DSP Systems

can be realized using

- Programmable processors
- Custom designed hardware

Maximize the performance

while keeping the cost down

Performance is measured in terms of

- amount of hardware circuitry and resources required
- speed of execution
- amount of power dissipation

DSP Algorithms and Applications

- End Products
  - Communication Systems
  - Control Systems
  - Electronic Appliances
- Higher-Level Algorithms
  - Modems
  - Codecs
  - Encryption/decryption
  - Image/Speech Compression
  - Image/Speech Synthesis and Recognition
  - Noise cancellation
- Elemental Algorithms
  - Digital Filters
  - Linear Transforms
  - Adaptive Filters

DSP Algorithms

- Scalar Type:
  - intensity mapping, etc.
- Vector Algebra Type:
  - filters, windowing, etc.
- Matrix Algebra Type:
  - singular value decomposition, geometric rotation, maximum likelihood estimation
- Transform Type:
  - Fourier transform, discrete cosine transform, etc.
- Sorter Type:
  - merge sort, etc.

Those are often combined

Example of Computational Requirement

- HDTV Image
- 1920x1080 pixels 30 frames/sec (1080i)

  6.2×10^7 samples/sec, 8×3 bit/sample
- Total 1.5 Gbps (cf. BS:24Mbps, TD: 17Mbps)
- Feature extraction with 3x3 window
- 5.6×10^8 computations and memory accesses per second, even for linear operations like spatial filtering, convolution, etc.
- TOPs will be necessary for recent applications

Features of DSP

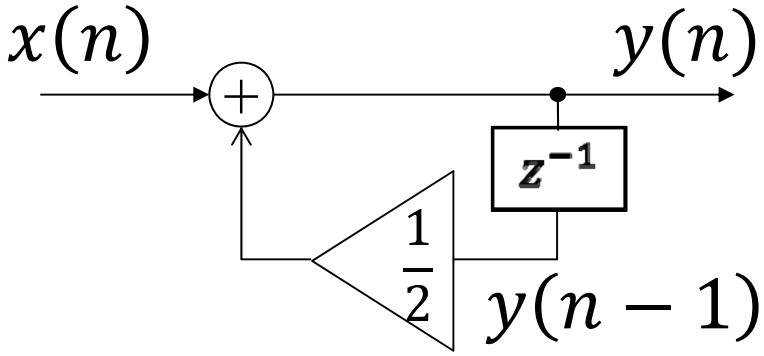
compared to other general purpose computations

- Real-time throughput requirement
  - hardware/software should be designed to meet the tight constraint of the real-time processing
- Data driven property
  - Any tasks or computations can be performed once all the input data are available

Graphical Representation of DSP Algorithms

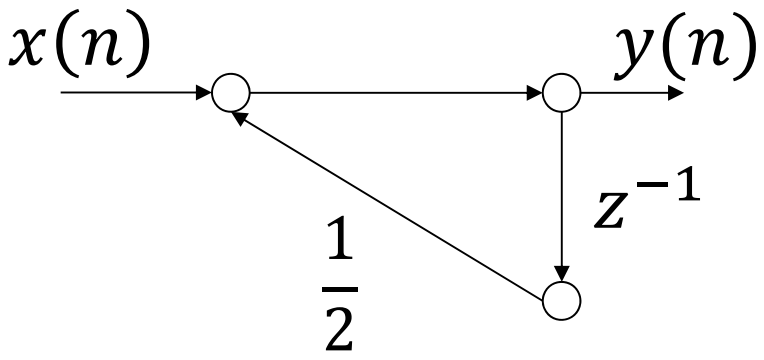
- Analysis of date flow properties
- Exploit the inherent parallelism
- Map the algorithm to hardware

- Block Diagram
  - using multipliers, adders, and delays



Signal Flow Graph

- Collection of Nodes and Directed Edges
  - Node: Computations or Tasks
  - Directed Edge: Connection of Tasks



