DSP Design

Graph representations and Iteration Bound

Steffen Malkowsky

DSP representations

Chapter 1.4

Definitions

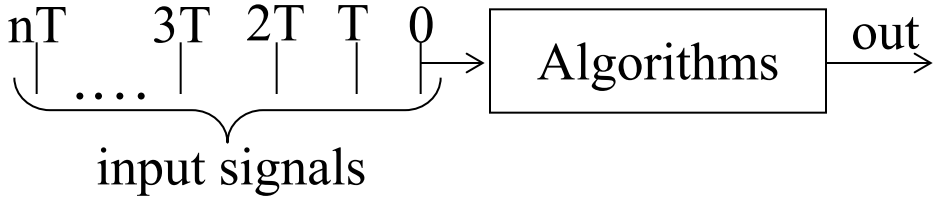
Some concepts that you must know!

- **Non-terminating**
- **Iteration (iteration period, iteration rate,…)**
- **Sampling rate**
- **Throughput**
- **Latency**
- **Critical path**

Non-terminating DSP algorithms

$$y(n) = \sum_{k=0}^{N-1} h(k)x(n-k)$$ **FIR-filter**

DSP algorithms are non-terminating



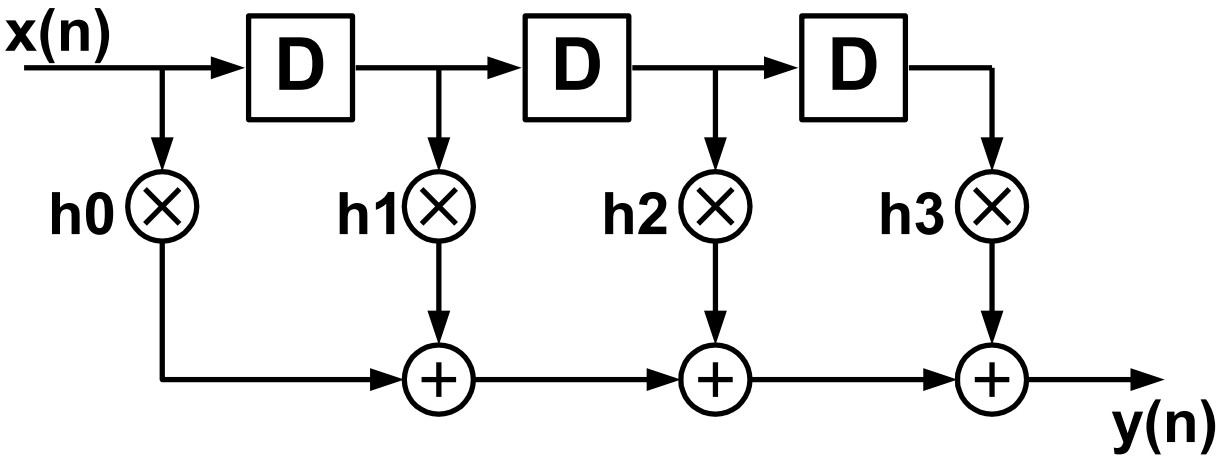
$$for\ n = 0\ to\ \infty$$

$$y(n) = h_0 x(n) + h_1 x(n-1) + h_2 x(n-2) + h_3 x(n-3)$$

Iteration

One execution of all computations in the algorithm once is equal to one *iteration*

Iteration period - the time required for the execution of one iteration of the algorithm.



One iteration

- 1 input
- 4 mult
- 3 adds
- 1 output

Sampling rate, Throughtput and Latency

Sampling rate **= nr. of samples processed/second (a.k.a. *throughput*)**

Latency **= time difference between output and corresponding input (eg. how long it takes to travel through the system), propagation delay**

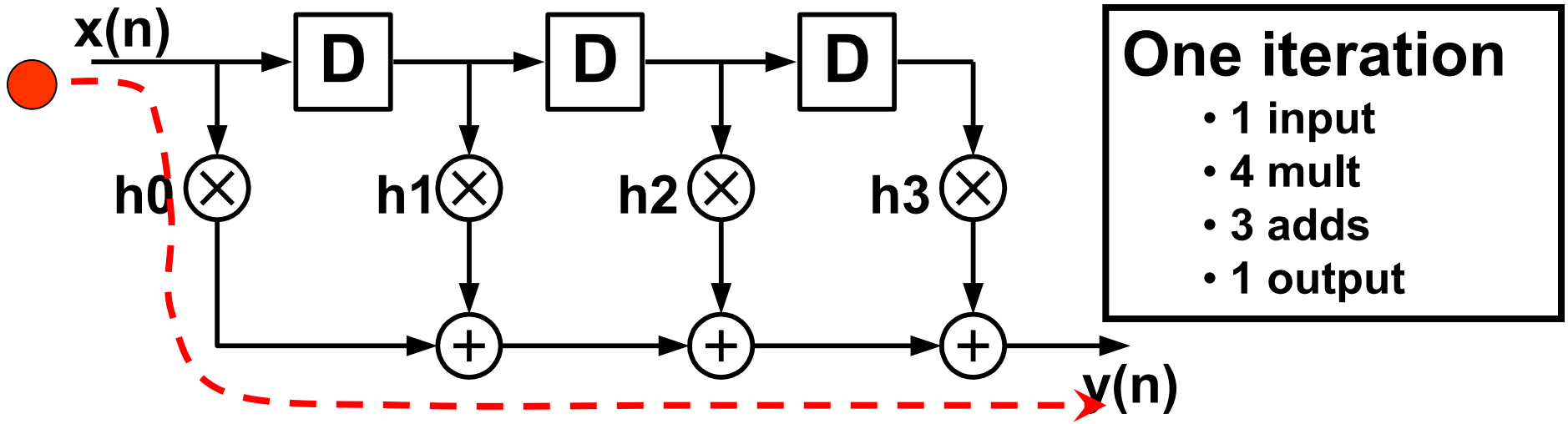
- **In combinatorial logic = gate delays**
- **In sequential logic = nr. of clock cycles**

Direct Form 4-tap FIR

Critical path = longest path without delay element

Critical path sets bound on clock frequency

$$T_{clk} \geq T_{critical}$$



Critical Path

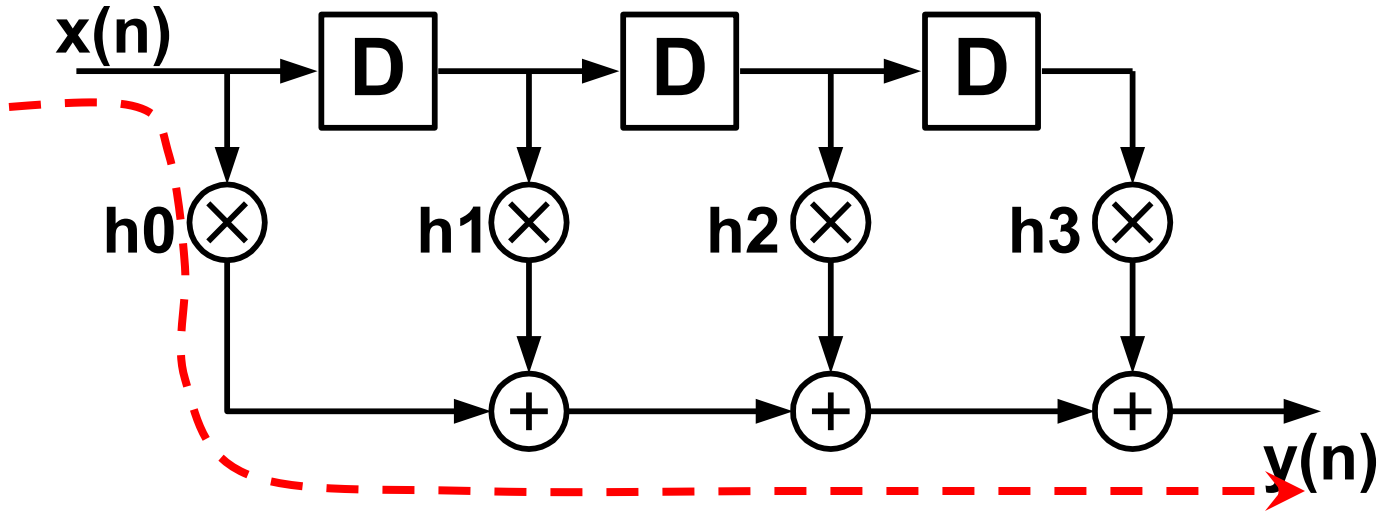
Traditionally the critical path was considered;

- **between delays**
- **from input to delay**
- **from delay to output**
- **from input to output**

However, with technology scaling we also need to consider the registers:

- **setup time (min time the input should be stable before the clock)**
- **hold time (min time the input should be held after the clock to reliably sample data)**
- **output delay (time to change output after clock edge)**

Direct Form 4-tap FIR

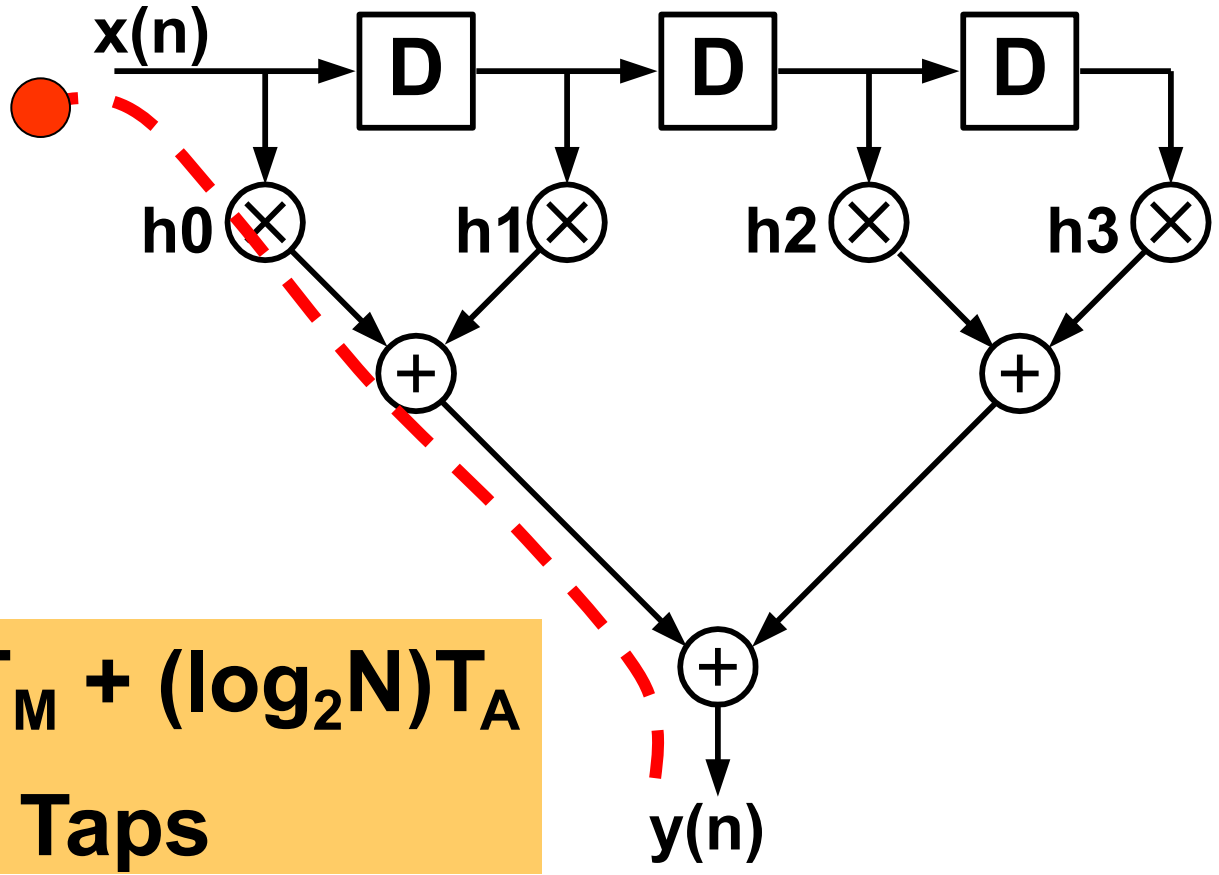


Clock speed limited by Critical Path!

