CSE4210 Architecture & Hardware for DSP

Chapter 4

Iterative Bound

Instructor: Prof. Peter Lian
Department of Electrical Engineering & Computer Science
Lassonde School of Engineering
York University

CSE4210 Architecture & Hardware for DSP

Graphical Representation

DSP Algorithms

- DSP algorithms are described by nonterminating programs, which execute the same code repetitively.

  $Y(n)=a*x(n)+b*x(n-1)+c*x(n-2)$, for n=1 to ∞
- An iteration – execution of all the computations in the algorithm once.
- Critical path – the longest path between inputs and outputs in combinational logic circuit.
- Latency – the difference between the time an output is generated and the time at which its corresponding input was received by the system.

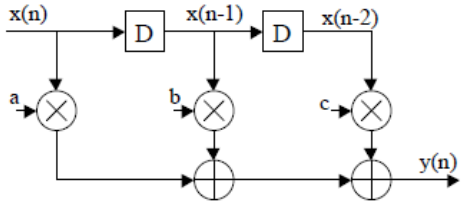
Representation of DSP Algorithms

- Graphical representations are efficient for investigating and analyzing data flow properties of DSP algorithms.
- Good for map DSP algorithms to hardware implementations
- Four methods for graphical representation
  - Block diagram
  - Signal-flow graph (SFG)
  - Data-flow graph (DFG)
  - Dependence graph (DG)

Block Diagram

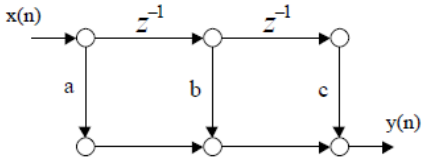
- Consists of functional blocks connected with directed edges, which represent data flow from its input block to its output block.
- Edges may or may not contain delay elements.
- Example

$Y(n)=a*x(n)+b*x(n-1)+c*x(n-2)$



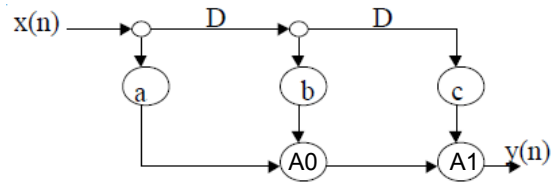
Signal-Flow Graph (SFG)

- SFG: a collection of nodes and directed edges
  - Nodes: represent computations and/or task, sum all incoming signals
  - Directed edge (j,k): denotes a linear transformation from the input signal at node j to the output signal at node k.
  - Linear SFGs can be transformed into different forms without changing the system functions.
  - Usually used for linear time-invariant DSP systems



Data-Flow Graph (DFG) (1)

- DFG: nodes represent computation(or functions or subtasks), while the directed edges represent data paths (data communications between nodes), each edge has a nonnegative number of delays associated with it.
- DFG: captures the data-driven property of DSP algorithm: any node can fire (perform its computation) whenever all its input data are available.



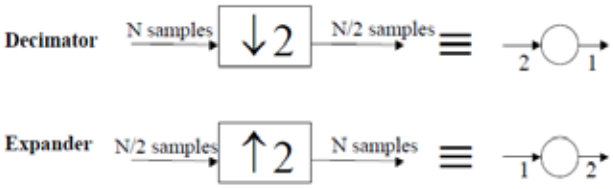
DFG Constraints(2)

- Each edge describes a precedence constraint between two nodes
  - Intra-iteration precedence constraint: if the edge has zero delays.
  - Inter-iteration precedence constraint: if the edge has one or more delays.
- DFGs and Block Diagrams can be used to describe both linear single-rate and nonlinear multi-rate DSP systems.

Examples of DFG

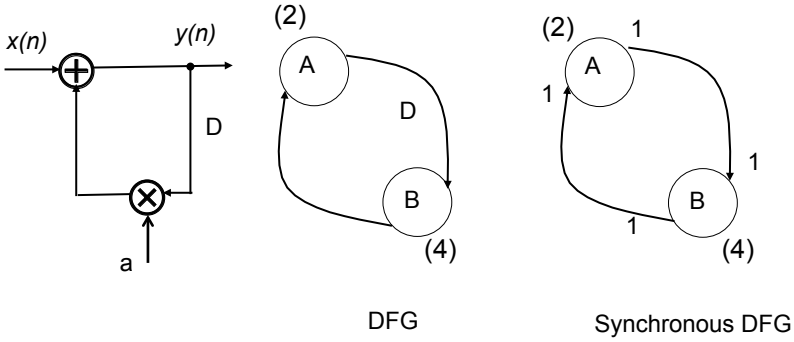
- Nodes are complex blocks (coarse-grain)

- Nodes can describe expanders/decimators in multi-rate DFGs



Example of DFG(2)

