Multiple-Data Processing

- Create a program with more than one iteration, e.g., J loops unrolling
- Example: Loop unrolling + software pipelining

<table>
<thead>
<tr>
<th>Operation</th>
<th>Clock Cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
</tr>
</tbody>
</table>

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

- Parallel processing
- Pipelined processing

Data Dependence

- Parallel processing requires NO data dependence between processors
- Pipelined processing will involve inter-processor communication
In a J-unfolded system, each delay is J-slow. That is, if input to a delay element is $x(kJ+m)$, then the output is $x((k-1)J+m) = x(kJ+m-J)$.

- Block processing
  - The number of inputs processed in a clock cycle is referred to as the block size.

- At the $k$-th clock cycle, three inputs $x(3k)$, $x(3k+1)$, and $x(3k+2)$ are processed simultaneously to generate $y(3k)$, $y(3k+1)$, and $y(3k+2)$.
I/O Conversion

- Serial to parallel converter

\[ x(n) \rightarrow 3k \rightarrow T/3 \rightarrow x(3k+1) \rightarrow x(3k+2) \]

- Parallel to serial converter

\[ y(n) \rightarrow 3k \rightarrow T/3 \rightarrow y(3k) \rightarrow y(3k+1) \rightarrow y(3k+2) \]

General approach for block processing
Mathematical Formulation

- e.g. \( y(n) = ay(n-9) + x(n) \)
- 2-parallel
  \[
  Y(2k) = ay(2k-9) + x(2k) \\
  Y(2k+1) = ay(2k-8) + x(2k+1)
  \]
- In 2-parallel SDFG, one active clock edge leads two samples
  \[
  Y(2k) = ay(2(k-5)+1) + x(2k) \\
  Y(2k+1) = ay(2(k-4)+0) + x(2k+1)
  \]
- Dependency with less than # parallelism of sample delays can be implemented with internal routing

Unfolding the DFG

\[ T = T_s \]

\[ T = J T_s \]

Not trivial, even for a simple graph
Block Processing for FIR Filter

- One form of vectorized parallel processing of DSP algorithms. (Not the parallel processing in most general sense)
- Block vector: \([x(3k) \ x(3k+1) \ x(3k+2)]\)
- Clock cycle: can be 3 times longer
- Original (FIR filter):
  \[y(n) = ax(n) + bx(n-1) + cx(n-2)\]
- Rewrite 3 equations at a time:
  \[
  \begin{bmatrix}
  y(3k) \\
y(3k + 1) \\
y(3k + 2)
  \end{bmatrix} =
  \begin{bmatrix}
  x(3k) \\
x(3k + 1) \\
x(3k + 2)
  \end{bmatrix}
  \begin{bmatrix}
  a \\
b \\
c
  \end{bmatrix}
  \begin{bmatrix}
  x(3k - 1) \\
x(3k - 1) \\
x(3k)
  \end{bmatrix}
  \begin{bmatrix}
  + \\
  + \\
  +
  \end{bmatrix}
  \begin{bmatrix}
  y(3k - 2) \\
y(3k - 1) \\
y(3k)
  \end{bmatrix}
  \]

Block Processing
Block Processing for IIR Digital Filter

- **Original formulation:**
  \[ y(n) = a \cdot y(n - 2) + x(n) \]

- **Rewrite:**
  \[ y(2k) = ay(2k - 2) + x(2k) \]
  \[ y(2k + 1) = ay(2k - 1) + x(2k + 1) \]

- **Vector formulation:**
  \[
  \begin{bmatrix}
  x(k) \\
  y(k)
  \end{bmatrix}
  =
  \begin{bmatrix}
  x(2k) \\
  x(2k + 1)
  \end{bmatrix}
  \begin{bmatrix}
  a \\
  1
  \end{bmatrix}
  \]
  \[ y(k) = ay(k - 1) + x(k) \]

Block IIR Filter

Clock period not equal to sampling period

\[ T_{\text{clk}} \neq T_{\text{sample}} \]
Timing Comparison

- Pipelining
  - MAC
    - $x(1)$ $x(2)$ $x(3)$ $x(4)$
    - $y(1)$ $y(2)$ $y(3)$ $y(4)$
  - Add
    - $x(1)$ $x(2)$ $x(3)$ $x(4)$ $x(5)$ $x(6)$ $x(7)$ $x(8)$
    - $y(1)$ $y(2)$ $y(3)$ $y(4)$ $y(5)$ $y(6)$ $y(7)$ $y(8)$
  - Mul
    - $x(2)$ $x(4)$ $x(6)$ $x(8)$
    - $x(1)$ $x(3)$ $x(5)$ $x(7)$

