Innocenzo Mungiello

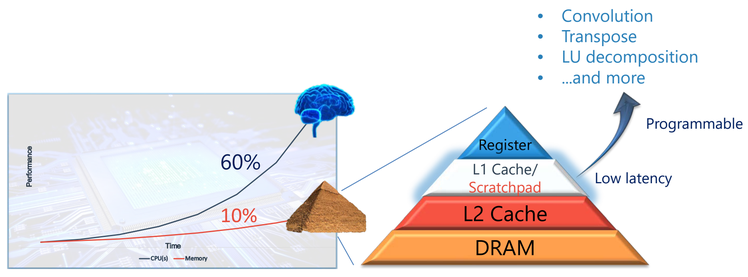
Tutor: Alessandro Cilardo

XXXI Cycle - II year presentation

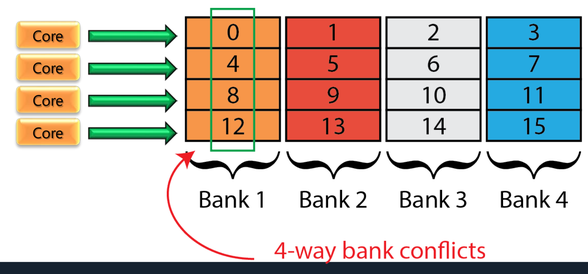
Improving Multibank Memory Access Parallelism on SIMT Architectures

The Unbroken Memory Wall:

- Memory wall remains a fundamental limit for high-performance computing systems.
- Also in terms of performance per watt
  - Only the 15% of energy consumption is used for useful computation
- The gap between compute and memory performance is called *Memory Wall*



- *Scratchpad* memories can improve memory subsystems performance
  - They are on-chip multi-banked, programmable and low latency memories
- A suitable placement strategy is essential for ensuring the maximum memory bandwidth to applications, as made available by the underlying multi-banked memory, minimizing or avoiding bank conflicts, which cause potentially parallel accesses to be serialized in time.



We propose a methodology for exploring conflict-free memory mapping schemes, focusing on a recurrent access pattern in many performance-critical applications, which we called **Transpose-Like**. In this pattern, store operations are performed row wise while load operations are performed column-wise, or vice versa. Because of the finite number of banks in the local memory, different store/load operations can incur conflicts. Existing programming practices for reducing or avoiding conflicts, like *padding*, involve limited modifications to the code but incurs some memory overhead.

Methodology: the proposed methodology relies on an Integer Linear Programming (ILP) model to describe the problem in terms of linear equalities ensuring optimal bank mapping strategies.

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

$$subject\ to:$$

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

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

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

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

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

Where N_{BK} is the Number of Scratchpad banks, N_{TH} is the number of threads that perform memory accesses and N_{IT} is the number of iteration performed by threads in order to load and store all data.

All feasible solutions of our ILP model guarantee that there aren't conflicts and memory overhead. We can represent a feasible solution using a mapping matrix S_n:

$$S_n = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \\ 3 & 4 & 1 & 2 \\ 4 & 1 & 2 & 3 \end{pmatrix}$$

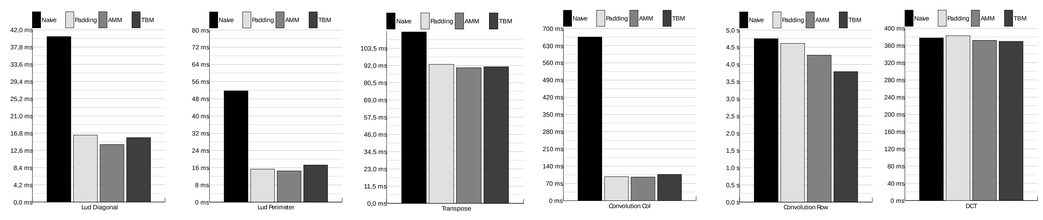
where rows represent the threads, columns represent the iterations, and each cell contains the bank index accessed by the corresponding thread/iteration.
Interpreting cells value as bank indices and the columns as the iterations of the loop facilitates the construction of a memory access function. In fact, we must simply subtract each column in S_n from the following one modulo *k*, meaning that if we obtain a negative value we need to add *k*.

Results: For our experiments we chose an NVIDIA Jetson TK1 GPU board as the test architecture. Using our ILP we found these four feasible solutions:

$$S_n^1 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \\ 3 & 4 & 1 & 2 \\ 4 & 1 & 2 & 3 \end{pmatrix} \quad S_n^2 = \begin{pmatrix} 1 & 4 & 3 & 2 \\ 2 & 1 & 4 & 3 \\ 3 & 2 & 1 & 4 \\ 4 & 3 & 2 & 1 \end{pmatrix}$$

$$S_n^3 = \begin{pmatrix} 1 & 2 & 4 & 3 \\ 2 & 3 & 1 & 4 \\ 3 & 4 & 2 & 1 \\ 4 & 1 & 3 & 2 \end{pmatrix} \quad S_n^4 = \begin{pmatrix} 1 & 4 & 2 & 3 \\ 2 & 1 & 3 & 4 \\ 3 & 2 & 4 & 1 \\ 4 & 3 & 1 & 2 \end{pmatrix}$$

We call S_n^1 Adaptive Modular Mapping (AMM), S_n^2 Inverse Adaptive Modular Mapping (IAMM), S_n^3 Triangular Based Mapping (TBM) and S_n^4 Inverse Triangular Based Mapping (ITBM).



Padding, AMM, and TBM solve all bank conflicts, but Padding waste memory and this lead to decreased performance as shown in *DCT*. The TBM technique must perform more arithmetic operations in order to compute memory access indices and in *Convolution Col* and *Lud Perimeter* kernels has a higher execution time than Padding. The *Convolution Row* kernel is the only one with non-coalesced global memory accesses. In this case, a higher utilization of the compute units by the TBM technique leads to a better execution time.

DIETI UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II
DIPARTIMENTO DI INGEGNERIA ELETTRICA E DELLE TECNOLOGIE DELL'INFORMAZIONE

CeR ICT

MANGO

Future Work:

- Automate the entire process, from the discovery of the mapping scheme to the source code transformation;
- Analyse the impact of memory performance in new fields like Artificial Intelligence and Approximate Computing.