Chapter 2
Iteration Bound

CSE4210 Winter 2012

Mokhtar Aboelaze

Discrete Real Time Systems

- A discrete real time system usually is a continuously running program that receives some input and produce an output.
- In many designs, data is processed in fixed size chunks.
- The system should be fast enough to complete processing a chunk before it acquires the next one.
- Usually, an analog signal is captured, digitized and then processed by a CPU, DSP of FPGA

Discrete Real Time Systems

- The system could be a single rate or multirate.
- In a single rate system, the number of samples per second at the input and output of the system is the same.
- In a multi rate system, that number is different.
- For example in a digital front end of a receiver, the samples go through multiple stages of decimation decreasing the number of samples per second in every stage. Transmitter if the opposite

Representation of DSP Algorithms

- Many ways to represent DSP algorithms
- Kahn Process Network
- Data flow graph
- Signal flow graph
- Dependence Graph

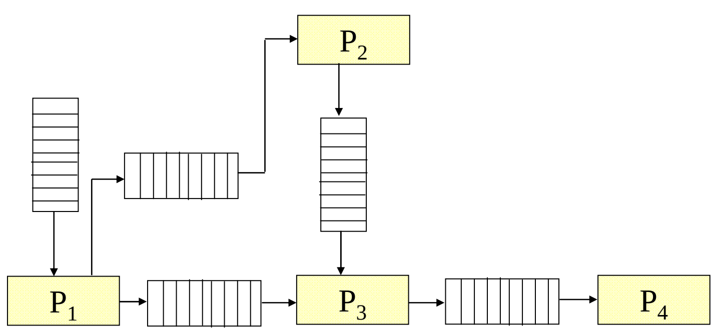
YORK UNIVERSITY CSE4210

Kahn Process Network

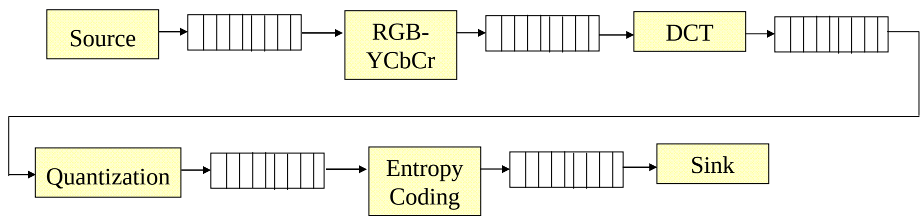
- KPN is a set of concurrently running autonomous processes.
- Processes communicate among themselves in a point-to-point manner over unbounded buffers.
- A process may read from a buffer, process data, and write the result to another buffer.
- Reading is a blocking operation, writes are non-blocking

YORK UNIVERSITY CSE4210

Example of a LPN



JPEG as KPN



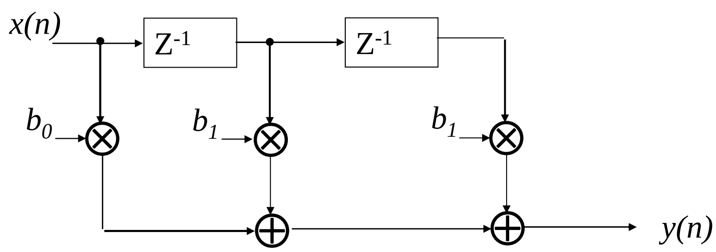
Limitations on KPN

- Reading is done from a FIFO, some DSP algorithms requires non FIFO reading (FFT).
- Once the data is read from the fifo, it is gone, some applications require multiple reading of the same data
- All values written in a FIFO will be read, some algorithms may not read all the values produced by a process.

