Unfolding

Unfolding creates a program with more than one iteration, \( J = \text{unfolding factor} \)

Unfolding is a structured way to achieve parallel processing

Applications
- sample period reduction, reach \( T_\infty \)
- Parallel processing
- Bit-serial and Digit-serial

Unfolding = Loop unrolling
- assembly programming
- compiler theory

Example: Loop unrolling + Software Pipelining

GSM Speechcoder
- Org. C-code = 250k cc
- Mod. C-code = 90k cc
- Hand Opt. = 50k cc

Unfolding \( \equiv \) Parallel Processing

In a ‘\( J \)’ unfolded system each delay is \( J \)-slow \( \Rightarrow \)
if input to a delay element is \( x(kJ + m) \) \( \Rightarrow \)
the output is \( x((k-1)J + m) = x(kJ + m - J) \).

\[ (1) \]
\[ (1) \]

\[ \text{2-unfolded} \]

\[ (1) \]
\[ (1) \]

\[ 0, 2, 4, \ldots \]

\[ \text{4 nodes & 4 edges} \]
\[ T_\infty = 2/2 = 1 \text{ut} \]
Unfolding, example

\[ y(n) = ay(n-9) + x(n) \]

Unfolding J=2, 2-times parallel \( \Rightarrow \)

\[
\begin{cases}
  y(2k) = ay(2k-9) + x(2k) \\
  y(2k+1) = ay(2k-8) + x(2k+1)
\end{cases}
\]

Not trivial even for a simple graph!

Definitions

\[
\begin{align*}
  \lfloor x \rfloor & \text{ is the floor of } x, \text{ largest integer } \leq x \\
  \lceil x \rceil & \text{ is the ceiling of } x, \text{ smallest integer } \geq x \\
  a \% b & \text{ remainder after } a/b
\end{align*}
\]
Algorithm for unfolding

• For each node $U$ in the original DFG, draw $J$ nodes $U_0, U_1, U_2, \ldots, U_{J-1}$.

\[
\begin{bmatrix}
(i + w) \\
J
\end{bmatrix} = \begin{bmatrix}
(i + 37) \\
4
\end{bmatrix} = \begin{cases}
9, & i = 0, 1, 2 \\
10, & i = 3
\end{cases}
\]

• 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 and Iteration Bound

**Example:**

- $T_A = 3$, $T_M = 6$
- $T_\infty = \frac{18}{9} = 2$
- $T_\infty = \frac{9}{9} = 1$

Properties of unfolding

- Unfolding preserves the number of delays in a DFG:
  \[\left\lfloor \frac{w}{J} \right\rfloor + \left\lceil \frac{(w+1)}{J} \right\rceil + \ldots + \left\lceil \frac{w + J - 1}{J} \right\rceil = w\]
- Unfolding preserves precedence constraints.
- $J$-unfolding of a loop with $w$ delays in the original DFG results in $\gcd(w, J)$ loops in the unfolded DFG. Each loop contains $\frac{w}{\gcd(w, J)}$ delays and $\frac{J}{\gcd(w, J)}$ copies of each node.
- Unfolding a DFG with iteration bound $T_\infty$ results in a $J$-unfolded DFG with iteration bound $JT_\infty$. 

$\gcd = \text{greatest common divisor}$
The Critical Path

If edge with \( w < J \) \( \Rightarrow \) \( (J-w) \) paths with zero delay and \( w \) paths with 1 delay

Can lead to increased critical path!

Edge with \( w \geq J \) will not create new critical path!

Sample Period Reduction

• Case 1: A node in the DFG having computation time greater than \( T_\infty \).

• Case 2: Iteration bound is not an integer.

• Case 3: Longest node computation is larger than the iteration bound \( T_\infty \), and \( T_\infty \) is not an integer.

Sample Period Reduction: case 1

The original DFG cannot have sample period equal to the iteration bound because a node computation time is more than iteration bound

\[
T_\infty = \max_{l \in L} \left\{ \frac{t_I}{w_I} \right\}
\]

\[
= \max \left\{ \frac{6}{3}, \frac{6}{2} \right\} = 3
\]

\(<4\), max node time

IIR-filter from Lab1

\[
X(n) + (0) \quad b_2
\]

\[
(4)
\]

\[
\Delta
\]

\[
+ \quad u(n)
\]

\[
(1)
\]

\[
(0)
\]
Sample Period Reduction: case 1

If the computation time of a node ‘U’, \( t_u \), is greater than the iteration bound \( T_\infty \), then \( \lfloor t_u/T_\infty \rfloor \) - unfolding should be used.

\[ t_u = 4 \text{ and } T_\infty = 3 \]

\[ \lfloor 4/3 \rfloor = 2 \text{ - unfolding} \]

But two Samples!

Sample Period Reduction: case 2

The original DFG cannot have sample period equal to the iteration bound because the iteration bound is not an integer

\[ T_\infty = \max_{l \in L} \frac{t_l}{w_l} = \frac{4}{3} \]

If a critical loop bound is of the form \( t_l/w_l \) where \( t_l \) and \( w_l \) are mutually co-prime, then \( w_l \)-unfolding should be used.

Unfolding of 3

Sample Period Reduction: case 3

The original DFG cannot have sample period equal to the iteration bound because the longest node computation is larger than the iteration bound \( T_\infty \), and \( T_\infty \) is not an integer

\[ T_\infty = 4 \]

and 3 samples gives minimum sample period \( 4/3 \)
Parallel processing can be performed by unfolding, chapter 3

Another FIR-filter, J=3

Bit-Level Parallel Processing

Digit-Serial

Digit-Serial (Digit-size = 2)

Bit-Serial
Bit-serial adder

Bit-serial can be seen as a time-multiplexed architecture, in this example on addition (i.e. 1 iteration) takes 4cc.

\[ a_3 a_2 a_1 a_0 \quad \rightarrow \quad s_3 s_2 s_1 s_0 \]

Switch for carry signal

How to unfold switches?

