

Symbolic Energy Analysis for Nested Loop Accelerators

Avinash Mahesh Nirmala, Dominik Walter, Frank Hannig, and Jürgen Teich

Department of Computer Science, Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Germany

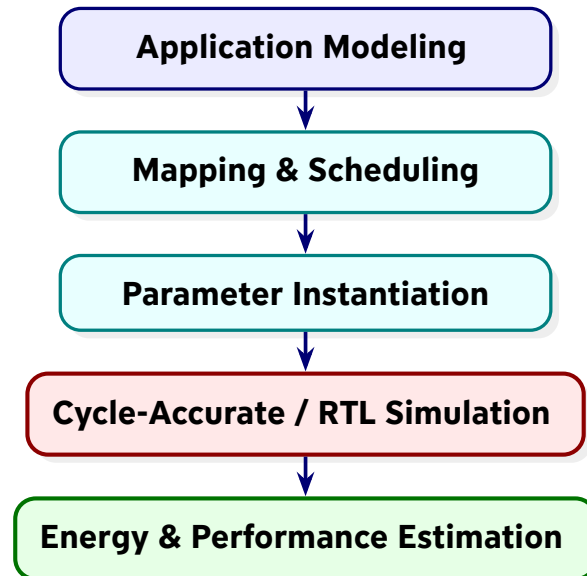
IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP),
Imperial College London, UK, Sept. 2-4, 2026



- **Growing AI workloads** demand higher computational performance, especially on the edge devices.
- Energy depends on **computation** and **data movement**.
- The choice of application mapping and scheduling strongly influences both **performance** and **energy** consumption.
- The large mapping design space requires evaluating many alternatives.
- Simulation-based energy and performance evaluation is too time-consuming for exhaustive exploration.

Can we avoid repeated simulation?

Conventional Simulation-Based Evaluation

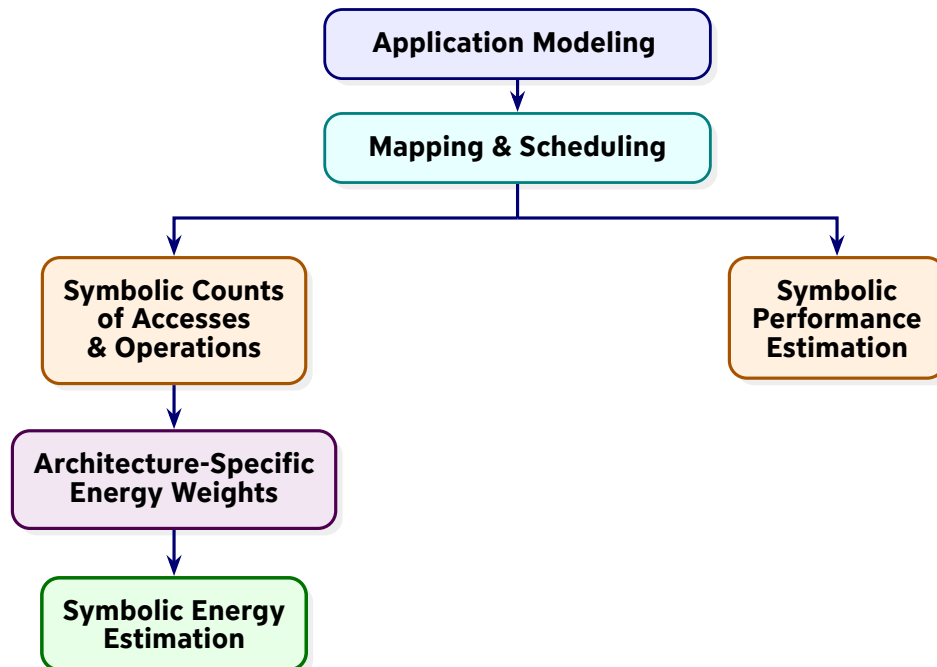


Repeated simulation for varying problem sizes

Context

- AI workloads, as well as many other digital signal processing and scientific computing applications, are usually represented as **nested loop programs**.
- These can be accelerated on massively parallel processor arrays (e.g., CGRAs, TCPAs, GPUs).
- Conventional mapping and scheduling exploration of loop-nest applications follows the workflow shown on the left.

Proposed Symbolic Approach



No time-intensive simulation for each problem size of application required!

Context

- AI workloads, as well as many other digital signal processing and scientific computing applications, can be represented as **nested loop programs**.
- These can be accelerated on massively parallel processor arrays (e.g., CGRAs, TCPAs, GPUs).
- The energy impact of mapping and scheduling decisions is analyzed symbolically.
- Symbolic operation and memory-access counts are combined with energy weights for fast energy estimation.

-
- Tightly Coupled Processor Arrays (TCPAs)
 - Loop-Nest Representation
 - Symbolic Tiling
 - Symbolic Scheduling
 - Energy Analysis
 - Experimental Results