Representation of DSP Algorithms

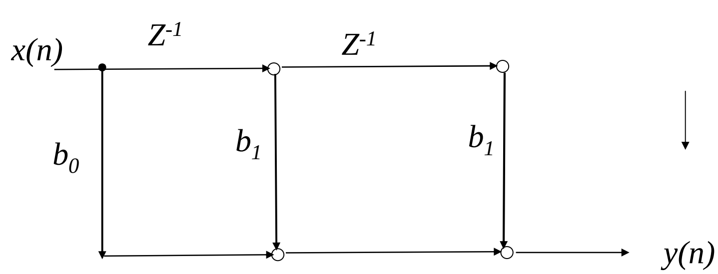
- Block Diagram

$Y(n)=b_0x(n)+b_1x(n-1)+b_2x(n-2)$



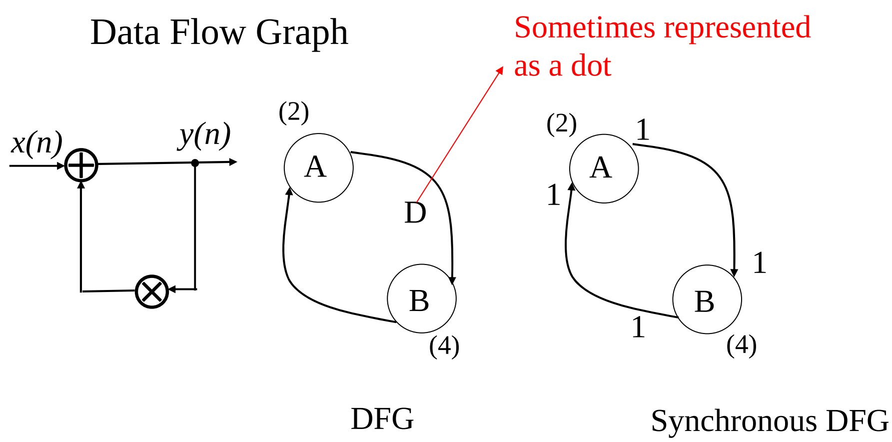
Representation of DSP Algorithms

- Signal Flow Graph



Representation of DSP Algorithms

Data Flow Graph



Representation of DSP Algorithms

- DFG
  - Nodes represents computations (functions) and directed edges represent data paths (communication).
  - Associated with every node its execution time (in parenthesis),
  - Edges have a non-negative delay
  - Nodes can fire (perform the computations) if all input data are available.

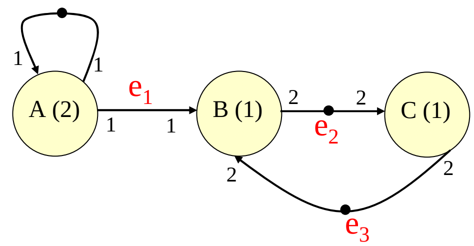
Representation of DSP Algorithms

- Imposes a constraints on the DFG.
- For example, the k^{th} iteration of A must be completed before the $k+1^{st}$ iteration of B *inter-iteration precedence*.
- The k^{th} iteration of B must be completed before the k^{th} iteration of A *intra-iteration precedence*.

Representation of DSP Algorithms

- In synchronous DFG, the number of data samples produced or consumed are specified apriori.
- For example, node B needs 1 data unit to fire and produces one data unit after completeion.
- 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.

Synchronous DFG



$$\begin{array}{c} \quad A \quad\; B \quad\; C \\ \begin{bmatrix} 1 & -1 & 0 \\ 0 & 2 & -2 \\ 0 & -2 & 2 \end{bmatrix} \begin{array}{c} e_1 \\ e_2 \\ e_3 \end{array} \end{array}$$

Topology Matrix: each column represent a node, and each row represent an edge.

The entry is node ***i*** produces (+) a number of tokens in edge ***j*** or consumes (-)

Synchronous DFG

- An SDFG is said to be consistent if the nodes neither starve for data or require an unbounded FIFO's on its edges.
- An inconsistent SDFG may suffer from deadlock (starvation) or requires unbounded FIFO's
- An SDFG is consistent if the rank of its topology graph =n-1, where n =number of nodes.

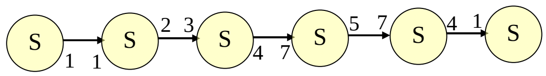
Balanced Firing equation for SDFG

- If nodes S and D are directly connected
- Node S produces PS tokens and Node D produces PD tokens.
- If the firing rate of S and D is fs and fd
- Then $f_S P_S = f_D P_D$ where f_S and f_D are non zero numbers
- Constructing this for every 2 connected nodes, solving for non trivial solution. If exists this is a consistent SDFG

SDFG

- We can use self-timed firing: As a node gets the required number of tokens, it fires.
- If mapped to H/W we can use self-timed execution nodes.
- Also, we can calculate a repetition vector, then we can use this vector to fire the nodes.