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



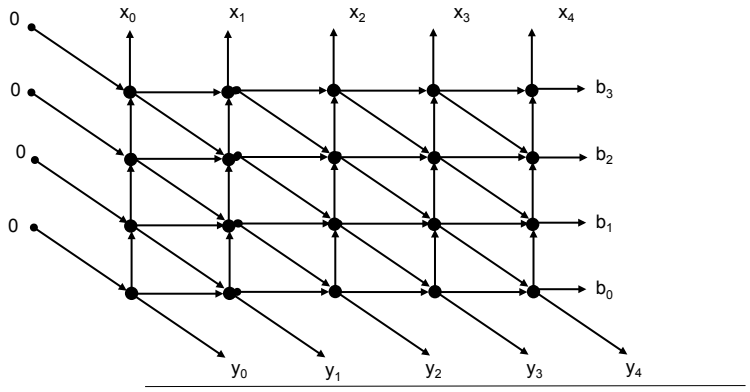
Synchronous DFG (SDFG)

- SDFG is a special case of data-flow graph.
- In SDFG, the number of data samples produced or consumed are specified a priori.
- For example, node B needs 1 data unit to fire and produces one data unit after completion.
- In multi-rate systems, that number could be greater than 1.
- By using node replication, a multi-rate system could be changed to a single-rate system.

Dependence Graph (DG)

- DG is a directed graph that shows the dependence of the computations in an algorithm
- The nodes represent computations and the edges represent precedence constraints among nodes.
- The DFG nodes are executed repetitively, while nodes in a dependence graph contains computations for all iterations.
- DFs are used for systolic array design.

Dependence Graph



Iteration Bound

Iteration bound

- Iteration: execution of all computations in the algorithm once.
- Iteration period: the time required to perform the iteration (sample period).
- Feedback imposes an inherent bound on the iteration period,
- Iteration bound is a characteristic of the representation of an algorithm in the form of DFG. Different representations of the same algorithm may lead to different iteration bounds.

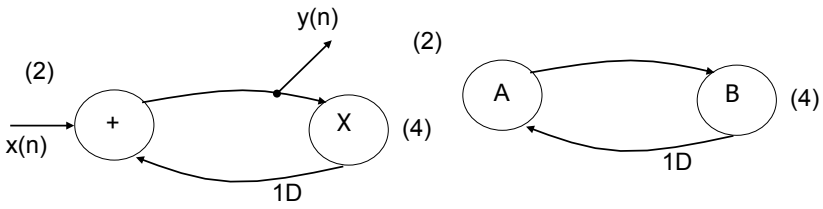
Iteration bound

- The feedback imposes an inherent fundamental lower bound on the achievable iteration period.
- It is not possible to achieve iteration period less than the iteration bound even if we have an infinite processing power.

Determine Iteration Bound

- Edges describe a precedence constraints
  - intra-iteration denotes "→"
  - inter-iteration denotes "⇒"
- Critical path is the path with the longest computation time among all paths that contains no delay.
- A non-recursive DFG contains no loops
- A recursive DFG contains at least one loop.
- A loop is a directed path that begins and ends at the same node.

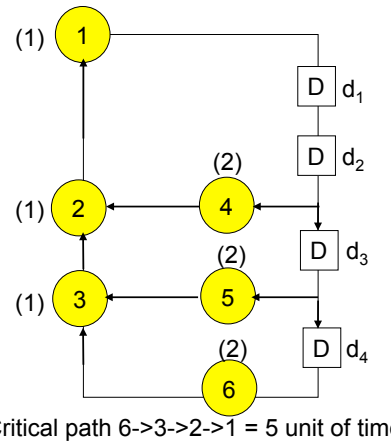
Precedence Constraints



- The edge from A to B enforces the intra-iteration precedence, the k^{th} iteration of A must be done before the k^{th} iteration of B. $A_K \rightarrow B_K$
- The edge from B to A enforces the inter-iteration precedence. The k^{th} iteration of B must be executed before the $(k+1)^{th}$ iteration of A. $B_K \Rightarrow A_{K+1}$
- $A_0 \rightarrow B_0 \Rightarrow A_1 \rightarrow B_1 \Rightarrow A_2 \rightarrow B_2$ ….

Critical Path

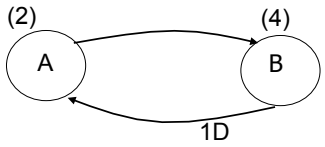
Non-recursive DFG



Critical path 6->3->2->1 = 5 unit of time (u.t.)

5->3->2->1 5 u.t.

Recursive DFG

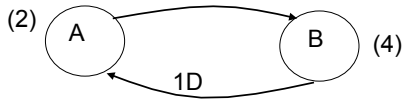


Critical Path A->B 6 u.t.

Determine Iteration Bound

- For recursive DFG, there is a fundamental lower bound "*iteration bound*" T_∞
- Loop bound of the *l*-th loop = t_l/w_l
  - t_l is loop computation time,
  - w_l is the delay in the loop.
- The critical loop is the loop with the maximum loop bound.
- The loop bound of the critical loop is the iteration bound.