Tightly Coupled Processor Arrays

Loop-Nest Representation

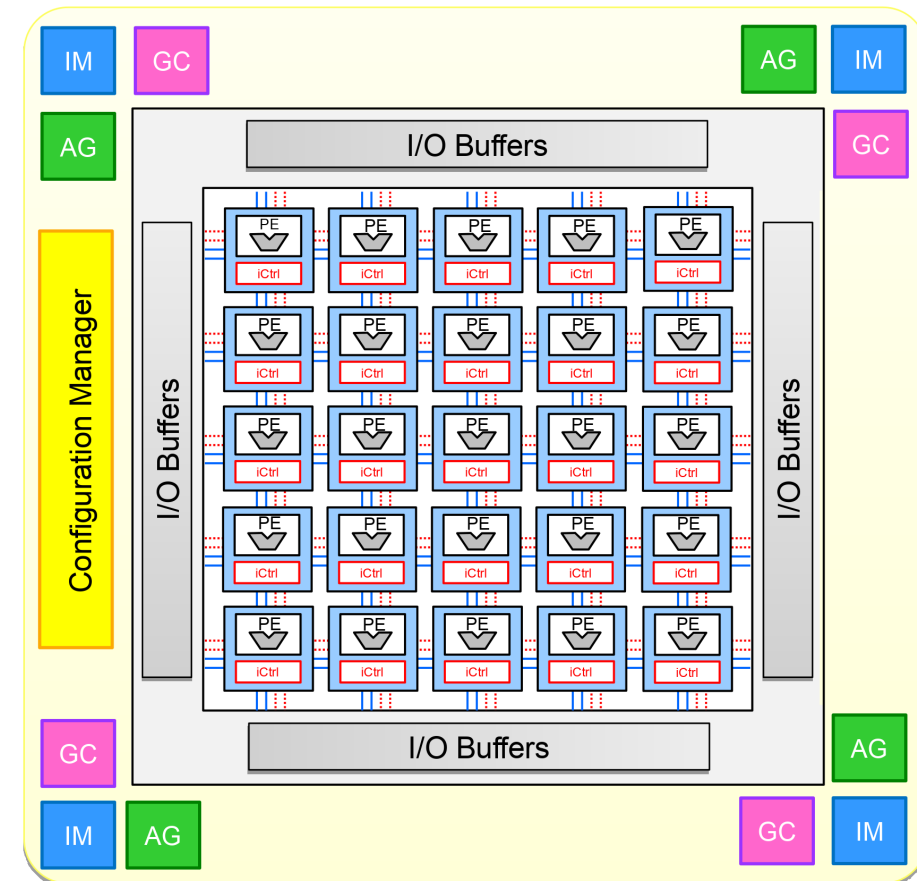
Symbolic Tiling

Symbolic Scheduling

Loop Nest Energy Analysis

Experimental Results

- Two-dimensional processor array for accelerating multidimensional loop applications.
- Programmable **Processing Elements (PEs)** connected through a configurable neighbor-to-neighbor interconnect.
- Bordered by **I/O buffers** with dedicated Address Generators (AGs) for efficient data transfer.
- Operates as a DMA-based coprocessor alongside a host system.



Example of a Tightly Coupled Processor Array (TCPA) architecture [1, 2, 3].

Tightly Coupled Processor Arrays

Loop-Nest Representation

Symbolic Tiling

Symbolic Scheduling

Loop Nest Energy Analysis

Experimental Results

Iteration Space

An n -dimensional loop nest is represented by the iteration space

$$\mathcal{I} \subseteq \mathbb{Z}^n, \quad \mathbf{i} = (i_0, i_1, \dots, i_{n-1})^T \in \mathcal{I}$$

with the polyhedral description

$$\mathcal{I} = \{\mathbf{i} \mid A\mathbf{i} \geq b\}, \quad A \in \mathbb{Z}^{m \times n}, \quad b \in \mathbb{Z}^m$$

Loop Program Representation

Loop nests are represented as Piecewise Linear Algorithms (PLAs),

$$S_q : x_q[\mathbf{P}_q \mathbf{i} + \mathbf{f}_q] = F_q(\dots, x_{q,r}[\mathbf{Q}_{q,r} \mathbf{i} - \mathbf{d}_{q,r}], \dots) \quad \text{if } \mathbf{i} \in \mathcal{I}_q \subseteq \mathcal{I}$$

A **Piecewise Regular Algorithm (PRA)** is a special case of a PLA with

$$\mathbf{P}_q = \mathbf{Q}_{q,r} = I, \quad \mathbf{f}_q = 0.$$

The statement S_q can be expressed as

$$S_q : x_q[\mathbf{i}] = F_q(\dots, x_{q,r}[\mathbf{i} - \mathbf{d}_{q,r}], \dots) \quad \text{if } \mathbf{i} \in \mathcal{I}_q \subseteq \mathcal{I}$$

GESUMMV [4] computes the sum of two matrix-vector products involving an input vector $X \in \mathbb{Z}^{N_1}$ and two matrices $A, B \in \mathbb{Z}^{N_0 \times N_1}$:

$$Y[i_0] = \sum_{i_1=0}^{N_1-1} (A[i_0, i_1] \cdot X[i_1] + B[i_0, i_1] \cdot X[i_1]), \quad 0 \leq i_0 < N_0.$$

This computation can be represented as a two-dimensional PRA over the iteration space

$$\mathcal{I} = \{(i_0, i_1) \mid 0 \leq i_0 < N_0, 0 \leq i_1 < N_1\},$$

using the following set of statements:

$$\begin{aligned} S_1 : x[i_0, i_1] &= X[i_1] && \text{if } i_0 = 0, \\ S_2 : x[i_0, i_1] &= x[i_0 - 1, i_1] && \text{if } i_0 > 0, \\ S_3 : a[i_0, i_1] &= A[i_0, i_1] \cdot x[i_0, i_1], \\ S_4 : b[i_0, i_1] &= B[i_0, i_1] \cdot x[i_0, i_1], \\ S_5 : s_A[i_0, i_1] &= a[i_0, i_1] && \text{if } i_1 = 0, \\ S_6 : s_A^*[i_0, i_1] &= s_A[i_0, i_1 - 1] && \text{if } i_1 > 0, \\ S_7 : s_A[i_0, i_1] &= s_A^*[i_0, i_1] + a[i_0, i_1] && \text{if } i_1 > 0, \\ S_8 : s_B[i_0, i_1] &= b[i_0, i_1] && \text{if } i_1 = 0, \\ S_9 : s_B^*[i_0, i_1] &= s_B[i_0, i_1 - 1] && \text{if } i_1 > 0, \\ S_{10} : s_B[i_0, i_1] &= s_B^*[i_0, i_1] + b[i_0, i_1] && \text{if } i_1 > 0, \\ S_{11} : Y[i_0] &= s_A[i_0, i_1] + s_B[i_0, i_1] && \text{if } i_1 = N_1 - 1. \end{aligned}$$

Tightly Coupled Processor Arrays

Loop-Nest Representation

Symbolic Tiling

Symbolic Scheduling

Loop Nest Energy Analysis

Experimental Results

Partitioning of the Iteration Space [5, 6, 7]

The n -dimensional iteration space $\mathcal{I}$ is partitioned into congruent tiles using a partitioning matrix

$$P = \text{diag}(p_0, p_1, \dots, p_{n-1})$$

such that

$$\mathcal{I} \subseteq \mathcal{J} \oplus P\mathcal{K}$$

The intra-tile iteration space $\mathcal{J}$ is defined as:

$$\mathcal{J} = \{\mathbf{j} = (j_0, \dots, j_\ell, \dots, j_{n-1})^T \mid 0 \leq j_\ell < p_\ell\}$$

- $\mathbf{j}$: iteration position within a tile
- p_ℓ : tile size along dimension ℓ

The tile-index space $\mathcal{K}$ is defined as:

$$\mathcal{K} = \{\mathbf{k} = (k_0, \dots, k_\ell, \dots, k_{n-1})^T \mid 0 \leq k_\ell < t_\ell\}$$

- $\mathbf{k}$: tile index
- t_ℓ : number of tiles along dimension ℓ

After tiling, each dependence $\mathbf{d} \in \mathbb{Z}^n$ is decomposed into the intra-tile dependence $\mathbf{d}_J$ and inter-tile dependence $\mathbf{d}_K$.

