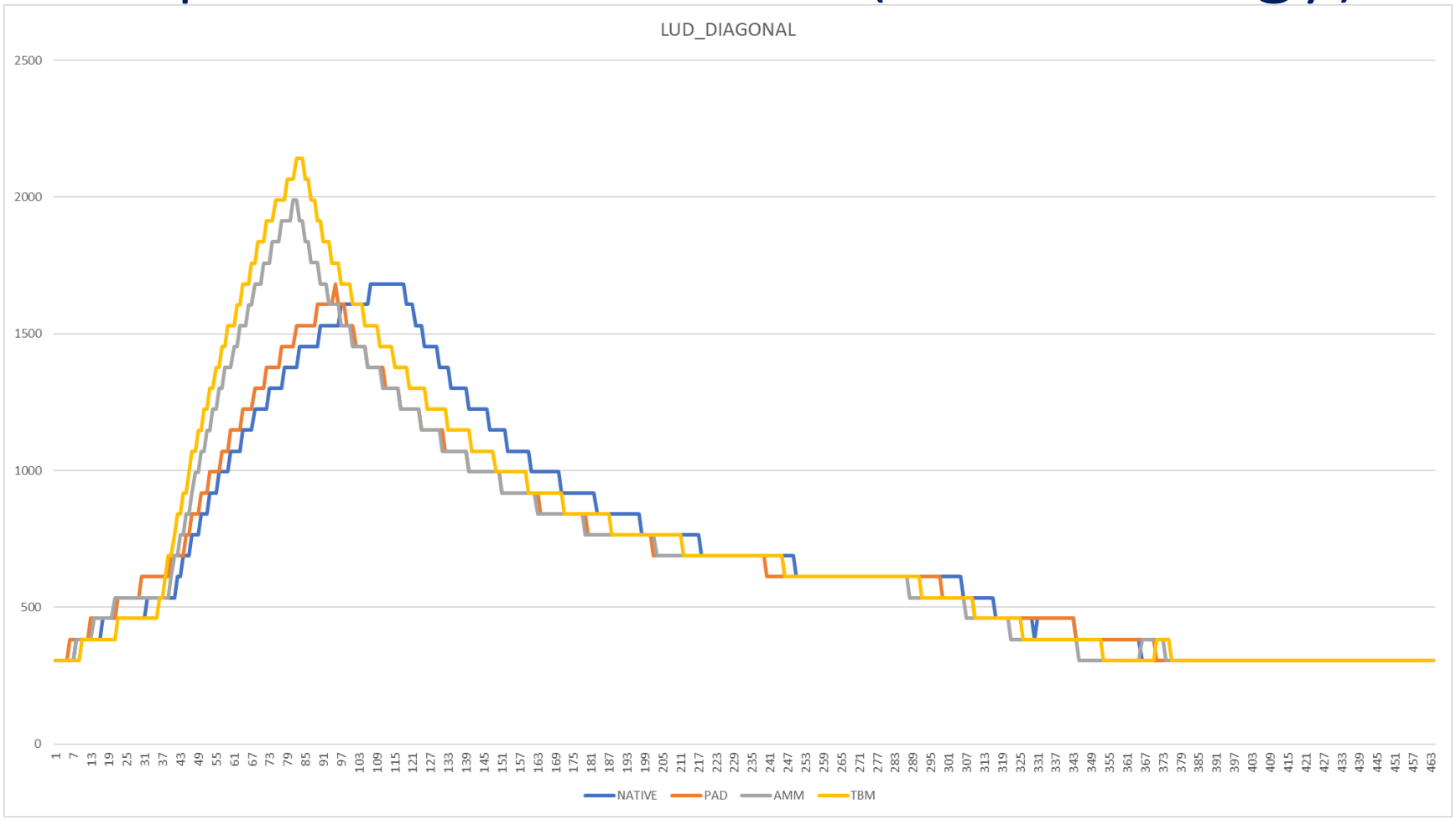
# Linear Programming Model: Experimental Results (TX2 - Energy)



