Unfolding

Lan-Da Van (范倫達), Ph. D.
Department of Computer Science
National Chiao Tung University
Taiwan, R.O.C.
Fall, 2010

ldvan@cs.nctu.edu.tw
http://www.cs.nctu.tw/~ldvan/
Outline

- Introduction
- An Algorithm for Unfolding
- Properties of Unfolding
- Critical Path, Unfolding, and Retiming
- Applications of Unfolding
- Conclusions
Introduction

Unfolding is a transformation technique
- Applied to a DSP algorithm to create a new program describing **more than one iteration** of the original program

Unfolding is also referred to as loop unrolling
- Unfolding factor $J$ describes $J$ consecutive iterations of the original program

Applications to high-speed & low-power VLSI architectures
- Reveal hidden concurrencies so that the program can be scheduled to a smaller iteration period
- Design parallel architectures at the **word level** and **bit level**
Example of A DSP System

- $y(2k) = ay(2k-9) + x(2k)$
- $y(2k+1) = ay(2k-8) + x(2k+1)$
- $y(n) = ay(n-9) + x(n)$
- $y(2k) = ay(2(k-5)+1) + x(2k)$
- $y(2k+1) = ay(2(k-4)+0) + x(2k+1)$
Outline

- Introduction
- *An Algorithm for Unfolding*
- Properties of Unfolding
- Critical Path, Unfolding, and Retiming
- Applications of Unfolding
- Conclusions
Unfolding Algorithm (1/3)

Unfolding is a **graph-based technique** for directly unfolding the DFG to create the J-unfolded DFG.

**Definitions**
- \( \lfloor x \rfloor \) is the floor operation of x, which is largest integer less than or equal to x. Ex: \( \lfloor 11 / 4 \rfloor = 2 \)
- \( a \% b \) is the remainder after dividing a by b, where a and b are integers. Ex: \( 11 \% 4 = 3 \) and \( 23 \% 3 = 2 \).

**Construct a J-unfolded DFG,**
- For each node U in the original DFG, draw the 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 \( \lfloor i + w / J \rfloor \) delays for \( i = 0, 1, \ldots, J-1 \).
Unfolding Algorithm (2/3)

While an edge with $w<J$ delays in the original DFG, the J-unfolded DFG will have:
- $J-w$ edges with no delays
- $w$ edges with 1 delay

Source: Prof. S. Y. Huang (EE/NTHU)
Unfolding Algorithm (3/3)

Proof:

- k-iteration of $U_i \rightarrow (Jk+i)$-th iteration of $U$ ---- (1)
- k-iteration of $U_i$ is consumed by $(k+\text{floor}((i+w)/J))$-th iteration of $V_{(i+w)\%J}$ ---- (2)
- $(k+\text{floor}((i+w)/J))$-th iteration of $V_{(i+w)\%J} \rightarrow (J(k+\text{floor}((i+w)/J))+(i+w)\%J)$-th iteration of $V$ ---- (3)
- Perform (3) – (1), we can obtain following term

$$J(k+\text{floor}((i+w)/J))+(i+w)\%J - (Jk+i)$$

$$= J(\text{floor}((i+w)/J))+(i+w)\%J - i$$

$$= i + w - i = w$$
4-Unfolded Example (1/2)

- $U_0 \rightarrow V_{(0+37)\%4=1}$ with $\left\lfloor \frac{0 + 37}{4} \right\rfloor = 9$ delays
- $U_1 \rightarrow V_{(1+37)\%4=2}$ with $\left\lfloor \frac{1 + 37}{4} \right\rfloor = 9$ delays
- $U_2 \rightarrow V_{(2+37)\%4=3}$ with $\left\lfloor \frac{2 + 37}{4} \right\rfloor = 9$ delays
- $U_3 \rightarrow V_{(3+37)\%4=0}$ with $\left\lfloor \frac{3 + 37}{4} \right\rfloor = 10$ delays
3-Unfolded Example (2/2)

