VLSI Digital Signal Processing Systems

Keshab K. Parhi

VLSI Digital Signal Processing Systems

- Textbook:
  - K.K. Parhi, VLSI Digital Signal Processing Systems: Design and Implementation, John Wiley, 1999

- Buy Textbook:
  - http://www.bn.com
  - http://www.amazon.com
  - http://www.bestbookbuys.com

Chapter 1. Introduction to DSP Systems

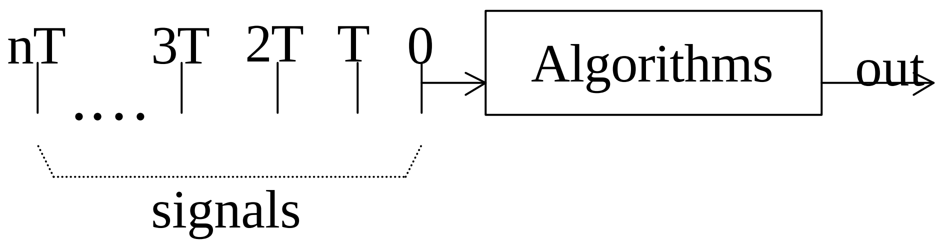
- Introduction (Read Sec. 1.1, 1.3)
- Non-Terminating Programs Require Real-Time Operations
- Applications dictate different speed constraints (e.g., voice, audio, cable modem, settop box, Gigabit ethernet, 3-D Graphics)
- Need to design Families of Architectures for specified algorithm complexity and speed constraints
- Representations of DSP Algorithms (Sec. 1.4)

Typical DSP Programs

- Usually highly real-time, design hardware and/or software to meet the application speed constraint

- Non-terminating
  - Example:

$$
\begin{aligned}
&\textit{for} \quad n = 1 \quad \textit{to} \quad \infty \\
&\qquad y(n) = a \cdot x(n) + b \cdot x(n-1) + c \cdot x(n-2) \\
&\textit{end}
\end{aligned}
$$



Area-Speed-Power Tradeoffs

- 3-Dimensional Optimization (Area, Speed, Power)
- Achieve Required Speed, Area-Power Tradeoffs
- Power Consumption

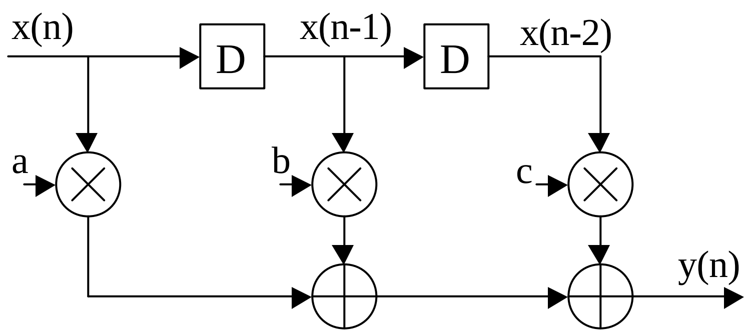
$$P = C \cdot V^2 \cdot f$$

- Latency reduction Techniques => Increase in speed or power reduction through lower supply voltage operation
- Since the capacitance of the multiplier is usually dominant, reduction of the number of multiplications is important (this is possible through strength reduction)