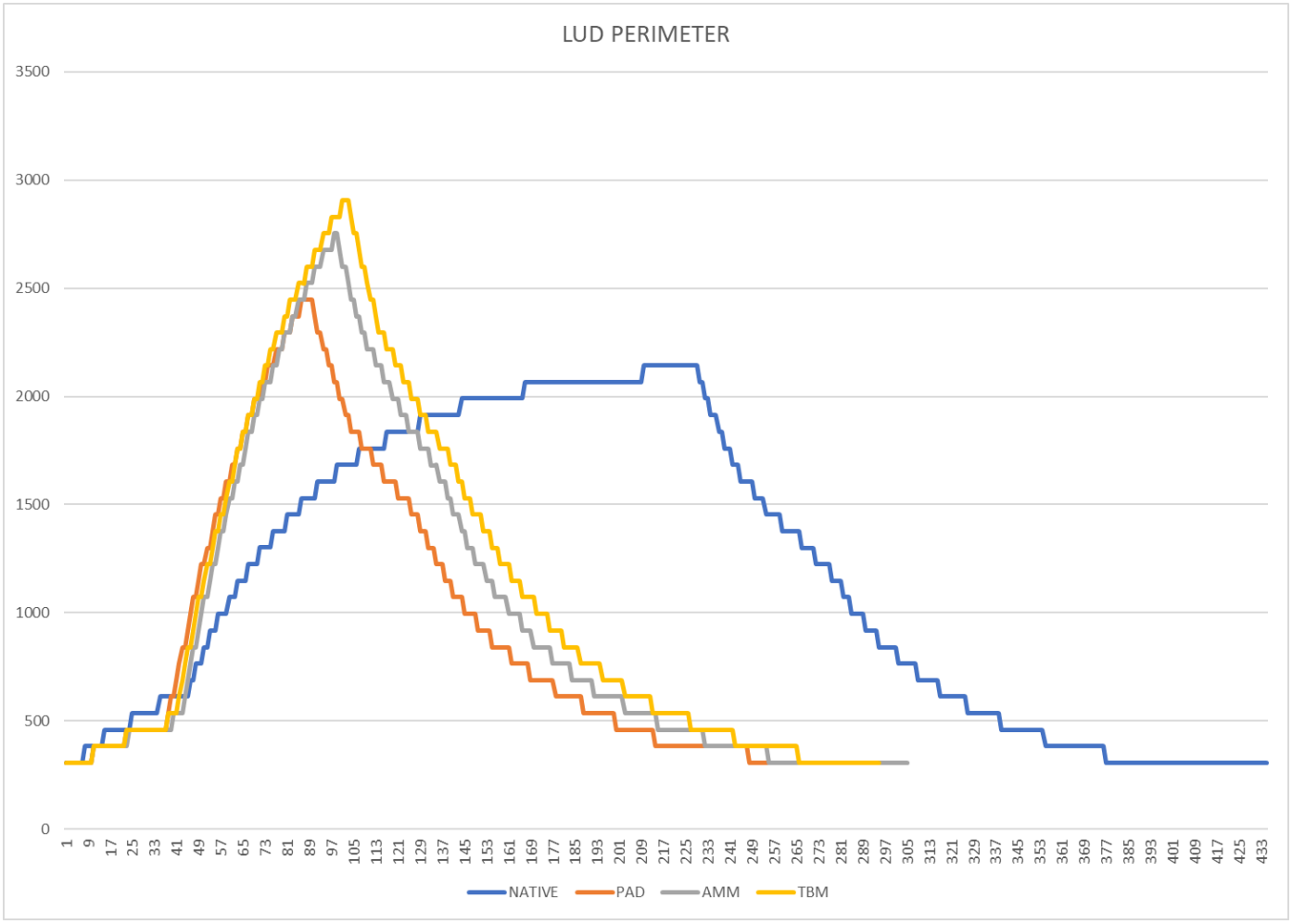
# Linear Programming Model: Experimental Results (TX2 - Energy)