\[ U_0 \rightarrow V_{(0+1)\%3=1}^{[0D]} \rightarrow T_{(1+6)\%3=1}^{[2D]} \rightarrow U_{(1+5)\%3=0}^{[2D]} \]

\[ U_1 \rightarrow V_{(1+1)\%3=2}^{[0D]} \rightarrow T_{(2+6)\%3=2}^{[2D]} \rightarrow U_{(2+5)\%3=1}^{[2D]} \]

\[ U_2 \rightarrow V_{(2+1)\%3=0}^{[1D]} \rightarrow T_{(0+6)\%3=0}^{[2D]} \rightarrow U_{(0+5)\%3=2}^{[D]} \]
Outline

- Introduction
- An Algorithm for Unfolding
- Properties of Unfolding
- Critical Path, Unfolding, and Retiming
- Applications of Unfolding
- Conclusions
Property 5.3.1

Unfolding preserves the number of delays in a DFG, i.e.
\[
\left\lfloor \frac{w}{J} \right\rfloor + \left\lfloor \frac{w+1}{J} \right\rfloor + \ldots + \left\lfloor \frac{w+J-1}{J} \right\rfloor = w
\]

Proof:

Let \( w = mJ + n \), where \( m \) and \( n \) are integers, and \( 0 \leq n \leq (J - 1) \). Thus,
\[
\left\lfloor \frac{w}{J} \right\rfloor = \left\lfloor \frac{mJ + n}{J} \right\rfloor = m,
\]
\cdots
\[
\left\lfloor \frac{w + J - n - 1}{J} \right\rfloor = \left\lfloor \frac{mJ + n + J - n - 1}{J} \right\rfloor = m,
\]
Property 5.3.1

\[
\begin{align*}
\left\lfloor \frac{w + J - n}{J} \right\rfloor &= \left\lfloor \frac{mJ + n + J - n}{J} \right\rfloor = m + 1, \\
\vdots
\end{align*}
\]

Calculate the sum of the above equations, we get

\[
\left\lfloor \frac{w}{J} \right\rfloor + \left\lfloor \frac{w + 1}{J} \right\rfloor + \ldots + \left\lfloor \frac{w + J - 1}{J} \right\rfloor = m \times (J - n) + (m + 1) \times n = mJ + n = w
\]
Lemmas on Loop

Unfolded Loop
- Let $l$ be a loop with $w_l$ delays in the original DFG, and let $A$ be a node in $l$.
- The loop $l$ can be denoted as the path $A \rightarrow A$ with $w_l$ delays.
- If the loop $l$ is traversed $p$ times ($p \geq 1$), this results in the path $A \rightarrow A \rightarrow A \rightarrow \ldots \rightarrow A$ with $pw_l$ delays.
- In the J-unfolded DFG, $A_i \rightarrow A_{(i+pw_l) \mod J}$ forms a loop if $i = (i+pw_l) \mod J$.

Lemma 5.3.1:
- $i = (i+pw_l) \mod J \iff pw_l = qJ$ for an integer $q$.

Lemma 5.3.2
- The smallest positive integer $p$ that satisfies $pw_l = qJ$ is $J/gcd(w_l, J)$.
Lemma 5.3.2

The smallest positive integer \( p \) that satisfies \( pw_{\ell} = qJ \) is \( J/\gcd(w_{\ell}, J) \)

Proof:

- The expression \( pw_{\ell} = qJ \) can be written as \( (a/w_{\ell})w_{\ell} = (a/J)J \).
- The values \( p = a/w_{\ell} \) and \( q = a/J \) must be integers and \( a \) is to be as small as possible because \( p \) is to be as small as possible.
- So \( a = \text{lcm}\{w_{\ell}, J\} \).
- Since \( \text{lcm}\{w_{\ell}, J\} \) \( \gcd\{w_{\ell}, J\} = w_{\ell}J \), the minimum value of \( p \) is \( p = \text{lcm}\{w_{\ell}, J\} / w_{\ell} = J/\gcd(w_{\ell}, J) \).
Example