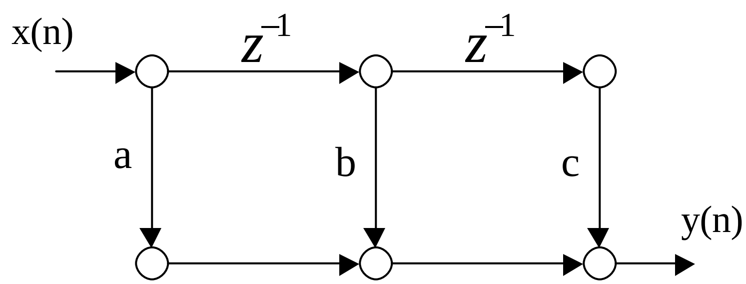
Representation Methods of DSP systems

Example: $y(n)=a*x(n)+b*x(n-1)+c*x(n-2)$

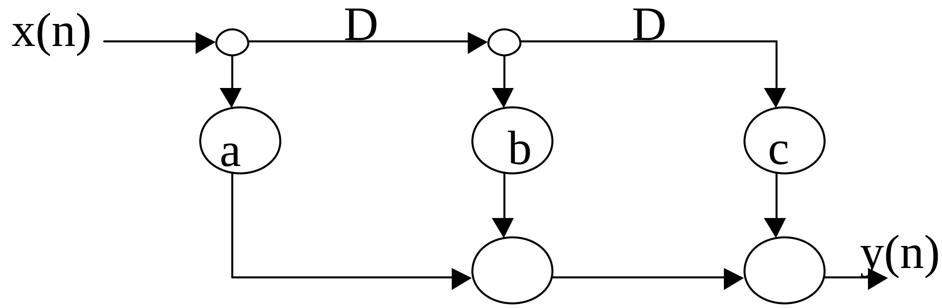
- Graphical Representation Method 1: Block Diagram
  - Consists of functional blocks connected with directed edges, which represent data flow from its input block to its output block



- Graphical Representation Method 2: Signal-Flow Graph
  - 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. For example, ***Flow graph reversal*** or ***transposition*** is one of these transformations (Note: only applicable to single-input-single-output systems)
  - Usually used for linear time-invariant DSP systems representation



- Graphical Representation Method 3: Data-Flow Graph
  - DFG: nodes represent computations (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 perform its computation whenever all its input data are available.
  - Each edge describes a precedence constraint between two nodes in DFG:
    - 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
    - Fine-Grain DFG

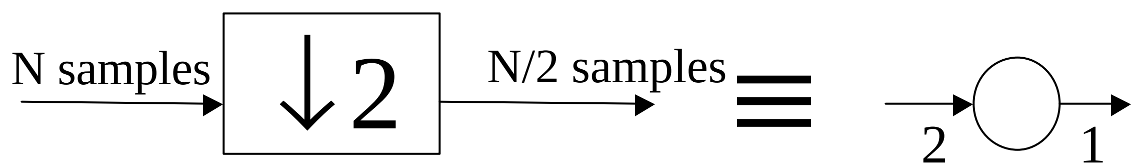


Examples of DFG

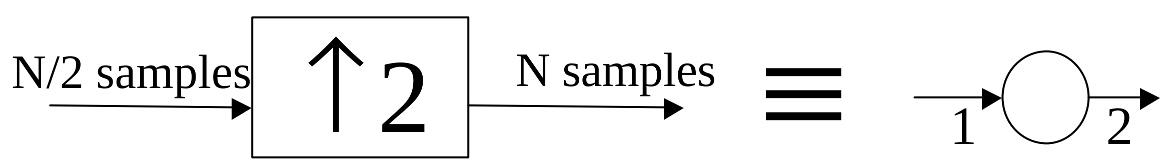
– Nodes are complex blocks (in Coarse-Grain DFGs)

– Nodes can describe expanders/decimators in Multi-Rate DFGs

Decimator



Expander

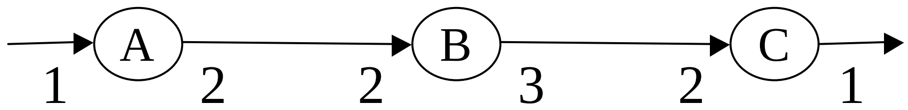


Chapter 2: Iteration Bound

- Introduction
- Loop Bound
  - Important Definitions and Examples
- Iteration Bound
  - Important Definitions and Examples
  - Techniques to Compute Iteration Bound

Introduction

- Iteration: execution of all computations (or functions) in an algorithm once
  - Example 1:



