Lattice-Based Memory Allocation

Gilles Villard
CNRS, Laboratoire LIP, ENS Lyon France

Joint work with Alain Darte (CNRS, LIP) and Rob Schreiber (HP Labs)

MOCAA, May 6, 2004, University of Waterloo
Introduction
Example 1.

\[
\text{do } i = 0, N - 1 \\
\text{do } j = 0, N - 1 \\
\text{S: } A(i, j) = \ldots \\
\text{end} \\
\text{end}
\]

\[
\text{do } i = 0, N - 1 \\
\text{do } j = 0, N - 1 \\
\text{T: } B(i, j) = A(i, j) + \ldots \\
\text{end} \\
\text{end}
\]

**Schedule:**  \( \theta(S, i, j) = Ni + j \)

\( \theta(S, i, j) = Ni + j + 1 \)

i.e., one “clock-cycle” later
Example 1.

do \ i = 0, \ N - 1
  do \ j = 0, \ N - 1
    S: \ A(i, j) = \ldots
  end
end

\textbf{Schedule:} \ \theta(S, i, j) = Ni + j

\begin{align*}
do \ i = 0, \ N - 1
  do \ j = 0, \ N - 1
    T: \ B(i, j) = A(i, j) + \ldots
  end
end
\end{align*}

\[ \theta(S, i, j) = Ni + j + 1 \]
i.e., one “clock-cycle” later

Design \textcolor{red}{\textbf{intermediate buffers}} for \( A \) with \textcolor{red}{\textbf{memory reuse}}?
Some array values cannot share the same buffer location, e.g. \( A(i, j) \) and \( A(i, j + 1) \) since \( A(i, j) \) is required later by the second loop, we say that corresponding indices are conflicting (relation \( \Join \)):

\[
(i, j) \Join (i, j + 1)
\]

\[
(i, N - 1) \Join (i + 1, 0) \quad (N = 6)
\]
The allocation function

\[ \sigma : (i, j) \mapsto Ni + j \mod 2, \]

which stores \( A(i, j) \) in Buffer[\( \sigma(i, j) \)], is a valid allocation (1D), indeed,

\( (i, j) \not\equiv (i, j + 1) : Ni + j \neq Ni + j + 1 \mod 2 \)

\( (i, N - 1) \not\equiv (i + 1, 0) : Ni + N - 1 \neq Ni + N \mod 2 \)

Conflicting indices are stored in different memory locations

Preserves the program semantics
Example 2. DCT-like code

\[
\begin{align*}
\text{do } b_r &= 0, 63 \\
\text{do } b_c &= 0, 63 \\
\text{do } r &= 0, 7 \\
S: A(b_r, b_c, r, *) &= \ldots
\end{align*}
\]

end

end

end

Pipelined with

\[
\begin{align*}
\text{do } b_r &= 0, 63 \\
\text{do } b_c &= 0, 63 \\
\text{do } c &= 0, 7 \\
T: \ldots &= A(b_r, b_c, *, c)
\end{align*}
\]

end

end

end

How to allocate elements of \( A \) in local memory, and minimize the size?

\[\leadsto\text{ Full array: } 64 \times 64 \times 8 \times 8 = 2^{18} = 256\text{K} \]

\[\leadsto\text{Optimal linear allocation: } 112 \text{ elements}, \sigma : \begin{cases} r \mod 4 \\
16(b_r + b_c) + 2r + c \mod 28.\end{cases}\]
How a compiler can automatically find a valid allocation?

Main constraints:

- Optimization of the size of the allocation (size of the buffer)
- Simplicity of the addressing function for implementation aspects

General context:

Compilers, parallelizers (static optimization, loop transformation, . . . )
Application-specific circuit, communicating hardware processes
Automatic synthesis of hardware accelerators
PICO: Program In Chip Out

Program In

PICO
Architecture Synthesis

Compiler

VHDL for processors

CAD Tools

Logical Synthesis
Physical Design

Chip Code Out

Input C code

Output "code"

– synthesizable VHDL
– netlists for FPGA
– VLIW code

Similar tools: MMAlpha (INRIA), Atomium (IMEC), Compaan (Leiden)

Introduction and context
Outline

Introduction and context

I - Problem statement, and previous heuristic limitations

II - Model: Integral lattices and linear allocations

III - Application: Memory allocation constructions and heuristics

Conclusion
Outline

Introduction and context

I - Problem statement, and previous heuristic limitations

II - Model: Integral lattices and linear allocations

III - Application: Memory allocation constructions and heuristics

Conclusion
Scheduled program or communicating processes

+ Dependence analysis (lifetime)

+ Choice of vector indices (e.g., loop indices or array, etc.)
Scheduled program or communicating processes

+ Dependence analysis (lifetime)

+ Choice of vector indices (e.g., loop indices or array, etc.)