- In original DFG:
  - Single loop: A->B->C->A with 6 delays
- In 3-unfolded DFG:
  - \( i = (i+6p)\%3 \Rightarrow p=1 \) for \( i=0,1,2 \).
  - Three loops and each loop with 2 delays
- In 4-unfolded DFG:
  - \( i = (i+6p)\%4 \Rightarrow p=2 \) for \( i=0,1,2,3 \).
  - Two loops and each with 3 delays
Property 5.3.2

- **J**-unfolding of a loop with \( w, \) delays in the original DFG leads to
  - each loop in **J**-unfolded DFG contains \( J / \gcd(w, J) \) copies of each node that appears in loop /
  - \( \gcd(w, J) \) loops in the unfolded DFG
  - each loop in **J**-unfolded DFG contains \( w / \gcd(w, J) \) delays

**Proof:**
- From Lemma 5.3.1, the path \( A_i \rightarrow A_{(i+pwl)} \mod J \) forms a loop if and only if \( pw = qJ \) holds.
- Using Lemma 5.3.2, the minimum value of \( p = J / \gcd(w, J) \). That means an unfolded loop contains \( J / \gcd(w, J) \) copies of each node in loop.
- The unfolded DFG contains a total of \( J \) copies of each node, so there must be \( J / (J / \gcd(w, J)) = \gcd(w, J) \) unfolded loops in the **J**-unfolded DFG
- Using Property 5.3.1, each unfolded loop contains \( w / \gcd(w, J) \) delays.
Property 5.3.3

Unfolding a DFG with iteration bound $T_\infty$ results in a J-unfolded DFG with iteration bound $T'_\infty = JT_\infty$.

Proof:

- The iteration bound of the original DFG is

$$T_\infty = \max_l \left\{ \frac{t_l}{w_l} \right\}$$

- The iteration bound of the J-unfolded DFG is

$$T'_\infty = \max_l \left\{ \frac{\left( J / \gcd(w_l, J) \right) t_l}{w_l / \gcd(w_l, J)} \right\} = J \max_l \left\{ \frac{t_l}{w_l} \right\} = JT_\infty$$
Outline

- Introduction
- An Algorithm for Unfolding
- Properties of Unfolding
- Critical Path, Unfolding, and Retiming
- Applications of Unfolding
  - Sample Period Reduction
  - Parallel Processing
- Conclusions
Property on Critical Path

Property 5.4.1

- Consider a path with \( w \) delays in the original DFG. \( J \)-unfolding of this path leads to \((J-w)\) paths with no delays and \( w \) path 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.
Lemma 5.4.1: Unfolding + Retiming

Any feasible clock cycle period that can be obtained by retiming the $J$-unfolded DFG, $G_J$, can be achieved by retiming the original DFG, $G$, directly and then unfolding it by unfolding factor $J$. i.e. $(G_u)_r = (G_r)_u$

Proof:
Outline

- Introduction
- An Algorithm for Unfolding
- Properties of Unfolding
- Critical Path, Unfolding, and Retiming
- Applications of Unfolding
  - Sample Period Reduction
  - Parallel Processing
- Conclusions
Sample Period Reduction

In some cases:

- The DSP program cannot be implemented with the iteration period equal to the iteration bound without the use of unfolding.
- The unfolded DFG can have a sample period equal to the iteration bound of the original DFG.

Case I

- When a node in the DFG possesses the computation time $t_{U,\text{max}} > T_\infty$
  - If the computation time of node U, $t_U > T_\infty$, then $\left\lceil \frac{t_u}{T_\infty} \right\rceil$-unfolding should be used.

