

High-Performance Embedded Systems-on-a-Chip

Sanjay Rajopadhye
Computer Science, Colorado State
University

Lecture 6: GKT Array (concl.)

Problem Specification

Compute $C[1, n + 1]$, where, for $1 \leq i < j \leq n + 1$

$$C[i, j] = \begin{cases} i + 1 = j : w_{i,j} \\ i + 1 < j : w_{i,j} + \min_{i < k < j} (C[i, k] + C[k, j]) \end{cases}$$

Observations/Questions

- Total volume of computation: $\approx \frac{1}{6}n^3$
- Total I/O volume : $\approx \frac{1}{2}n^2$ (actually n^2)
- Triangular array: PE $[i, j]$ computes $C[i, j]$
- What values are used to compute a given $C[i, j]$?
- Where is a given $C[i, j]$ used?
- With whom is a given $C[i, j]$ “combined”?
- Where is it combined?

How the array works

- $C[i, j]$ is computed at time $2(j - i)$
- It is then sent horizontally and vertically, travelling at a rate of one PE per cycle for exactly $j - i$ more cycles.
- Afterwards, it travels at a slower rate of one PE every *two* cycles.
- So every value meets exactly the right “partner” at the right place.

How the array works

- $C[i, j]$ is computed at time $2(j - i)$
- It is then sent horizontally and vertically, travelling at a rate of one PE per cycle for exactly $j - i$ more cycles.
- Afterwards, it travels at a slower rate of one PE every *two* cycles.
- So every value meets exactly the right “partner” at the right place.
- But *why on earth* does it work

How the array works

- $C[i, j]$ is computed at time $2(j - i)$
- It is then sent horizontally and vertically, travelling at a rate of one PE per cycle for exactly $j - i$ more cycles.
- Afterwards, it travels at a slower rate of one PE every *two* cycles.
- So every value meets exactly the right “partner” at the right place.
- But ***why on earth*** does it work and ***how on earth*** did they design it?

(Quadratic) Scheduling

- If reductions (of \min) can be computed *instantaneously*

(Quadratic) Scheduling

- If reductions (of min) can be computed *instantaneously* $t(i, j) = j - i$.

(Quadratic) Scheduling

- If reductions (of min) can be computed ***instantaneously*** $t(i, j) = j - i$. But realistically, reduction for $C[i, j]$ needs $\Theta(j - i)$ steps

(Quadratic) Scheduling

- If reductions (of min) can be computed **instantaneously** $t(i, j) = j - i$. But realistically, reduction for $C[i, j]$ needs $\Theta(j - i)$ steps
- So when can each reduction **start**?

(Quadratic) Scheduling

- If reductions (of min) can be computed *instantaneously* $t(i, j) = j - i$. But realistically, reduction for $C[i, j]$ needs $\Theta(j - i)$ steps
- So when can each reduction *start*? Oops :-)

(Quadratic) Scheduling

- If reductions (of min) can be computed **instantaneously** $t(i, j) = j - i$. But realistically, reduction for $C[i, j]$ needs $\Theta(j - i)$ steps
- So when can each reduction **start**? Oops :-(
- What is the **serialization order**: $k = i + 1$ to $j - 1$

(Quadratic) Scheduling

- If reductions (of min) can be computed **instantaneously** $t(i, j) = j - i$. But realistically, reduction for $C[i, j]$ needs $\Theta(j - i)$ steps
- So when can each reduction **start**? Oops :-(
- What is the **serialization order**: $k = i + 1$ to $j - 1$
 or $k = j - 1$ down to $i + 1$?

(Quadratic) Scheduling

- If reductions (of min) can be computed **instantaneously** $t(i, j) = j - i$. But realistically, reduction for $C[i, j]$ needs $\Theta(j - i)$ steps
- So when can each reduction **start**? Oops :-(
- What is the **serialization order**: $k = i + 1$ to $j - 1$ or $k = j - 1$ down to $i + 1$? In either case, **quadratic schedule**.

Affine Schedule

- Solution: **Middle serialization** (let $\Delta = j - i$):

$$C_{i,j} = \min_{k=1}^{\Delta/2} ((C_{i,i+k} + C_{i+k,j}), (C_{i,j-k} + C_{j-k,j}))$$

- Serialize along **decreasing** k . Computation of $C_{i,j}$ **takes** $\Delta/2$ (twice as expensive) steps.

