Chapter 5
Unfolding

CSE4210 Winter 2012

Mokhtar Aboelaze

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.

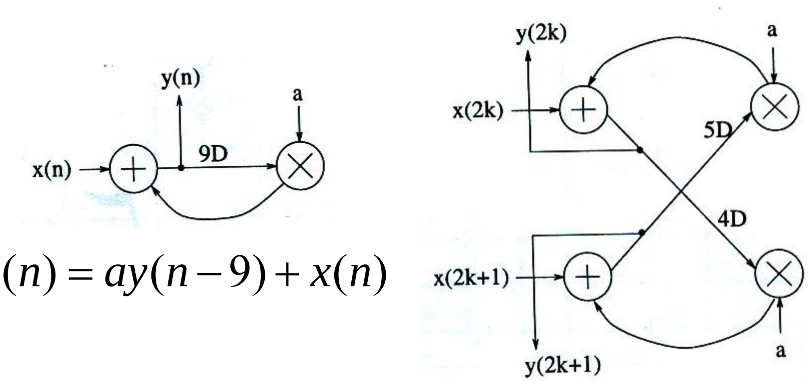
Unfolding

- 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



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

$$y(2k) = ay(2(k-5)+1) + x(2k)$$
$$y(2k+1) = ay(2(k-4)+0) + x(2k+1)$$

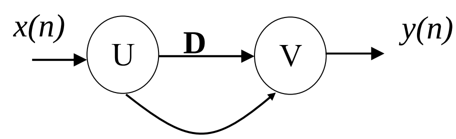
Unfolding

- 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,..,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

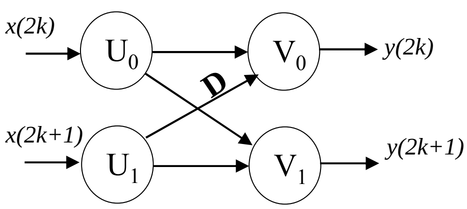
Examples



y(n)=x(n)+x(n-1)

y(8)=x(8)+x(7)

y(9)=x(9)+x(8)



y(2k)=x(2k)+x(2(k-1)+1)

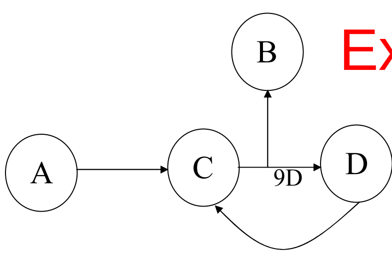
y(2k+1)=x(2k)+x(2k+1)

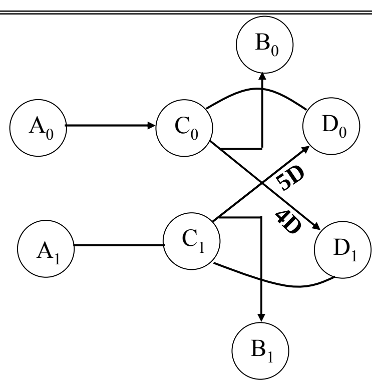
k=4

y(8)=x(8)+x(7)

y(9)=x(8)+x(9)

Example





$U_i \rightarrow V_{(i+w)\%j}$ with $\left\lfloor \frac{i+w}{j} \right\rfloor$

Example

YORK UNIVERSITY CSE4210

Example

YORK UNIVERSITY CSE4210

Unfolding

- Prove that the unfolded graph *preserves dependence* of the DSP program.

iteation k of U consumed by $(k + w)$ of V

In the unfolded graph

$$\text{iteration } k \text{ of } U_i \text{ consumed by} \left(\mathrm{k} + \left\lfloor \frac{i+w}{j} \right\rfloor \right) \text{of } V_{(i+w)\% j}$$

- Show they are the same

YORK UNIVERSITY CSE4210

Unfolding

- Proof

the k^{th} iteration of node U_i in the $j-$folded graph execcutes the $(jk+i)$ iteration of the original graph.

$j(k + \lfloor (i+w)/j \rfloor) + (i+w)\% j - (jk+i)$

$j(\lfloor (i+w)/j \rfloor) + (i+w)\% j - i = w$

Properties

- Unfolding preserves the number of delays in a DFG $\left\lfloor \frac{w}{J} \right\rfloor + \left\lfloor \frac{w+1}{J} \right\rfloor + \ldots + \left\lfloor \frac{w+J-1}{J} \right\rfloor = w$

Properties

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

Properties

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

Properties

- 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:

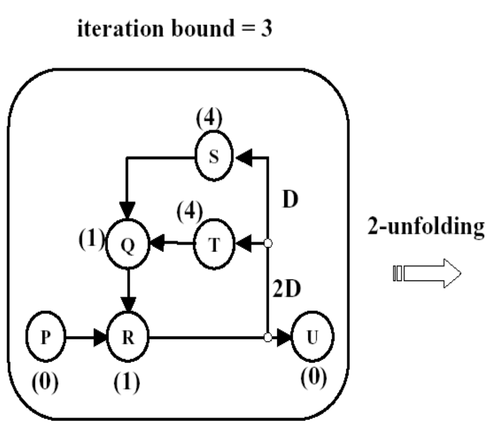
Applications

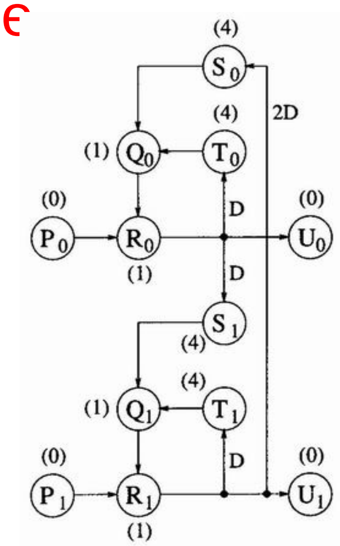
- Unfolding can be used in
  - Sample period reduction
  - Parallel processing

Sample Period Reduction

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

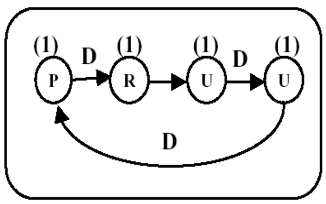
Example



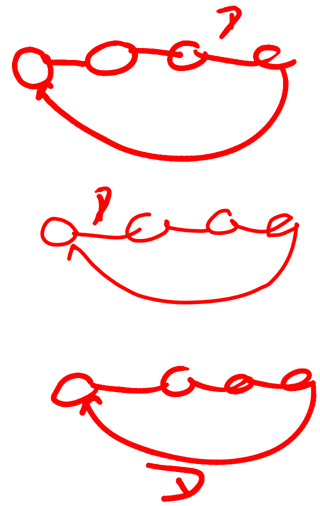


Example

iteration bound = 4/3



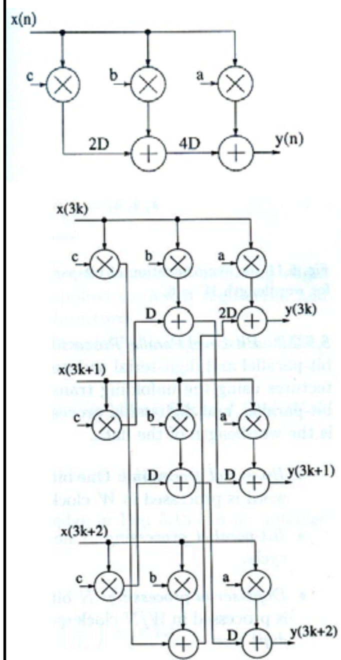
3-unfolding

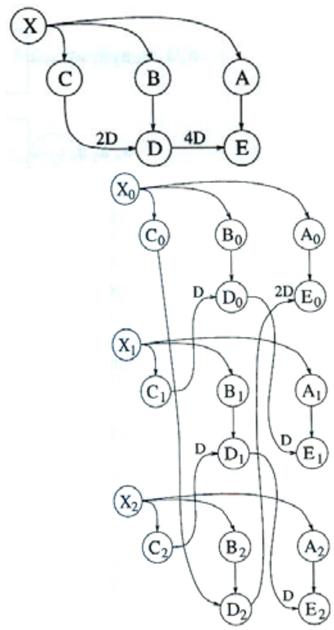


YORK UNIVERSITY CSE4210

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



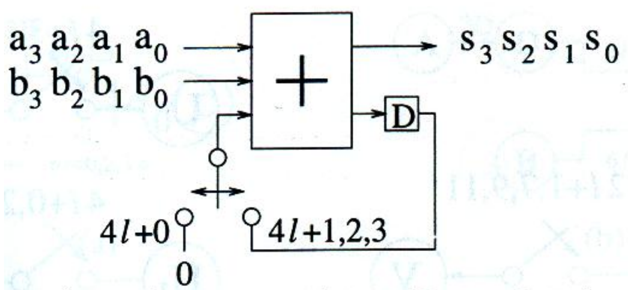


Bit Level Parallel Processing

- Bit level parallel processing can increase the speed (reduce the sample time) be 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