$T_{Critical} = T_M + (N-1)T_A$

N = Nr. of Taps

Direct Form 4-tap FIR with Adder Tree



$T_{Critical} = T_M + (\log_2 N) T_A$

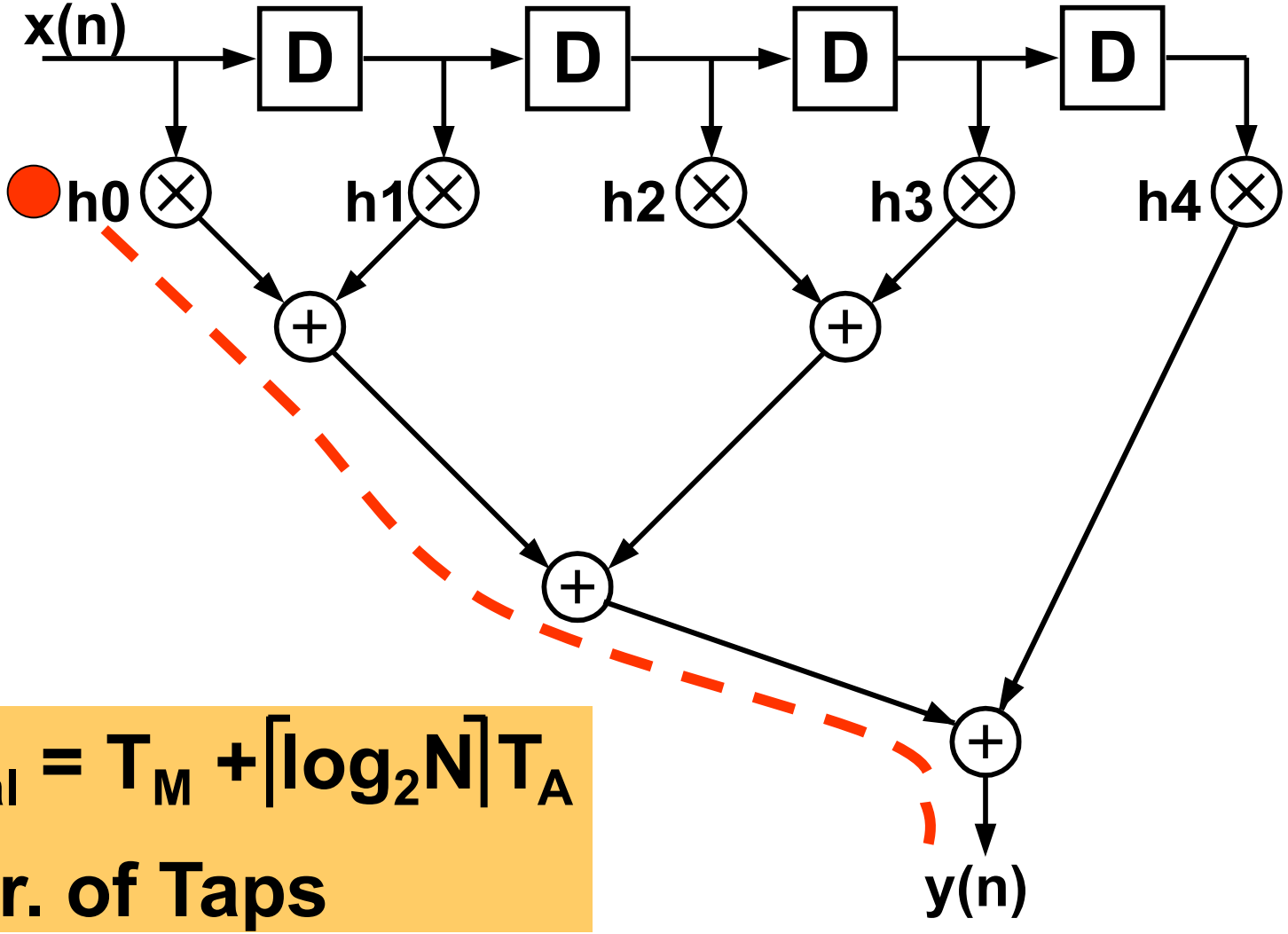
N = Nr. of Taps

Balanced Path

Always try to have a balanced path:

- reduced critical path
- less jitter, i.e. not wanted transitions

Direct Form 5-tap FIR with Adder Tree



$$T_{Critical} = T_M + \lceil \log_2 N \rceil T_A$$

N = Nr. of Taps

Graphical Representations

There are 3 main methods for representing a DSP algorithm in graphical form.

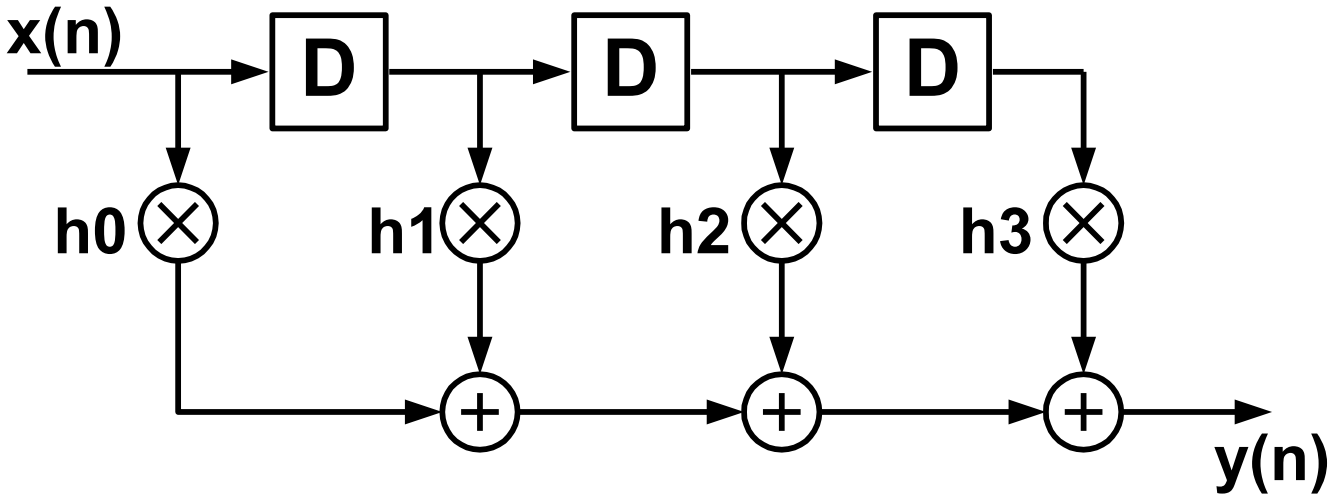
Method 1: Block diagram

Method 2: Signal-Flow Graph (SFG)

Method 3: Data-Flow Graph (DFG)

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

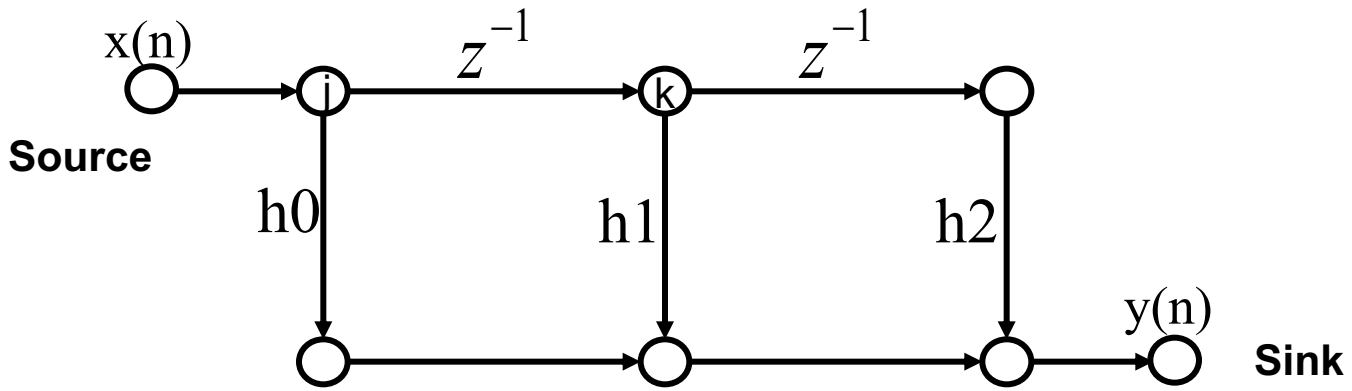


Block diagrams

- Easy to understand.
- Most frequently used to represent DSP systems.
- Captures the exact functionality of the system.
- Various block diagrams can be derived for the same system.

Graphical Representation
Method 2: Signal-Flow Graph (SFG)

- Nodes: represent computations and/or task
- Directed edge *(j, k)* denotes a linear transformation from the input signal at node *j* to the output signal at node *k*
  - in digital usually limited to delays and constant mults
- Source = no entering edge
- Sink = only entering edges



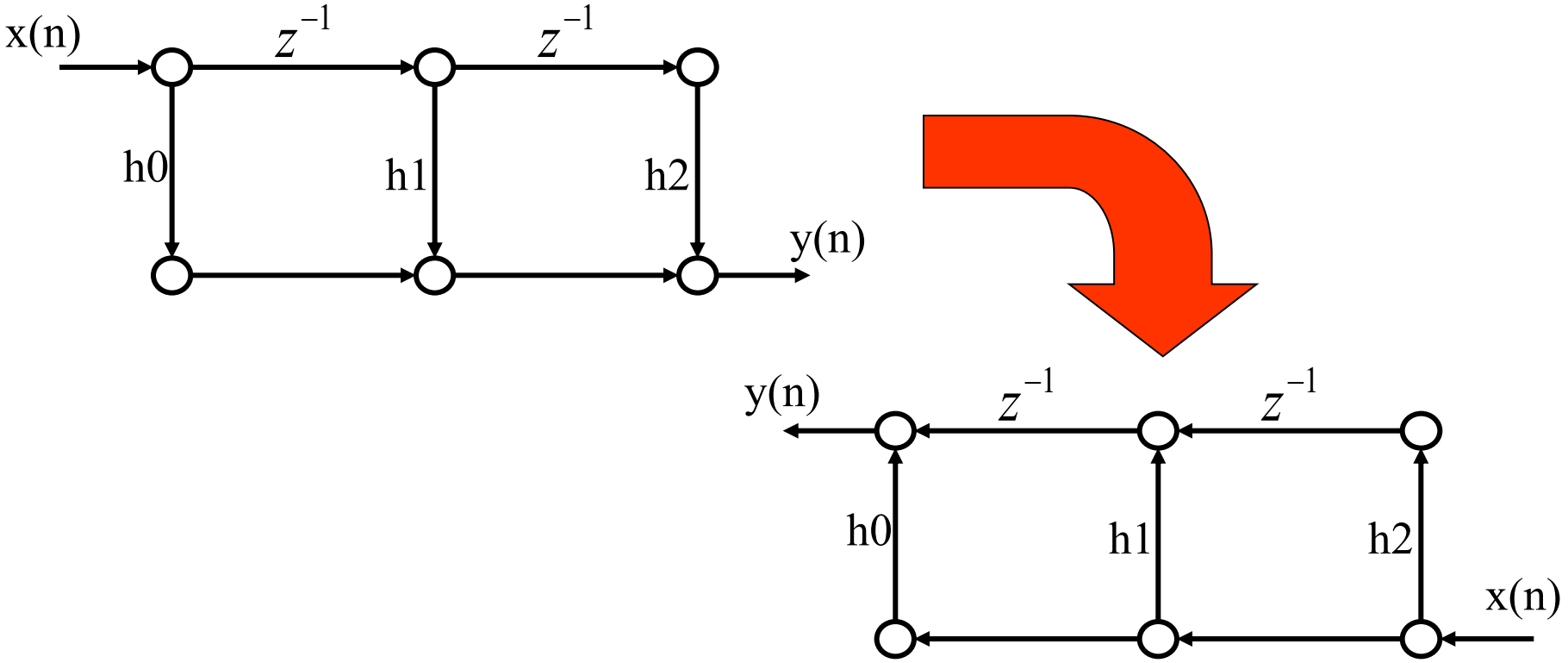
Transposition

Linear SFGs can be transformed without changing the system functions. For example, *Flow graph reversal* or *transposition*