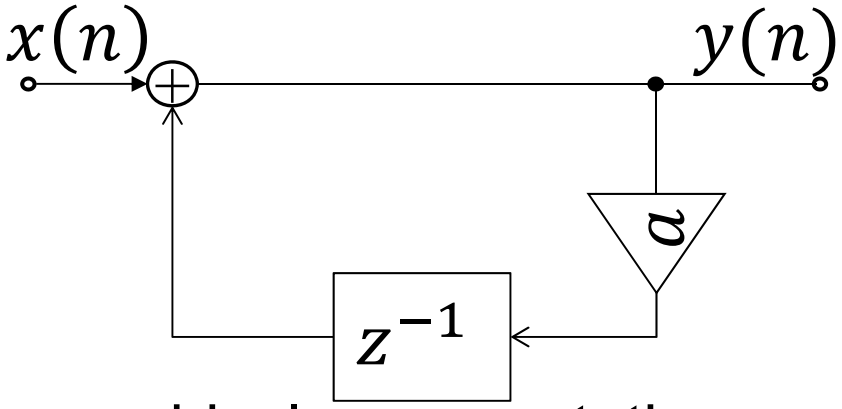
Signal Flow Graph

- Used for analysis, representation, design, and evaluation of linear systems
- Linear SFGs can be transformed into different forms without changing the system functions
  - Flow Reversal or Transposition
- Mason's rule can be used to calculate the system function

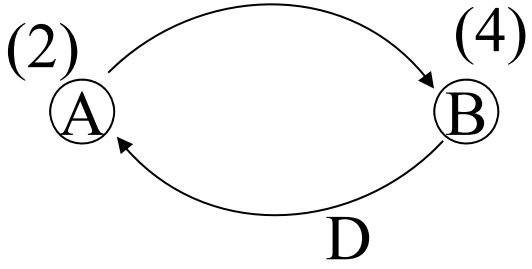
Data Flow Graph

- DSP:non terminating program
- DFG:directed graph that describes the program

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



graphical representation



DFG

Data Flow Graph

- node:task or computation
  - each node has an execution time
    - A:addition, B:multiplication
- edge:communication between the nodes
  - each edge has a nonnegative delays
    - A→B:zero delays, B → A:one delay
- Iteration of a node: execution of the node exactly once
- Iteration of the DFG: execution of each node in the DFG exactly once

Precedence

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

Together, the order in which the nodes in the DFG can be executed is specified.

Precedence

example

- The k-th iteration of A must be executed before the k-th iteration of B

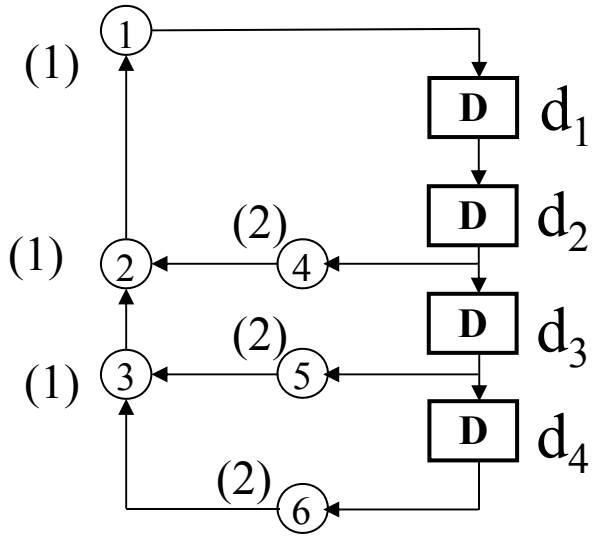
$$A_k \rightarrow B_k$$

- The k-th iteration of B must be executed before the $(k+1)$ -th iteration of A

$$B_k \Longrightarrow A_{k+1}$$

Critical Path

- The path with the longest computation time among all paths that contain zero delays.
- Computation time of the critical path is the minimum computation time for one iteration of the DFG.



The computation time along the critical path 5 u.t.