Example



Solving for repetition vector gives us

$$\begin{bmatrix} 147 & 147 & 98 & 56 & 40 & 160 \end{bmatrix}$$

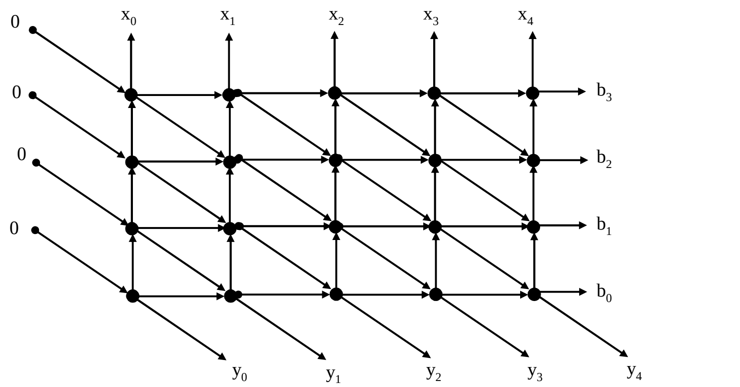
The size of buffer we need?

What if self-timied firing?

Dependence graph

- Dependence Graph is a directed graph that shows the dependence on the computations in an algorithm
- The nodes represent computations and the edges represent precedence constraints.
- The DFG nodes are executed repetitively, while nodes in a dependence graph contains computations for all iterations.

Dependence Graph



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,
- A characteristic of the representation of the algorithm (DFG). Different representations of the same algorithms may lead to different iteration bounds.

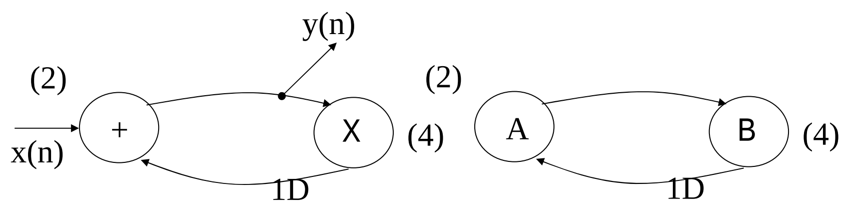
Iteration bound

- 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.

Iteration Bound

- Edges describe a precedence constraints both intra-iteration $\rightarrow$ and inter-iteration $\Rightarrow$
- Critical path is the path with the longest computation time among all paths that contains no delay.
- For recursive (contains loops) DFG, there is a fundamental lower bound "*iteration bound*" T_∞
- Loop bound: t_l/w_l , t_l= loop computation time, w_l is the delay in the loop.
- The critical loop is the loop with the max. loop bound.
- The loop bound of the critical loop is the iteration bound

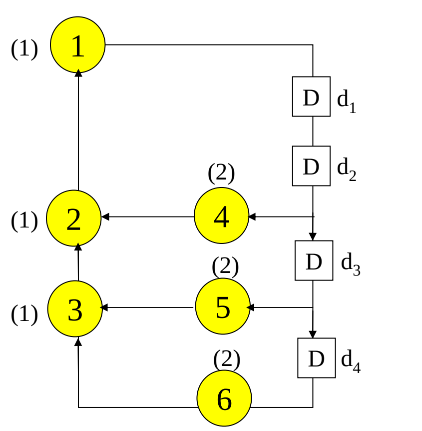
Iteration Bound



- 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. $B_K \Rightarrow A_{K+1}$
- $A_0 \rightarrow B_0 \Rightarrow A_1 \rightarrow B_1 \Rightarrow A_2 \rightarrow B_2$ ….

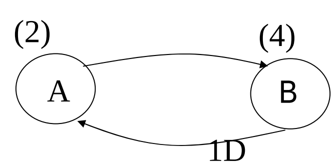
YORK UNIVERSITY CSE4210

Critical Path



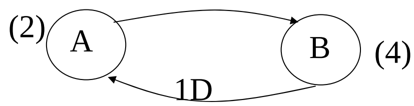
Critical path 6->3->2->1 = 5 tu

5->3->2->1 5 tu's



Critical Path A->B 6 tu's

Iteration bound



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 D; loop bound =6/2=3

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

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

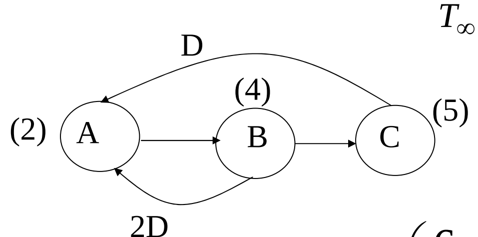
YORK UNIVERSITY CSE4210

Iteration bound

- Iteration bound

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

-



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

Longest path Matrix Algorithm "Iteration bound"

