High-Performance Embedded Systems-on-a-Chip

Lecture 17: Scheduling

Sanjay Rajopadhye

Computer Science, Colorado State University

Limitations of Systolic Arrays

Only a (very small) proper subset of SAREs:
Those that

- are Serializable,
- are Localizable,
- correspnod to a Single Equation, and
- admit a One-dimensional schedule.

Question: What is beyond systolic arrays?

(Silicon) Compilation

- For each point in domain of each variable, determine:
  - A time instant $\Rightarrow$ schedule
  - A place $\Rightarrow$ processor and memory allocation
- Transform P-SARE so that indices denote either
  - time,
  - processor, or
  - memory address
- Generate code (or HDL, we hope)

Two Orthogonal Issues

- Static Analysis: what transformation to apply
  - scheduling
  - processor (& memory) allocation
- Program Transformation: manipulating the SARE
  - Rules to modify the SARE (Change of Basis)
  - Code Generation (how to interpret the transformed SARE)

Golden Rule of Static Analysis

The dependence graph cannot be explicitly constructed

- Too large
- Not (fully) known at compile time – parameters
- Explicitly constructed results are not useful

Implication: use compact information (reduced dependence graph)

Key Problem: Scheduling

- Definition: A function t such that whenever $U[x]$ depends on $V[y]$, then $t(U,x) > t(V,y)$.
- Affine schedules:

$$\begin{aligned} t(U,x) &\equiv \lambda^U x + \alpha^U \\ &= \lambda_1^U x_1 + \ldots + \lambda_n^U x_n + \alpha^U \end{aligned}$$

Geometric interpretation: all points executed at time $t_0 = \lambda^U z + \alpha^U$ belong to isotemporal hyperplane with normal vector λ^U

Scheduling a (single) URE

$$V[z] = \{z \in D\} \ : \ f(V[z+d_1], \ldots, V[z+d_s]$$

- $\langle \lambda, \alpha \rangle$ is valid iff for $k = 1 \ldots s$, and $\forall z \in \mathcal{D}$

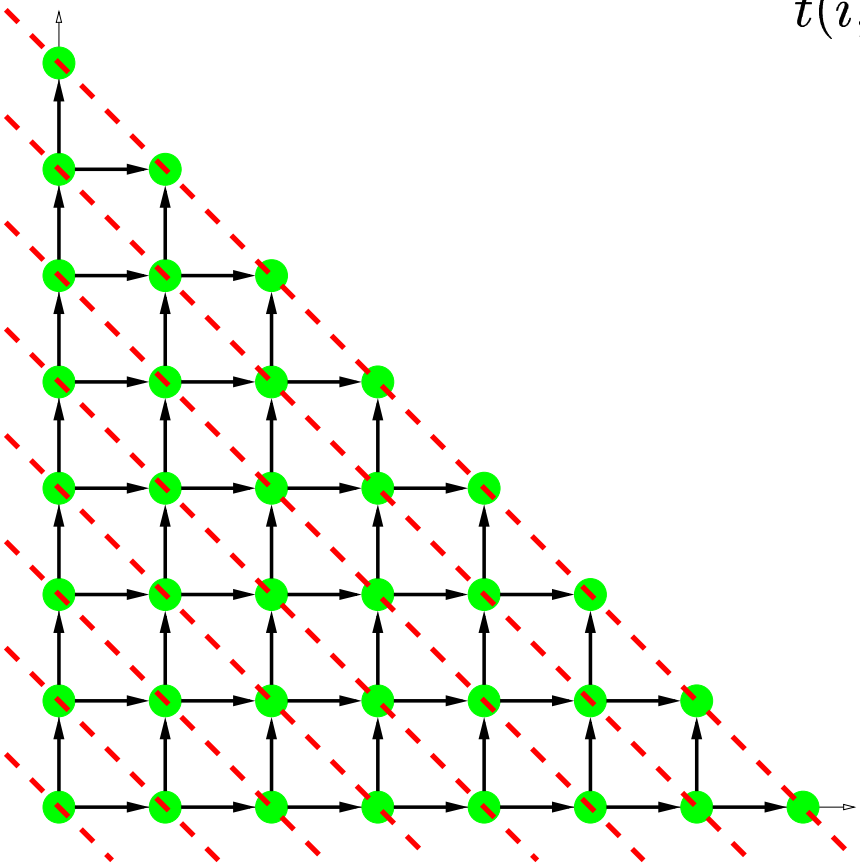
$$\begin{aligned} \lambda z + \alpha &> \lambda(z + d_k) + \alpha \\ &= \lambda z + \lambda d_k \end{aligned}$$

i.e., $\lambda d_k < 0$

- Finite number of constraints, independent of domain size. Scheduling $\equiv$ Linear Programming

  Geometric view: Choose the hyperplanes so that dependences point backwards

Example



$$X[i,j] = g(X[i-1,j], X[i,j-1]$$
$$t(i,j) \equiv ai + bj + \alpha$$

Schedule validity conditions

$$[a,b][0,-1]^T < 0$$
$$[a,b][-1,0]^T < 0$$
$$\alpha \geq 0$$

i.e., $\{\mathtt{a}, \mathtt{b}, \alpha \mid \mathtt{a}, \mathtt{b} > 0, \alpha \geq 0\}$

Optimal schedule: $t(i,j) = i + j$

Scheduling an SURE

- Single schedule for all variables
  Not general enough: some well defined SURE's don't admit such a schedule (e.g. the convolution example)
- Shifted linear schedules
  Allow the α^U to be different for each variable, U, but same λ
- Variable dependent schedules: different slopes for different variables)
- Multidimensional schedules