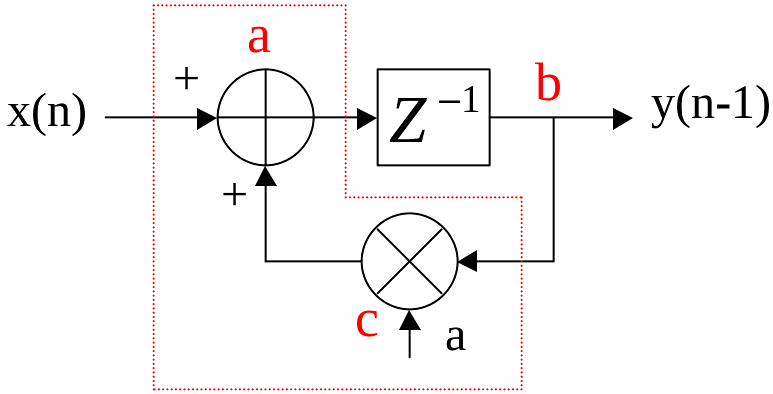
    - For 1 iteration, computations are:

A	B	C
2 times	2 times	3 times

- Iteration period: the time required for execution of one iteration of algorithm (same as sample period)
  - Example:

$$y(n) = a \cdot y(n-1) + x(n)$$

$$i.e. \quad H(z) = \frac{1}{1 - a \cdot z^{-1}}$$



Introduction (cont'd)

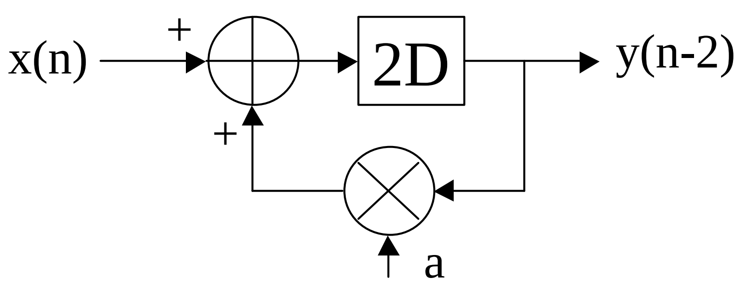
- Assume the execution times of multiplier and adder are T_m & T_a, then the iteration period for this example is $T_m + T_a$ (assume 10ns, see the red-color box). so for the signal, the sample period (T_s) must satisfy:

$$T_s \geq T_m + T_a$$

- Definitions:
  - Iteration rate: the number of iterations executed per second
  - Sample rate: the number of samples processed in the DSP system per second (also called throughput)

Iteration Bound

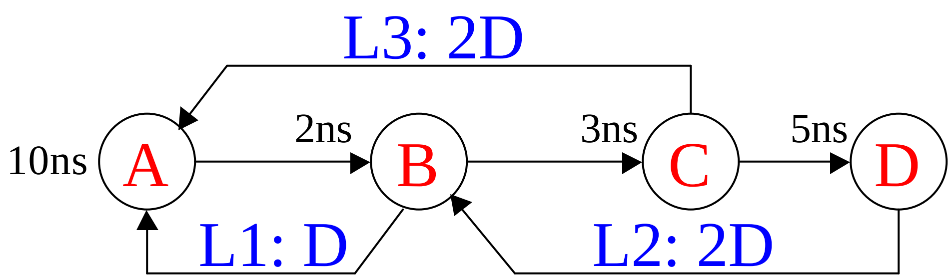
- Definitions:
  - Loop: a directed path that begins and ends at the same node
  - Loop bound of the j-th loop: defined as Tj/Wj, where Tj is the loop computation time & Wj is the number of delays in the loop
  - **Example 1:** a→ b→ c→ a is a loop (see the same example in Note 2, PP2), its loop bound: $T_{loopbound} = T_m + T_a = 10ns$
  - **Example 2:** y(n) = a*y(n-2) + x(n), we have:



$$T_{loopbound} = \frac{T_m + T_a}{2} = 5ns$$

Iteration Bound (cont'd)

- **Example 3:** compute the loop_bounds of the following loops:



$$T_{L1} = (10+2)/1 = 12ns$$
$$T_{L2} = (2+3+5)/2 = 5ns$$
$$T_{L3} = (10+2+3)/2 = 7.5ns$$