Iteration bound: Example



Precedence

$A_0 \rightarrow B_0 \Rightarrow A_1 \rightarrow B_1 \Rightarrow A_2 \rightarrow B_2 \Rightarrow A_3 \rightarrow B_3$

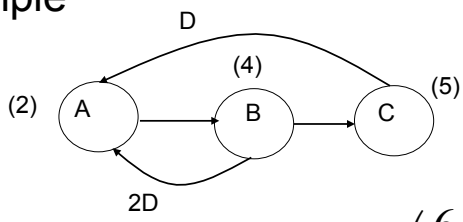
If 2D instead of 1D; loop bound =6/2=3

$A_0 \rightarrow B_0 \Rightarrow A_2 \rightarrow B_2 \Rightarrow A_4 \rightarrow B_4 \Rightarrow A_6 \rightarrow B_6$

$A_1 \rightarrow B_1 \Rightarrow A_3 \rightarrow B_3 \Rightarrow A_5 \rightarrow B_5 \Rightarrow A_7 \rightarrow B_7$

Iteration bound: Example

- Iteration bound $T_\infty = \max_{l \in L} \left\{ \frac{t_l}{w_l} \right\}$
- Example



$$T_\infty = \max\left(\frac{6}{2}, \frac{11}{1}\right) = 11$$

Algorithms for Computing Iteration Bound

Longest Path Matrix Algorithm

- A series of matrices are constructed $\mathbf{L}^{(m)}$, $m=1,2,..d$, where d is the number of delays in the DFG.
- The value of $\ell_{i,j}^{(m)}$ is the longest computation time of all paths from delay element d_i to delay element d_j that passes through m-1 delay elements, if no such path, then $\ell_{i,j}^{(m)}$ = -1.

Longest Path Matrix Algorithm

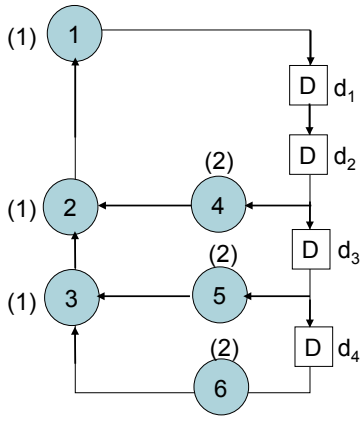
- First determine $\ell_{i,j}^{(1)}$ → $L^{(1)}$
- Then high order matrices are computed by

$$\ell_{i,j}^{(m+1)} = \max_{k \in K}\left(-1, \ell_{i,k}^{(1)} + \ell_{k,j}^{(m)}\right)$$

where *K* is the set of integers *k* in the interval [1,d] such that neither $\ell_{i,k}^{(1)} = -1$ nor $\ell_{k,j}^{(1)} = -1$ holds

$$T_\infty = \max_{i,m \in \{1,2,..d\}} \left\{ \frac{\ell_{i,i}^{(m)}}{m} \right\}$$

Longest Path Matrix Algorithm



$$\ell_{3,1}^{(1)} = 2+1+1+1 = 5$$

$$L^{(1)} = \begin{bmatrix} -1 & 0 & -1 & -1 \\ 4 & -1 & 0 & -1 \\ 5 & -1 & -1 & 0 \\ 5 & -1 & -1 & -1 \end{bmatrix}$$

$$L^{(2)} = \begin{bmatrix} 4 & -1 & 0 & -1 \\ 5 & 4 & -1 & 0 \\ 5 & 5 & -1 & -1 \\ -1 & 5 & -1 & -1 \end{bmatrix}$$

Longest Path Matrix Algorithm

$$\mathbf{L}^{(1)} = \begin{bmatrix} -1 & 0 & -1 & -1 \\ 4 & -1 & 0 & -1 \\ 5 & -1 & -1 & 0 \\ 5 & -1 & -1 & -1 \end{bmatrix} \qquad \mathbf{L}^{(3)} = \begin{bmatrix} 5 & 4 & -1 & 0 \\ 8 & 5 & 4 & -1 \\ 9 & 5 & 5 & -1 \\ 9 & -1 & 5 & -1 \end{bmatrix}$$

$$\mathbf{L}^{(2)} = \begin{bmatrix} 4 & -1 & 0 & -1 \\ 5 & 4 & -1 & 0 \\ 5 & 5 & -1 & -1 \\ -1 & 5 & -1 & -1 \end{bmatrix} \qquad \mathbf{L}^{(4)} = \begin{bmatrix} 8 & 5 & 4 & -1 \\ 9 & 8 & 5 & 4 \\ 10 & 9 & 5 & 5 \\ 10 & 9 & -1 & 5 \end{bmatrix}$$

$$T_\infty = \max\left\{\frac{4}{2}, \frac{4}{2}, \frac{5}{3}, \frac{5}{3}, \frac{5}{3}, \frac{8}{4}, \frac{8}{4}, \frac{5}{4}, \frac{5}{4}, \right\} = 2$$