- Block processing
  - $x(1)$ $x(2)$ $x(3)$ $x(4)$ $x(5)$ $x(6)$ $x(7)$ $x(8)$

Definitions

- Unfolding is the process of unfolding a loop so that several iterations are unrolled into the same iteration.
- Also known as (a.k.a.)
  - Loop unrolling (in compilers for parallel programs)
  - Block processing
- Applications
  - Reducing sampling period to achieve iteration bound (desired throughput rate) $T_c$.
  - Parallel (block processing) to execute several iterations concurrently.
  - Digit-serial or bit-serial processing
Unfolding the DFG

- $y(n) = ay(n-9) + x(n)$

- Rewrite the algorithm formulation:
  
  $y(2k) = ay(2k-9) + x(2k)$
  
  $y(2k+1) = ay(2k-8) + x(2k+1)$

  $y(2k) = ay(2(k-5)+1) + x(2k)$
  
  $y(2k+1) = ay(2(k-4)) + x(2k+1)$

- After $J$-folded unfolding, the clock period $T = J T_s$, where $T_s$ is the data sampling period.

Timing Diagram

- Above timing diagram is obtained assuming that the sampling period $T_s$ remains unchanged. Thus, the clock period $T$ is increased $J$-fold.
- Since $9/2$ is not an integer, output $(y(0), y(1))$ will be needed by two different future iterations, $4T$ and $5T$ later.
Another DFG Unfolding Example

Step 1.Duplicate J copies of each node

<table>
<thead>
<tr>
<th>i</th>
<th>w</th>
<th>((i+w)%J)</th>
<th>((i+w)/J)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

Step 2. Add all edges with 0 delay on them.

Another DFG Unfolding Example

Step 1. Duplicate J copies of each node

<table>
<thead>
<tr>
<th>i</th>
<th>w</th>
<th>((i+w)%J)</th>
<th>((i+w)/J)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

Step 2. Add all edges with 0 delay on them.
Another DFG Unfolding Example

<table>
<thead>
<tr>
<th>i</th>
<th>w</th>
<th>((i+w)%J)</th>
<th>((i+w)/J)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

Step 3. Use table on the left to figure out edges with delays.

Unfolding Transformation

- For each node U in the original DFG, draw J node U₀, U₁,..., U₉⁻₁.
- For each edge UV with w delays in the original DFG, draw the J edges UᵢV((i+w)\%J) with floor\(((i+w)/J)\) delays for i=0,1,..., J-1.

Example

- Unfolding of an edge with w delays in the original DFG produces J-w edges with no delays and w edges with 1 delay in J-unfolded DFG for w < J.
- Unfolding preserves precedence constraints of a DSP algorithm.
Precedence Preservation

Original DFG

\[ U \xrightarrow{m} V \]

J-unfolded DFG

\[ U \xrightarrow{\frac{i+w}{J}} V_{(i+w)\times J} \]

<table>
<thead>
<tr>
<th>Iteration</th>
<th>( U )</th>
<th>( V )</th>
</tr>
</thead>
<tbody>
<tr>
<td>J-unfolded DFG</td>
<td>( k )</td>
<td>( k + \frac{i+w}{J} )</td>
</tr>
<tr>
<td>Original DFG</td>
<td>( Jk + l )</td>
<td>( J\left(k + \frac{i+w}{J}\right) + (i+w)\times J )</td>
</tr>
</tbody>
</table>

\[ \frac{i+w}{J} + (i+w)\times J - l = (i+w) - l = w \]

Delay Preservation

- Unfolding preserves the number of delays in a DFG
- Let \( w = m \times J + n \), where \( m, n \in \{0 \cup N\} \quad 0 \leq n \leq (J-1) \)

\[
\begin{align*}
\frac{w}{J} &= m \\
\frac{w + (J - n - 1)}{J} &= m \\
\frac{w + (J - n)}{J} &= m \times J + n \\
\frac{w + (J - 1)}{J} &= m + 1 \\
\end{align*}
\]
Example

- Unfold the following DFG using folding factor 2 and 5

![DFG Diagram]