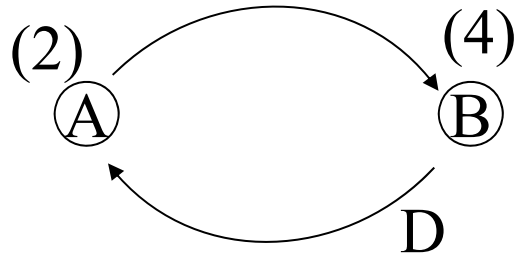
Loop

a directed path that begins and ends at the same node

- recursive DFGs have loops (IIR filters)
- non recursive DFGs have no loops (FIR filters)

The amount of time required to execute a loop

$A_0 \rightarrow B_0 \Rightarrow A_1 \rightarrow B_1 \Rightarrow A_2 \rightarrow B_2 \cdots$

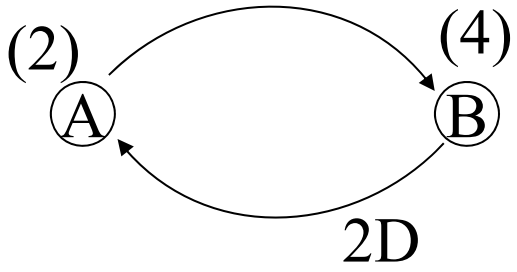


one iteration of the loop requires 6.u.t

Loop Bound

The loop bound for the ℓ-th loop

$$\frac{t_\ell}{w_\ell} \quad \begin{cases} t_\ell\text{: loop computational time} \\ w_\ell\text{: the number of delays in the loop} \end{cases}$$



the loop bound is $\frac{6}{2} = 3$ u.t.

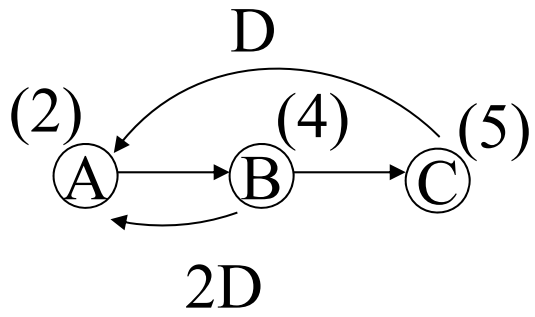
$$A_0 \to B_0 \Rightarrow A_2 \to B_2 \Rightarrow A_4 \to B_4 \Rightarrow \cdots$$
$$A_1 \to B_1 \Rightarrow A_3 \to B_3 \Rightarrow A_5 \to B_5 \Rightarrow \cdots$$

Iteration Bound

$$T_\infty = \max_\ell \frac{t_\ell}{w_\ell}$$

the loop bound of the critical loop, which is the loop with the maximum loop bound

Lower bound on iteration or sample period of the DSP program regardless of the amount of computing resources available

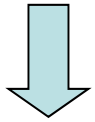


$$T_\infty = \max_\ell \left\{\frac{6}{2}, \frac{11}{1}\right\} = 11 \text{ u.t.}$$

Different representation of the same algorithm may lead to different iteration bound

Computation of Iteration Bound

- Straightforward technique to locate all the loops and directly compute T_∞
- The number of loops may grow exponentially with respect to the number of nodes



- Systematic algorithm is necessary

Longest Path Matrix Algorithm

- Paths from one delay to another
- d: # of delays
- $d \times d$ matrix $L(m), m = 1, \cdots, d$
- $l_{ij}(m)$: longest computing time from output of d_i to input of d_j with exactly $m - 1$ delays in between
- If no such a path exists, $l_{ij}(m) = -1$
- $L(1)$ is computed from DFG
- $L(m+1)$ is computed recursively by $l_{ij}(m+1) = \max\left(-1, l_{ik}(m) + l_{kj}(m)\right), k = 1, \cdots, d$
- $T_\infty = \max\limits_{0 \le i,m \le d} \frac{l_{ii}(m)}{m}$

