High-Performance Embedded Systems-on-a-Chip

Sanjay Rajopadhye
Computer Science, Colorado State University

Lecture 2: Birds-eye view (contd.)
Buzzwords

- Systolic Arrays
- Affine Control Loops
- Recurrence Equations
- Polyhedra
- Alpha
- Tiling
Source Programs: Affine Control Loops

- **loop bounds:**
  - affine expressions of surrounding loops;
  - or max/min of such expressions.

- **loop body:**
  - a loop;
  - an assignment statement;
  - a sequence of (any of) these two.

- **Variables:** index-vars or params, or (multidimensional) arrays

- **Assignment statements:**
  - **lhs** \(\Rightarrow\) array variable, accessed by affine function of indices;
  - **rhs** \(\Rightarrow\) expression containing such (indexed) array vars.
Example: Forward Substitution

S1: \( x[1] := \frac{b[1]}{a[1,1]}; \)
for \( i = 2 \) to \( n \) do
S2: \( s := 0; \)
for \( j = 1 \) to \( i-1 \) do
S3: \( s := s + x[j] \times a[i, j]; \)
enddo
S4: \( x[i] := \frac{(b[i] - s)}{a[i,i]} \)
enddo
Recurrence Equations: Definition

\[ X[z] = \{ \forall z \in D \} : g(\ldots X[f(z)]\ldots) \]
Recurrence Equations: Definition

\[ X(z) = \{ \forall z \in \mathcal{D} : g(\ldots X[f(z)]\ldots) \} \]

- \( X \) \( n \)-dimensional data variable;
Recurrence Equations: Definition

\[ X[z] = \{ \forall z \in \mathcal{D} : g(\ldots X[f(z)]\ldots) \} \]

- \( X \) \( n \)-dimensional data variable;
- \( z \) \( n \)-dimensional index variable;
Recurrence Equations: Definition

\[ X[z] = \{ \forall z \in D \} : g(... X[f(z)] ... ) \]

- \( X \) \n-dimensional data variable;
- \( z \) \n-dimensional index variable;
- \( g \) strict, ("atomic") computation function;
Recurrence Equations: Definition

\[ X[z] = \{ \forall z \in D \} : g(\ldots X[f(z)] \ldots) \]

- **X** \( n \)-dimensional data variable;
- **z** \( n \)-dimensional index variable;
- **g** strict, (“atomic”) computation function;
- **f(z)** dependency function \( f : \mathbb{Z}^n \rightarrow \mathbb{Z}^n \);
Recurrence Equations: Definition

\[ X[z] = \{ \forall z \in \mathcal{D} : g(\ldots X[f(z)]\ldots) \} \]

- \( X \) \( n \)-dimensional data variable;
- \( z \) \( n \)-dimensional index variable;
- \( g \) strict, (“atomic”) computation function;
- \( f(z) \) dependency function \( f : \mathbb{Z}^n \rightarrow \mathbb{Z}^n \);
- “…” other such arguments;
Recurrence Equations: Definition

\[ X[z] = \{ \forall z \in D \} : g(\ldots X[f(z)] \ldots) \]

- **X** \( n \)-dimensional data variable;
- **z** \( n \)-dimensional index variable;
- **g** strict, (“atomic”) computation function;
- **f(z)** dependency function \( f : \mathbb{Z}^n \rightarrow \mathbb{Z}^n \);
- “…” other such arguments;
- **D \subseteq \mathbb{Z}^n** domain of the equation.
Recurrence Equations: Taxonomy
Recurrence Equations: Taxonomy

- **Uniform Recurrence Equations (URE’s):** $f(z)$ has the form $z + \delta$
- **Affine Recurrence Equations (ARE’s):** $f(z)$ is an affine function, $Az + a$. 
Recurrence Equations: Taxonomy

- **Uniform Recurrence Equations (URE’s):** $f(z)$ has the form $z + \delta$

- **Affine Recurrence Equations (ARE’s):** $f(z)$ is an affine function, $Az + a$.