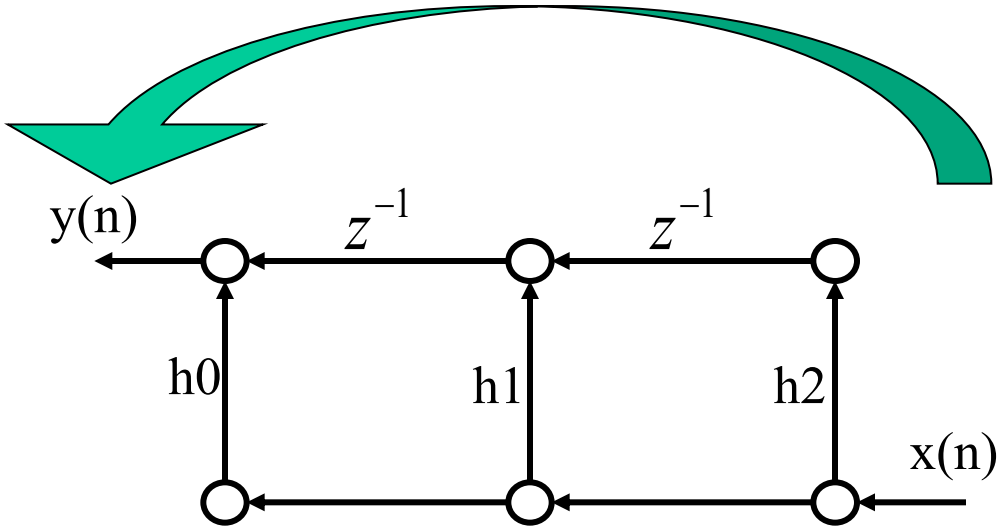
- **Note: only applicable to single-input-single-output systems**

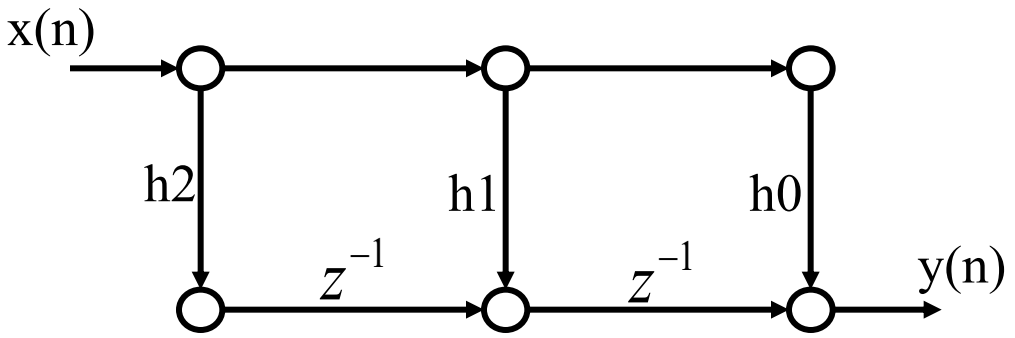
Algorithm: Step 1: Reverse the direction of all edges

Step 2: Exchange input and output

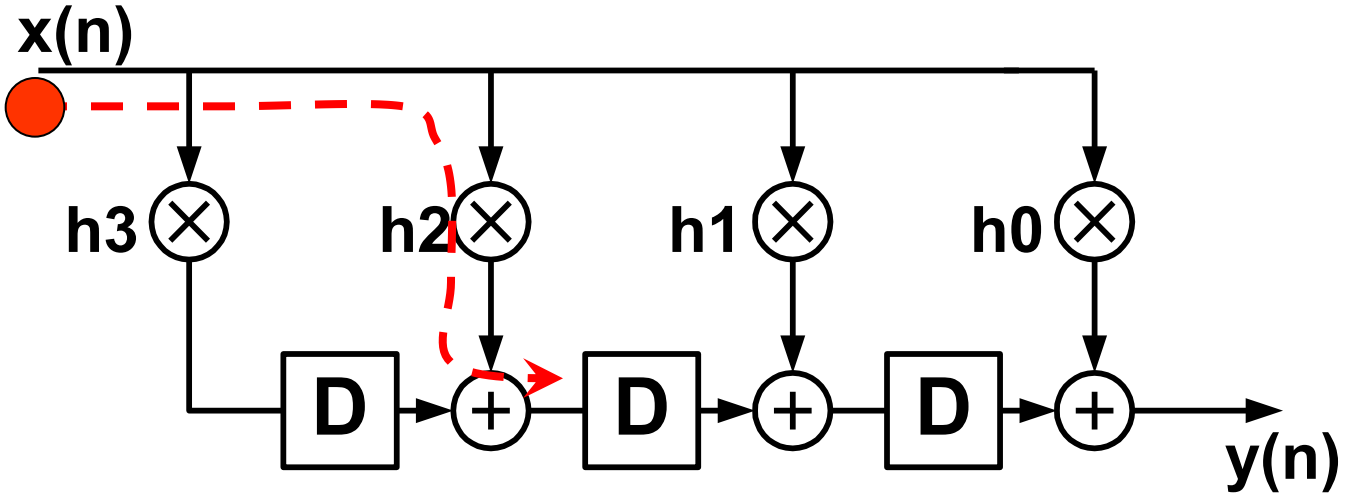


Transposition, *continued*





Transposed Form 4-tap FIR



$T_{Critical} = T_M + T_A$

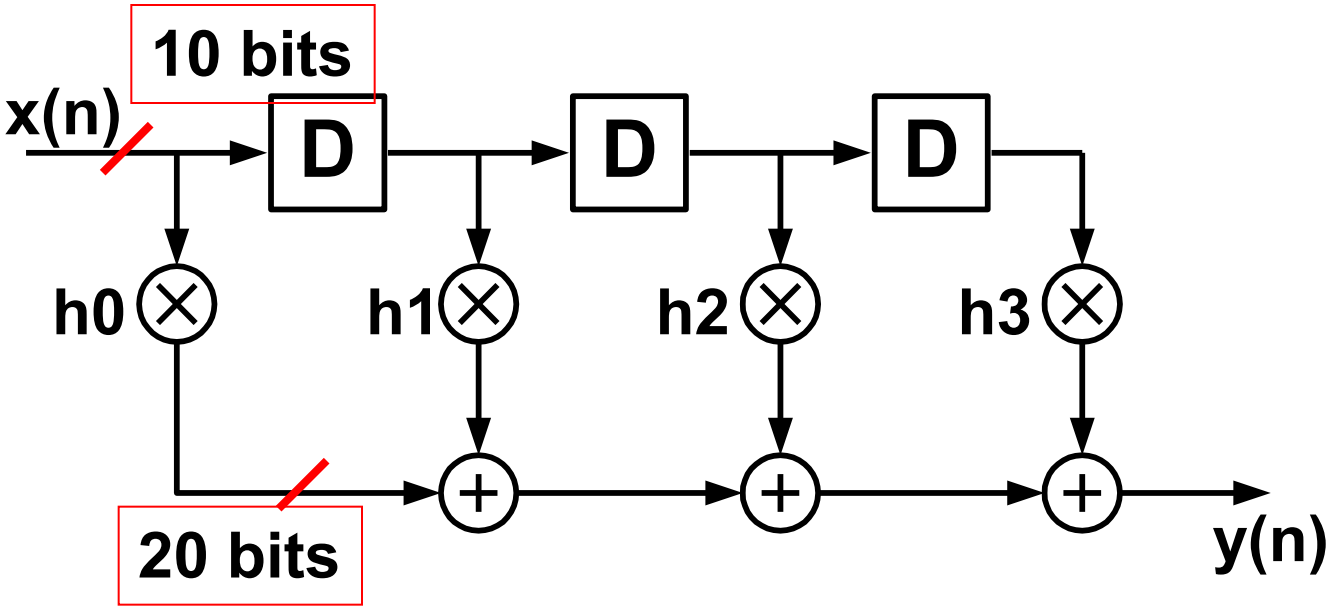
N = Nr. of Taps

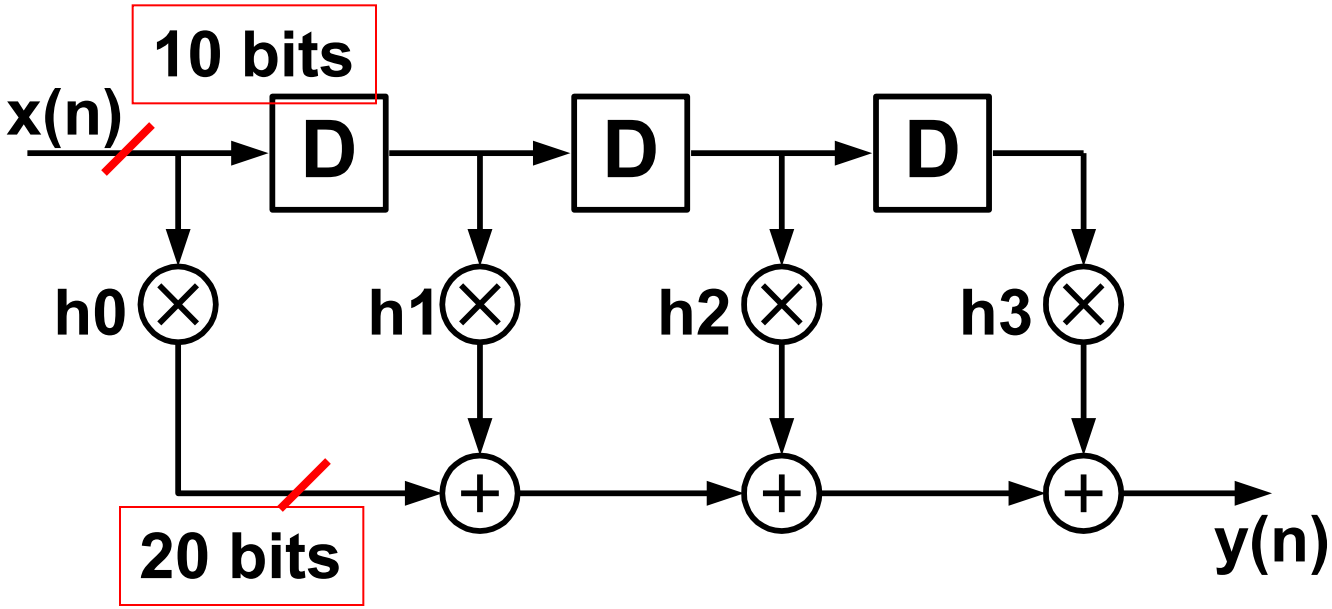
Short critical path!!
Downsides??

Fixed Point Implementation

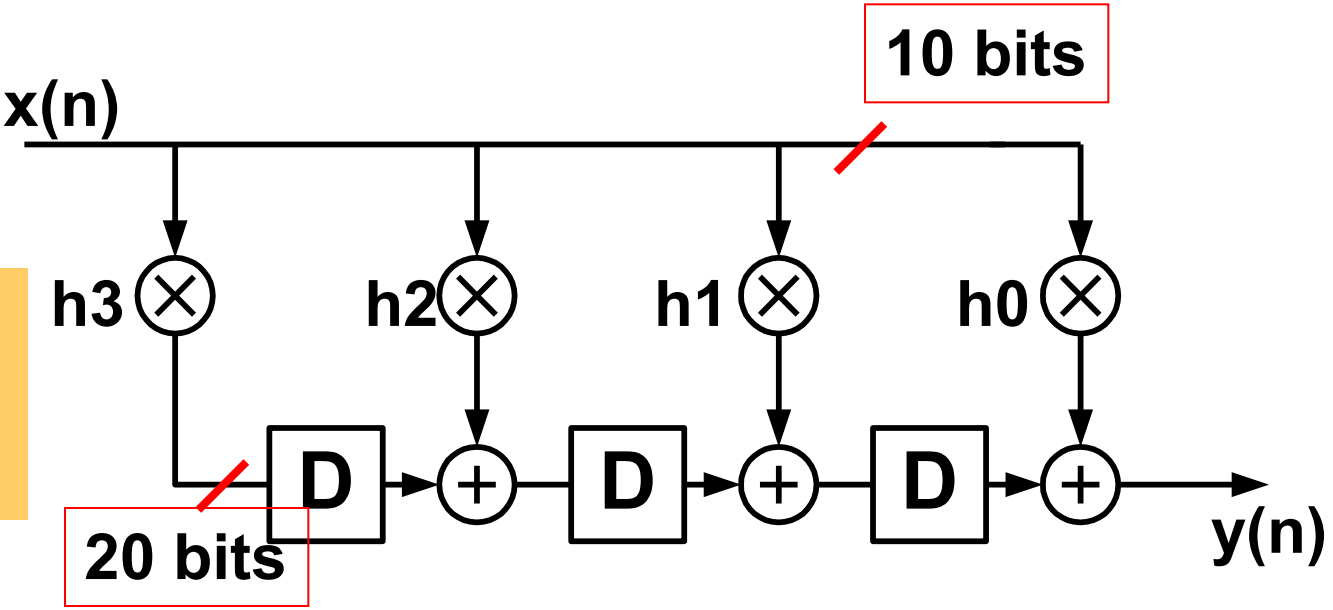
Wordlength is a crucial parameter for:

- Algorithm Performance.
- Power Consumption.
- Clock Speed.
- Area.