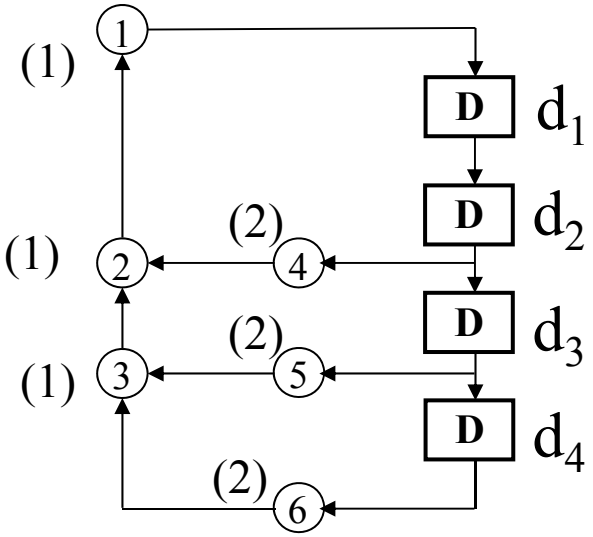
Longest Path Matrix Algorithm

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

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

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

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



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

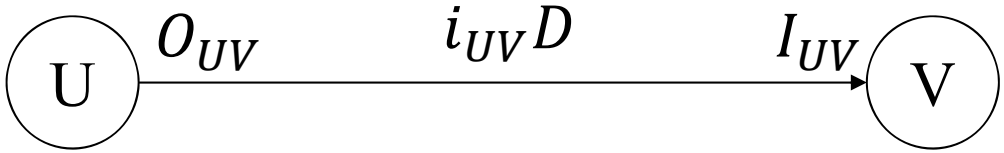
Multirate DFG

- Node can be executed more than once per iteration
- Two nodes are not required to execute the same number of times in an iteration

- Two step approach

1. Construct an equivalent single rate DFG
2. Compute the iteration bound for that equivalent DFG

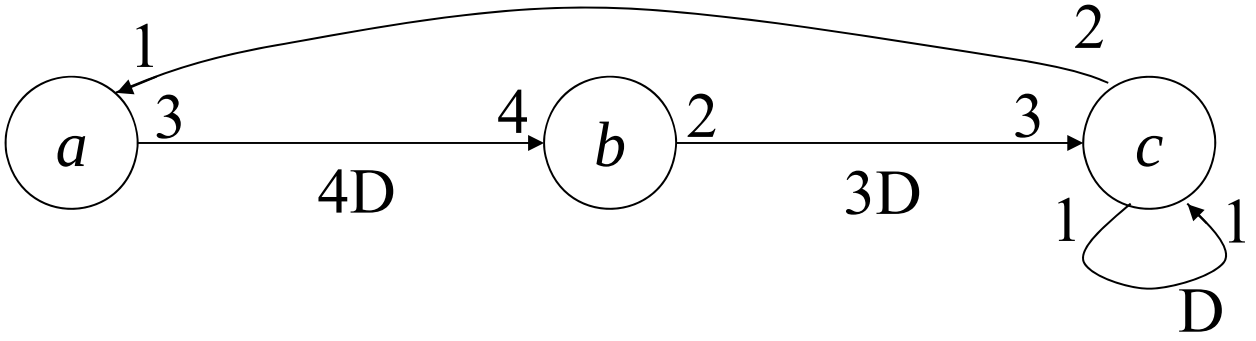
Edge in Multirate DFG



- O_{UV} is the number of samples produced on the edge by an invocation node U
- I_{UV} is the number of samples consumed from the edge by an invocation node V
- i_{UV} is the number of delays on the edge
- U is invoked k_U times in an iteration
- V is invoked k_V times in an iteration

$$O_{UV}k_U = I_{UV}k_V$$

How many times invoked?



