

# High-Performance Embedded Systems-on-a-Chip

Sanjay Rajopadhye  
Computer Science, Colorado State  
University

Lecture 9: I/O & Control; Alpha (1)

## Input/Output of Systolic Arrays

- Main Rule: I/O must occur at **boundary processors**.

## Input/Output of Systolic Arrays

- Main Rule: I/O must occur at boundary processors.
- In the original SURE

## Input/Output of Systolic Arrays

- Main Rule: I/O must occur at boundary processors.
- In the original SURE ,
  - I/O variables have their own domain: usually, at least one dimension less than computation variables (compute bound applications)
  - boundary  $\equiv$  boundary of polyhedron defining the computation domain.
- In the Transformed SURE, boundary processors  $\equiv$  boundary of image, by the allocation function of the computation domain.

## Where does the I/O occur

In original SURE:

- Inputs  $\equiv$  first (only?) use of an input variable.
- Outputs  $\equiv$  RHS expression of the output vars (i.e., some subdomain of computation domain)

In the final (transformed) SURE: I/O is the image of these (sub) domains by the CoB.

## Handling I/O & Control Signals

- Ensure that this happens at boundary PEs.
- If not, add auxiliary variables (similar to localization)
- Change of PE behavior occurs at a **boundary plane**

## Handling I/O & Control Signals

- Ensure that this happens at boundary PEs.
- If not, add auxiliary variables (similar to localization)
- Change of PE behavior occurs at a **boundary plane**, i.e., linear (in)equality of the indices.
- Control signal generation  $\equiv$  localization

# Outline

- Programming with SAREs+Reductions
- Convolution
- Forward Substitution
- LU Decomposition
- Cholesky Factorization
- Algebraic Path Problem
- Knapsack Problem(s)

# Advantages & Disadvantages

## Pros:

- Very high level of programming
- Functional
- Amenable to formal manipulation, verification, etc.

# Advantages & Disadvantages

## Pros:

- Very high level of programming
- Functional
- Amenable to formal manipulation, verification, etc.

## Cons:

- Functional

# Advantages & Disadvantages

## Pros:

- Very high level of programming
- Functional
- Amenable to formal manipulation, verification, etc.

## Cons:

- Functional
- Restrictive
- Inefficient

# High Level Programming (examples)

*n*-point convolution of a sequence:

$$i = \sum_j w_j x_{i-j}$$

## Forward Substitution

Given an  $n \times n$  lower triangular matrix,  $L$ , and an  $n$ -vector,  $b$ , solve the equation  $Lx = b$ .

By definition,

$$\begin{aligned} b_i &= \sum_{j=1}^n L_{i,j} x_j = \sum_{j=1}^i L_{i,j} x_j \\ &= \begin{cases} i = 1 & \Rightarrow L_{1,1} x_1 \\ i > 1 & \Rightarrow L_{i,i} x_i + \sum_{j=1}^{i-1} L_{i,j} x_j \end{cases} \end{aligned}$$

Hence,

$$x_i = \begin{cases} i = 1 & \Rightarrow b_1 / L_{1,1} \\ i > 1 & \Rightarrow \frac{b_i - \sum_{j=1}^{i-1} L_{i,j}}{L_{i,i}} \end{cases}$$

# LU Decomposition

Given an  $n \times n$  square matrix,  $A$ , find its factorization into  $A = LU$ , where  $L$  is (strictly) lower-triangular (with unit diagonal) and  $U$  is upper-triangular.

# LU Decomposition

Given an  $n \times n$  square matrix,  $A$ , find its factorization into  $A = LU$ , where  $L$  is (strictly) lower-triangular (with unit diagonal) and  $U$  is upper-triangular. By definition,

$$A_{i,j} = \sum_{k=1}^n L_{i,k} U_{k,j}$$

# LU Decomposition

Given an  $n \times n$  square matrix,  $A$ , find its factorization into  $A = LU$ , where  $L$  is (strictly) lower-triangular (with unit diagonal) and  $U$  is upper-triangular. By definition,

$$A_{i,j} = \sum_{k=1}^n L_{i,k} U_{k,j} = \sum_{k=1}^{\min(i,j)} L_{i,k} U_{k,j}$$