- Definitions (Important):
  - Critical Loop: the loop with the maximum loop bound
  - Iteration bound of a DSP program: the loop bound of the critical loop, it is defined as

$$T_{\infty} = \max_{j \in L} \left\{ \frac{T_j}{W_j} \right\}$$

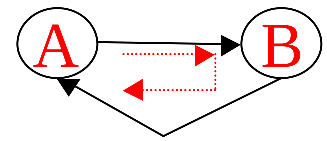
where L is the set of loops in the DSP system,
T_j is the computation time of the loop j and
W_j is the number of delays in the loop j

  - **Example 4:** compute the iteration bound of the example 3:

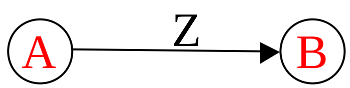
$$T_{\infty} = \max_{l \in L} \{12, \quad 5, \quad 7.5\}$$

Iteration bound (cont'd)

- If no delay element in the loop, then $T_\infty = T_L / 0 = \infty$
  - Delay-free loops are non-computable, see the example:



- Non-causal systems cannot be implemented

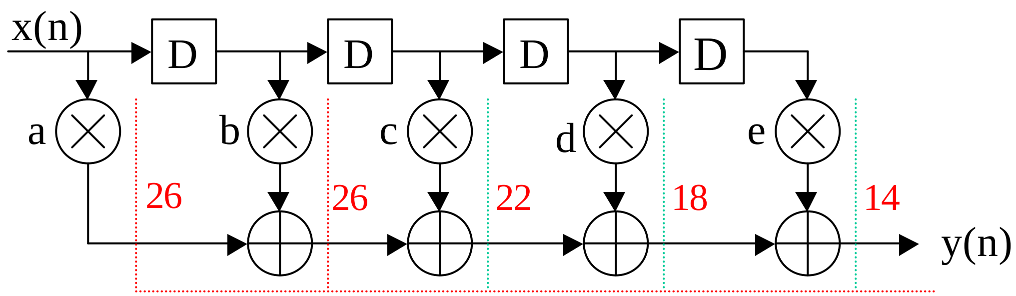


$$\left\{\begin{array}{ll} B = A \cdot Z & non-causal \\ A = B \cdot Z^{-1} & causal \end{array}\right\}$$

- Speed of the DSP system: depends on the "critical path comp. time"
  - Paths: do not contain delay elements (4 possible path locations)
    - (1) input node $\rightarrow$delay element
    - (2) delay element's output $\rightarrow$ output node
    - (3) input node $\rightarrow$ output node
    - (4) delay element $\rightarrow$ delay element
  - Critical path of a DFG: the path with the longest computation time among all paths that contain zero delays
  - Clock period is lower bounded by the critical path computation time

Iteration Bound (cont'd)

– **Example:** Assume Tm = 10ns, Ta = 4ns, then the length of the critical path is 26ns (see the red lines in the following figure)



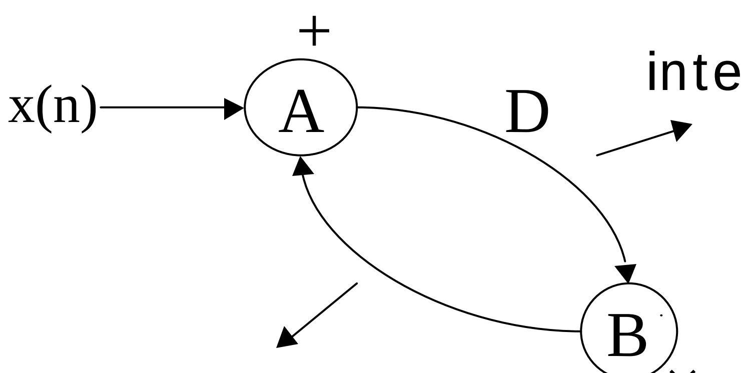
– Critical path: the lower bound on clock period

– To achieve high-speed, the length of the critical path can be reduced by *pipelining and parallel processing (Chapter 3).*

Precedence Constraints

- Each edge of DFG defines a precedence constraint
- Precedence Constraints:
  - Intra-iteration $\Rightarrow$ edges with no delay elements
  - Inter-iteration $\Rightarrow$ edges with non-zero delay elements
- Acyclic Precedence Graph(APG) : Graph obtained by deleting all edges with delay elements.