Properties of Unfolding

- Unfolding preserves the number of registers (delays) in a DFG.
- For a loop with \( w \) delays in a DFG that has been unfolded \( J \) times, it leads to:
  - \( \text{g.c.d.}(w, J) \) loops in the unfolded DFG, with each of these loops containing \( W/\text{g.c.d.}(w,J) \) delays and \( J/\text{g.c.d.}(w,J) \) copies of each node that appear in the original loop.
- Unfolding a DFG with iteration bound \( T_n \) results in a \( J \)-folded DFG with iteration bound \( JT_n \).
- A path with \( w \) (\( \leq J \)) delays in a DFG will lead to \( J-w \) paths with no delays, and \( w \) paths with 1 delay each in the \( J \)-unfolded DFG.
- Any clock period that can be achieved by retiming a \( J \)-unfolded DFG can be achieved by retiming the original DFG and followed by \( J \)-unfolding.
When a Loop is Unfolded

- A loop \( \ell \) with \( w \) delays in a DFG
- Travel the loop \( A \rightarrow A \) \( p \) times \( \rightarrow \) also a loop with \( pw \) delays
- In \( J \)-unfolded DFG, consider the path \( A_i \rightarrow A_{(i+pw) \% J} \). It is a loop if \( i=(i+pw)\%J \). This implies that \( J \mid pw \)
- The smallest \( p = \frac{J}{\gcd(J, w)} \). That is, in \( J \)-unfolded DFG, one can travel the loop \( A \rightarrow A \frac{J}{\gcd(J, w)} \) times.
- Recall that there are totally \( J \) copies of node \( A \). Hence, there are \( J/\gcd(J, w) \) loops and each loop contains \( \frac{w}{\gcd(J, w)} \) delays.
- The iteration bound in \( J \)-unfolded DFG is then

\[
T_w' = \max \left\{ \frac{J}{\gcd(j, w_i) t_j} \right\} = \max \left\{ \frac{J}{w_i t_j} \right\} = JT_w
\]

When a Path is Unfolded

- If \( w < J \), then a path containing \( w \) delays within a DFG will lead to \((J-w)\) paths with no delays and \( w \) paths with 1 delay in the \( J \)-unfolded DFG.
- If \( w \geq J \), then the path leads to \( J \) paths with one or more delays in the \( J \)-unfolded DFG. This implies that these paths are not critical.
- Assume that the critical path of the \( J \)-unfolded DFG is \( c \). If \( D(U, V) \geq c \), then \( W_t(UV) = W(UV) + r(V) - r(U) \geq J \)
- Any feasible clock cycle period that can be obtained by retiming the \( J \)-unfolded DFG can be achieved by retiming the original DFG directly and followed by \( J \)-unfolding.
When a Path is Unfolded

- Suppose $r'$ is a legal retiming for the J-unfolded DFG, $G_J$, which leads to critical path $c$.
- Let $r(U) = \sum_{i} r'(U_i), 0 \leq i \leq J-1$.
  - $r$ is a feasible retiming for the original DFG, $G$.
  - The retiming leads to a critical path $c$.

Consider an edge $U \rightarrow V$ with $w$ delays in $G$.

Since $r'$ is legal retiming for $G_J$ and leads to a critical path $c$, then

1. $r'(U_i) - r'(V_{(i+1)}) \leq \frac{t + w}{J}$ feasible constraint
2. $r'(U_i) - r'(V_{(i+1)}) \leq \frac{t + w}{J} - 1$, if $D(U_i \rightarrow V_{(i+1)}) > c$

Sample Period Reduction

- **Case1**: A node in the DFG having computation time greater than $T$
- **Case2**: Iteration bound is not an integer
- **Case3**: Longest node computation is larger than the iteration $T$, and $T$ is not an integer
Case 1

• Critical path dominates, since a node computation time is more than iteration bound

\[ T_\infty = \max_{l \in L} \left\{ \frac{t_l}{w_l} \right\} \]

\[ = \max_{l \in L} \left\{ \frac{6}{3}, \frac{6}{2} \right\} = 3 \]

<4, max node time

Retiming cannot be used to reduce sample period

Sample Period Reduction

• Rule of Thumb: \( \left[ \frac{t_r}{T_\infty} \right] \) – unfolding should be used

\( t_r = 4 \) and \( T_\infty = 3 \)

\( \left[ \frac{4}{3} \right] = 2 \) - unfolding

But two Samples!