# LU Decomposition

Given an  $n \times n$  square matrix,  $A$ , find its factorization into  $A = LU$ , where  $L$  is (strictly) lower-triangular (with unit diagonal) and  $U$  is upper-triangular. By definition,

$$\begin{aligned}
 A_{i,j} &= \sum_{k=1}^n L_{i,k} U_{k,j} = \sum_{k=1}^{\min(i,j)} L_{i,k} U_{k,j} \\
 &= \begin{cases} i \leq j \Rightarrow \sum_{k=1}^i L_{i,k} U_{k,j} \\ j < i \Rightarrow \sum_{k=1}^j L_{i,k} U_{k,j} \end{cases}
 \end{aligned}$$

$$= \left\{ \begin{array}{ll} 1 = i \leq j & \Rightarrow L_{i,i}U_{i,j} \\ 1 < i \leq j & \Rightarrow L_{i,i}U_{i,j} + \sum_{k=1}^{i-1} L_{i,k}U_{k,j} \\ 1 = j < i & \Rightarrow L_{i,j}U_{j,j} \\ 1 < j < i & \Rightarrow L_{i,j}U_{j,j} + \sum_{k=1}^{j-1} L_{i,k}U_{k,j} \end{array} \right.$$

Hence,

$$U_{i,j} = \begin{cases} 1 = i \leq j & \Rightarrow A_{1,j} \\ 1 < i \leq j & \Rightarrow A_{i,j} - \sum_{k=1}^{i-1} L_{i,k} U_{k,j} \end{cases}$$

$$L_{i,j} = \begin{cases} 1 = j < i & \Rightarrow A_{i,j} / U_{j,j} \\ 1 < j < i & \Rightarrow \left( A_{i,j} - \sum_{k=1}^{j-1} L_{i,k} U_{k,j} \right) / U_{j,j} \end{cases}$$

# Cholesky Factorization

Given an  $n \times n$  **symmetric positive semi-definite** matrix,  $A$ , factorize it into  $A = LL^T$ , where  $L$  is lower-triangular.

**Definition** (of positive semi-definite):  $x \neq 0 \Rightarrow x^t Ax \geq 0$

**Important property:** Principal submatrices of positive semi-definite matrices are also positive semi-definite

**Theorem:** If  $A$  is positive semi-definite, there exists a unique lower-triangular matrix  $L$  (with positive diagonal elements) such that  $A = LL^T$

**Proof:** By induction:

- If  $A = [a]$  then  $L = [\sqrt{a}]$

- Let theorem hold for matrices of order  $n - 1$  and let

$A' = \begin{pmatrix} A & a \\ a^T & \alpha \end{pmatrix}$  be a symmetric positive semi-definite matrix of order  $n$  ( $A$ , of order  $n$ , is itself positive semi-definite).

We seek to factorize  $A'$  as  $A' = L' L'^T$ ;  $L' = \begin{pmatrix} L & 0 \\ l^T & \lambda \end{pmatrix}$ . So,

$$\begin{pmatrix} A & a \\ a^T & \alpha \end{pmatrix} = \begin{pmatrix} L & 0 \\ l^T & \lambda \end{pmatrix} \begin{pmatrix} L^T & l \\ 0 & \lambda \end{pmatrix}. \text{ Hence}$$

$$A = LL^T$$

$$a = Ll$$

$$a^T = l^T L^T$$

$$\alpha = l^T l + \lambda^2$$

# Cholesky Factorization Equations

$$L_{i,j} = \begin{cases} i = j = 1 & \Rightarrow \sqrt{A_{i,j}} \\ 1 = j < i \leq n & \Rightarrow A_{i,j} / L_{j,j} \\ 1 < j < i \leq n & \Rightarrow \left( A_{i,j} - \sum_{k=1}^{j-1} L_{i,k} L_{j,k} \right) / L_{j,j} \\ 1 < j = i \leq n & \Rightarrow \sqrt{a_{i,i} - \sum_{k=1}^{i-1} l_{i,k}^2} \end{cases}$$

**Why can't such equations be programs?**

**Why can't such equations be programs?**

Thank God for Alpha