$$3k_a = 4k_b$$
$$2k_b = 3k_c$$
$$k_c = k_c$$
$$2k_c = k_a$$

$$k_a = 4$$
$$k_b = 3$$
$$k_c = 2$$

Algorithm for conversion

For each node U in MRDFG

for $k = 0$ to $k_U - 1$

Draw a node U_k in the SRDFG

For each edge in MRDFG

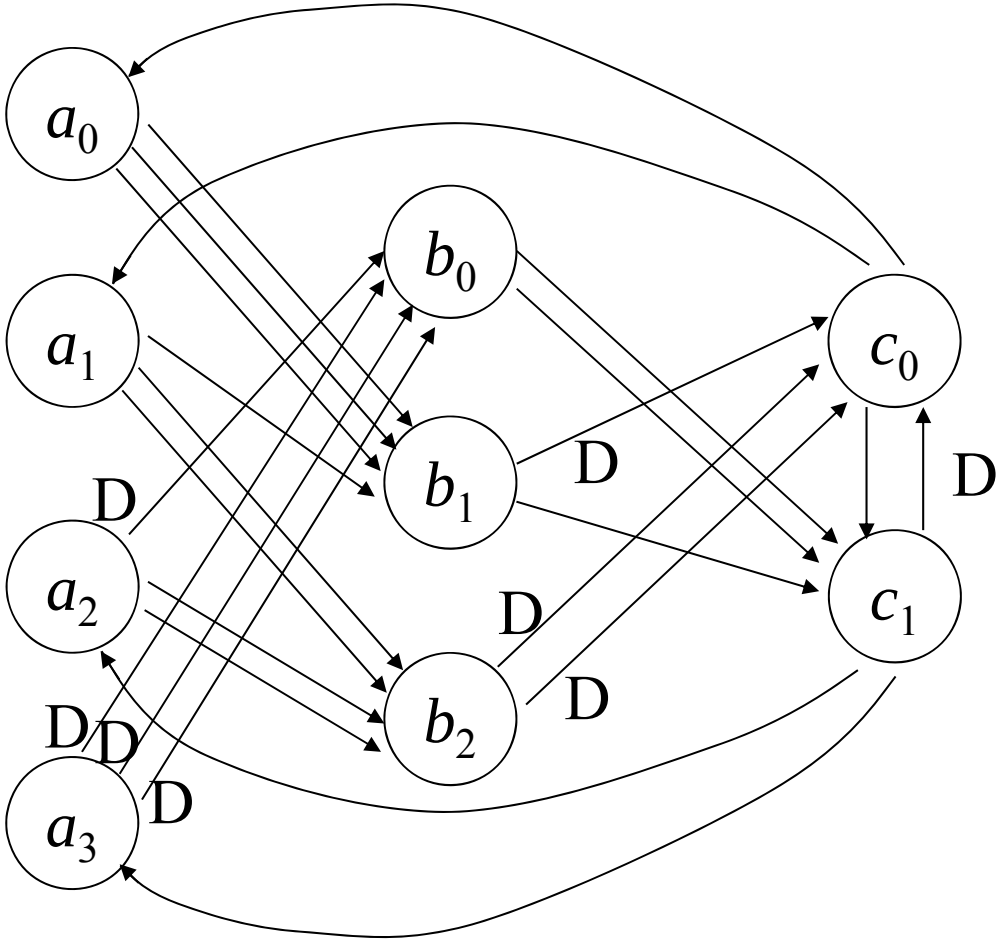
for $j = 0$ to $O_{UV} k_U - 1$

Draw an edge from $U_{j/O_{UV}}$ to $V_{((j+i_{UV})I_{UV})\%k_V}$ with $(j + i_{UV})/(I_{UV} k_V)$ delays

a/b denotes integer part of division

$a\%b$ denotes the remainder

Equivalent Single Rate DFG



Iteration Bound can now be calculated.

Exercise 11

1. Draw a DFG for a 4th-order IIR digital filter implemented as cascade of two 2nd-order sections. Assume each multiplication requires T_M and addition requires T_A. What is the critical path of this DFG? What is the iteration bound of this DFG?
2. Repeat the previous question for a direct form 4th-order IIR filter.
3. Consider an FIR filter

   $y(n) = a_0 x(n) + a_2 x(n-2) + a_3 x(n-3)$.

   Assume that the time required for 1 multipliy operation is T_M and the time required for 1 add operation is T_A. Draw two types of direct-form realization structures of the above filter. What is the critical path length? What is the iteration bound?