\( T_\infty = 6 \)

But two Samples!
Case 2

- Iteration period cannot achieve the iteration bound

$$T_\infty = \max_{t \in L} \frac{I_t}{w_t} = \frac{4}{3}$$

If a critical loop bound is of the form $t/w$, where $t$ and $w$ are mutually co-prime, then $w$-unfolding should be used.

Unfolding of 3

Sample Period Reduction

$$T_\infty = 4$$

and 3 samples gives minimum sample period $4/3$
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 $T_{nn}$, and $T_{nn}$ is not an integer

The minimum $J$ that achieves the iteration bound is the minimum value of $J$ such that $J T_{nn}$ is an integer and is greater or equal to the longest node computation time

Parallel Processing

- Parallel processing can be performed by unfolding

\[ x_{2k+1} \rightarrow x_{2k} \rightarrow D \rightarrow b_2 \times x_{2k-2} \rightarrow b_1 \rightarrow b_2 \times y_{2k} \rightarrow y_{2k-1} \]
Bit-Level Parallel Processing

Bit-Parallel

Bit-Serial
(Digit-size = 2)

Digit-Serial

Bit-Serial

Digit-Serial

Bit-Parallel
Bit-Serial Adder

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 $W'I + u = J(W'I + \lfloor u/J \rfloor) + (u \% J)$.
  - Draw an edge from the node $U_{u \% J}$ to $V_{u \% J}$, which is switched at time instance $(W'I + \lfloor u/J \rfloor)$. 

\[ U \quad \bigg\downarrow \quad V \]
Example

Write the switching instance as
\[ WI + u = J(\lfloor WI + \lfloor u/J \rfloor \rfloor) + (u \% J) \]
\[ 9 + 1 = 3(3l + \lfloor 1/3 \rfloor) + (1 \% 3) = 3(3l + 0) + 1 \]
\[ 9 + 5 = 3(3l + \lfloor 5/3 \rfloor) + (5 \% 3) = 3(3l + 1) + 2 \]

Switched at time instances

Edges between Nodes

Example

Write the switching instance as
\[ WI + u = J(\lfloor WI + \lfloor u/J \rfloor \rfloor) + (u \% J) \]
\[ 9 + 1 = 3(3l + \lfloor 1/3 \rfloor) + (1 \% 3) = 3(3l + 0) + 1 \]
\[ 9 + 5 = 3(3l + \lfloor 5/3 \rfloor) + (5 \% 3) = 3(3l + 1) + 2 \]

Edges between Nodes

Draw an edge from the nodes \( u \equiv v \), i.e., \( U_1 \equiv V_1 \) and \( U_2 \equiv V_2 \)
Example

\[ 9t+1 = 3\{3t + \lfloor 1/3 \rfloor \} + (1\%3) = 3\{3t + 1\} + 1 \]
\[ 9t+1 = 3\{3t + \lfloor 5/3 \rfloor \} + (5\%3) = 3\{3t + 1\} + 2 \]

Switched at time instances

Switched at time instance \( W \{ U/J \} \), i.e.

\[ U_1 \neq V_1 \text{ at } (3t+0) \text{ and } U_2 \neq V_2 \text{ at } (3t+1) \]

Example:

\[ 12t + 1, 7, 9, 11 \]

Unfolding by 3

To unfold the DFG by \( J = 3 \), the switching instances are as follows

\[ 12t + 1 = 3(4t + 0) + 1 \]
\[ 12t + 7 = 3(4t + 2) + 1 \]
\[ 12t + 9 = 3(4t + 3) + 0 \]
\[ 12t + 11 = 3(4t + 3) + 2 \]
Switches with Delays

Unfolding a DFG containing an edge having a switch and a positive number of delays is done by introducing a dummy node.

Switch with Delays

Unfolding a DFG containing an edge having a switch and a positive number of delays is done by introducing a dummy node.
If Wordlength is not a Multiple of J

- If the word-length, \( W \), is not a multiple of the unfolding factor, \( J \), then expand the switching instances with periodicity \( \text{lcm}(W,J) \)
- Example: Consider \( W=4 \), \( J=3 \). Then \( \text{lcm}(4,3) = 12 \). For this case, \( 4l = 12l + \{0,4,8\} \), \( 4l+1 = 12l + \{1,5,9\} \), \( 4l+2 = 12l + \{2,6,10\} \), \( 4l+3 = 12l + \{3,7,11\} \). All new switching instances are now multiples of \( J=3 \).