Direct Form
Long Critical Path
Narrow Registers



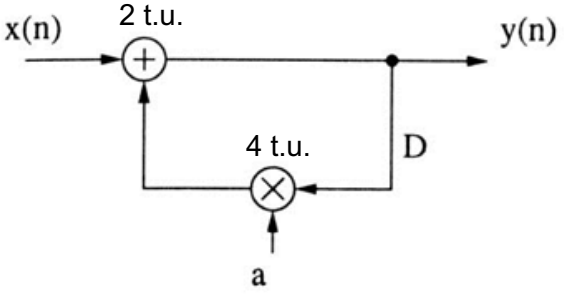
Transposed Form
Short Critical Path
Wide Registers

Graphical Representation
Method 3: Data-Flow Graph (DFG)

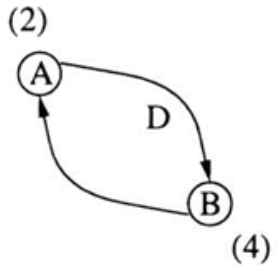
- DFGs capture the data-driven property of DSP algorithm.
- Nodes represent computations (or functions).
- Directed edges represent data flow with non-negative number of delays.
- A node can execute whenever all its input data are available.
- Each edge describes a precedence constraint between two nodes 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 (i.e. different sample rates) DSP systems

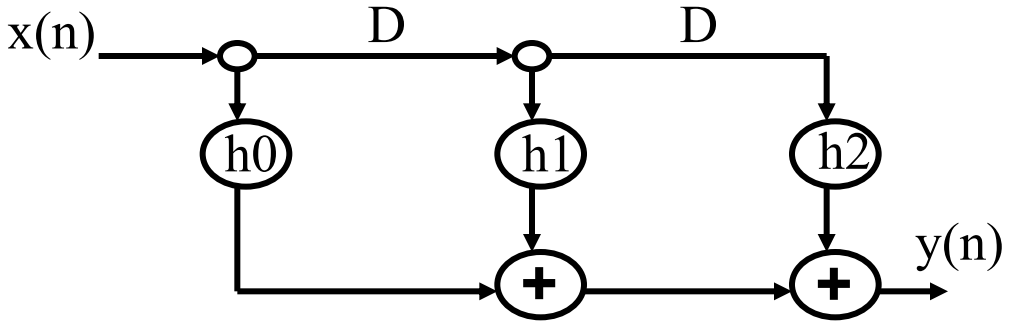
Block diagram - closer to HW
DFG - represents data flow

Block diagram

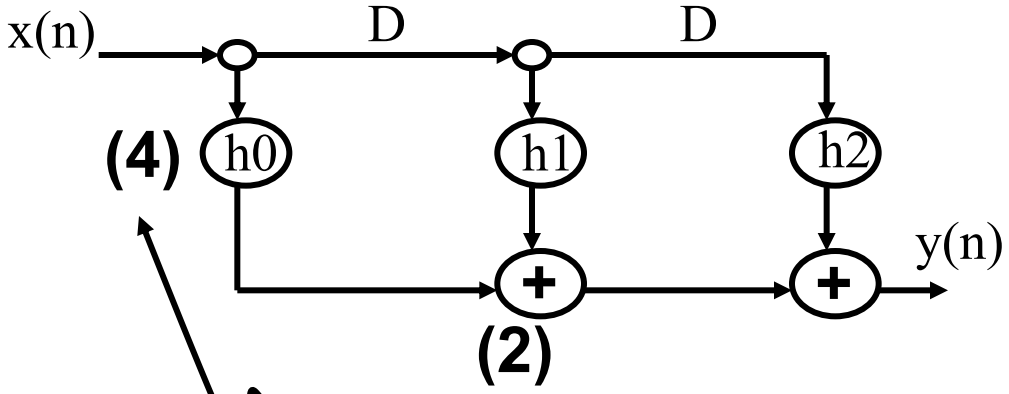


DFG

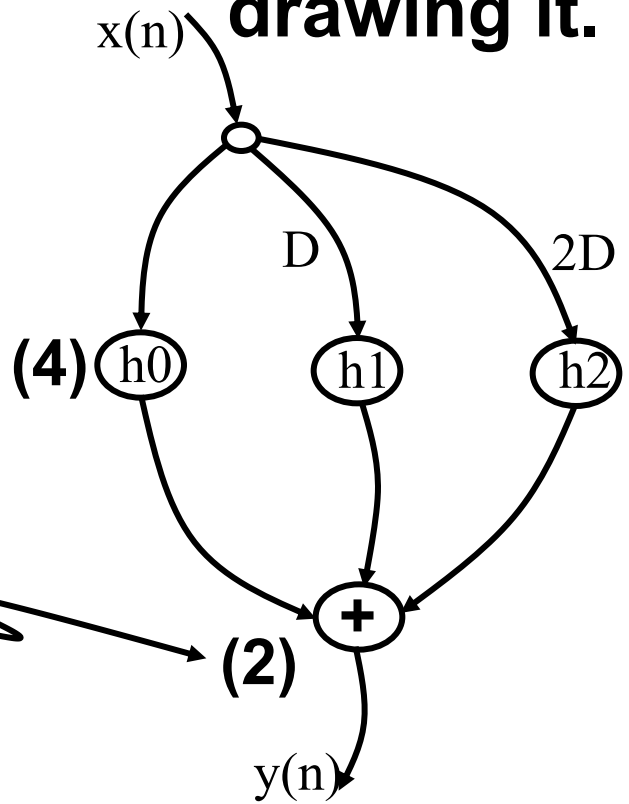




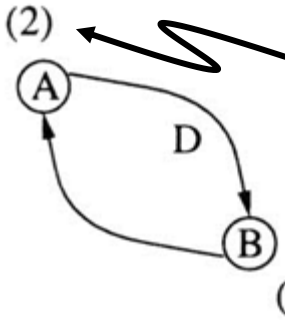
Data-Flow Graph (DFG)



Another way of drawing it.



Computation time

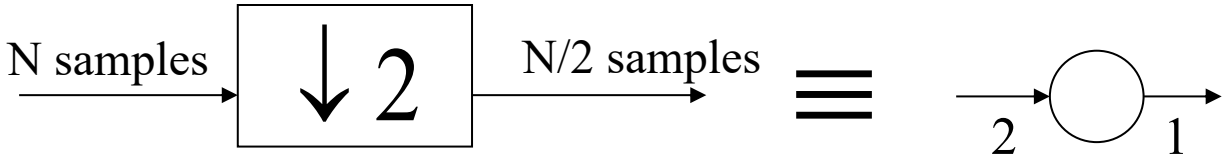


Examples of DFG

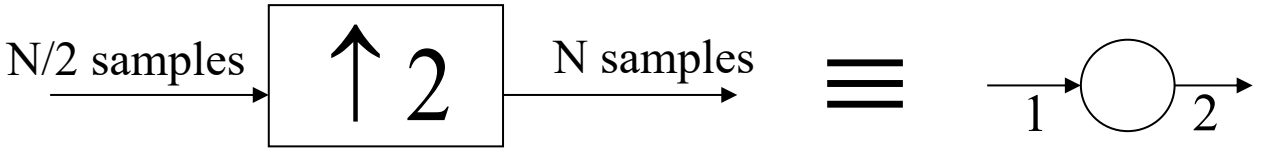
- If indivisible operations (add, logic operations) ⇨ atomic DFG
- Nodes are complex blocks ⇨ Coarse-Grain DFG

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

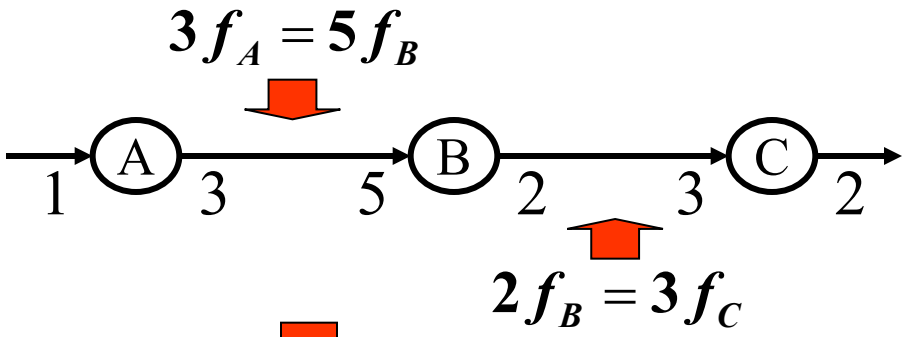
Decimator

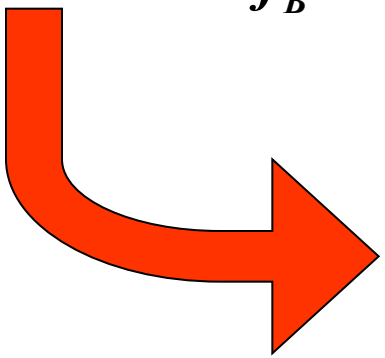


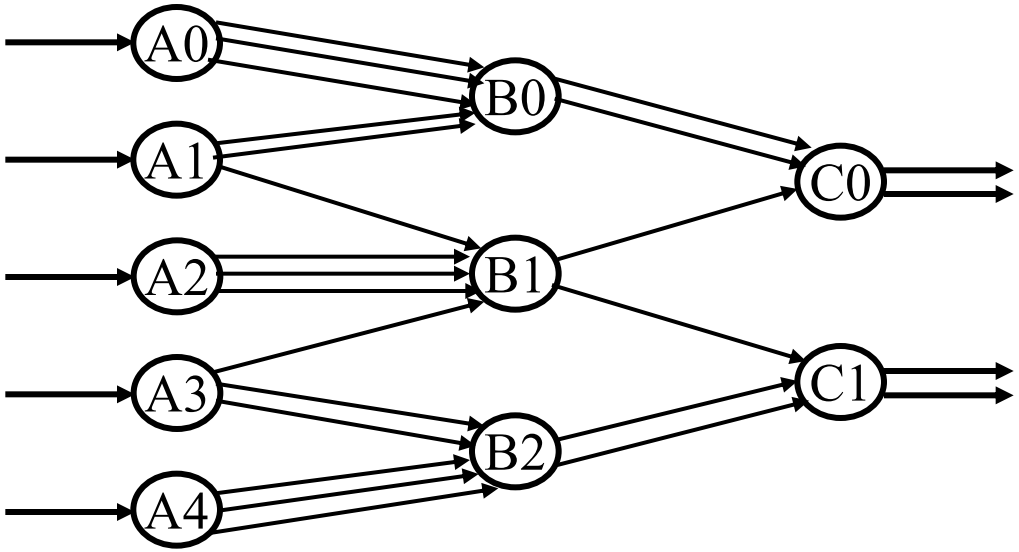
Expander



Multi Rate DFGs







Single-rate DFG representation
More on Unfolding

Iteration Bounds

Chapters 2.1 – 2.6

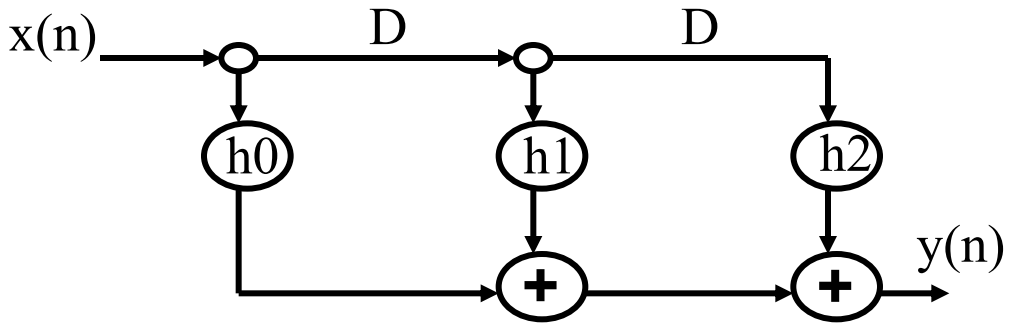
Iteration Bound

Feedback loops within recursive algorithms sets lower bound (limit) on the iteration or sample period

It is not possible to achieve iteration period lower than iteration bound even with INFINITE processing power.