Affine Schedule

- Solution: **Middle serialization** (let $\Delta = j - i$):

$$C_{i,j} = \min_{k=1}^{\Delta/2} ((C_{i,i+k} + C_{i+k,j}), (C_{i,j-k} + C_{j-k,j}))$$

- Serialize along **decreasing** k . Computation of $C_{i,j}$ **takes** $\Delta/2$ (twice as expensive) steps.
- $C_{i,j}$ is computed at $2\Delta = 2(j - i)$.
- Two (or four) fold slowdown, but **affine** schedule.

Validity of the schedule

$$C_{i,j} = \min_{k=1}^{\Delta/2} ((C_{i,i+k} + C_{i+k,j}), (C_{i,j-k} + C_{j-k,j}))$$

- Computation of $C_{i,j}$ **must start** at 1.5Δ

Validity of the schedule

$$C_{i,j} = \min_{k=1}^{\Delta/2} ((C_{i,i+k} + C_{i+k,j}), (C_{i,j-k} + C_{j-k,j}))$$

- Computation of $C_{i,j}$ **must start** at 1.5Δ
- Its k -th step

Validity of the schedule

$$C_{i,j} = \min_{k=1}^{\Delta/2} ((C_{i,i+k} + C_{i+k,j}), (C_{i,j-k} + C_{j-k,j}))$$

- Computation of $C_{i,j}$ **must start** at 1.5Δ
- Its k -th step, which is computed at $2\Delta - k$

Validity of the schedule

$$C_{i,j} = \min_{k=1}^{\Delta/2} ((C_{i,i+k} + C_{i+k,j}), (C_{i,j-k} + C_{j-k,j}))$$

- Computation of $C_{i,j}$ **must start** at 1.5Δ
- Its k -th step, which is computed at $2\Delta - k$, needs $C_{i,i+k}$, $C_{i+k,j}$, $C_{i,j-k}$ and $C_{j-k,j}$

Validity of the schedule

$$C_{i,j} = \min_{k=1}^{\Delta/2} ((C_{i,i+k} + C_{i+k,j}), (C_{i,j-k} + C_{j-k,j}))$$

- Computation of $C_{i,j}$ **must start** at 1.5Δ
- Its k -th step, which is computed at $2\Delta - k$, needs $C_{i,i+k}$, $C_{i+k,j}$, $C_{i,j-k}$ and $C_{j-k,j}$
- which are computed respectively at $2k$

Validity of the schedule

$$C_{i,j} = \min_{k=1}^{\Delta/2} ((C_{i,i+k} + C_{i+k,j}), (C_{i,j-k} + C_{j-k,j}))$$

- Computation of $C_{i,j}$ **must start** at 1.5Δ
- Its k -th step, which is computed at $2\Delta - k$, needs $C_{i,i+k}$, $C_{i+k,j}$, $C_{i,j-k}$ and $C_{j-k,j}$
- which are computed respectively at $2k$, $2(\Delta - k)$

Validity of the schedule

$$C_{i,j} = \min_{k=1}^{\Delta/2} ((C_{i,i+k} + C_{i+k,j}), (C_{i,j-k} + C_{j-k,j}))$$

- Computation of $C_{i,j}$ **must start** at 1.5Δ
- Its k -th step, which is computed at $2\Delta - k$, needs $C_{i,i+k}$, $C_{i+k,j}$, $C_{i,j-k}$ and $C_{j-k,j}$
- which are computed respectively at $2k$, $2(\Delta - k)$, $2(\Delta - k)$ and $2k$

Validity of the schedule

$$C_{i,j} = \min_{k=1}^{\Delta/2} ((C_{i,i+k} + C_{i+k,j}), (C_{i,j-k} + C_{j-k,j}))$$

- Computation of $C_{i,j}$ **must start** at 1.5Δ
- Its k -th step, which is computed at $2\Delta - k$, needs $C_{i,i+k}$, $C_{i+k,j}$, $C_{i,j-k}$ and $C_{j-k,j}$
- which are computed respectively at $2k$, $2(\Delta - k)$, $2(\Delta - k)$ and $2k$
- They are all smaller than $2\Delta - k$

Usage

- $C_{i,j}$ is used by all points to its right (on the same row) and above it (on the same column)

Usage

- $C_{i,j}$ is used by all points to its right (on the same row) and above it (on the same column)
- Consider only the row (column is analogous)

Usage

- $C_{i,j}$ is used by all points to its right (on the same row) and above it (on the same column)
- Consider only the row (column is analogous) i.e., the ***first*** and ***third*** arguments.