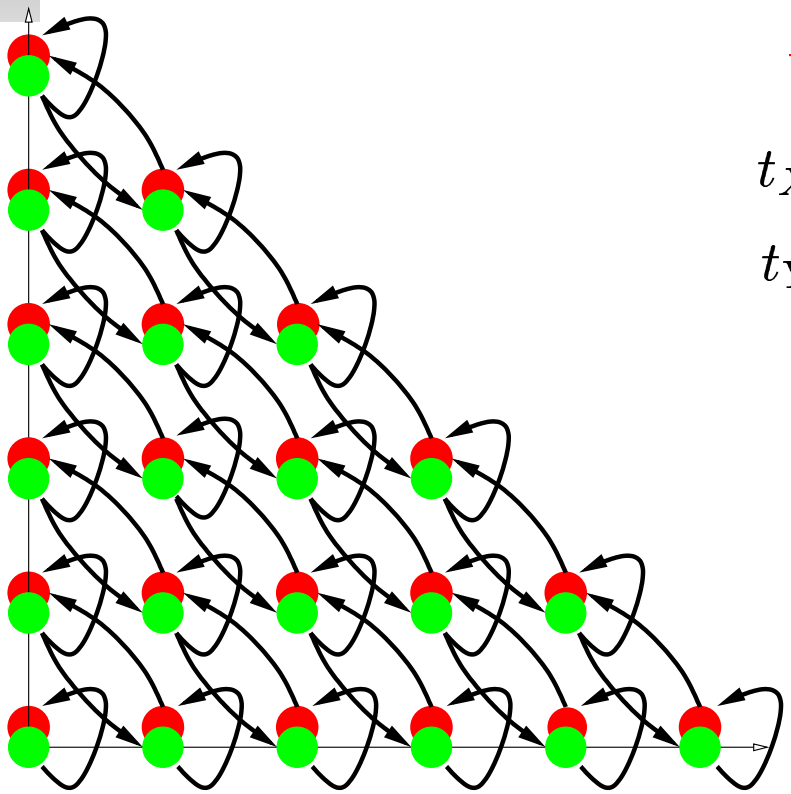
Limits of shifted linear schedules

$$X[i,j] = g(X[i-1,j+1])$$

$$Y[i,j] = h(Y[i+1,j-1])$$

This SURE cannot be scheduled with same-slope lines for both X and Y.

A less contrived example

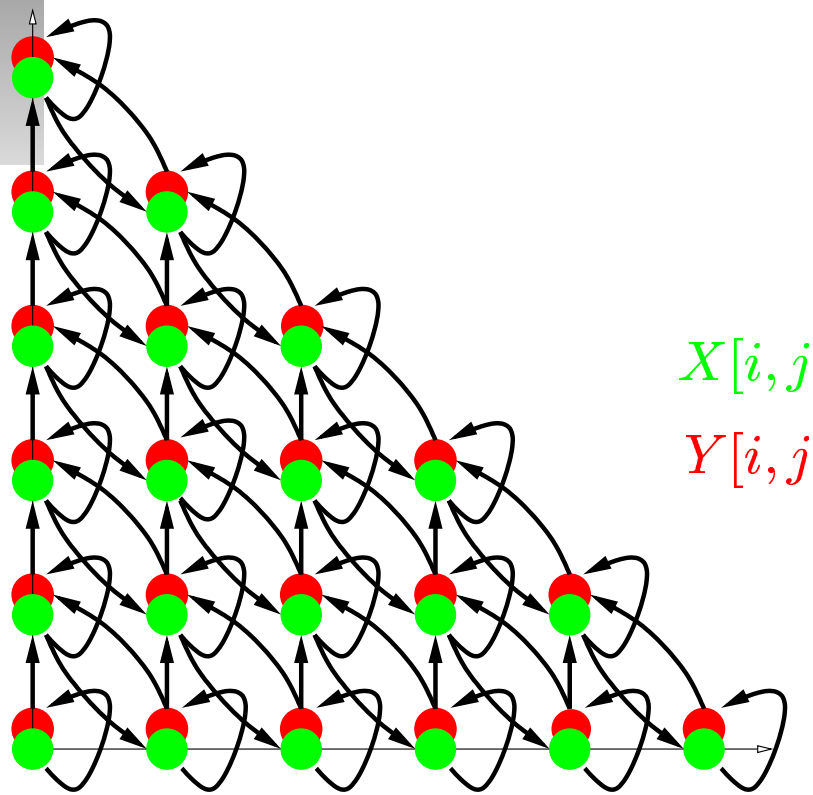


$$
\begin{aligned}
X[i,j] &= g(X[i-1,j+1] \\
Y[i,j] &= h(Y[i+1,j-1], X[i,j] \\
t_X(i,j) &= a_X i + b_X j + \alpha_X \\
t_Y(i,j) &= a_Y i + b_Y j + \alpha_Y
\end{aligned}
$$

Optimal solution

$$
\begin{aligned}
t_X(i,j) &= i \\
t_Y(i,j) &= i + 2j + 1
\end{aligned}
$$

Exercise



$$X[i,j] = f(X[i-1,j+1], Y[i,j-1])$$
$$Y[i,j] = g(X[i,j], Y[i+1,j-1])$$

Find length of longest path reaching the green (cf. red) node at $[i,j]$