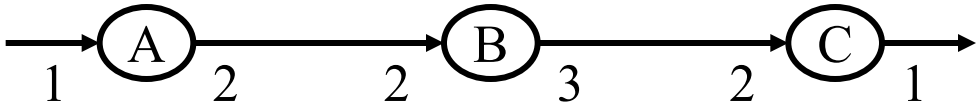
Iterations

- ***Iteration*: execution of all computations (or functions) of an algorithm once**
  - **Example 1, 3-tap FIR:**



One iteration
- **3 mult**
- **2 adds**

  - **Example 2, Multi-rate DFG:**



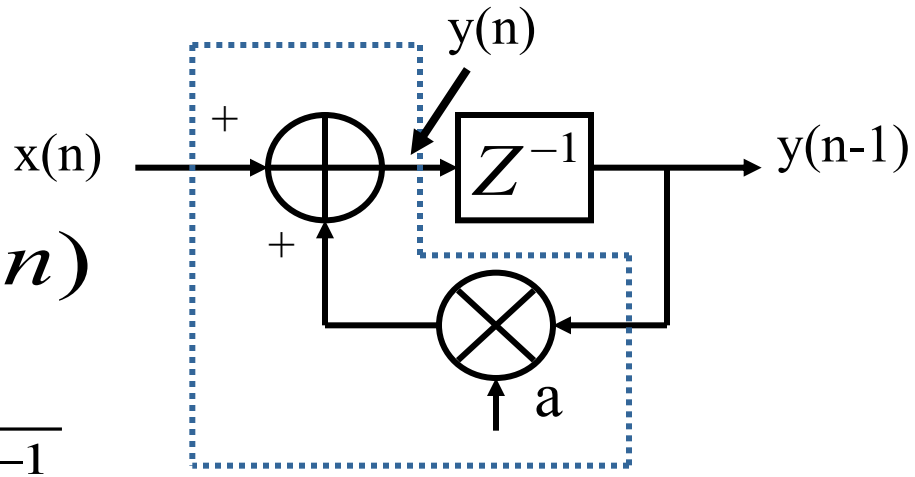
- **For 1 iteration:**

A	B	C
2 times	2 times	3 times

Iterations (cont'd)

***Iteration period*: the time required for execution of one iteration**

- **Example:**



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

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

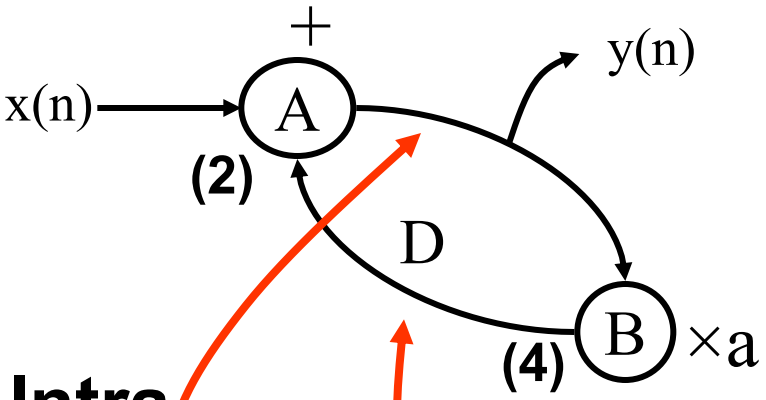
- **Assume the execution times of multiplier and adder are T_m and T_a, then the iteration period for this example is $T_m + T_a$**
- **So the sample period (T_s) must satisfy:**

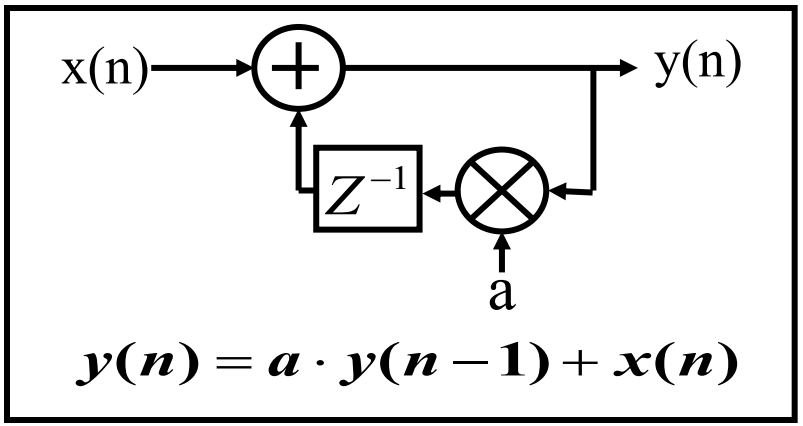
$$T_s \geq T_m + T_a$$

Definition Precedence Constraints

Gives order of execution of each node

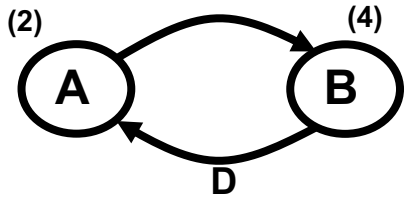
- Each edge of DFG defines a precedence constraint
  - Intra-iteration → edges with no delay elements
  - Inter-iteration ⇒ edges with non-zero delay elements



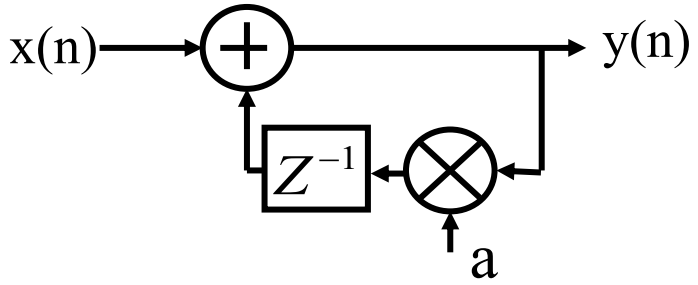


$$A_0 \rightarrow B_0 \Rightarrow A_1 \rightarrow B_1 \Rightarrow A_2 \ldots$$

Precedence Constraints



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



$$y(0) = a \cdot y(-1) + x(0)$$

$$y(1) = a \cdot y(0) + x(1) = a(a \cdot y(-1) + x(0)) + x(1)$$

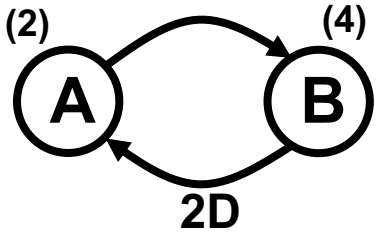
$$y(2) = a \cdot y(1) + x(2) = a(a \cdot y(0) + x(1)) + x(2) =$$

$$a(a(a \cdot y(-1) + x(0)) + x(1)) + x(2)$$

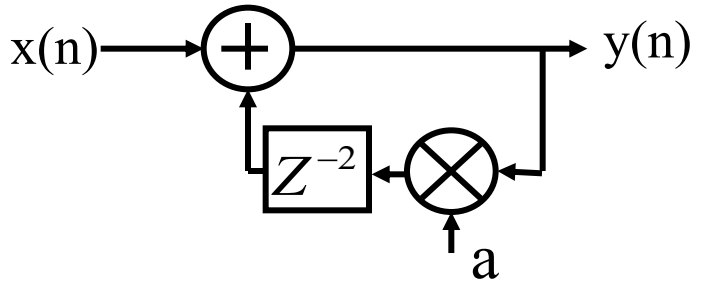
$$y(3) = \ldots$$

$$A_0 \rightarrow B_0 \Rightarrow A_1 \rightarrow B_1 \Rightarrow A_2 \ldots$$

Precedence Constraints



$$y(n) = x(n) + a\ y(n-2)$$



$$y(0) = a \cdot y(-2) + x(0)$$

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

$$y(2) = a \cdot y(0) + x(2) = a(a \cdot y(-2) + x(0)) + x(2)$$

$$y(3) = a \cdot y(1) + x(3) = a(a \cdot y(-1) + x(1)) + x(3)$$

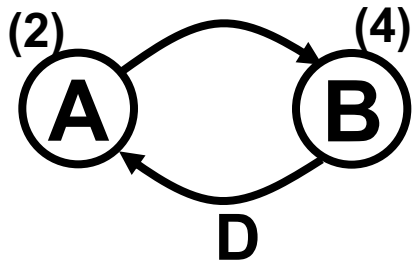
$$y(4) = a \cdot y(2) + x(4) = a(a(a \cdot y(-2) + x(0)) + x(2)) + x(4)$$

$$y(5) = \ldots$$

$$A_0 \rightarrow B_0 \Rightarrow A_2 \rightarrow B_2 \Rightarrow A_4 \ldots$$

$$A_1 \rightarrow B_1 \Rightarrow A_3 \rightarrow B_3 \Rightarrow A_5 \ldots$$

Precedence Constraints (3)



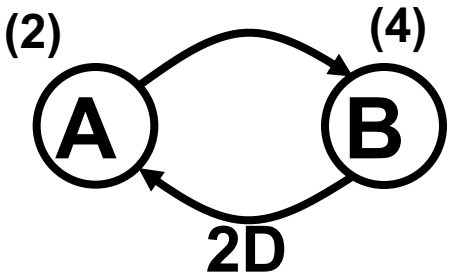
$$y(n) = x(n) + a\ y(n-1)$$

$$A_0 \rightarrow B_0 \Rightarrow A_1 \rightarrow B_1 \Rightarrow A_2 \ldots$$

$$y(1) = x(1) + a\ y(0)$$

$$y(2) = x(2) + a\ y(1)$$

$$y(3) = x(3) + a\ y(2)$$



$$y(n) = x(n) + a\ y(n-2)$$

$$A_0 \rightarrow B_0 \Rightarrow A_2 \rightarrow B_2 \Rightarrow A_4 \ldots$$

$$A_1 \rightarrow B_1 \Rightarrow A_3 \rightarrow B_3 \Rightarrow A_5 \ldots$$

$$y(1) = x(1) + a\ y(-1)$$ ⟸ **Only odd input samples!**

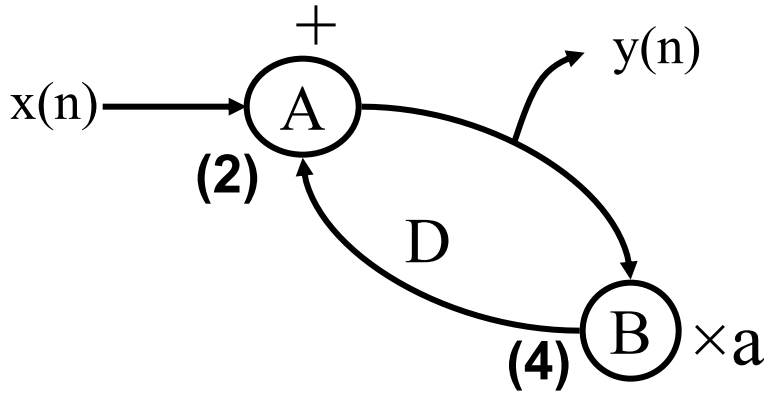
$$y(2) = x(2) + a\ y(0)$$ ⟸ **Only even input samples!**

$$y(3) = x(3) + a\ y(1)$$

Loop Bound

A loop begins and ends in the same node:

$$A_0 \rightarrow B_0 \Rightarrow A_1$$



Iteration k is $A_k + B_k$ and requires 2+4 = 6 t.u.

