Chapter 7 Unfolding

Introduction (1)

- **unfolding**
  - A transformation technique
  - Also known as loop unrolling
  - Create a new program describing more than one iteration of the original program

- **Unfolding example**
  - \( y(n) = ay(n-9) + x(n) \)
  - Unfolding factor \( J = 2 \)
  - \( y(2k) = ay(2k-9) + x(2k) \)
  - \( y(2k+1) = ay(2k-8) + x(2k+1) \)
  - Each delay is J-slow

2-unfolded version
Introduction (2)

Applications of unfolding
- Unfold the program to reveal hidden concurrency so that the program can be scheduled to a smaller iteration period
- Design parallel architectures at the word level and bit level for high speed or low power purposes

Unfolding algorithm (1)

Preliminary
- Each node in the original DFG $\Rightarrow J$ nodes with the same function in the unfolded DFG
- Each edge in the original DFG $\Rightarrow J$ edges in the unfolded DFG

Unfolding algorithm
- For each node $U$ in the original DFG, draw $J$ nodes $U_0$, $U_1$, $\ldots$, $U_{J-1}$
- For each edge $U \rightarrow V$ with $w$ delays in the original DFG, draw the $J$ edges $U_i \rightarrow V_{(i+w)\%J}$ with $\left\lfloor \frac{i+w}{J} \right\rfloor$ delays for $i=0,1,\ldots,J-1$
Unfolding algorithm (2)

- Unfolding using algorithm
  - $A$: input, $B$: output, $C$: ADD, $D$: MPY
  - Unfolding factor $J = 2$
  - 8 nodes after unfolding, $A_i$, $B_i$, $C_i$, $D_i$, $i=0,1$
  - $D \rightarrow C$ with no delay $\Rightarrow$ $D_0 \rightarrow C_0$, $D_1 \rightarrow C_1$
  - $C \rightarrow D$ with 9 delay $\Rightarrow$
    - $C_0 \rightarrow D_{(9+0)\%2}$ with $\left\lceil \frac{9+0}{2} \right\rceil$ delays $\Rightarrow C_0 \xrightarrow{4D} D_1$
    - $C_1 \rightarrow D_{(9+1)\%2}$ with $\left\lceil \frac{9+1}{2} \right\rceil$ delays $\Rightarrow C_1 \xrightarrow{5D} D_0$

Unfolding algorithm (3)

- Unfolding using algorithm (cont.)
  - The $k$-th iteration of node $A_i$ in the unfolded DFG executes the $(Jk+i)$-th iteration of node $A$ in the original DFG
  - $A_0$ corresponds to input $x(2k+0)$
  - $A_1$ corresponds to input $x(2k+1)$
  - $B_0$ corresponds to output $y(2k+0)$
  - $B_1$ corresponds to output $y(2k+1)$
Unfolding algorithm (4)

- Unfolding example with $J = 4$

Unfolding example with $J = 3$

Properties of Unfolding (1)

- General observation
  - Unfolding of an edge with $w$ delays produces $Jw$ edges with no delays and $w$ edges with 1 delay in $J$-folded DFG
  - Unfolding preserves precedence constraints

- Property 5.3.1
  - Unfolding preserves the number of delays in a DFG
  - The sum of the delays on the $J$ unfolded edges $U_i \rightarrow V_{(i+w)\%J}$ for $i = 0, 1, \ldots, J-1$ is the same as the number of delays on the edge $U \rightarrow V$ in the original DFG
  - i.e. $\left \lfloor \frac{w}{J} \right \rfloor + \left \lfloor \frac{w+1}{J} \right \rfloor + \cdots + \left \lfloor \frac{w+J-1}{J} \right \rfloor = w$
Properties of Unfolding (2)

Loop unfolding

- Assume a loop $A \rightarrow A$ with $w_l$ delay
- Traverse the loop $p$ times: $A \rightarrow A \cdots \rightarrow A$ with $pw_l$ delay
- The corresponding unfolded path starting at the node $A_i$, $0 \leq i \leq J-1$
- For a $J$-unfolded DFG, the starting and end points of a $p$ times traversed loop are $A_i \rightarrow A_{(i+pw_l)\%J}$
- This path forms a loop in the unfolded DFG if $i = (i+pw_l)\%J$
- Question: what is the minimum value of $p$?

Properties of Unfolding (3)

Loop unfolding example

- A single loop $l = A \rightarrow B \rightarrow C \rightarrow A$ with $w_l = 6$ delays
- 3 unfolded DFG
- $i = (i+pw_l)\%J \Rightarrow i = (i+6p)\%3$ holds for $i = 0,1,2$ for $p=1$
- $A_0 \rightarrow B_1 \rightarrow C_0 \rightarrow A_0$
- $A_1 \rightarrow B_2 \rightarrow C_1 \rightarrow A_1$
- $A_2 \rightarrow B_0 \rightarrow C_2 \rightarrow A_2$
Loop unfolding example (cont.)