Edges with Switches

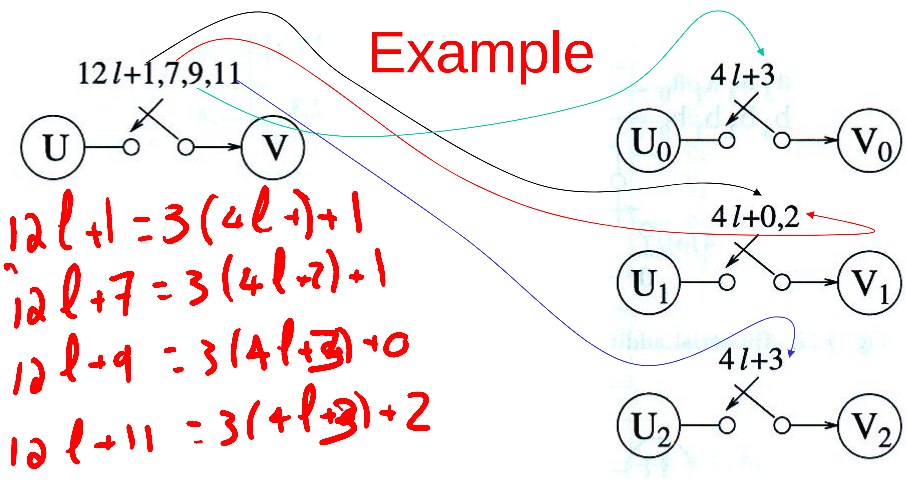
- Assumptions
  - The word-length **W** is a multiple of the unfolding factor **J,** i.e., **W=W'J**
  - All edges into and out of the switches have no delays
- Write the switching instance as

$$Wl + u = J(W'l + \left\lfloor \frac{u}{J} \right\rfloor) + (u\%J)$$

- Step2: 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

$$(W'l + \left\lfloor \frac{u}{J} \right\rfloor)$$

Example



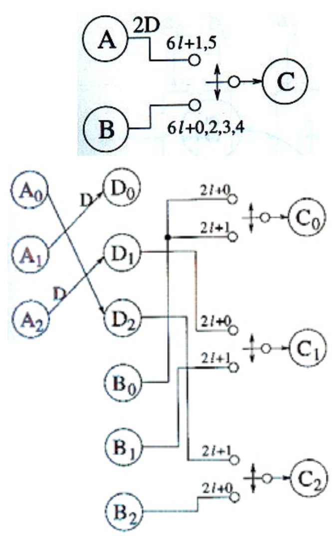
$12l+1 = 3(4l+0)+1$

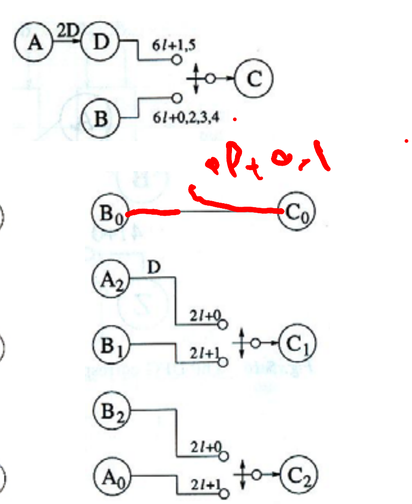
$12l+7 = 3(4l+2)+1$

$12l+9 = 3(4l+3)+0$

$12l+11 = 3(4l+3)+2$

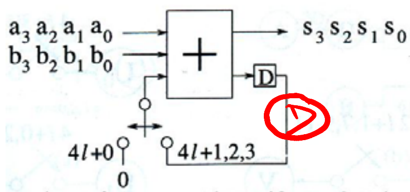
Example

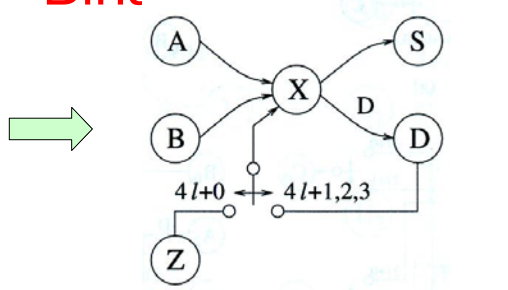


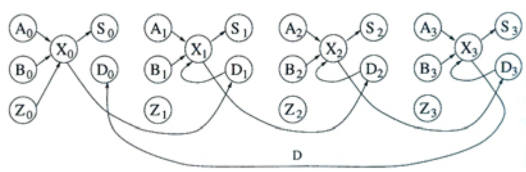


YORK UNIVERSITY CSE4210

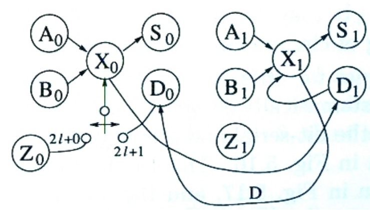
Example—Binary adder







J=4



J=2

YORK UNIVERSITY CSE4210

Example Binary adder