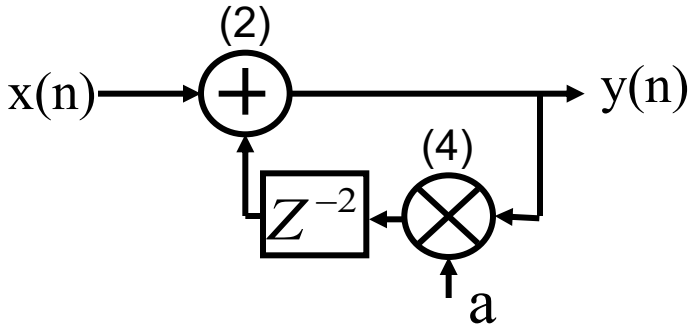
(t.u. = time units)

Loop Bound

- Definitions:
  - Loop bound of the j-th loop is defined as $\frac{T_j}{W_j}$
  - where T_j is the loop computation time
  - W_j is the number of delays in the loop
  - **Example:**

$$y(n) = a \cdot y(n-2) + x(n)$$

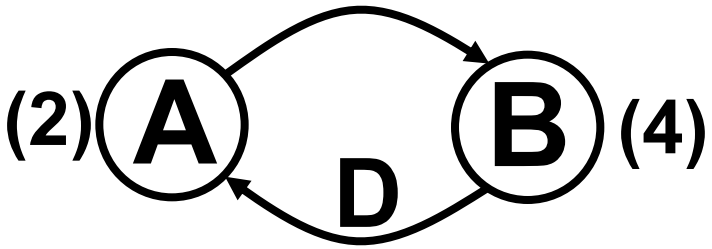
Obs!



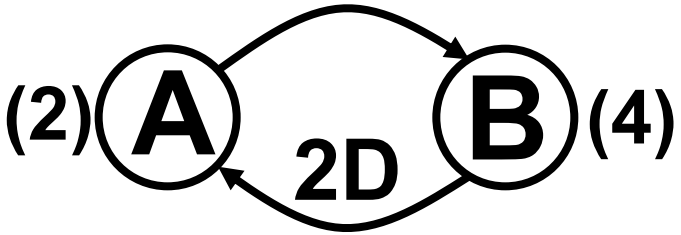
$$T_{loopbound} = \frac{T_m + T_a}{2} = \frac{4+2}{2} = 3t.u.$$

Loop Bound

is the lower bound on the loop computation time.

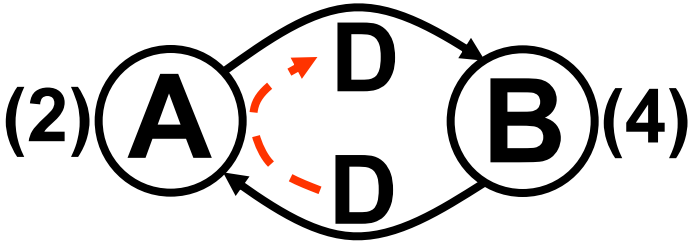


Critical path = 6
Loop bound = 6



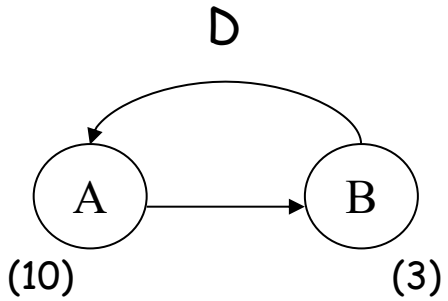
Critical path = 6
Loop bound = 6/2 = 3

Same loop bound
Different critical path

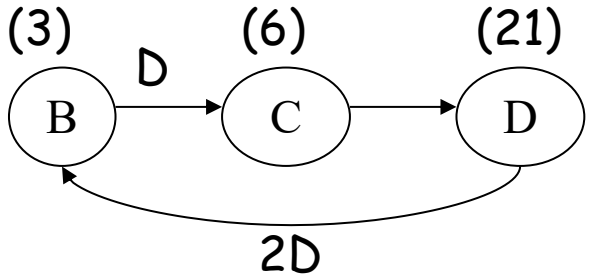


Critical path = 4
Loop bound = 6/2 = 3

- **Loop Bound examples**



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



$B_N \Rightarrow C_{N+1} \rightarrow D_{N+1} \Rightarrow B_{N+3} \Rightarrow C_{N+4} \rightarrow \ldots$

$B_0 \Rightarrow C_1 \rightarrow D_1 \Rightarrow B_3 \Rightarrow C_4 \rightarrow D_4 \Rightarrow B_6$
$B_1 \Rightarrow C_2 \rightarrow D_2 \Rightarrow B_4 \Rightarrow C_5 \rightarrow D_5 \Rightarrow B_7$
$B_2 \Rightarrow C_3 \rightarrow D_3 \Rightarrow B_5 \Rightarrow C_6 \rightarrow D_6 \Rightarrow B_8$

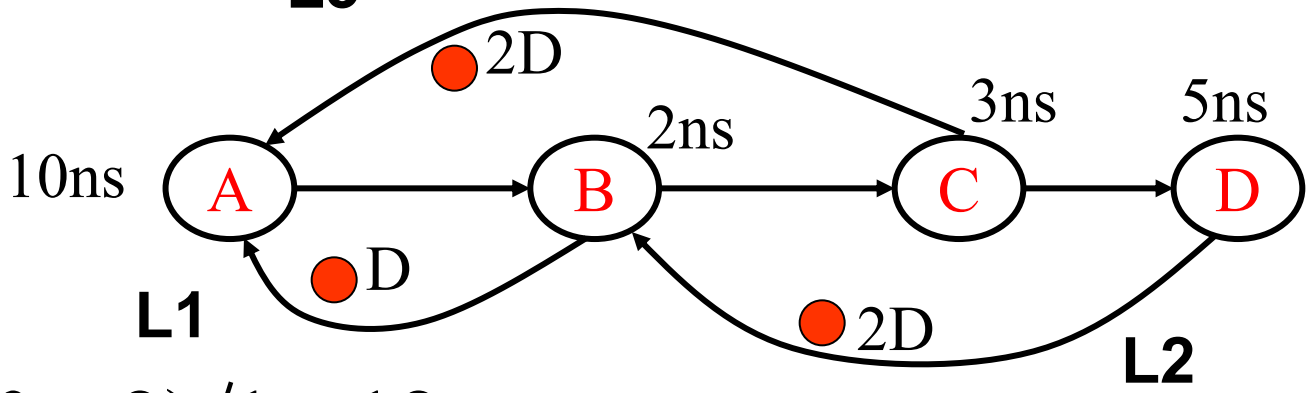
Loop contains three delay elements

loop bound = 30 / 3 =10 t.u. = (loop computation time) / (#of delay elements)

Iteration Bound

Question! What are the loop bounds of the following loops??

$$T_{L3} = (10+2+3)/2 = 7.5ns$$



$$T_{L1} = (10+2)/1 = 12ns$$

$$T_{L2} = (2+3+5)/2 = 5ns$$

Iteration Bound

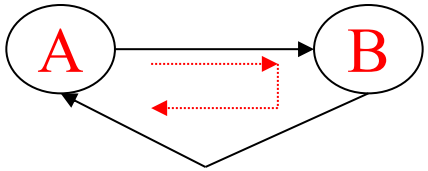
The critical loop is the loop with maximum loop bound which is the bound for the DSP program!!

This is called the Iteration Bound, T_∞!

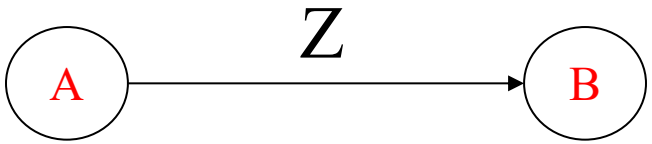
$$T_\infty = \max_{l \in L} \left\{ \frac{t_l}{w_l} \right\}$$

Iteration bound (cont'd)

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



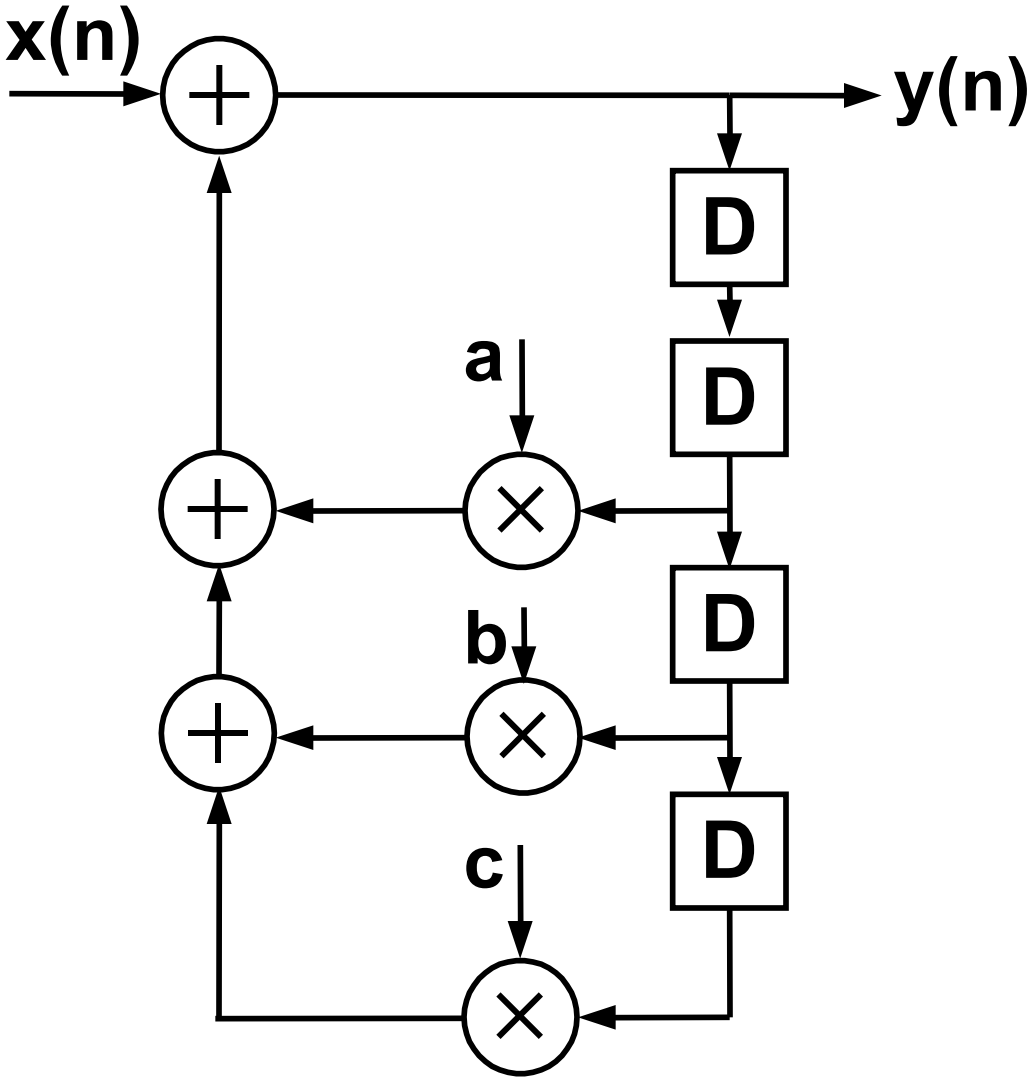
- Non-causal systems cannot be implemented



$$\begin{cases} B = A \cdot Z & non-causal \\ A = B \cdot Z^{-1} & causal \end{cases}$$

- Clock speed of the system: depends on the "critical path"
  - 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