- 4 unfolded DFG
- \( i = (i+p_{\text{wl}}) \% J \Rightarrow i = (i+6p) \% 4 \) holds for \( i = 0,1,2 \) for \( p=2 \)
- \( A_0 \rightarrow B_1 \rightarrow C_3 \rightarrow A_2 \rightarrow B_3 \rightarrow C_1 \rightarrow A_0 \)
- \( A_1 \rightarrow B_2 \rightarrow C_0 \rightarrow A_3 \rightarrow B_0 \rightarrow C_2 \rightarrow A_1 \)
- Consists of 2 loops in the unfolded DFG

\[
\begin{align*}
A_0 &\rightarrow B_1 & B_3 &\rightarrow C_1 & A_2 &\rightarrow B_3 & C_0 &\rightarrow A_0 \\
A_1 &\rightarrow B_2 & C_1 &\rightarrow A_0 & B_3 &\rightarrow C_2 & A_0 &\rightarrow B_1
\end{align*}
\]

Properties of Unfolding (5)

**Lemma 5.3.1**

- \( i = (i+p_{\text{wl}}) \% J \iff p_{\text{wl}} = qJ \) for an integer \( q \)

**Lemma 5.3.2**

- The smallest positive integer \( p \) that satisfies \( p_{\text{wl}} = qJ \) is \( J/\gcd(w_l, J) \)
- Proof: 
  \( \gcd \) is a common divisor of \( w_l \) and \( J \).
  \( \lcm \) is a common multiple of \( w_l \) and \( J \).
  Use Bézout's identity to express \( \gcd \) and \( \lcm \) in terms of integers.

\[
\begin{align*}
\gcd \{w_l, J\} &\text{ divides } w_l \Rightarrow \exists a \in \mathbb{Z} : a = \gcd \{w_l, J\}
\Rightarrow a = \lcm \{w_l, J\}
\Rightarrow q = \left( \frac{a}{J} \right) \in I
\Rightarrow p = \left( \frac{a}{w_l} \right) \in I
\Rightarrow minimum \; p = \frac{\lcm \{w_l, J\}}{\gcd \{w_l, J\}} = \frac{J}{w_l}
\end{align*}
\]
Properties of Unfolding (5)

Property 5.3.2
- \( J \)-unfolding of a loop \( l \) with \( w_l \) delays in the original DFG leads to
- \( \gcd(w_l, J) \) loops in the unfolded DFG
- Each loop contains \( w_l / \gcd(w_l, J) \) delays and \( J / \gcd(w_l, J) \) copies of each node in loop \( l \)

Property 5.3.3
- Unfolding a DFG with iteration bound \( T_\infty \) results in a \( J \)-unfolded DFG with iteration bound \( JT_\infty \)
  - proof \( T_\infty = \max_i \left\{ \frac{t_l}{w_l} \right\} \) for the original DFG, from Property 5.3.2
  \[ T'_\infty = \max_i \left\{ \frac{J / \gcd\{w_l, J\} t_l}{w_l / \gcd\{w_l, J\}} \right\} = J \max_i \left\{ \frac{t_l}{w_l} \right\} = JT_\infty \]

Critical path, unfolding & retiming (1)

Property 5.4.1
- Consider a path with \( w \) delays in the original DFG
  - \( J \)-unfolding leads to \((J-w)\) paths with no delay and \( w \) paths with 1 delay each, when \( w < J \)

Corollary 5.4.1
- Any path in the original DFG containing \( J \) or more delays leads to \( J \) paths with 1 or more delays in each path
- A path in the original DFG with \( J \) or more delays cannot create a critical path in the \( J \)-unfolded DFG
Critical path, unfolding & retiming (2)

Problem statement
- retime the original DFG such that the $J$-unfolded version of the retimed DFG will meet a critical path computation time $c$

Observation
- The critical path of the unfolded DFG can be $c$ if there exists a path in the original DFG with computation time $c$ and less than $J$ delays

Retiming constraint for critical path
- If $D(U, V) \geq c$, $\Rightarrow W_r(U, V) = W(U, V) + r(V) - r(U) \geq J$
- Or $r(U) - r(V) \leq W(U, V) - J$
- + Feasibility constraint $w(e) + r(V) - r(U) \leq 0$

Critical path, unfolding & retiming (3)

Lemma 5.4.1
- Any feasible clock cycle period that can be obtained by retiming the $J$-folded DFG, $G_J$
  - can be achieved by retiming the original DFG, $G$, directly and then unfolding it by unfolding factor $J$