- 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 it is set to -1

Longest path Matrix Algorithm "Iteration bound"

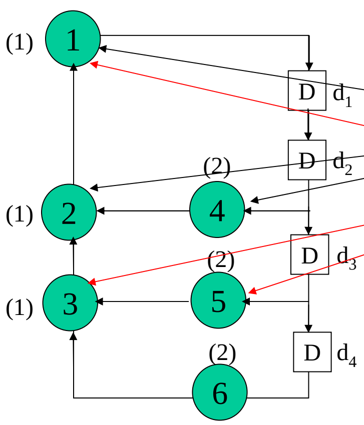
- High order matrices are computed

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

K = [1,d]; $\ell_{i,k}^{(1)}$, $\ell_{k,j}^{(m)}$ ≠-1

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

Longest path Matrix Algorithm “Iteration bound”



$$\ell_{31}(1) = 2+1+1+1 = 5$$

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

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

$$l_{i,j}^2 = \max_{k \in K}\left(-1, l_{i,k}^1 + l_{k,j}^1\right), l_{i,k}^1, l_{k,j}^1 \neq -1$$

$$L^2 = \begin{bmatrix} 4 & -1 & 0 & -1 \\ 5 & 4 & -1 & 0 \\ 5 & 5 & -1 & -1 \\ -1 & 5 & -1 & -1 \end{bmatrix} = \max \ldots\ldots \begin{bmatrix} -1 & 0 & -1 & -1 \\ 4 & -1 & 0 & -1 \\ 5 & -1 & -1 & 0 \\ 5 & -1 & -1 & -1 \end{bmatrix}, \begin{bmatrix} -1 & 0 & -1 & -1 \\ 4 & -1 & 0 & -1 \\ 5 & -1 & -1 & 0 \\ 5 & -1 & -1 & -1 \end{bmatrix}$$

$$L^3 = \begin{bmatrix} 5 & 4 & -1 & 0 \\ 8 & 5 & 4 & -1 \\ 9 & 5 & 5 & -1 \\ 9 & -1 & 5 & -1 \end{bmatrix} = \max\ldots\begin{bmatrix} -1 & 0 & -1 & -1 \\ 4 & -1 & 0 & -1 \\ 5 & -1 & -1 & 0 \\ 5 & -1 & -1 & -1 \end{bmatrix}, \begin{bmatrix} 4 & -1 & 0 & -1 \\ 5 & 4 & -1 & 0 \\ 5 & 5 & -1 & -1 \\ -1 & 5 & -1 & -1 \end{bmatrix}$$

Longest path Matrix Algorithm "Iteration bound"

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

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

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

The min. Cycle Mean Algorithm

- The cycle mean M(c), of a cycle c, is the average length of the edges in c. Calculated as the sum of weights of all edges divided by the number of edges in the cycle.
- The minimum cycle mean is the min of all c in the graph.
- The maximum cycle mean is the max of all c
- The cycle means of a new graph G_d is used to calculate the iteration bound.

The min. Cycle Mean Algorithm

- Construct a new graph G_d from G (SFG).
- A node in G_d for each delay element in G
- w(i,j) in G_d is the longest path in G between delay d_i to d_j that dos not pass through any delay elements (zero-delay)
- If no such pass exist, the edge does not exist in G_d ($L^{(1)}$ in LPM).
- The maximum cycle mean in G_d is the iteration bound.

- Construct the graph $\overline{G_d}$ from G_d by negating the values of the weights
- The maximum cycle mean of G_d is simply the minimum cycle mean of $\overline{G_d}$ multiplied by -1
- Find the minimum cycle mean of $\overline{G_d}$, multiply it by -1