Transformed PRA Statements

$$S_q : x_q[\mathbf{j}, \mathbf{k}] = F_q(\dots, x_{q,r}^*[\mathbf{j}, \mathbf{k}], \dots) \quad \text{if } \mathbf{j} + P\mathbf{k} \in \mathcal{I}_q \quad (1)$$

$$S_q^* : x_{q,r}^*[\mathbf{j}, \mathbf{k}] = x_{q,r}[\mathbf{j} - \mathbf{d} - P\gamma, \mathbf{k} + \gamma] \quad \text{if } \mathbf{j} + P\mathbf{k} \in \mathcal{I}_q \\ \wedge \mathbf{j} - \mathbf{d} - P\gamma \in \mathcal{J}. \quad (2)$$

For each solution γ satisfying

$$\{\gamma \in \mathbb{Z}^n : -e < \gamma + P^{-1}\mathbf{d} < e\}, \quad e = (1 \ 1 \ \dots \ 1)^T$$

a corresponding statement S_q^* is created.

In the following, we call statements S_q of the form in Eq. (1) *computational statements*, as they involve the computation of functions F_q , and statements S_q^* of the form in Eq. (2) *memory-related statements*, as only a read of a right-hand-side variable and a write of the left-hand-side variable (copy) is involved.

Running Example

Symbolic Tiling and Dependency Transformation

Consider: $N_0 \times N_1 = 4 \times 5$, $t_0 \times t_1 = 2 \times 2$

For the explanation of tiling transformation, consider statement S_6

$$S_6 : s_A^*[i_0, i_1] = s_A[i_0, i_1 - 1].$$

The original dependence is

$$\mathbf{d} = (0, 1)^T.$$

After tiling, the two valid solutions are

$$\gamma^{(1)} = (0, 0)^T, \quad \gamma^{(2)} = (0, -1)^T,$$

According to Eq. (19), we obtain the following transformed statement S_6^* .

$$S_6^{*1} : s_A^*[\mathbf{j}, \mathbf{k}] = s_A[\mathbf{j} - (0, 1)^T, \mathbf{k}]$$

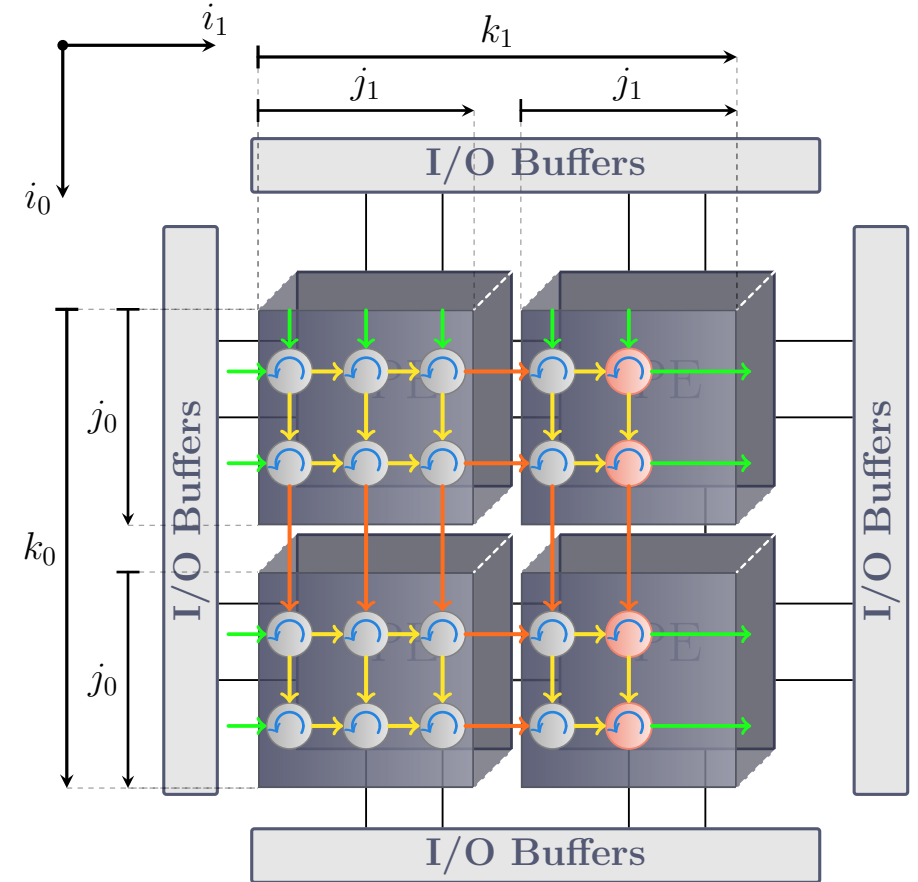
if $j_1 + p_1 k_1 > 0 \wedge \mathbf{j} - (0, 1)^T \in \mathcal{J}$ **(Intra-tile)**

$$S_6^{*2} : s_A^*[\mathbf{j}, \mathbf{k}] = s_A[\mathbf{j} - (0, 1 - p_1)^T, \mathbf{k} + (0, -1)^T]$$

if $j_1 + p_1 k_1 > 0 \wedge \mathbf{j} - (0, 1 - p_1)^T \in \mathcal{J}$ **(Inter-tile)**

resulting in

$$\mathbf{d}_6^* = \begin{cases} (0, 1, 0, 0)^T, \\ (0, 1 - p_1, 0, 1)^T. \end{cases}$$



Tiled GESUMMV iteration space mapped onto a 2×2 processor array.