Proof
- Let $r'$ be a legal retiming for the unfolded DFG, $G_J$, which leads to critical path $c$
- Let $r$ be a retiming for $G$ defined as $r(U) = \sum_{i=0}^{J-1} r'(U_i)$
- We next prove that $r$ is a feasible retiming on $G$ such that the $G_r$ will have a critical path $c$
Critical path, unfolding & retiming (4)

Proof (cont.)
- Proof of the feasibility of retiming $r$
- for $U \rightarrow V$ with delay $w$ in $G$, since $r'$ is a legal timing in $G_r$
  \[ r'(U_i) - r'(V_{(i+w)\%J}) \leq \left\lfloor \frac{i + w}{J} \right\rfloor \]
- Summing above for $i = 0$ to $J-1$, \( \Rightarrow r(U) - r(V) \leq w \)
- So $r$ is a legal retiming in $G$
- Next prove the critical path in $J$-unfolded DFG is $c$
  \[ r'(U_i) - r'(V_{(i+w)\%J}) \leq \left\lfloor \frac{i + w}{J} \right\rfloor - 1 \quad \text{if} \quad D(U_i \rightarrow V_{(i+w)\%J}) > c \]
- Summing above for $i = 0$ to $J-1$ \( \Rightarrow r(U) - r(V) \leq W(U,V) - J \)
- which is the desired critical path constraint shown in p4-15

Sample period reduction (1)

Cases when iteration bound equal to $T_\infty$ cannot be achieved without unfolding
- A node in the DFG with computation time greater than $T_\infty$ (assume the node cannot be further pipelined)
- When the iteration bound is not an integer

Case 1 example
- $T_\infty = 3$
- but node S & T require 4 u.t.
- Minimum iteration period after retiming is 4
- Can be reduced to 3 after unfolding
Sample period reduction (2)

Case 1 example after unfolding
- Unfolding factor = 2
- $T_{\infty}$ of the unfolded DFG is 6
- Critical path of the unfolded DFG is 6
- Sample period = $6/2 = T_{\infty}$ of the original DFG
- If the computation time of a node $U$, $t_U$ is greater than $T_{\infty}$, $\left\lceil t_U / T_{\infty} \right\rceil$ unfolding is needed

Sample period reduction (3)

Case 2 example
- $T_{\infty} = 4/3$
- Retiming without unfolding can only achieve iteration period 2
- Unfolding 3 times and $T_{\infty}$ of the unfolded DFG is 4
- Equivalent sample period is 4/3
- If a critical loop bound is $t_l/w_l$, then $w_l$ times unfolding should be used
Sample period reduction (4)

- If the longest node computation time is larger than $T_\infty$ and $T_\infty$ is not an integer, minimum unfolding factor $J$ is
  - $JT_\infty$ is an integer
  - $JT_\infty$ is greater than the longest node computation time

- A perfect rate DFG
  - Any DFG with 1 delay in each loop
  - Can always be scheduled such that the iteration bound is equal to $T_\infty$

Sample period reduction (5)

- Any DFG, if not perfect rate, can be unfolded to become a perfect rate one
  - Recall property 5.3.2
  - A loop with $w_l$ delays forms $\gcd(w_l,J)$ loops and each containing $w_l/\gcd(w_l,J)$ delays after $J$ time unfolding
  - Choose $J$ as a multiple of $w_l$, we have $\gcd(w_l,J) = w_l$ and $w_l/\gcd(w_l,J) = 1$, the unfolded DFG is a perfect rate one
  - For a DFG containing multiple loops, choose $J$ equal to the lcm of all loop delays
Periodic schedules (1)

- Period schedules
  - Can be constructed from the acyclic precedence graphs that are obtained by deleting all edges with delay elements from the DFG
  - Overlapped v.s. non-overlapped schedules

- Non-overlapped schedule
  - The period of the schedule is the same as the critical path of the acyclic precedence graph

- Overlapped schedule
  - The schedules of two successive iterations overlap
  - The period of the schedule is smaller than the critical path of the acyclic precedence graph

Periodic schedules (2)

- Non-overlapped schedule example
  - Assume $T_A$, $T_B$, $T_C$ are 1, 2, 4 u.t. respectively
  - The iteration bound is 3.5 u.t.
  - Schedule period = 6 = critical path
  - Retiming can only achieve an iteration period of 4 u.t.
Periodic schedules (2)

- overlapped schedule example
  - 2-unfolded DFG
  - Average iteration period is 3.5 u.t.

Parallel processing

- Parallel processing
  - Word serial → word parallel
  - Bit serial → bit parallel or digit serial

- Word-level parallel processing
  - original version: one sample per clock
  - Unfolded version: two samples per clock
In general, a $j$ unfolded DFG leads to a parallel architecture computing $j$ words per clock cycle.

Example

$$y(n) = ax(n) + bx(n-4) + cx(n-6)$$

Example (cont.)