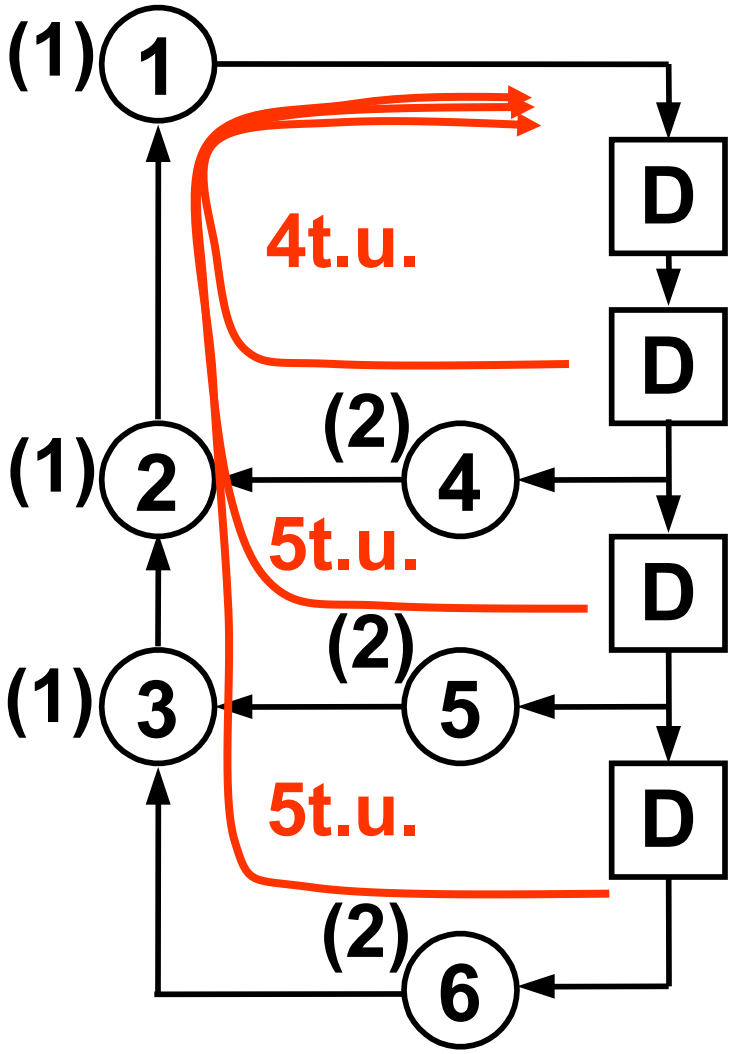
How to calculate Loop Bounds



An IIR-filter

T_{mult} = 2 t.u.

T_{add} = 1 t.u.



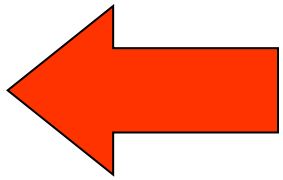
$\frac{4}{2}$

$\frac{5}{3}$

$\frac{5}{4}$

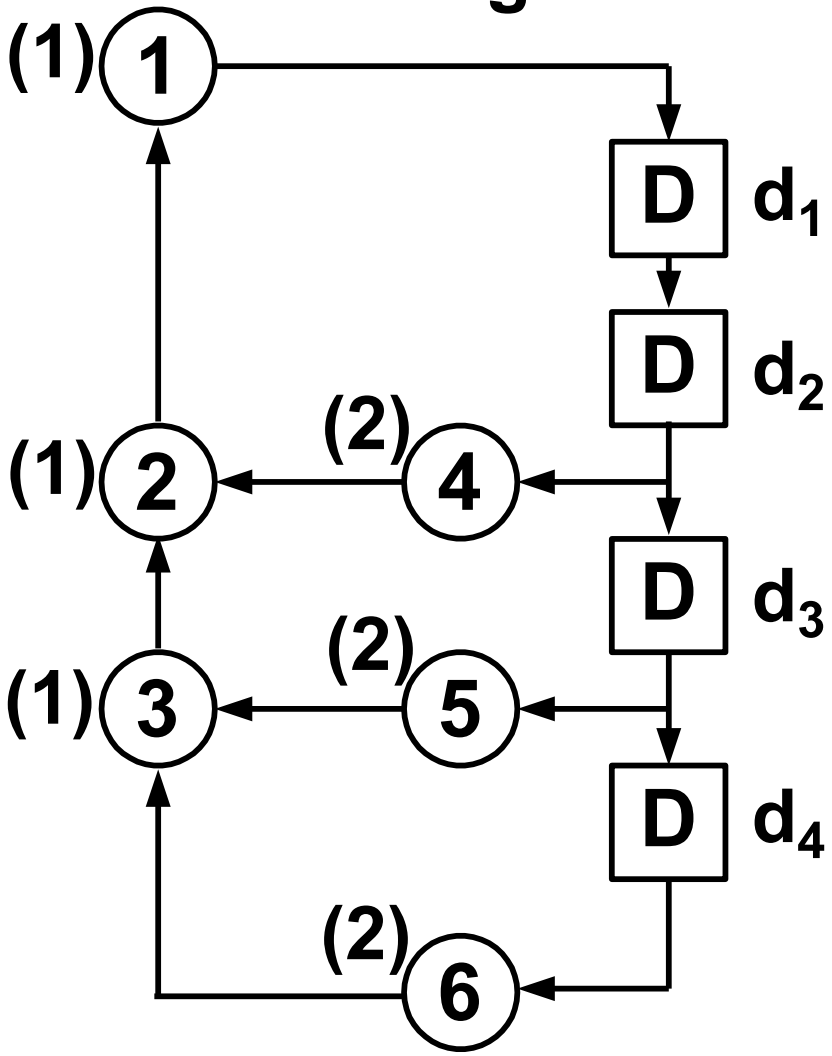
Iteration Bound

- **Algorithms to compute iteration bound**
  - **Longest Path Matrix (LPM)** (2.4.1)
  - **Minimum Cycle Mean (MCM)**



Longest Path Matrix

Algortihm to compute T_∞



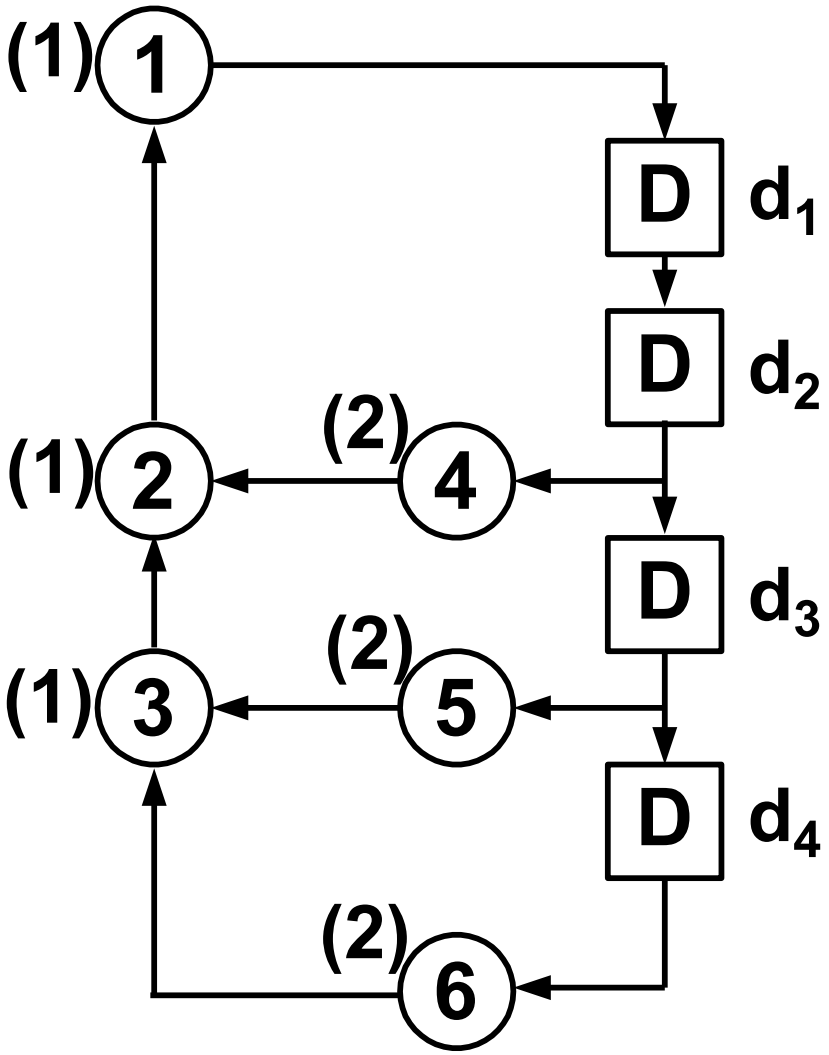
Construct Matrices

$L^{(m)}$, $m=1,2, ..., d$

where each element $l_{i,j}^{(m)}$

is the longest path from d_i **to** d_j **with** $m-1$ **delays (-1 if no path)**

Longest Path Matrix (2)

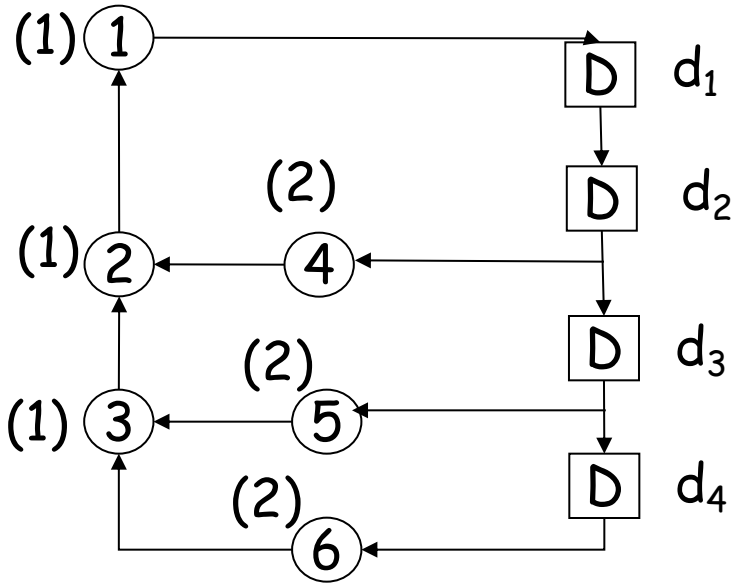


The Iteration Bound can be calculated from the $L^{(m)}$ matrices by

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

since the diagonal element represents the longest comp. time of all loops with m delays which contains d_i

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



Construct Matrices

$L^{(m)}, m=1,2, ..., d$

where

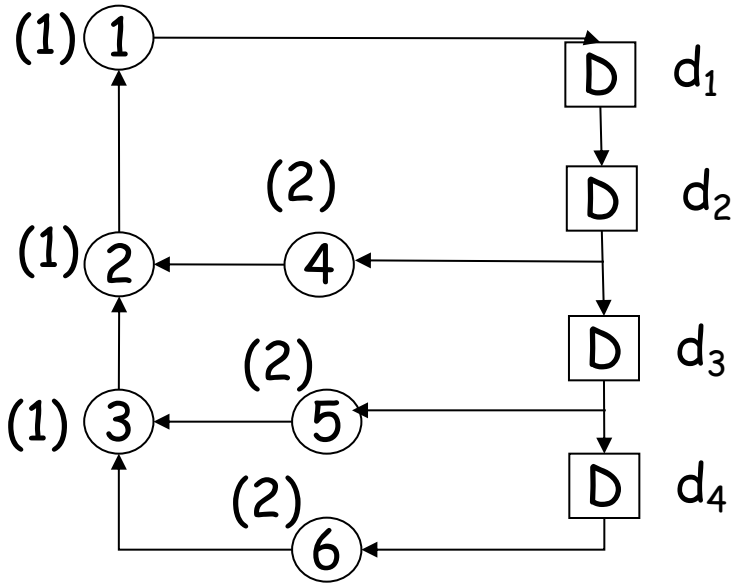
each element $l_{i,j}^{(m)}$

is the longest path from d_i **to** d_j **with** $m-1$ **delays (-1 if no path)**

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

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

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



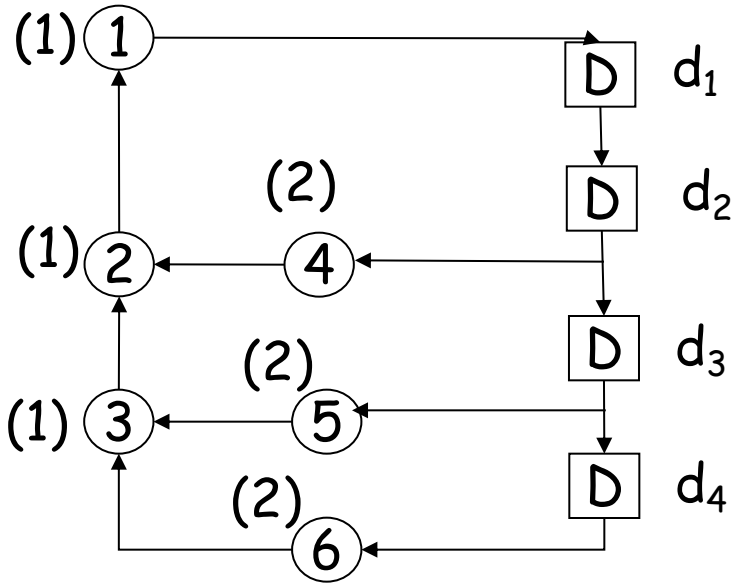
$$L^{(4)} = \begin{bmatrix} 8 & 5 & 4 & -1 \\ 9 & 8 & 5 & 4 \\ 10 & 9 & 5 & 5 \\ 10 & 9 & -1 & 5 \end{bmatrix}$$

$L^{(1)}$ is computed from DFG while higher order matrices are computed recursively.