↓

**Data storage optimization with respect to a representation**
Previous approaches

De Greef, Catthoor and De Man (1996-1997)
Lefebvre and Feautrier (1996-1997)
Strout, Carter, Ferrante and Simon (1998)
Thies, Vivien, Sheldon and Amarasinghe (2001)
All these approaches may be formalized using:

**Definition:** Two indices $\vec{i}$ and $\vec{j}$ of $\mathbb{Z}^n$ are **conflicting** ($\vec{i} \Join \vec{j}$) if they correspond to two values that are simultaneously alive during the execution with schedule $\theta$.

\[
\text{CS} = \{ (\vec{i}, \vec{j}) | \vec{i} \Join \vec{j} \} : \text{the set of all pairs of conflicting indices.}
\]

Ex: \{([0], [0]), ([0], [1]), \ldots, ([0], [1]), ([1], [2]), \ldots\}

**Definition:** a **linear allocation** of size $m$ is a homomorphism $\sigma : \mathbb{Z}^n \rightarrow \mathcal{M}$, where $\mathcal{M} \subset \mathbb{Z}^p$ is a finite abelian group of $m$ elements.

Problem statement
Valid linear allocation

For conflicting indices $\vec{i}$ and $\vec{j}$, $\vec{i} \neq \vec{j}$ one must have $\sigma(\vec{i}) \neq \sigma(\vec{j})$, i.e. $\sigma(\vec{i} - \vec{j}) \neq 0$

$DS = \{\vec{i} - \vec{j} \in \mathbb{Z}^n | i \not\sim j\}$

Definition: $\sigma$ is valid iff $DS \cap \ker \sigma = \{\vec{0}\}$.

Example of Difference Set

$(i,j)-(i,j+1) = (0,1)$

$(0,N-1)-(1,0) = (-1,N-1)$

Problem statement
For affine schedules, regular sets of iteration, and affine access functions, $CS$ is represented as all integral points in a union of polytopes.

Depending on the dependence analysis, $CS$ and $DS$ are super-approximated, let $CS \subseteq C$ and $DS \subseteq D$.

Let $D$ be an approximation of the difference set: $DS \subseteq D$.

$D$ is the set of integral points within a 0-symmetric polytope $K$: $D = \mathring{K}$ (or a body)

**Problem:** Minimize the size of linear allocations valid for $D$ (or $K$).
Previous heuristics: ex. storage in a 2d buffer

[Successive projections — Lefebvre and Feautrier, loop indices]
[Canonical linearizations — De Greef et al., array indices]

For a given index basis
Choice of **appropriate moduli** such that

\[ \sigma(\vec{i}) = \vec{i} \mod \vec{b} = \begin{bmatrix} i_1 \\ i_2 \end{bmatrix} \mod \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \]

or one of the \(2^n n!\) canonical linearization

\[ \sigma(\vec{i}) = \pm Ni_1 \pm i_2 \mod b \quad \text{or} \quad \sigma(\vec{i}) = \pm i_1 \pm Ni_2 \mod b \]

is a valid allocation.
Ex. **σ must be nonzero** on \( \mathcal{D} = \{(0, 1), (1, 1 - N), \ldots\} \)

Largest component along \( e_1 \):

\[
\begin{bmatrix}
1 & 0 \\
\cdot & \cdot \\
\end{bmatrix}
\begin{bmatrix}
1 \\
1 - N \\
\end{bmatrix}
= \begin{bmatrix}
1 \\
\cdot \\
\end{bmatrix}
\]

\[
\Rightarrow \begin{bmatrix}
1 & 0 \\
0 & 1 \\
\end{bmatrix}
\begin{bmatrix}
i_1 \\
i_2 \\
\end{bmatrix}
\bmod \begin{bmatrix} 2 \\
2 \\
\end{bmatrix}
\]