Usage

- $C_{i,j}$ is used by all points to its right (on the same row) and above it (on the same column)
- Consider only the row (column is analogous) i.e., the **first** and **third** arguments.

$$C_{i,j'} = \min_{k=1}^{(j'-i)/2} ((C_{i,i+k} + C_{i+k,j'}), (C_{i,j'-k} + C_{j'-k,j'}))$$

Usage

- $C_{i,j}$ is used by all points to its right (on the same row) and above it (on the same column)
- Consider only the row (column is analogous) i.e., the **first** and **third** arguments.

$$C_{i,j'} = \min_{k=1}^{(j'-i)/2} ((C_{i,i+k} + C_{i+k,j'}), (C_{i,j'-k} + C_{j'-k,j'}))$$

- For some $j' > j$, $C_{i,j'}$ uses $C_{i,j}$ as its **first** argument if $j = i + k$ and $1 \leq k \leq \frac{j'-i}{2}$

Usage

- $C_{i,j}$ is used by all points to its right (on the same row) and above it (on the same column)
- Consider only the row (column is analogous) i.e., the **first** and **third** arguments.

$$C_{i,j'} = \min_{k=1}^{(j'-i)/2} ((C_{i,i+k} + C_{i+k,j'}), (C_{i,j'-k} + C_{j'-k,j'}))$$

- For some $j' > j$, $C_{i,j'}$ uses $C_{i,j}$ as its **first** argument if $j = i + k$ and $1 \leq k \leq \frac{j'-i}{2}$, i.e., if $j - i \geq 1$ (obviously true) and

Usage

- $C_{i,j}$ is used by all points to its right (on the same row) and above it (on the same column)
- Consider only the row (column is analogous) i.e., the **first** and **third** arguments.

$$C_{i,j'} = \min_{k=1}^{(j'-i)/2} ((C_{i,i+k} + C_{i+k,j'}), (C_{i,j'-k} + C_{j'-k,j'}))$$

- For some $j' > j$, $C_{i,j'}$ uses $C_{i,j}$ as its **first** argument if $j = i + k$ and $1 \leq k \leq \frac{j'-i}{2}$, i.e., if $j - i \geq 1$ (obviously true) and $j' \geq j + \Delta$

Usage

- $C_{i,j}$ is used by all points to its right (on the same row) and above it (on the same column)
- Consider only the row (column is analogous) i.e., the **first** and **third** arguments.

$$C_{i,j'} = \min_{k=1}^{(j'-i)/2} ((C_{i,i+k} + C_{i+k,j'}), (C_{i,j'-k} + C_{j'-k,j'}))$$

- For some $j' > j$, $C_{i,j'}$ uses $C_{i,j}$ as its **first** argument if $j = i + k$ and $1 \leq k \leq \frac{j'-i}{2}$, i.e., if $j - i \geq 1$ (obviously true) and $j' \geq j + \Delta$, and as its **third** argument for $j + 1 \leq j' \leq j + \Delta$

Data Flow

- For some $j < j' \leq j + \Delta$, $C_{i,j'}$ uses $C_{i,j}$ as its ***third*** argument

Data Flow

- For some $j < j' \leq j + \Delta$, $C_{i,j'}$ uses $C_{i,j}$ as its **third** argument, (i.e., $j = j' - k$) at time instant $2(j' - i) - k$

Data Flow

- For some $j < j' \leq j + \Delta$, $C_{i,j'}$ uses $C_{i,j}$ as its **third** argument, (i.e., $j = j' - k$) at time instant $2(j' - i) - k = 2\Delta +$

Data Flow

- For some $j < j' \leq j + \Delta$, $C_{i,j'}$ uses $C_{i,j}$ as its **third** argument, (i.e., $j = j' - k$) at time instant $2(j' - i) - k = 2\Delta + (j' - j)$
- For some $j' \geq 2\Delta$, $C_{i,j'}$ uses $C_{i,j}$ as its **first** argument, (i.e., $j = i + k$) at time instant $2(j' - i) - k$