Tightly Coupled Processor Arrays

Loop-Nest Representation

Symbolic Tiling

Symbolic Scheduling

Loop Nest Energy Analysis

Experimental Results

A schedule assigns a start time $\tau_q(\mathbf{j}, \mathbf{k})$ to a statement S_q when the operation F_q is computed for each iteration $(\mathbf{j}, \mathbf{k})$ [8, 5].

- Iterations within a tile are executed sequentially or in a pipelined order with initiation interval π .
- Different tiles are scheduled in parallel across the processor array, resulting in a **locally sequential, globally parallel (LSGP)** schedule.
- The schedule of iterations within a tile is described by an intra-tile schedule vector λ^J , similarly, the start time of tiles is also described by a so-called inter-tile schedule vector λ^K .

The schedule vector (λ^J, λ^K) determines the global latency

$$G_L = \lambda^J \begin{pmatrix} p_0 - 1 \\ \vdots \\ p_{n-1} - 1 \end{pmatrix} + \lambda^K \begin{pmatrix} t_0 - 1 \\ \vdots \\ t_{n-1} - 1 \end{pmatrix} + L_c$$

where

$$L_c = \max_{1 \leq q \leq |S|} (\tau_q + w_q).$$

Running Example

Symbolic Scheduling of GESUMMV

For the tiled GESUMMV application, and assuming an **initiation interval** $\pi = 1$, we obtain the symbolic intra-tile schedule

$$\lambda^J = (1, p_0)^T$$

and the inter-tile schedule

$$\lambda^K = (p_0, p_0(p_1 - 1) + 1)^T.$$

The global latency is

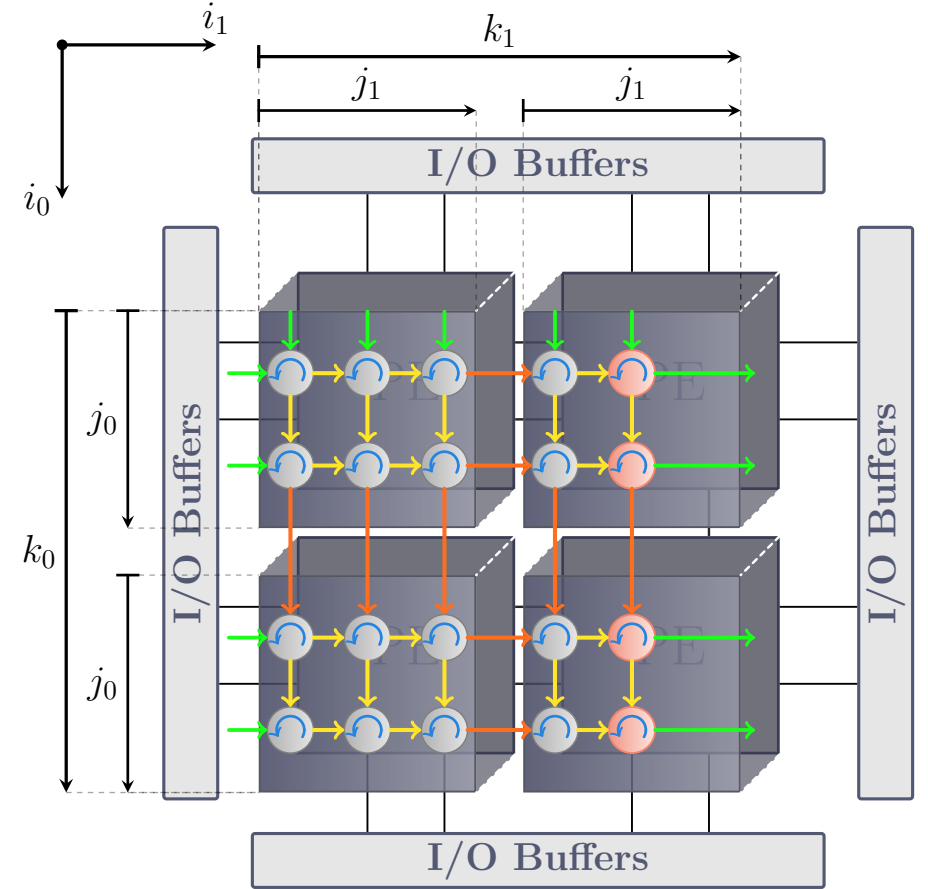
$$G_L = \lambda^J \cdot (p_0 - 1, p_1 - 1)^T + \lambda^K \cdot (t_0 - 1, t_1 - 1)^T + L_c.$$

For $L_c = 4$, $p_0 = 2$, $p_1 = 3$, $t_0 = t_1 = 2$,

$$G_L = (p_0 p_1 - 1) + p_0(t_0 - 1) + (p_0(p_1 - 1) + 1)(t_1 - 1) + 4,$$

and thus

$$G_L = 5 + 7 + 4 = 16.$$



Tightly Coupled Processor Arrays

Loop-Nest Representation

Symbolic Tiling

Symbolic Scheduling

Loop Nest Energy Analysis

Experimental Results

Let $L : x \rightarrow \mathcal{T}$ denote the function that returns the memory class $\mathcal{T} = \{\text{RD}, \text{FD}, \text{ID}, \text{OD}, \text{IOb}, \text{DR}\}$ of a variable x_q appearing on the left-hand side (write) or $x_{q,r}^*$ appearing on the right-hand side (read) of the statement S_q in Eq. (1):

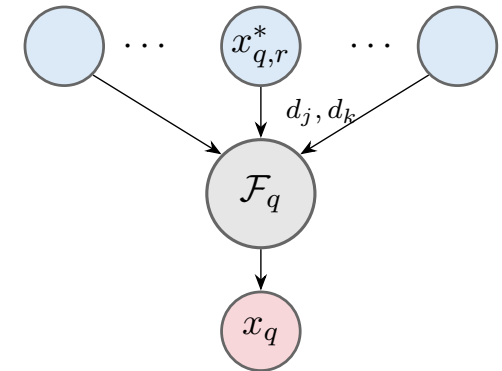
$$S_q : x_q[\mathbf{j}, \mathbf{k}] = F_q(\dots, x_{q,r}^*[\mathbf{j}, \mathbf{k}], \dots) \quad \text{if } \mathbf{j} + P\mathbf{k} \in \mathcal{I}_q.$$

The energy E_q^C of executing one iteration of S_q can be evaluated as

$$E_q^C = E(L(x_q)) + E(\mathcal{F}_q) + \sum_{r=1}^R E(L(x_{q,r}^*)),$$

where

- R is the number of RHS input variables,
- $E(\mathcal{F}_q)$ is the arithmetic-operation energy,
- $E(L(x))$ is the (nominal) energy of either writing ($x = x_q$) or reading ($x = x_{q,r}^*$) data of memory class $L(x)$.