Largest component in the orthogonal:

\[
\begin{bmatrix}
\cdot & \cdot \\
0 & 1 \\
\end{bmatrix}
\begin{bmatrix}
0 \\
1 \\
\end{bmatrix}
= \begin{bmatrix}
0 \\
1 \\
\end{bmatrix}
\]

or best canonical linearization

\[
\max_{\mathcal{D}} \{Ni + j\} = 1 \Rightarrow Ni + j \bmod 2, \quad \text{Size} = 2
\]
Limitations

\[ D = \{(1, 1), (N - 1, N), \ldots\} \]
Limitations

\[ D = \{(1,1), (N-1,N), \ldots \} \]

\[
\begin{bmatrix}
1 & 0 \\
. & . \\
N & 
\end{bmatrix}
\begin{bmatrix}
N-1 \\
N 
\end{bmatrix}
= 
\begin{bmatrix}
N-1 \\
. 
\end{bmatrix}
\]

\[ \Rightarrow \mod N \Rightarrow \text{Size} = N \]

or

\[ \max_D \{| \pm Ni \pm j | \} = \max_D \{| \pm i \pm Nj | \} = N(N-1) \Rightarrow \text{Size} = O(N^2) \]
Our contribution

- **Geometrical framework** for formalizing and studying heuristics
- **Lower and upper bounds on performance** with respect to $D$ and $K$
- **Guaranteed heuristics**, i.e., whose size cannot be “arbitrarily bad”
Outline

Introduction and context

I - Problem statement, and previous heuristic limitations

II - Model: Integral lattices and linear allocations

III - Application: Memory allocation constructions and heuristics

Conclusion
Geometrical interpretation

[Early work on skewing schemes: Budnik and Kuck 1971, Shapiro 78, Wijshoff and Van Leeuwen 1985]

Validity: \( K \cap \ker \sigma = \{ \vec{0} \} \)

Kernel of \( \sigma \): \( \vec{i}, \vec{j} \in \ker \sigma \subset \mathbb{Z}^n \Rightarrow u\vec{i} + v\vec{j} \in \ker \sigma, \ u, v \in \mathbb{Z} \).
Validity $\equiv$ strictly admissible lattice

**Definition:** The lattice $\Lambda = \ker \sigma$ is **strictly admissible** for the polytope $K$ iff

\[ K \cap \Lambda = \{0\} \]
“Good” allocation $\equiv$ “accurate” strictly admissible lattice

Optimal size: 2
Up to equivalence (same kernel),

\[
\sigma : \vec{i} \mapsto U \cdot \vec{i} \mod \vec{s} = \begin{cases} 
    u_{11} i_1 + \ldots + u_{1n} i_n \mod s_1 \\
    \ldots \\
    u_{n1} i_1 \ldots + u_{nn} i_n \mod s_n 
\end{cases}
\]

with \( U \) unimodular and \( \text{diag}(\vec{s}) \) in Smith normal form.
Up to equivalence (same kernel),

\[ \sigma : \vec{i} \mapsto U \cdot \vec{i} \mod \vec{s} = \begin{cases} 
  u_{11}i_1 + \ldots + u_{1n}i_n \mod s_1 \\
  \ldots \\
  u_{n1}i_1 \ldots + u_{nn}i_n \mod s_n 
\end{cases} \]

with \( U \) unimodular and \( \text{diag}(\vec{s}) \) in Smith normal form.

**Storage**

Size = \( s_1s_2\ldots s_n \)

| \( A \) | \( \begin{bmatrix} 2 \\ 4 \\ 4 \\ 8 \end{bmatrix} \mod \) | Size = 256 (dim 4) |

**Underlying lattice \( \Lambda \) (the kernel)**

\[ \det \Lambda = s_1s_2\ldots s_n \]

\[ \Lambda : U^{-1} \begin{bmatrix} s_1 \\ \ldots \\ s_n \end{bmatrix} \]

Integral lattices and linear allocations
\[ \Lambda : \begin{bmatrix} N & 0 \\ 0 & 1 \end{bmatrix}, \quad \det \Lambda = N \]

\[ \Lambda : \begin{bmatrix} 1 - N & 2 \\ -N & 2 \end{bmatrix}, \quad \det \Lambda = 2 \]

Integral lattices and linear allocations
$K$ a 0-symmetric polytope (or a body)

**Problem:** Find a lattice, **integral** and **strictly admissible** for $K$, of **small determinant**

Nota: then, one constructs a valid allocation whose kernel is $\Lambda$ (always possible).
Admissible lattice and lattice packing

Admissible lattice for $K$  ↔  Lattice packing for $K/2$

Integral lattices and linear allocations
**Admissible lattice and lattice packing**