The min. Cycle Mean Algorithm

- Choose any node arbitrarily and set

$$f^{(m)}(j) = \min_{i \in I}\left(f^{(m-1)}(i) + \overline{w}(i,j)\right)$$

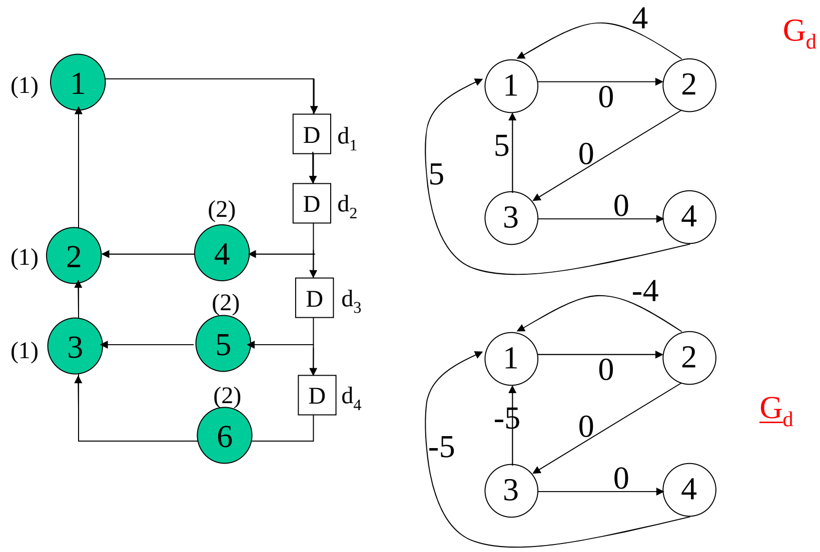
$\overline{w}(i,j)$ is the weight of the edge $i \to j$ in $\overline{G}_d$, I is th set of nodes in $\overline{G}_d$ such that there exist an edge from $i \to j$

$$f^{(0)} = \begin{bmatrix} 0 \\ \infty \\ \infty \\ \infty \end{bmatrix}$$

$$T_\infty = -\min_{i \in \{1,2,\ldots,d\}}\left(\max_{m \in \{0,1,\ldots,d-1\}}\left(\frac{f^{(d)}(i) - f^{(m)}(i)}{d-m}\right)\right)$$

d is the number of nodes in $\overline{G}_d$

Example Fig 2.2

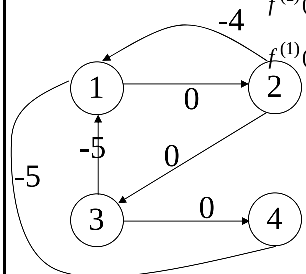


$$f^{(1)}(1) = \min_{3,4}\left\{f^{(0)}(2) + \bar{w}(2,1),\ f^{(0)}(3) + \bar{w}(3,1),\ f^{(0)}(4) + \bar{w}(4,1)\right\} = \min\{\infty, \infty\} =$$

$$f^{(1)}(2) = \min_{1}\left\{f^{(0)}(1) + \bar{w}(1,2)\right\} = 0 - 0 = 0$$

$$f^{(1)}(3) = \min_{2}\left\{f^{(0)}(2) + \bar{w}(2,3)\right\} = \infty - 0 = \infty$$

$$f^{(1)}(4) = \min_{1}\left\{f^{(0)}(3) + \bar{w}(3,4)\right\} = \infty - 0 = \infty$$



$$f^{(0)} = \begin{bmatrix} 0 \\ \infty \\ \infty \\ \infty \end{bmatrix} \quad f^{(1)} = \begin{bmatrix} \infty \\ 0 \\ \infty \\ \infty \end{bmatrix} \quad f^{(2)} = \begin{bmatrix} -4 \\ \infty \\ 0 \\ \infty \end{bmatrix} \quad f^{(3)} = \begin{bmatrix} -5 \\ -4 \\ \infty \\ 0 \end{bmatrix} \quad f^{(4)} = \begin{bmatrix} -8 \\ -5 \\ -4 \\ \infty \end{bmatrix}$$

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

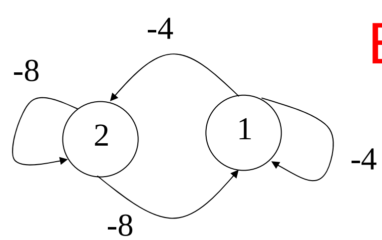
$$f^{(0)}=\begin{bmatrix}0\\ \infty\\ \infty\\ \infty\end{bmatrix}\quad f^{(1)}=\begin{bmatrix}\infty\\ 0\\ \infty\\ \infty\end{bmatrix}\quad f^{(2)}=\begin{bmatrix}-4\\ \infty\\ 0\\ \infty\end{bmatrix}\quad f^{(3)}=\begin{bmatrix}-5\\ -4\\ \infty\\ 0\end{bmatrix}\quad f^{(4)}=\begin{bmatrix}-8\\ -5\\ -4\\ \infty\end{bmatrix}$$