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



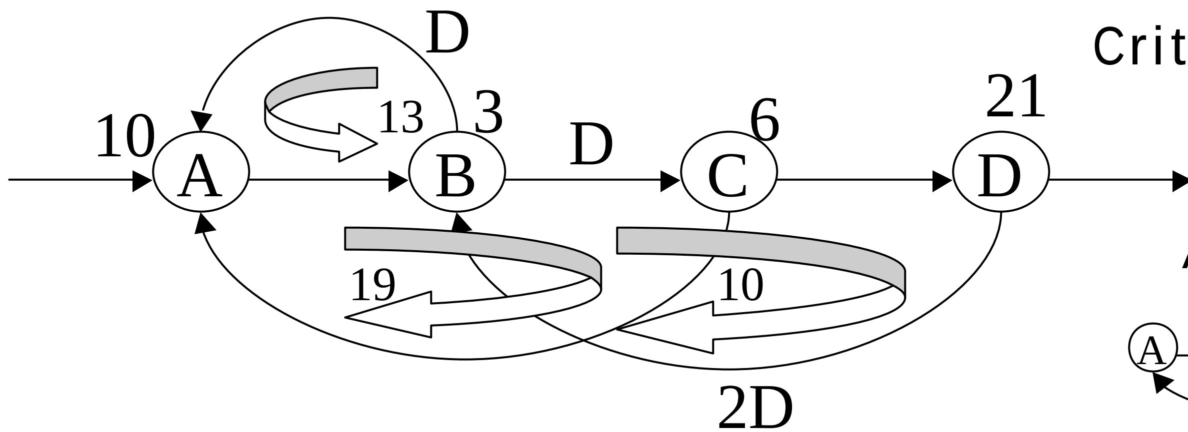
inter-iteration precedence constraint

$A_1 \rightarrow B_2$

$A_2 \rightarrow B_3$

$B_1 \rightarrow A_1 \Rightarrow B_2 \rightarrow A_2 \Rightarrow B_3 \rightarrow A_3 \Rightarrow \ldots$

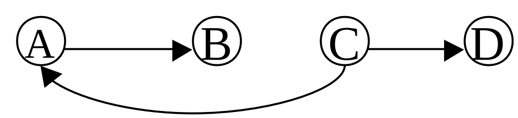
intra-iteration precedence constraint



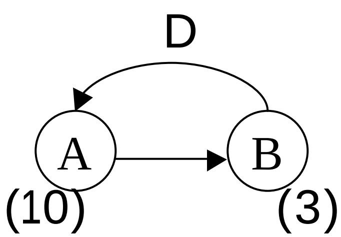
Critical Path = 27ut

T_{clk} >= 27ut

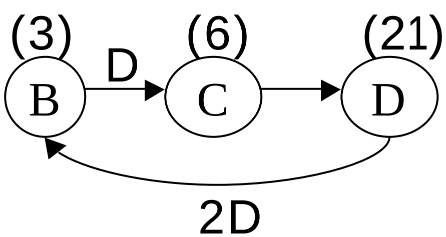
APG of this graph is



- Achieving Loop Bound



T_{loop}= 13ut $\quad A_1 \rightarrow B_1 \Rightarrow A_2 \rightarrow B_2 \Rightarrow A_3 \ldots.$



$B_1 \Rightarrow C_2 \rightarrow D_2 \Rightarrow B_4 \Rightarrow C_5 \rightarrow D_5 \Rightarrow B_7$
$B_2 \Rightarrow C_3 \rightarrow D_3 \Rightarrow B_5 \Rightarrow C_6 \rightarrow D_6 \Rightarrow B_8$
$C_1 \rightarrow D_1 \Rightarrow B_3 \Rightarrow C_4 \rightarrow D_4 \Rightarrow B_6$