3-parallel DSP program
**Bit level parallel processing (1)**

- **Bit serial processing**
  - One bit is processed per clock cycle
  - A complete word is processed in $W$ clock cycles

- **Bit parallel processing**
  - One word of $W$ bits is processed every clock cycle

- **Digit serial processing**
  - $N$ (digit size) bits are processed per clock cycle
  - A word is processed in $W/N$ clock cycles

![Bit-parallel](image1)

![Digit-serial](image2)

$W = 6$

$N = 2$

**Bit level parallel processing (2)**

- **Bit serial adder example**
  - A switch is required
  - ![Adder Circuit](image3)

- **Switch unfolding**
  - Assumption 1: $W$ is a multiple of the unfolding factor $J$, i.e. $W = W'J$
  - Assumption 2: all edges into and out of the switch have no delays
Switch unfolding (cont.)

- \( Wl + u = J(W'l + \left\lfloor u / J \right\rfloor) + (u \% J) \)
- Draw an edge with no delays in the unfolded graph from node \( U_{u \% J} \) to node \( V_{u \% J} \),
- which is switched at time instance \( (W'l + \left\lfloor u / J \right\rfloor) \)

Example

- \( W = 12, u = 7, J = 3 \)
- \( W = W'J \Rightarrow W' = 4 \)
- \( 12l + 7 = 3(4l + \left\lfloor 7 / 3 \right\rfloor) + (7 \% 3) = 3(4l + 2) + 1 \)
- Means from node \( U_1 \) to \( V_1 \) and switching at \( 4l + 2 \)
- \( 12l + 1 = 3(4l + 0) + 1 \)
- \( 12l + 7 = 3(4l + 2) + 1 \)
- \( 12l + 9 = 3(4l + 3) + 0 \)
- \( 12l + 11 = 3(4l + 3) + 2 \)
Bit level parallel processing (5)

- Unfolding a switch with delays
  - Employing a dummy node

\[
\begin{align*}
2)12(356 + 2D_{6l+1,5} + \sum_{i=0}^{4} 2D_{6l+0,2,3,4} &= 0 & \text{ll} \\
1)12(346 + 2D_{6l+1,5} + \sum_{i=0}^{4} 2D_{6l+0,2,3,4} &= 1 & \text{ll} \\
0)12(336 + 2D_{6l+1,5} + \sum_{i=0}^{4} 2D_{6l+0,2,3,4} &= 2 & \text{ll} \\
2)02(326 + 2D_{6l+1,5} + \sum_{i=0}^{4} 2D_{6l+0,2,3,4} &= 3 & \text{ll} \\
1)02(316 + 2D_{6l+1,5} + \sum_{i=0}^{4} 2D_{6l+0,2,3,4} &= 4 & \text{ll} \\
0)02(306 + 2D_{6l+1,5} + \sum_{i=0}^{4} 2D_{6l+0,2,3,4} &= 5 & \text{ll}
\end{align*}
\]

- For J = 3 unfolding

\[
\begin{align*}
6l + 0 &= 3(2l + 0) + 0 \\
6l + 1 &= 3(2l + 0) + 1 \\
6l + 2 &= 3(2l + 0) + 2 \\
6l + 3 &= 3(2l + 1) + 0 \\
6l + 4 &= 3(2l + 1) + 1 \\
6l + 5 &= 3(2l + 1) + 2
\end{align*}
\]

Bit level parallel processing (6)

- Unfolding a switch with delays (cont.)
  - Dead node \( A_1 \) is eliminated
  - Number of delays is not preserved
Digital serial processing example

- 4-bit bit serial adder

\[
\begin{align*}
A_3 & \rightarrow A_2 & \rightarrow A_1 & \rightarrow A_0 \\
b_3 & \rightarrow b_2 & \rightarrow b_1 & \rightarrow b_0
\end{align*}
\]

- 4-unfolded version (bit parallel ripple adder)

Unfolded DFG

Architecture design

Digital serial processing example (cont.)

- 2-unfolded version

- Digit size = 2
Switch unfolding when $W$ is not a multiple of $J$

- determining $L = \text{lcm}\{W, J\}$
- Replacing switching instance (not switch) $Wl + u$ with $L/W$
  switching instances $Ll + u + wW$ for $w = 0$ to $L/W - 1$
- Switch periodicity is changed from $W$ to $L$
- Each switching instance is expanded by a factor $L/W$
- Switching periodicity is now a multiple of $J$
- Proceeds with the previous approach

Example

- $W = 4, J = 3 \implies L = 12$
- Switching instance $4l$ is expanded to $12l, 12l + 4, 12l + 8$
- Switching instance $4l+1$ is expanded to $12l+1, 12l + 5, 12l+9$
Example (cont.)

- Process 3 words in every 4 clock cycles