Unfolding

- Unfolding is a transformation technique to change the program into another program such that one iteration in the new program describes more than one iteration in the original program.
- Unfolding, AKA loop unrolling in CSE4201.
- Unfolding factor of $j$ means that one iteration in the new program describes $j$ iterations in the old one.
Unfolding

- Also used to design bit parallel and word parallel architectures from bit serial and word serial architecture.

\[
y(n) = ay(n - 9) + x(n)
\]
\[
y(2k) = ay(2k - 9) + x(2k)
\]
\[
y(2k + 1) = ay(2k - 8) + x(2k + 1)
\]
\[
y(2k) = ay(2k - 9) + x(2k)
= ay(2(k - 5) + 1) + x(2k)
\]
\[
y(2k + 1) = ay(2k - 8) + x(2k + 1)
= ay(2(k - 4) + 0) + x(2k + 1)
\]
Unfolding

- In the unfolded system, each delay is $J$-slow.
- That means if the input to a delay element is the signal $x(kJ+m)$, the output is $x((k-1)J+m) = x(kJ-J+m)$

Algorithm for Unfolding

- 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 $\left\lfloor \frac{i+w}{J} \right\rfloor$ delays for $i = 0, 1, \ldots, J-1$
- For the input nodes, $A_i$ corresponds to input signal $x(jk+i)$
- For $J > w$, an edge with $w$ delays will result in $J-w$ edges with zero delay and $w$ edges with 1 delay
Examples

Example
Properties

• Unfolding a DFG with iteration bound $T_\infty$ results in a $J$-unfolded DFG with iteration bound $T_\infty' = JT_\infty$

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.
Properties

• 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:

Applications

• Unfolding can be used in
  - Sample period reduction
  - Parallel processing
Sample Period Reduction

- Any implementation of a DSP program can never achieve an iteration period less than $T_\infty$.
- Sometimes we can not achieve that lower bound (2 reasons):
  - When one node has a computation time greater than $T_\infty$ (can not be split).
  - $T_\infty$ is a fraction.

Example
Word Level Parallel Processing

- We start with a DSP at the word level
- We can use unfolding to replicate the design $J$ times ($J$ unfolding).
- Example
Bit Level Parallel Processing

- Bit level parallel processing can increase the speed (reduce the sample time) by processing more than one bit at a time.
- Digit serial parallel processing is when we process W bits at a time where W is the word length.
- Usually involves switching (multiplexers)
Bit Serial Adder

- Consider the following adder, \( W=4 \)

\[
\begin{align*}
\text{Edges with Switches} \\
\text{Assumptions} \\
&\text{The word-length } W \text{ is a multiple of the unfolding factor } J, \text{ i.e., } \\
&W=W'J \\
&\text{All edges into and out of the switches have no delays} \\
\text{Write the switching instance as} \\
Wl + u &= J(W'l + \left\lfloor \frac{u}{J} \right\rfloor) + (u\%J) \\
\text{Step2: Draw an edge with no delays in the unfolded} \\
&\text{graph from the node } U_{u\%J} \text{ to the node } V_{u\%J}, \text{ which is} \\
&\text{switched at time instance } (W'l + \left\lfloor \frac{u}{J} \right\rfloor). 
\end{align*}
\]
Example—Binary adder

York University CSE

\[
\begin{align*}
&\begin{array}{c}
a_3 a_2 a_1 a_0 \\
b_3 b_2 b_1 b_0
\end{array} \\
&\begin{array}{c}
s_3 s_2 s_1 s_0
\end{array}
\end{align*}
\]

\[
\begin{align*}
&\begin{array}{c}
4l+0 \quad 4l+1,2,3
\end{array} \\
&\begin{array}{c}
0 \quad 0
\end{array}
\end{align*}
\]

\[
\begin{align*}
&\begin{array}{c}
J=4
\end{array} \\
&\begin{array}{c}
J=2
\end{array}
\end{align*}
\]

Example Binary adder

York University CSE

\[
\begin{align*}
&\begin{array}{c}
a_3 a_2 a_1 a_0 \\
b_3 b_2 b_1 b_0
\end{array} \\
&\begin{array}{c}
s_3 s_2 s_1 s_0
\end{array}
\end{align*}
\]

\[
\begin{align*}
&\begin{array}{c}
4l+0 \quad 4l+1,2,3
\end{array} \\
&\begin{array}{c}
0 \quad 0
\end{array}
\end{align*}
\]

\[
\begin{align*}
&\begin{array}{c}
\text{carry out}
\end{array} \\
\end{align*}
\]