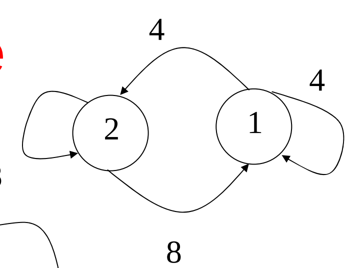
$$\begin{bmatrix}(-8-0)/4 & (-8-\infty)/3 & (-8-4)/2 & -8+5\\ (-5-\infty)/4 & (-5-0)/3 & (-5-\infty)/2 & -5+4\\ (-4-\infty)/4 & (-4-\infty)/3 & (-4-0)/2 & -4-\infty\\ (\infty-\infty)/4 & (\infty-\infty)/3 & (\infty-\infty)/2 & \infty-0\end{bmatrix}=\begin{bmatrix}-2\\ -1\\ -2\\ \infty\end{bmatrix}$$

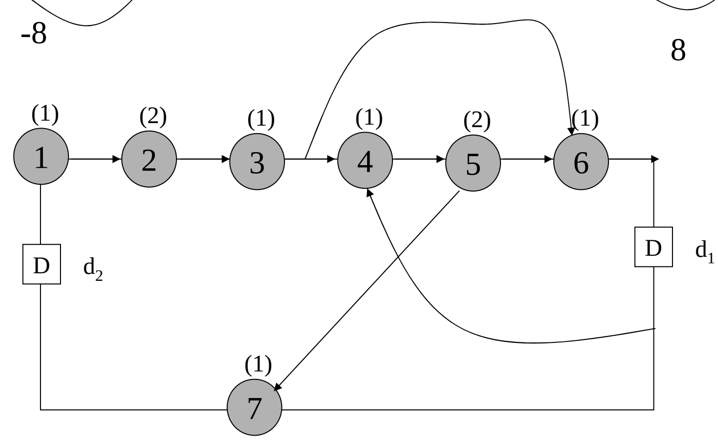
T_∞=-min(-2,-1,-1, ∞)=-(-2)=2

YORK UNIVERSITY CSE4210

Example







$$f^{(m)}(j) = \min_{i \in I}\left(f^{(m-1)}(i) + \overline{w}(i,j)\right), \; f^{(0)} = \begin{bmatrix} 0 \\ \infty \end{bmatrix}$$

$$f^{(1)}(1) = \min\left(f^{(0)}(1) + \overline{w}(1,1),\; f^{(0)}(2) + \overline{w}(2,1)\right) = \min(0-4, \infty - 8) = -4$$

$$f^{(1)}(2) = \min\left(f^{(0)}(1) + \overline{w}(1,2),\; f^{(0)}(2) + \overline{w}(2,2)\right) = \min(0-4, \infty - 8) = -4$$

$$f^{(2)}(1) = \min\left(f^{(1)}(1) + \overline{w}(1,1),\; f^{(1)}(2) + \overline{w}(2,1)\right) = \min(-4-4, -4-8) = -12$$

$$f^{(2)}(1) = \min\left(f^{(1)}(1) + \overline{w}(1,1),\; f^{(1)}(2) + \overline{w}(2,1)\right) = \min(-4-4, -4-8) = -12$$

$$f^{(0)}\begin{bmatrix} 0 \\ \infty \end{bmatrix}, \; f^{(1)} = \begin{bmatrix} -4 \\ -4 \end{bmatrix}, \; f^{(0)} = \begin{bmatrix} -12 \\ -12 \end{bmatrix}$$

$$T_{\infty} = -\min_{i \in \{1,2,\ldots,d\}}\left(\max_{m \in \{0,1,\ldots,d-1\}}\left(\frac{f^{(d)}(i) - f^{(m)}(i)}{d-m}\right)\right)$$

$$f^{(0)}\begin{bmatrix} 0 \\ \infty \end{bmatrix}, \; f^{(1)} = \begin{bmatrix} -4 \\ -4 \end{bmatrix}, \; f^{(0)} = \begin{bmatrix} -12 \\ -12 \end{bmatrix}$$

$$\max\left(\begin{bmatrix} (-12-0)/2 \\ (-12-\infty)/2 \end{bmatrix}, \begin{bmatrix} -12+4 \\ -12+4 \end{bmatrix}\right) = \begin{bmatrix} -6 \\ -8 \end{bmatrix}$$

$$-\min(-6,-8) = 8$$

Multirate DFG

- Change the MRDFG into SRDFG
- Calculate the iteration bound of the SRDFG, which is the same as the iteration bound of the MRDFG