Loop contains three delay elements

loop bound = 30 / 3 =10ut = (loop computation time) / (#of delay elements)

- Algorithms to compute iteration bound
  - Longest Path Matrix (LPM)
  - Minimum Cycle Mean (MCM)

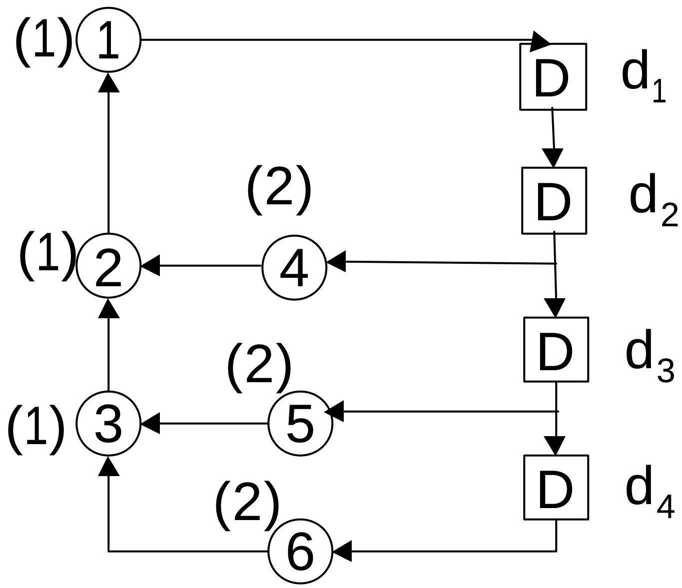
- Longest Path Matrix Algorithm
  - Let 'd' be the number of delays in the DFG.
  - A series of matrices $L^{(m)}$, m = 1, 2, ... , d, are constructed such that $l_{i,j}^{(m)}$ is the longest computation time of all paths from delay element d_i to d_j that passes through exactly (m-1) delays. If such a path does not exist $l_{i,j}^{(m)} = -1$.
  - The longest path between any two nodes can be computed using either Bellman-Ford algorithm or Floyd-Warshall algorithm (Appendix A).
  - Usually, $L^{(1)}$ is computed using the DFG. The higher order matrices are computed recursively as follows :

$$l_{i,j}^{(m+1)} = \max(-1,\ l_{i,k}^{(1)} + l_{k,j}^{(m)}) \quad \text{for } k \in K$$

    where K is the set of integers k in the interval [1,d] such that neither $l_{i,k}^{(1)} = -1$ nor $l_{k,j}^{(m)} = -1$ holds.
  - The iteration bound is given by,

$$T_\infty = \max\{l_{i,i}^{(m)}/m\}, \text{ for } i, m \in \{1, 2, ..., d\}$$

- Example :



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

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

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

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

$$T_\infty = \max\{4/2, 4/2, 5/3, 5/3, 5/3, 8/4, 8/4, 5/4, 5/4\} = 2.$$

- Minimum Cycle Mean :
  - The cycle mean m(c) of a cycle c is the average length of the edges in c, which can be found by simply taking the sum of the edge lengths and dividing by the number of edges in the cycle.
  - Minimum cycle mean is the min{m(c)} for all c.
  - The cycle means of a new graph G_d are used to compute the iteration bound. G_d is obtained from the original DFG for which iteration bound is being computed. This is done as follows:
    - # of nodes in G_d is equal to the # of delay elements in G.
    - The weight w(i,j) of the edge from node i to j in G_d is the longest path among all paths in G from delay d_i to d_j that do not pass through any delay elements.
    - The construction of G_d is thus the construction of matrix $L^{(1)}$ in LPM.
  - The cycle mean of G_d is obtained by the usual definition of cycle mean and this gives the maximum cycle bound of the cycles in G that contain the delays in c.
  - The maximum cycle mean of G_d is the max cycle bound of all cycles in G, which is the iteration bound.