![Diagram of admissible lattice and lattice packing](image)

| Admissible lattice for $K$ | ←→ | Lattice packing for $K/2$ |

**Density** of a lattice packing of $K$:

$$\delta(K, \Lambda) = \frac{\text{Vol}(K)}{\det \Lambda}$$

**Hard question:** densest lattice packing? [Rogers 64, Gruber and Lekkerkerker 87]
The critical determinant of $K$:

$$\Delta(K) = \inf_\Lambda \{ \det \Lambda \mid \Lambda \text{ is admissible for } K \}$$

[Minkowski 1rst Theorem, Minkowski-Hlawka]

$$\frac{\text{Vol}(K)}{2^n} \leq \Delta(K) \leq \text{Vol}(K)$$
The **critical determinant** of $K$:

$$\Delta(K) = \inf_\Lambda \{ \det \Lambda \mid \Lambda \text{ is admissible for } K \}$$

[Minkowski 1st Theorem 1893, Minkowski-Hlawka]

$$\frac{\text{Vol}(K)}{2^n} \leq \Delta(K) \leq \text{Vol}(K)$$

---

**Best memory allocation** (linear):

$$\Delta_Z(K) = \inf_\Lambda \text{integral} \{ \det \Lambda \mid \Lambda \text{ is strictly admissible for } K \}$$

$$\frac{\text{Vol}(K)}{2^n} \leq \Delta_Z(K) \leq ?$$

Integral lattices and linear allocations
Outline

Introduction and context

I - Problem statement, and previous heuristic limitations

II - Model: Integral lattices and linear allocations

III - Application: Memory allocation constructions and heuristics

Conclusion
Scheme I

Input: \( K \)
Output: an integral lattice \( \Lambda \), strictly admissible for \( K \)

1. Start from an integral lattice with basis \( (\vec{c}_1, \ldots, \vec{c}_n) \)

2. 

3. Compute appropriate integer \textbf{scaling factors} \( \rho_i, 1 \leq i \leq n \)

Return the lattice with basis \( (\rho_1 \vec{c}_1, \ldots, \rho_n \vec{c}_n) \)
Scheme I

Input: $K$
Output: an integral lattice $\Lambda$, strictly admissible for $K$, $\det(\Lambda) \leq c_n \text{Vol}(K)$

1. Start from an integral lattice with basis $(\vec{c}_1, \ldots, \vec{c}_n)$
2. “Improve” the basis
3. Compute appropriate integer scaling factors $\rho_i$, $1 \leq i \leq n$

Return the lattice with basis $(\rho_1 \vec{c}_1, \ldots, \rho_n \vec{c}_n)$
Arbitrary basis for the ball
Arbitrary working basis for the ball

Memory allocation constructions and heuristics
Arbitrary working basis for the ball

Memory allocation constructions and heuristics
Arbitrary working basis for the ball

Memory allocation constructions and heuristics
Arbitrary basis for a polytope
Arbitrary working basis for a polytope

Memory allocation constructions and heuristics
Arbitrary working basis for a polytope

"Depth" of the second projection

$F_2 (c_2) = 0.14$

Memory allocation constructions and heuristics
**Definition: \(i\)th “depth”** [Lovász and Scarf 1992]

- \( F(\vec{c}) = \inf \{ \rho > 0 \mid \vec{c} \in \rho K \} \)

- \( F_i(\vec{c}_i) = \inf \{ F(\vec{x}) \mid \vec{x} \in \vec{c}_i + \text{Vect}(\vec{c}_1, \ldots, \vec{c}_{i-1}) \} \)

or, working in the dual \(K^*\) of \(K\),

**Definition: \(i\)th “width”**

- \( F^*_i(\vec{c}_i) = \sup \{ \vec{c}_i \cdot \vec{y} \mid \vec{y} \in K, \vec{y} \cdot \vec{c}_1 = \vec{y} \cdot \vec{c}_2 = \ldots = \vec{y} \cdot \vec{c}_{i-1} = 0 \} \)
Memory allocation constructions and heuristics
Memory allocation constructions and heuristics
Arbitrary working basis for a polytope