Unfolding of Switches

- The following assumptions are made when unfolding an edge \( U \rightarrow V \) containing a switch:
  - The wordlength \( W \) is a multiple of the unfolding factor \( J \), i.e. \( W = W'J \).
  - All edges into and out of the switch have no delays.
- With the above two assumptions an edge \( U \rightarrow V \) can be unfolded as follows:
  - Write the switching instance as
    \[ WI + u = J( W'I + \lfloor u/J \rfloor ) + (u\%J) \]
  - Draw an edge from the node \( U_{u\%J} \Rightarrow V_{u\%J} \)
    which is switched at time instance \( ( W'I + \lfloor u/J \rfloor ) \).

Example: Unfolding of Switches, \( J=3 \)

\[ U \quad \rightarrow \quad V \]

- Write the switching instance as
  \[ WI + u = J( W'I + \lfloor u/J \rfloor ) + (u\%J) \]
  \[ 9l+1 = 3(3l + 1) + (1\%3) = 3(3l + 1) + 1 \]
  \[ 9l+5 = 3(3l + 5/3) + (5\%3) = 3(3l + 1) + 2 \]

Switched at time instances

Edges between Nodes

Example: Unfolding of Switches, \( J=3 \)

\[ U \quad \rightarrow \quad V \]

- Write the switching instance as
  \[ WI + u = J( W'I + \lfloor u/J \rfloor ) + (u\%J) \]
  \[ 9l+1 = 3(3l + 1) + (1\%3) = 3(3l + 1) + 1 \]
  \[ 9l+5 = 3(3l + 5) + (5\%3) = 3(3l + 1) + 2 \]

Edges between Nodes

Draw an edge from the node \( U_{u\%J} \Rightarrow V_{u\%J} \)

\( U_1 \Rightarrow V_1 \) and \( U_2 \Rightarrow V_2 \)
Example: Unfolding of Switches, J=3

Switch with multiple instances

Switch with multiple instances

Switches with Delays
Bit-serial Adder

INPUTS

A

X

D

B

4/0

4/1,2,3

Z

Reset

Carry = 0

Carry

S

Output

Dummy node

Unfold Bit-serial Adder, J=2

Unfold Bit-serial Adder, J=2

For each node U in the original DFG, draw J nodes U₀, U₁, U₂,..., Uᵣ with

For each edge U → V with w delays in the original DFG, draw the J edges Uᵢ → Vᵢ with ⌊(i+w)/J⌋ delays for i = 0, 1,..., J-1

If edge has w=0 ⇔ Uᵢ → Vᵢ with 0 delays

Viktor Öwall, Dept. of Electrical and Information Technology, Lund University, Sweden-www.eit.lth.se

Viktor Öwall, Dept. of Electrical and Information Technology, Lund University, Sweden-www.eit.lth.se
Unfold the Switch, J=2

Write the switching instance as
\[ Wl + u = J\left( Wl + \left\lfloor \frac{u}{J} \right\rfloor \right) + (u \% J) \]

Viktor Öwall, Dept. of Electrical and Information Technology, Lund University, Sweden-www.eit.lth.se
Remove Dead and Dummy Nodes

\[ Z_0 \rightarrow X_0 \text{ at time } 2l+0 \]
\[ D_0 \rightarrow X_0 \text{ at time } 2l+1 \]
\[ D_1 \rightarrow X_1 \text{ at time } 2l+0,1 \]
\[ i.e. \text{ always closed} \]

The Digit Serial Adder

Carry within iteration
\[ D=1 \]

Fully Parallel Adder, i.e. \( J=4 \)

Unfold the Switch, \( J=4 \)

For each node \( U \) in the original DFG, draw \( J \) nodes \( U_0, U_1, U_2, \ldots, U_{J-1} \)
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 \( \lfloor (i+w)/J \rfloor \) delays for \( i = 0, 1, \ldots, J-1 \)

Write the switching instance as
\[ W_l + u = J \left( W'_l + \left\lfloor u/J \right\rfloor \right) + (u\%J) \]
Unfold the Switch, J=4

$Z \Rightarrow X$
$4l+0 \rightarrow 4(1l+0)+0$

$D \Rightarrow X$
$4l+0 \rightarrow 4(1l+0)+0$
$4l+1 \rightarrow 4(1l+0)+1$
$4l+2 \rightarrow 4(1l+0)+2$
$4l+3 \rightarrow 4(1l+0)+3$

Only 1 time instance 0, i.e. fully parallel
$Z_0 \rightarrow X_0, D_1 \rightarrow X_1, D_2 \rightarrow X_2$ and $D_3 \rightarrow X_3$

Bit-parallel Adder

Only 1 time instance 0, i.e. fully parallel
$Z_0 \rightarrow X_0, D_1 \rightarrow X_1, D_2 \rightarrow X_2$ and $D_3 \rightarrow X_3$

Bit-parallel Adder

"Dead" nodes

"Dead" nodes can be removed

"Dead" nodes can be removed
Bit-parallel Adder

Carry from MSB as overflow or if to be used as a 4-bit module

Switch if to be used as a 4-bit module

Carry Ripple Adder

If Wordlength is not a multiple of J

• determine $lcm(W,J)$, $lcm$ = least common multiple
• replace switching instance $Wl+u$ with $L/W$ instances $Ll+u+wW$, for $w=0..L/W-1$
  i.e. the switching periodicity has been changed from $W$ to $L$
• perform the unfolding as previously
• identify the correspondence between original instances and expanded instances