- To compute the maximum cycle mean of G_d the MCM of G_d' is computed and multiplied with –1. G_d' is similar to G_d except that its weights negative of that of G_d.

Algorithm for MCM :

- Construct a series of d+1 vectors, $f^{(m)}$, m=0, 1, … , d, which are each of dimension $d \times 1$.
- An arbitrary reference node s is chosen and $f^{(0)}$is formed by setting $f^{(0)}(s)=0$ and remaining entries of $f^{(0)}$ to ∞.
- The remaining vectors $f^{(m)}$, m = 1, 2, … , d are recursively computed according to

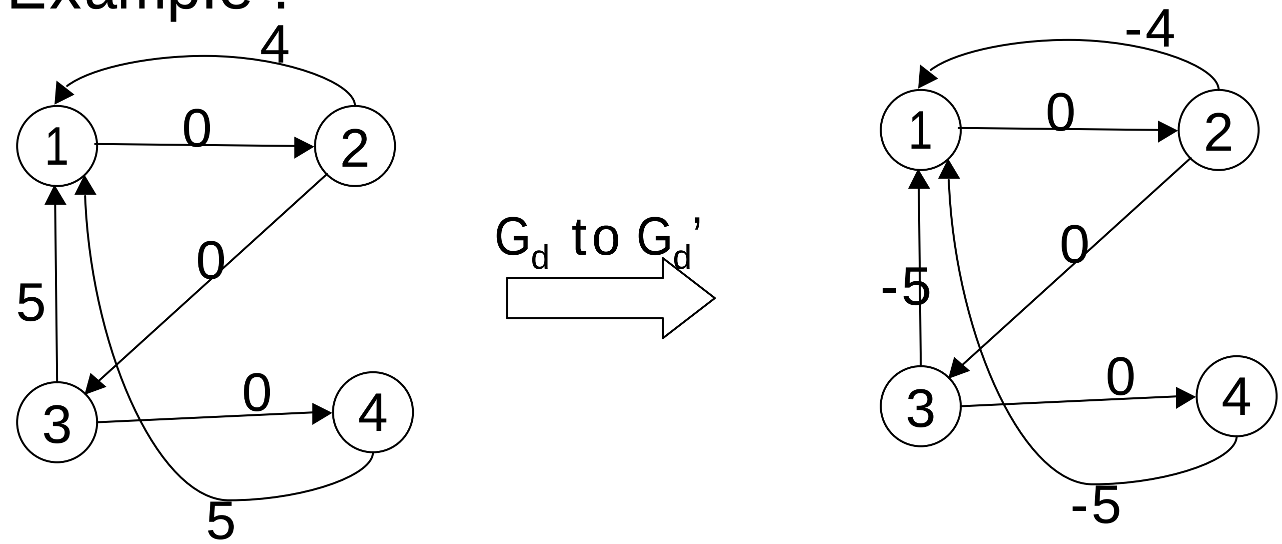
$$f^{(m)}(j) = \min(f^{(m-1)}(i) + w'(i,j)) \quad \text{for } i \in I$$

where, I is the set of nodes in G_d' such that there exists an edge from node i to node j.

- The iteration bound is given by :

$$T_\infty = -\min_{i \in \{1,2,\dots,d\}} (\max_{m \in \{0,1,\dots,d-1\}} ((f^{(d)}(i) - f^{(m)}(i))/(d-m)))$$

- Example :



	m=0	m=1	m=2	m=3	$\max_{m \in \{0,1,\ldots,d-1\}}((f^{(d)}(i) - f^{(m)}(i))/(d-m))$
i=1	-2	$-\infty$	-2	-3	-2
i=2	$-\infty$	-5/3	$-\infty$	-1	-1
i=3	$-\infty$	$-\infty$	-2	$-\infty$	-2
i=4	$\infty-\infty$	$\infty-\infty$	$\infty-\infty$	∞	∞

$$T_\infty = -\min\{-2, -1, -2, \infty\} = 2$$