Reduced Dependence Graph

Similarly, considering memory-related statement S_q^* in Eq. (2):

$$S_q^* : x_{q,r}^*[\mathbf{j}, \mathbf{k}] = x_{q,r}[\mathbf{j} - \mathbf{d} - P\gamma, \mathbf{k} + \gamma] \quad \text{if } \mathbf{j} + P\mathbf{k} \in \mathcal{I}_q \\ \wedge \mathbf{j} - \mathbf{d} - P\gamma \in \mathcal{J}.$$

The energy E_q^M of execution of one memory-related statement S_q^* can be evaluated as

$$E_q^M = E(L(x_q)) + E(L(x_{q,r}^*))$$

The energy of a memory access $E(L(x))$ depends on the role of the variable and involved dependence vectors:

$$E(L(x)) = \begin{cases} E(\text{DR}) + E(\text{IOb}) + E(\text{ID}) & \text{if } x \in I \quad // \text{ input variable} \\ E(\text{DR}) + E(\text{IOb}) + E(\text{OD}) & \text{if } x \in O \quad // \text{ output variable} \\ E(\text{RD}) & \text{if } \mathbf{d}_j = \mathbf{d}_k = 0 \\ E(\text{FD}) & \text{if } \mathbf{d}_j \neq 0 \wedge \mathbf{d}_k = 0 \\ E(\text{ID}) & \text{if } \mathbf{d}_k \neq 0 \end{cases}$$

Finally, the total energy E_{tot} for executing a parametric loop nest is obtained by summing up the energy of each statement S_q after tiling, multiplied by the volume of S_q , being given by the number of integer points of the polyhedral space where statement S_q is defined:

$$E_{\text{tot}} = \sum_{S_q \in C} \text{Vol}(S_q) \cdot E_q^C + \sum_{S_q \in M} \text{Vol}(S_q) \cdot E_q^M$$

Running Example

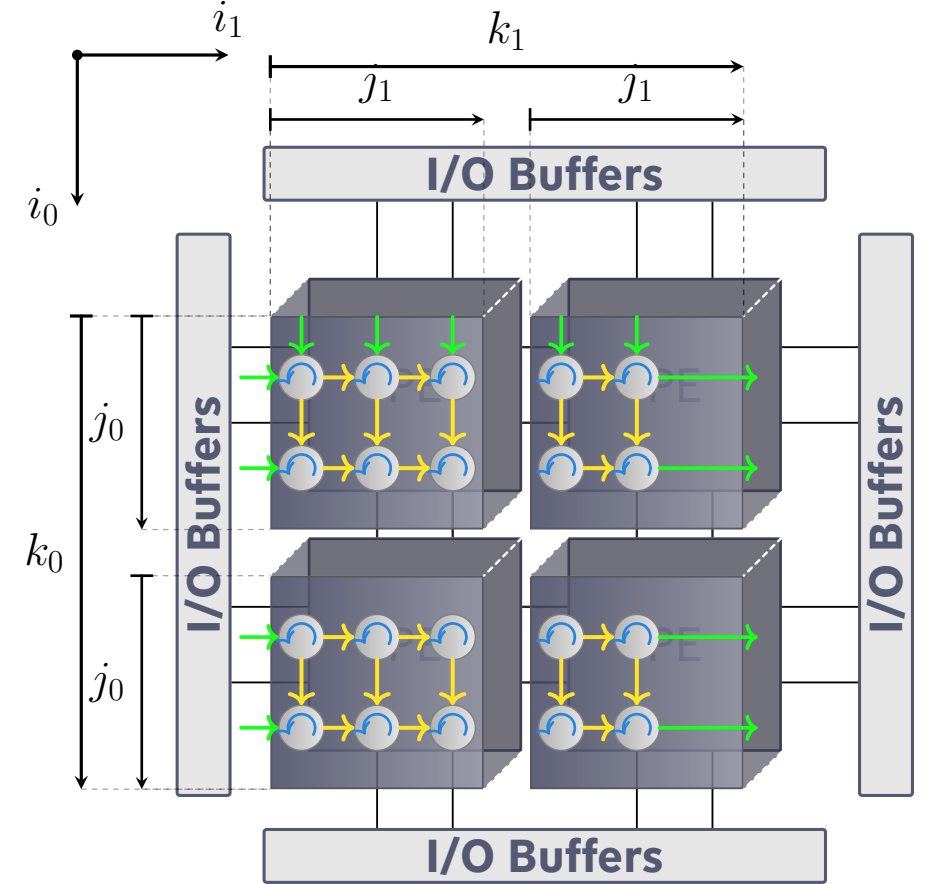
Symbolic Volume Computation

From the transformed statements S_6^{*1} and S_6^{*2} , we obtain two corresponding iteration spaces:

- $\mathcal{I}_6^{*1}$ for the intra-tile dependence,
- $\mathcal{I}_6^{*2}$ for the inter-tile dependence.

$$\mathcal{I}_6^{*1} = \left\{ (j_0, j_1, k_0, k_1) \in \mathbb{Z}^4 \left| \begin{array}{l} 0 \leq j_0 < p_0, \quad 0 \leq j_1 < p_1, \\ 0 \leq k_0 < t_0, \quad 0 \leq k_1 < t_1, \\ 0 \leq j_0 + p_0 k_0 < N_0, \\ 0 < j_1 + p_1 k_1 < N_1, \\ 0 \leq j_1 - 1 < p_1 \end{array} \right. \right\}$$

$$\mathcal{I}_6^{*2} = \left\{ (j_0, j_1, k_0, k_1) \in \mathbb{Z}^4 \left| \begin{array}{l} 0 \leq j_0 < p_0, \quad 0 \leq j_1 < p_1, \\ 0 \leq k_0 < t_0, \quad 0 \leq k_1 < t_1, \\ 0 \leq j_0 + p_0 k_0 < N_0, \\ 0 < j_1 + p_1 k_1 < N_1, \\ 0 \leq j_1 + p_1 - 1 < p_1 \end{array} \right. \right\}$$