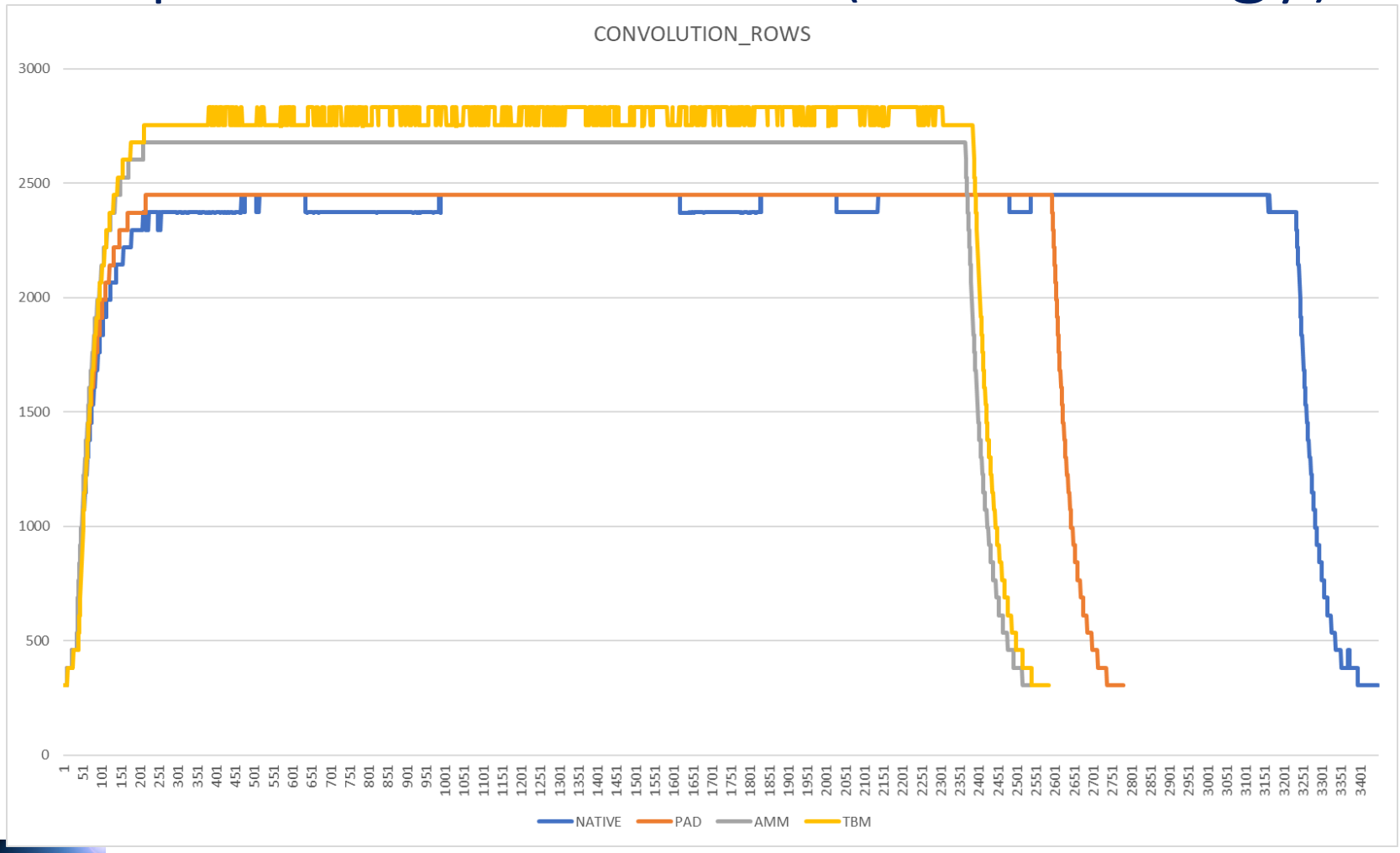
# Linear Programming Model: Experimental Results (TX2 - Energy)



# Linear Programming Model: Experimental Results (TX2 - Energy)



# Linear Programming Model: Experimental Results (TX2 - Energy)



# Linear Programming Model: Experimental Results (TX2 - Energy)