- **System of Recurrence Equations (SRE’s):** set of (mutually recursive) equations. Could be uniform or affine (SURE or SARE).
Recurrence Equations: Taxonomy

- **Uniform Recurrence Equations (URE’s):** \( f(z) \) has the form \( z + \delta \)

- **Affine Recurrence Equations (ARE’s):** \( f(z) \) is an affine function, \( Az + a \).

- **System of Recurrence Equations (SRE’s):** set of (mutually recursive) equations. Could be uniform or affine (SURE or SARE).

- **Reductions:** associative/commutative operators applied to collections of data values
Parameterized Polyhedral Domains
Parameterized Polyhedral Domains

- Integral Polyhedron: points satisfying a finite number of linear (in)equality constraints

\[ \{ z \in \mathbb{Z}^n \mid Qz \geq q \} \]
Parameterized Polyhedral Domains

- **Integral Polyhedron**: points satisfying a finite number of linear (in)equality constraints
  \[ \{ z \in \mathbb{Z}^n \mid Qz \geq q \} \]

- **Parameterized family**: constraints involve size parameters
  \[ \{ z \in \mathbb{Z}^n \mid Qz \geq q - Pp \} \]

Alternatively:

\[ \begin{pmatrix} z \\ p \end{pmatrix} \in \mathbb{Z}^{n+m} \mid \begin{bmatrix} Q & P \end{bmatrix} \begin{pmatrix} z \\ p \end{pmatrix} \geq q \]
Parameterized Polyhedral Domains

- **Integral Polyhedron**: points satisfying a finite number of linear (in)equality constraints
  \[ \{ z \in \mathbb{Z}^n \mid Qz \geq q \} \]

- **Parameterized family**: constraints involve size parameters \( \{ z \in \mathbb{Z}^n \mid Qz \geq q - Pp \} \).
  Alternatively:
  \[
  \left\{ \begin{pmatrix} z \\ p \end{pmatrix} \in \mathbb{Z}^{n+m} \mid \begin{pmatrix} Q & P \end{pmatrix} \begin{pmatrix} z \\ p \end{pmatrix} \geq q \right\}
  \]

- **Dual Representation**: In terms of
♦ **constraints:** linear (in)equalities
♦ **constraints:** linear (in)equalities

♦ **or generators:** vertices (& rays)

\[
\{ z \in \mathbb{Z}^n \mid z = a^T G; \sum_i a_i = 1 \}
\]
Change of Basis Transformation

Given an SRE

\[
U[z] = z \in D^u : g_u( U[f_{uu}(z)]), \\
V[f_{uv}(z)], \ldots
\]

\[
V[z] = z \in D^v : g_v( U[f_{vu}(z)]), \\
V[f_{vv}(z)], \ldots
\]
Change of Basis Transformation

Given an SRE

\[
U[z] = z \in D^u : g_u(U[f_{uu}(z)], V[f_{uv}(z)], \ldots)
\]

\[
V[z] = z \in D^v : g_v(U[f_{vu}(z)], V[f_{vv}(z)], \ldots)
\]

and a function, \( \mathcal{T} \), mapping original to new indices, construct an equivalent SRE
Change of Basis

$D^u$ $b$ $a$ $c$ $f_{uu}$ $f_{vu}$

$D^v$ $l$ $n$ $m$ $f_{vv}$
Change of Basis

\[ T(D^u) \]
Transformed SRE

\[\begin{align*}
U[z] &= z \in \mathcal{T}(D^u) : g_u( U[\mathcal{T} \circ f_{uu} \circ \mathcal{T}'(z)], \\
&\quad V[f_{uu} \circ \mathcal{T}'(z)], \ldots ) \\
V[z] &= z \in D^v : g_v( U[\mathcal{T} \circ f_{vv}(z)], \\
&\quad V[f_{vv}(z)], \ldots )
\end{align*}\]
Closure Properties

Domains \equiv \text{Abstract Data Type (ADT)}, closed under:

- Intersection
- Union
- Preimage by the class of dependence functions
- Image by the class COB transformations

Also

- Transformations \subseteq \text{Dependence functions}
- Dependence functions closed under composition
Partition the domains (iteration spaces of loops) into tiles (or blocks or supernodes)

Tiles are usually (hyper) parallelepiped shaped

Used in many contexts:
Tiling/Partitioning/Loop Blocking

Partition the domains (iteration spaces of loops) into tiles (or blocks or supernodes).