Running Example

Symbolic Volume Computation

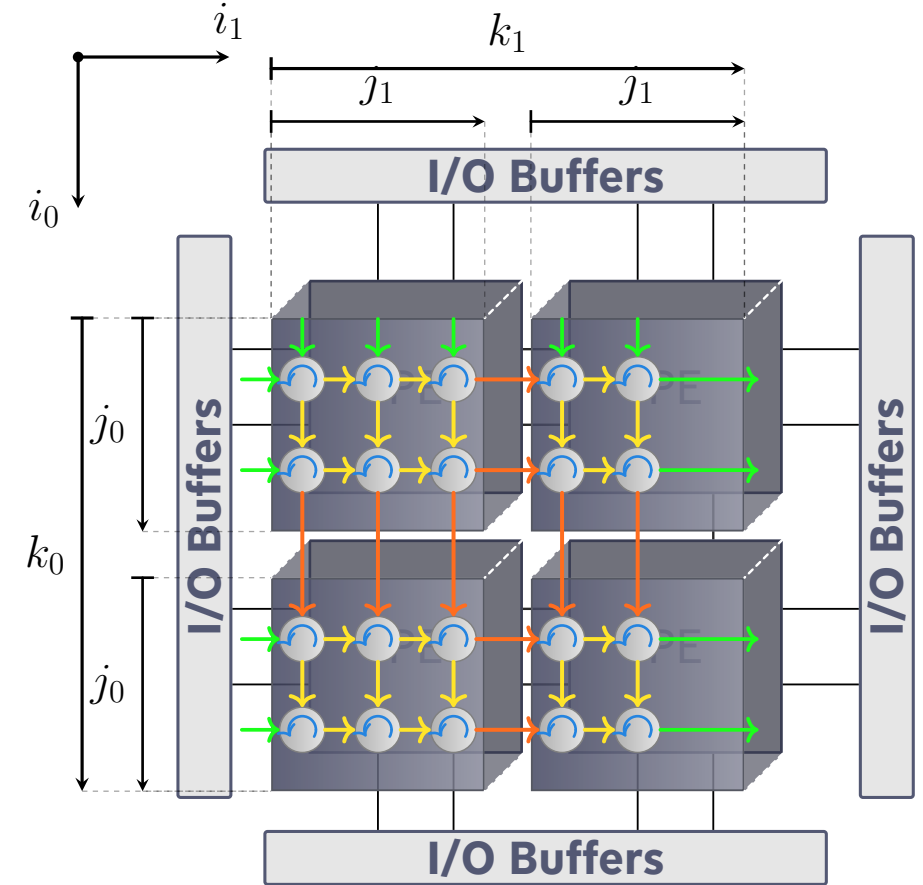
Applying Barvinok's algorithm [9] as part of the Integer Set Library (ISL) [10] is able to compute the number of integer-point counts for the two cases symbolically.

Intra-tile case: S_6^{*1}

$$\text{Vol}(S_6^{*1}) = \begin{cases} 4p_0(p_1 - 1), & 0 < p_0, 2p_0 < N_0 \wedge p_1 \geq 2 \wedge 2p_1 < N_1 \\ 2N_0(p_1 - 1), & N_0 > 0, 2p_0 \geq N_0 \wedge p_1 \geq 2 \wedge 2p_1 < N_1 \\ (2N_1 - 4)p_0, & 0 < p_0, 2p_0 < N_0 \wedge p_1 \leq N_1 - 2 \wedge 2p_1 \geq N_1 \\ N_0(N_1 - 2), & N_0 > 0, 2p_0 \geq N_0 \wedge p_1 \leq N_1 - 2 \wedge 2p_1 \geq N_1 \\ 0, & \text{otherwise.} \end{cases}$$

Inter-tile case: S_6^{*2}

$$\text{Vol}(S_6^{*2}) = \begin{cases} 2p_0, & 0 < p_0 < N_0/2 \wedge 0 < p_1 < N_1, \\ N_0, & N_0 > 0 \wedge p_0 \geq N_0/2 \wedge 0 < p_1 < N_1, \\ 0, & \text{otherwise.} \end{cases}$$



Running Example

Symbolic Volume Computation

Interpretation

Substituting $N_0 \times N_1 = 4 \times 5$ and $p_0 \times p_1 = 2 \times 3$ into the symbolic volume expressions, we obtain

- $\text{Vol}(S_6^{*1}) = 12$ **intra-tile dependencies** $\rightarrow$ 12 yellow arrows in the j_1 direction.
- $\text{Vol}(S_6^{*2}) = 4$ **inter-tile dependencies** $\rightarrow$ 4 orange arrows in the k_1 direction.

For S_6^{*1} , we obtain

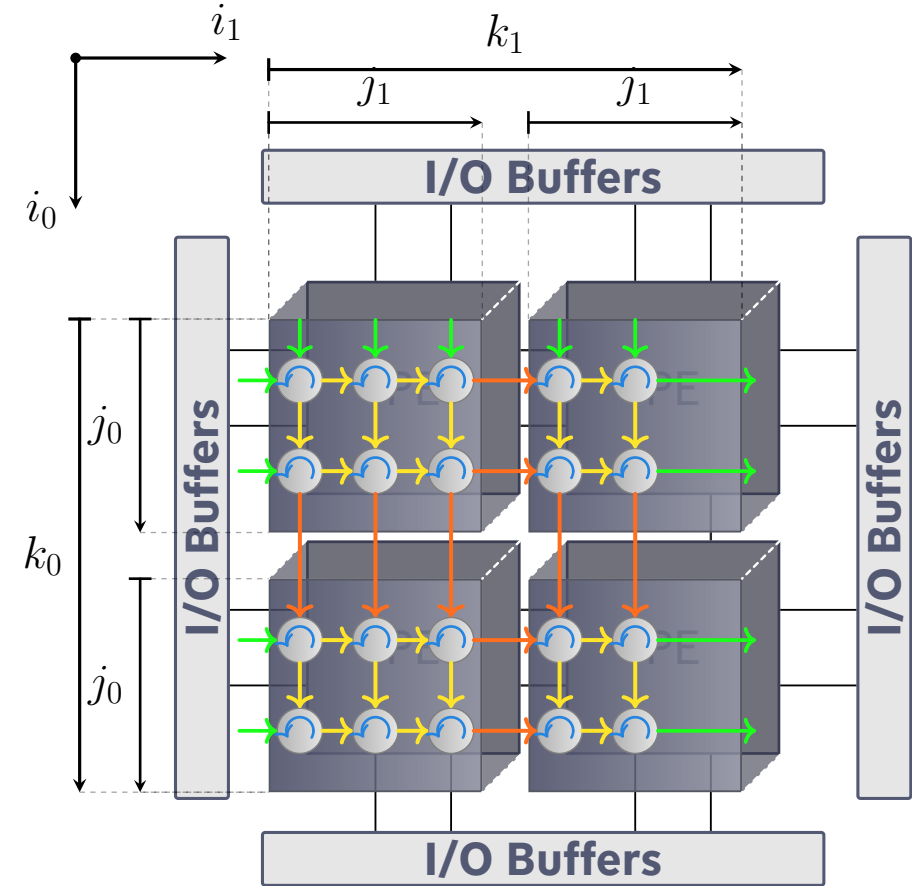
$$E_{6^{*1}}^M = E(\text{FD}) + E(\text{RD}) = 0.35 + 0.12 = 0.47 \text{ pJ.}$$