Case II

- When the iteration bound is not an integer
  - If a critical loop bound is $t_r/w_r$, where $t_r$ and $w_r$ are mutually coprime, then $w_r$-unfolding should be used.
Case I Example

iteration bound = 3

2-unfolding

3x2/2 = 3 u.t.
Case II Example

iteration bound = 4/3

(4/3)x3/3=4/3 u.t.
Case 3 = Case I + Case II

When the longest node computation time $t_{U,\text{max}}$ is greater than $T_\infty$, and $T_\infty$ is not an integer

- The minimum unfolding factor $J$ that allows the iteration period to equal the iteration bound can be determined by the following equation

$$J \cdot T_\infty \geq t_{U,\text{max}}$$

- Ex: Assume $T_\infty = 4/3$ and $t_{U,\text{max}} = 6$

Sol:

$$J \cdot \frac{4}{3} \geq 6 \implies J = 6$$
Parallel Processing

Word-level Parallel Processing

- Unfolding a word-serial architecture by J creates a word-parallel architecture that processes J words per clock cycle

Bit-level Parallel Processing

- Bit-serial processing
  - One bit is processed per clock cycle and a complete word is processed in W clock cycles, where W is the word-length.

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

- Digit-serial processing
  - N bits are processed per clock cycle and a word is processed in W/N clock cycles, where N is referred to as the digit size
Bit-level Processing Representation

Bit-parallel

Bit-serial

Digit-serial (Digit-size = 2)
Word-Level Parallel Processing
Unfolding Bit-Serial Architecture Issues

A bit-serial adder $s = a + b$ with word-length $W = 4$

How to obtain the bit-level parallel architecture?
How to apply unfolding algorithm for bit-level parallel processing?
How to unfold the edge with a switch?
Two basis assumptions

- The word-length $W$ is a multiple of the unfolding factor $J$, i.e., $W = W'J$.
- All edges into and out of the switches have no delays.

Two steps to unfold

- Step 1: Write the switching instance as
  \[ Wl + u = J(W'l + \left\lfloor \frac{u}{J} \right\rfloor) + (u \% J) \]

- Step 2: Draw an edge with no delays in the unfolded graph from the node $U_{u \% J}$ to the node $V_{u \% J}$, which is switched at time instance $(W'l + \left\lfloor \frac{u}{J} \right\rfloor)$.
Bit-level Parallel Processing
(J=3, W=12)

\[12l + 1 = 3(4l + \left\lfloor \frac{1}{3} \right\rfloor) + (1 \mod 3) = 3(4l + 0) + 1\]

\[12l + 7 = 3(4l + \left\lfloor \frac{7}{3} \right\rfloor) + (7 \mod 3) = 3(4l + 2) + 1\]

\[12l + 9 = 3(4l + \left\lfloor \frac{9}{3} \right\rfloor) + (9 \mod 3) = 3(4l + 3) + 0\]

\[12l + 11 = 3(4l + \left\lfloor \frac{11}{3} \right\rfloor) + (11 \mod 3) = 3(4l + 3) + 2\]
Bit-level Parallel Processing
(J=3, W=6)

6l+0 = 3(2l+0) + 0 (B0)
6l+1 = 3(2l+0) + 1 (D1)
6l+2 = 3(2l+0) + 2 (B2)
6l+3 = 3(2l+1) + 0 (B0)
6l+4 = 3(2l+1) + 1 (B1)
6l+5 = 3(2l+1) + 2 (D2)
4 Bit-Serial Adder Unfolding (2/2)

- Bit-parallel adder
- Digit-serial adder
Conclusions

- Unfolding algorithm is a graph-based transformation technique
  - Reveal hidden concurrences of the program
  - Retrieve a smaller iteration period of the algorithm

- Unfolding and retiming

- Applications for sample period reduction

- Applications for parallel processing
  - Word-level
  - Bit-level
    - Bit-serial
    - Bit-parallel
    - Digit-serial