Data Flow

- For some $j < j' \leq j + \Delta$, $C_{i,j'}$ uses $C_{i,j}$ as its **third** argument, (i.e., $j = j' - k$) at time instant $2(j' - i) - k = 2\Delta + (j' - j)$
- For some $j' \geq 2\Delta$, $C_{i,j'}$ uses $C_{i,j}$ as its **first** argument, (i.e., $j = i + k$) at time instant $2(j' - i) - k = 2j' - i - j$

Data Flow

- For some $j < j' \leq j + \Delta$, $C_{i,j'}$ uses $C_{i,j}$ as its **third** argument, (i.e., $j = j' - k$) at time instant $2(j' - i) - k = 2\Delta + (j' - j)$
- For some $j' \geq 2\Delta$, $C_{i,j'}$ uses $C_{i,j}$ as its **first** argument, (i.e., $j = i + k$) at time instant $2(j' - i) - k = 2j' - i - j = 2\Delta$

Data Flow

- For some $j < j' \leq j + \Delta$, $C_{i,j'}$ uses $C_{i,j}$ as its **third** argument, (i.e., $j = j' - k$) at time instant $2(j' - i) - k = 2\Delta + (j' - j)$
- For some $j' \geq 2\Delta$, $C_{i,j'}$ uses $C_{i,j}$ as its **first** argument, (i.e., $j = i + k$) at time instant $2(j' - i) - k = 2j' - i - j = 2\Delta + \Delta$

Data Flow

- For some $j < j' \leq j + \Delta$, $C_{i,j'}$ uses $C_{i,j}$ as its **third** argument, (i.e., $j = j' - k$) at time instant $2(j' - i) - k = 2\Delta + (j' - j)$
- For some $j' \geq 2\Delta$, $C_{i,j'}$ uses $C_{i,j}$ as its **first** argument, (i.e., $j = i + k$) at time instant $2(j' - i) - k = 2j' - i - j = 2\Delta + \Delta + (2(j' - (j + \Delta)))$

Exercise (CoB and SUREs)

Recall the rules for the change-of-basis transformations of SRE's. Consider the following SRE:

$$X[z] = \{z \in D_X\} : g(\dots, X[f_{xx}(z)], Y[f_{xy}(z)])$$

$$Y[z] = \{z \in D_Y\} : h(\dots, X[f_{yx}(z)], Y[f_{yy}(z)])$$

and a bijective affine function, $\mathcal{T} : z \mapsto Tz + t$ where T is an integer unimodular matrix, and t is a vector. Show that when all the dependence functions are *uniform* (i.e., the SRE is actually a SURE) applying the *same* transformation, \mathcal{T} , to *both* the variables X and Y , the resulting system remains uniform. Determine the new dependence vectors. Can you think of a (slightly) more general transformation that retains this closure property?

Solution

Note: $\mathcal{T}^{-1}(z) = T^{-1}z - T^{-1}t$, since

Solution

Note: $\mathcal{T}^{-1}(z) = T^{-1}z - T^{-1}t$, since $T^{-1}(Tz + t) - T^{-1}t = z$

Solution

Note: $\mathcal{T}^{-1}(z) = T^{-1}z - T^{-1}t$, since $T^{-1}(Tz + t) - T^{-1}t = z$

Applying CoB by \mathcal{T} to both variables

$$X[z] = \{z \in \mathcal{T}(D_X)\} : g(\dots, X[\mathcal{T} \circ f_{xx} \circ \mathcal{T}^{-1}(z)], \\ Y[\mathcal{T} \circ f_{xy} \circ \mathcal{T}^{-1}(z)])$$

$$Y[z] = \{z \in \mathcal{T}(D_Y)\} : h(\dots, X[\mathcal{T} \circ f_{yx} \circ \mathcal{T}^{-1}(z)], \\ Y[\mathcal{T} \circ f_{yy} \circ \mathcal{T}^{-1}(z)])$$

Each new dependence is of the form $\mathcal{T} \circ f \circ \mathcal{T}^{-1}(z)$, where f is uniform, i.e., $f(z) = z + \delta$

$$\begin{aligned}\mathcal{T} \circ f \circ \mathcal{T}^{-1}(z) &= T f(T^{-1}z - T^{-1}t) + t \\ &= T(T^{-1}z - T^{-1}t + \delta) + t \\ &= (z - t + T\delta) + t \\ &= z + T\delta\end{aligned}$$