For S_6^{*2} , we obtain

$$E_{6^{*2}}^M = E(\text{ID}) + E(\text{RD}) = 0.24 + 0.12 = 0.36 \text{ pJ.}$$

Therefore:

$$\begin{aligned} E_{S_6^*} &= 12(0.47) + 4(0.36) \\ &= 7.08 \text{ pJ.} \end{aligned}$$



Tightly Coupled Processor Arrays

Loop-Nest Representation

Symbolic Tiling

Symbolic Scheduling

Loop Nest Energy Analysis

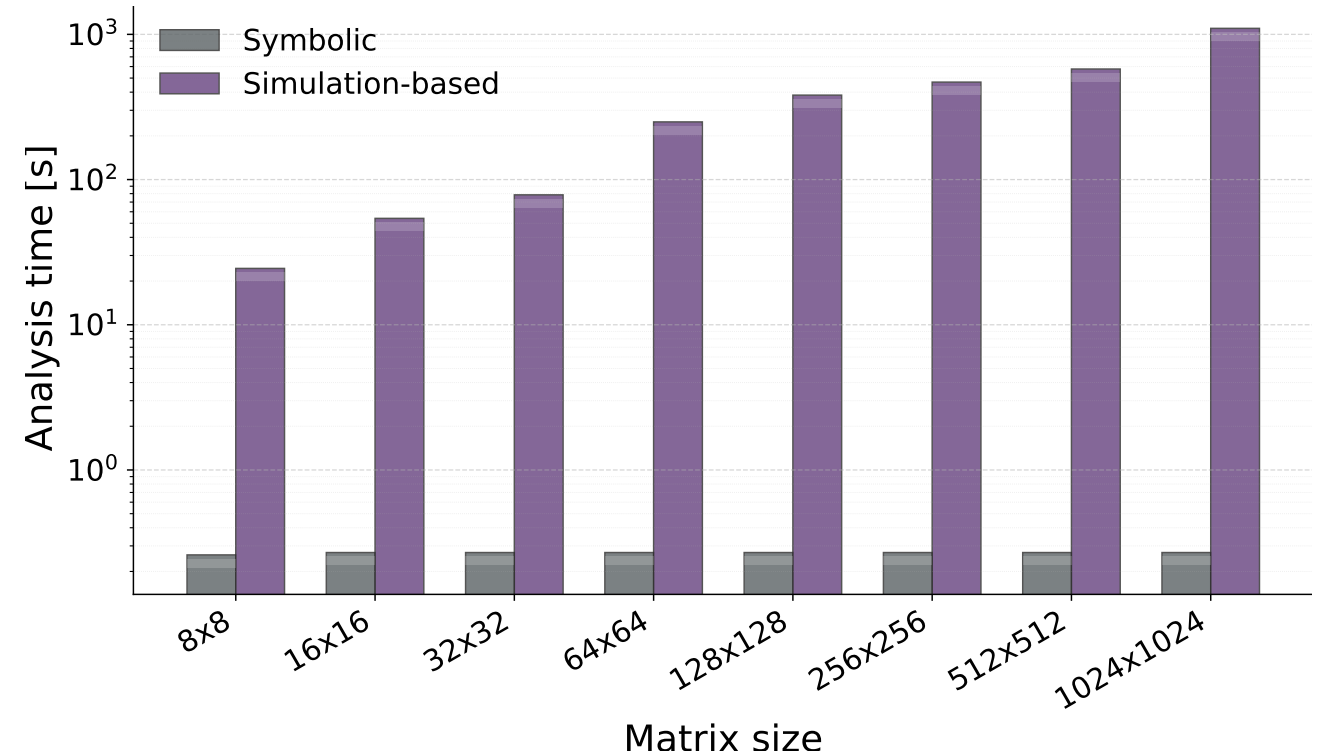
Experimental Results

Experimental Results

Accuracy and Analysis Time

Validation of Symbolic Analysis

- **Reference:** One cycle-accurate simulation run needed for each problem size, executing every iteration, computation and data access individually.
- **Validation:** Symbolic computation and memory-access volumes are derived as closed-form expressions and instantiated using the same problem sizes as in the simulation.