Tiles are usually (hyper) parallelepiped shaped.

Used in many contexts:

- granularity on parallel machines
Tiling/Partitioning/Loop Blocking

Partition the domains (iteration spaces of loops) into tiles (or blocks or supernodes).

Tiles are usually (hyper) parallelepiped shaped.

Used in many contexts:

- granularity on parallel machines
- locality optimization for caches
Tiling/Partitioning/Loop Blocking

Partition the domains (iteration spaces of loops) into tiles (or blocks or supernodes).

Tiles are usually (hyper) parallelepiped shaped.

Used in many contexts:

- granularity on parallel machines
- locality optimization for caches
- bandwidth adaptation for FPGA co-processors
Tiling/Partitioning/Loop Blocking

Partition the domains (iteration spaces of loops) into tiles (or blocks or supernodes)

Tiles are usually (hyper) parallelepiped shaped

Used in many contexts:

- granularity on parallel machines
- locality optimization for caches
- bandwidth adaptation for FPGA co-processors
- power reduction in embedded systems.
Optimal Tiling Problem

Optimally choose the tile parameters:
Optimal Tiling Problem

Optimally choose the tile parameters:

- **Shape**: tile hyperplane normal vectors
Optimal Tiling Problem

Optimally choose the tile parameters:

- **Shape**: tile hyperplane normal vectors
- **Form**: hyper parallelepiped aspect ratio
Optimal Tiling Problem

Optimally choose the tile parameters:

- **Shape**: tile hyperplane normal vectors
- **Form**: hyper parallelepiped aspect ratio
- **Size**: hyper parallelepiped dimensions
Optimal Tiling Problem

Optimally choose the tile parameters:

- **Shape**: tile hyperplane normal vectors
- **Form**: hyper parallelepiped aspect ratio
- **Size**: hyper parallelepiped dimensions

Tile Sizing Problem: given the shape, determine the size to optimize a cost measure (running time)
Previous Work

Previous Work

- Xue (1997) foundations and extensions
Previous Work

  definitions and foundations

- Xue (1997)
  foundations and extensions

  shape optimization
Previous Work

- Xue (1997) foundations and extensions
- Hodzic & Shang (1996) size and shape optimization
Previous Work

- Xue (1997) foundations and extensions
- Hodzic & Shang (1996) size and shape optimization
Previous Work

- Xue (1997) - foundations and extensions
- Hodzic & Shang (1996) - size and shape optimization

Tile Sizing: Assuming tiles parallel to boundaries, choose tile size to minimize running time
Previous Work


- Xue (1997) foundations and extensions

- Boulet et al. (1994), Calland & Risset (1995) shape optimization

- Hodzic & Shang (1996) size and shape optimization


Tile Sizing: Assuming tiles parallel to boundaries, choose tile size to minimize running time

- King, Chou & Ni (1990) 2-D, square
Previous Work

- Xue (1997) foundations and extensions
- Hodzic & Shang (1996) size and shape optimization
- Dagstuhl Seminar (1998) [www.dagstuhl.de/DATA/Reports/98341](http://www.dagstuhl.de/DATA/Reports/98341)

Tile Sizing: Assuming tiles parallel to boundaries, choose tile size to minimize running time

- King, Chou & Ni (1990) 2-D, square
Our Results

- Andonov, et. al (2000, 2001) 2-D oblique
- Derrien & Rajopadhye (2000) tiling for FPGA’s
- Derrien & Rajopadhye (2001) tiling for power
Overview of the approach

- Build the tile graph
- Map to a $p$-processor parallel machine
- Determine running time with abstract parameters period, $\mathcal{P}$ and latency, $\mathcal{L}$
- Instantiate $\mathcal{P}$ and $\mathcal{L}$ with machine-specific model
- Solve discrete nonlinear optimization problem
- Experimental Validation