Memory allocation constructions and heuristics
What’s wrong with the working basis?
The determinant of the output basis is related to

\[ \prod_{i=1}^{n} \rho_i \approx \prod_{i=1}^{n} \frac{1}{F_i(\tilde{c}_i)}, \]

hence, for upper bounding the determinant,

\[ \prod_{i=1}^{n} F_i(\tilde{c}_i) \geq ? \]
The determinant of the output basis is related to

$$\prod_{i=1}^{n} \rho_i \approx \prod_{i=1}^{n} \frac{1}{F_i(\vec{c}_i)},$$

hence, for upper bounding the determinant,

$$\prod_{i=1}^{n} F_i(\vec{c}_i) \geq ?$$

⇒ use a \textbf{generalized reduced basis} [Lovász and Scarf 92] with

$$F_i(\vec{c}_i) \geq \lambda_i(K)\left(\frac{1}{2} - \epsilon\right)^{i-1}$$

and for the successive minima $\lambda_i(K)$, use the Second Theorem of Minkowski.
Generalized lattice basis reduction

Memory allocation constructions and heuristics
Reduce $c_2$ using $c_1$
New norm $\sim 0.4$

Generalized lattice basis reduction
Generalized lattice basis reduction

Memory allocation constructions and heuristics
Generalized lattice basis reduction

$c_2$ is small compared to $c_1$

swap $c_1$ and $c_2$

Memory allocation constructions and heuristics
Generalized lattice basis reduction

Memory allocation constructions and heuristics
Application to memory allocations

1. Better understanding of previous heuristics
   Based on “fixed” bases (loops, arrays, schedule, . . .)
   \(\Rightarrow\) may fail if the basis is not adequate with respect to \(DS\)

2. Upper bound for the strictly admissible determinant \(\Delta_Z\)

3. Provides heuristics with guaranteed size
Improved basis for a polytope
Improved basis for a polytope

Memory allocation constructions and heuristics
Improved basis for a polytope

Memory allocation constructions and heuristics
Improved basis for a polytope

Memory allocation constructions and heuristics
Improved basis for a polytope

Memory allocation constructions and heuristics
For a given $K$, the critical determinant ($\Lambda \subseteq \mathbb{R}^n$) satisfies [Minkowski-Hlawka]

$$\Delta(K) \leq \text{Vol}(K)$$

**Scheme II**

Using the **successive minima** of $K$ we establish that there exists a **strictly admissible and integer lattice** such that

$$\Delta_{\mathbb{Z}}(K) \leq n! \text{ Vol}(K)$$
**Guaranteed heuristics**

Full dimensional polytope, arbitrary set in some cases

\[ \det \Lambda \leq c_n \text{Vol}(K) \]

Enumeration, \( \Lambda \) such that \( \det(\Lambda) \leq n! \text{Vol}(K) \)

Using the successive minima (Scheme II) (adapting [Rogers])

Based on \( K \) (Scheme I, \( F_i(\vec{a}_i) \leq 1 \))

Generalized reduction (Scheme I)

Based on \( K^* \) (Scheme I, \( F_i^*(\vec{c}_i) \leq 1 \)) (cf [Lefebvre and Feautrier])

Lenstra-Lenstra-Lovász reduction (ellipsoid approximation)

+ 1D allocations, and power of two moduli

Optimal linear

\[ c_n = n! \]

\[ c_n = (n!)^2 \]

\[ c_n = 2^{n^2} n! \]

\[ c_n = (n!)^2 \]

\[ c_n = 2^{n(n+3)/4} n^n \]

Memory allocation constructions and heuristics
Cf Limitations

Previous heuristics: size $O(N)$ or $O(N^2)$

Guaranteed heuristics, $n = 2$:

Size $= \text{det } \Lambda \leq 2 \text{ Vol}(K) = 4$. 

Memory allocation constructions and heuristics
Outline

Introduction and context

I - Problem statement, and previous heuristic limitations

II - Model: Integral lattices and linear allocations

III - Application: Memory allocation constructions and heuristics

Conclusion
**In practice**

Performance is guaranteed as soon as the basis is appropriate w.r.t \( K \)

- access functions to arrays are “simple”
- scheduling functions are not “too degenerated”
- writing domains are “not too skewed”

\( \Rightarrow \) Mixing Lefebvre-Feautrier and Quilleré-Rajopadhye (schedule basis)

**Computational aspects**

Integer matrix manipulation for enumerative construction
Generalized basis reduction (Linear Programming)
Integer Linear Programming

Discussion and open questions
Questions

Another approach for obtaining integral and strictly admissible lattices?

Power of linear allocations with respect to the optimum?

More general allocations, e.g. multi-periodic schemes?

More general conflicting indices set?