GESUMMV: Symbolic vs. simulation-based analysis time

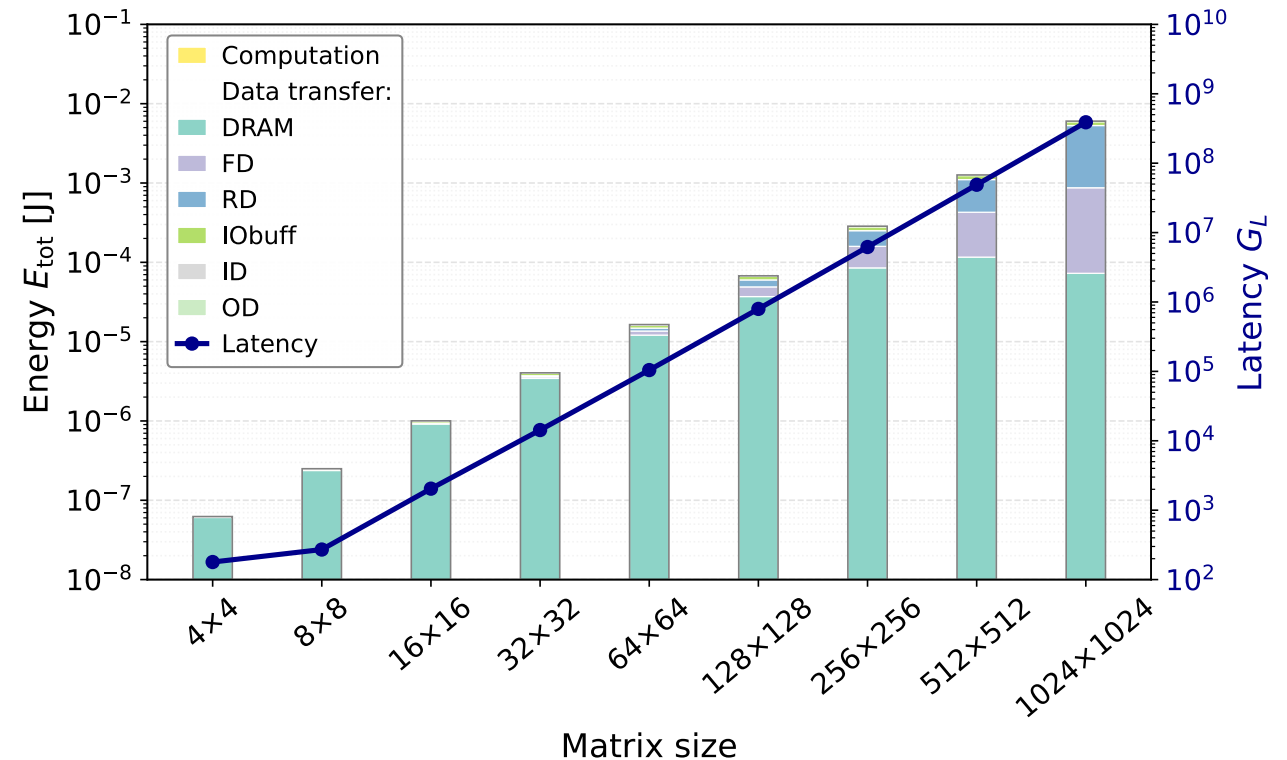
Key Observation

Different memory and register accesses do **not necessarily scale at the same rate**.

Their scaling depends on

tile sizes + data dependencies

Consequently, extrapolating activity counts from a small simulated workload may not accurately capture scaling at larger loop bounds.



Total energy E_{tot} and global latency G_L vs. matrix size for GEMM

Summary

We presented a framework for symbolic evaluation of energy and latency of tiled loop nests mapped onto processor-array accelerators. Instead of repeatedly executing cycle-accurate simulations, our approach derives reusable closed-form symbolic expressions.

- Symbolic computation and memory-access analysis using parametric loop bounds.
- One-time performed symbolic analysis allowing polynomial time evaluation of energy and latency for arbitrary loop bounds and accelerator configurations.
- Enabling efficient architectural design space exploration.

- [1] Dmitrij Kissler, Frank Hannig, Alexey Kupriyanov, and Jürgen Teich. "A highly parameterizable parallel processor array architecture". In: *IEEE Int. Conf. on Field Programmable Technology (FPT)*, Bangkok, Thailand. IEEE, 2006, pp. 105–112. DOI: 10.1109/FPT.2006.270293.
- [2] Frank Hannig, Vahid Lari, Srinivas Boppu, Alexandru Tanase, and Oliver Reiche. "Invasive Tightly-Coupled Processor Arrays: A Domain-Specific Architecture/Compiler Co-Design Approach". In: *ACM Trans. Embed. Comput. Syst.* 13.4s (2014), 133:1–133:29. DOI: 10.1145/2584660.
- [3] Frank Hannig, Vahid Lari, Srinivas Boppu, Alexandru Tanase, and Oliver Reiche. "Invasive Tightly-Coupled Processor Arrays: A Domain-Specific Architecture/Compiler Co-Design Approach". In: *ACM Trans. Embedded Comput. Syst.* 13.4s (2014), 133:1–133:29. DOI: 10.1145/2584660.
- [4] Louis-Noël Pouchet. *PolyBench: The Polyhedral Benchmark Suite*. Accessed: Oct. 10, 2025.
- [5] M. Witterauf, D. Walter, F. Hannig, and J. Teich. "Symbolic Loop Compilation for Tightly Coupled Processor Arrays". In: *ACM Trans. Embedded Comput. Syst. (TECS)* 20.5 (2021), pp. 1–31.
- [6] Jürgen Teich and Lothar Thiele. "Partitioning of processor arrays: a piecewise regular approach". In: *Integr.* 14.3 (1993), pp. 297–332. DOI: 10.1016/0167-9260(93)90013-3.
- [7] Jürgen Teich, Lothar Thiele, and Lee Z. Zhang. "Partitioning Processor Arrays under Resource Constraints". In: *J. VLSI Signal Process.* 17.1 (1997), pp. 5–20. DOI: 10.1023/A:1007935215591.
- [8] Jürgen Teich, Alexandru Tanase, and Frank Hannig. "Symbolic Mapping of Loop Programs onto Processor Arrays". In: *J. Signal Process. Syst.* 77.1-2 (2014), pp. 31–59. DOI: 10.1007/S11265-014-0905-0.
- [9] Alexander I. Barvinok. "A polynomial time algorithm for counting integral points in polyhedra when the dimension is fixed". In: *Mathematics of Operations Research* 19.4 (1994), pp. 769–779.
- [10] Sven Verdoolaege. "*isl*: An Integer Set Library for the Polyhedral Model". In: *Third Int. Congress on Mathematical Software (ICMS)*. Ed. by Komei Fukuda et al. Vol. 6327. Springer, 2010, pp. 299–302.

Thank you!

Avinash Mahesh Nirmala, Dominik Walter, Frank Hannig, and Jürgen Teich

Department of Computer Science, Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Germany

IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP),
Imperial College London, UK, Sept. 2-4, 2026

Symbolic Energy Analysis for Nested Loop Accelerators

avinash.avi.mahesh@fau.de