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

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

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



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

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

- **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,

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

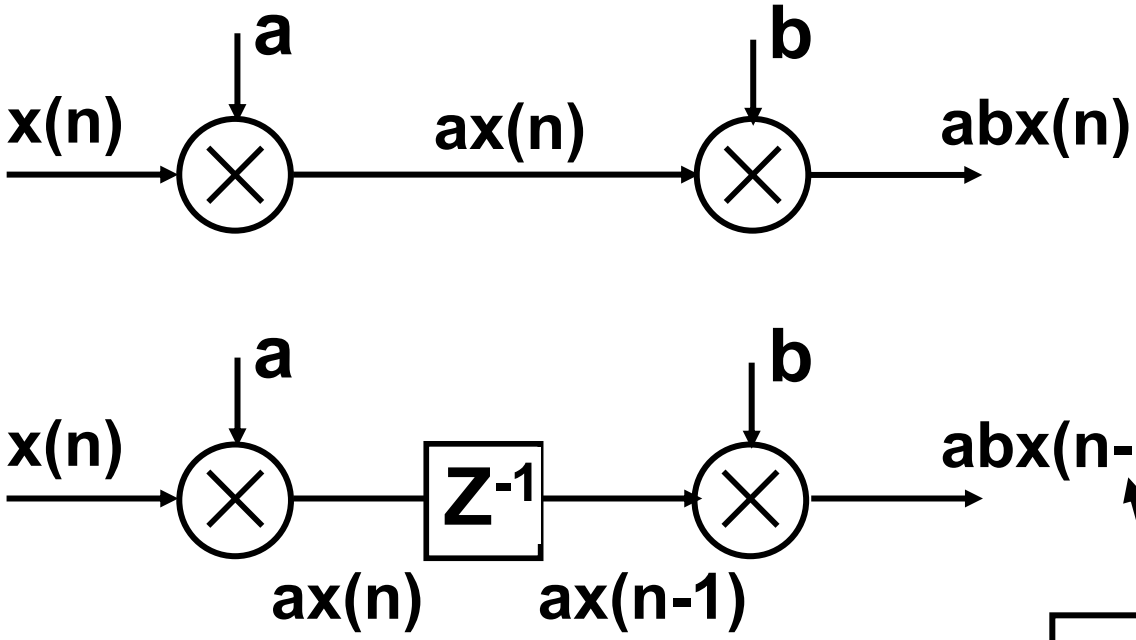
Pipelining and Parallel Processing

Introduction

- **Pipelining**
  - **Splits the logic path by introducing pipeline registers.**
  - **Leads to a reduction of the critical path but introduce latency.**
  - **Either increases the clock speed (or sampling speed) or reduces the power consumption at same speed in a DSP system.**

- **Parallel Processing**
  - **Multiple outputs are computed in parallel in a clock period.**
  - **The effective sampling speed is increased by the level of parallelism.**
  - **Can also be used to reduce the power consumption.**

Pipelining



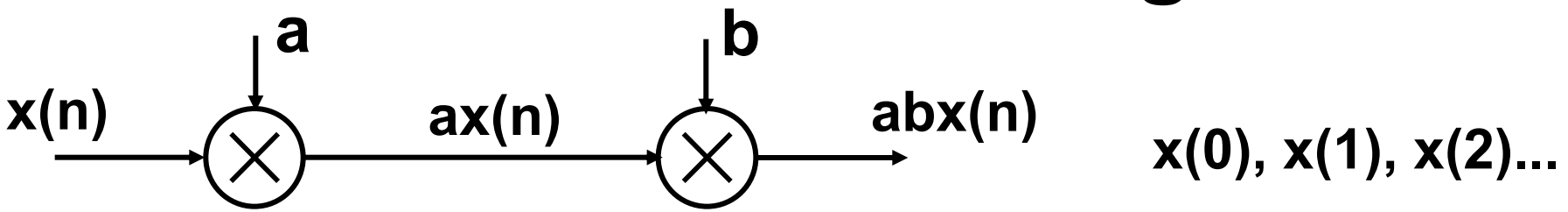
Critical path cut in half

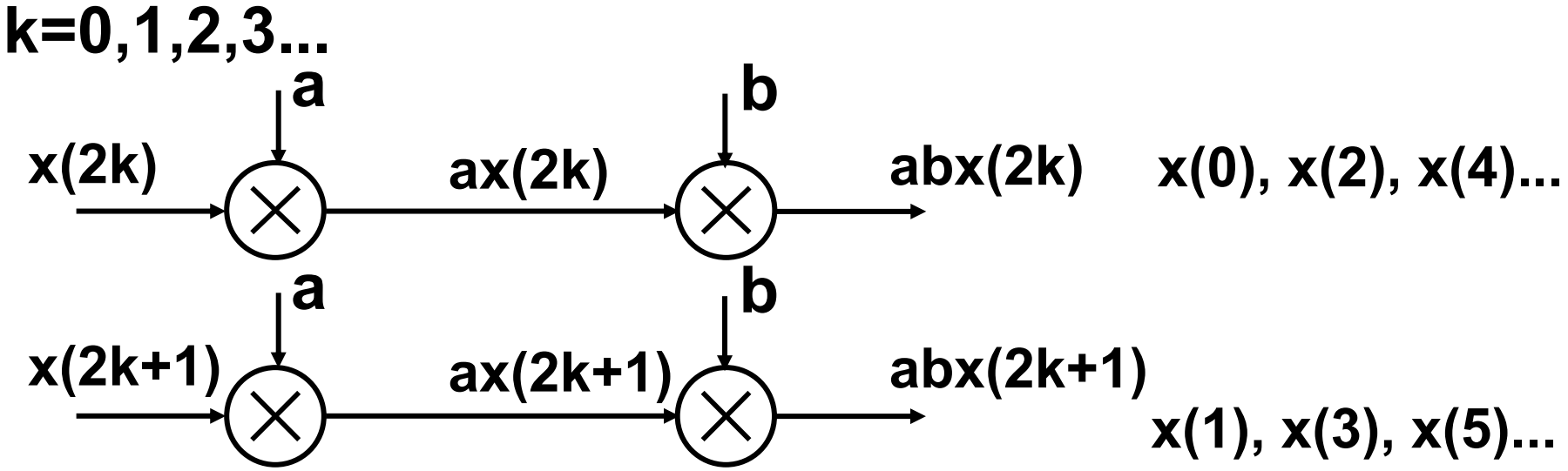
- **double clock speed**

or

- **lower power consumption due to reduced V_{DD}**

Parallel Processing





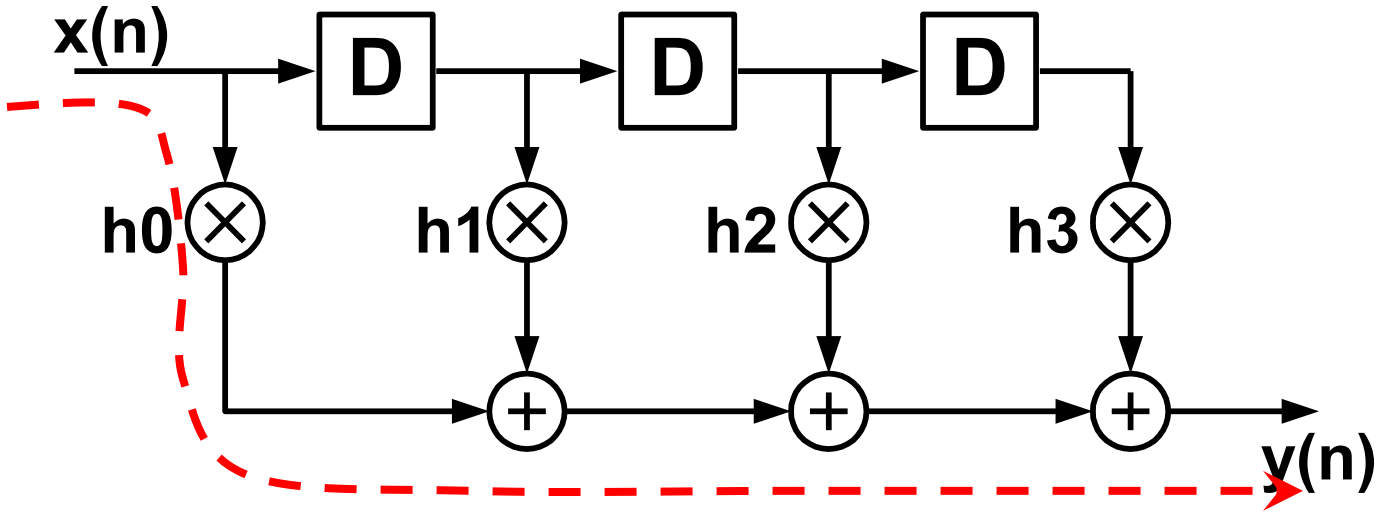
Two samples are processed in parallel

- double throughput

or

- lower power consumption due to reduced V_{DD}

Direct Form 4-tap FIR



Clock speed limited by Critical Path!

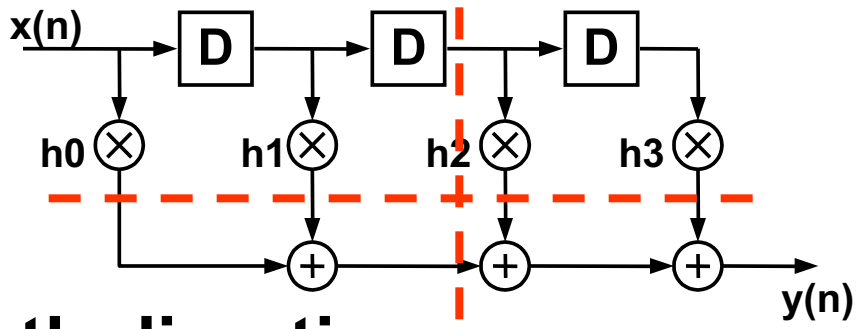
$T_{Critical} = T_M + (N-1)T_A$

N = Nr. of Taps

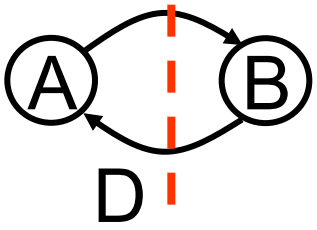
Definitions

Cutset: A set of edges that if removed, or cut, results in two disjoint graphs.

Feedforward Cutset: if data is moved in forward direction on all cutsets.

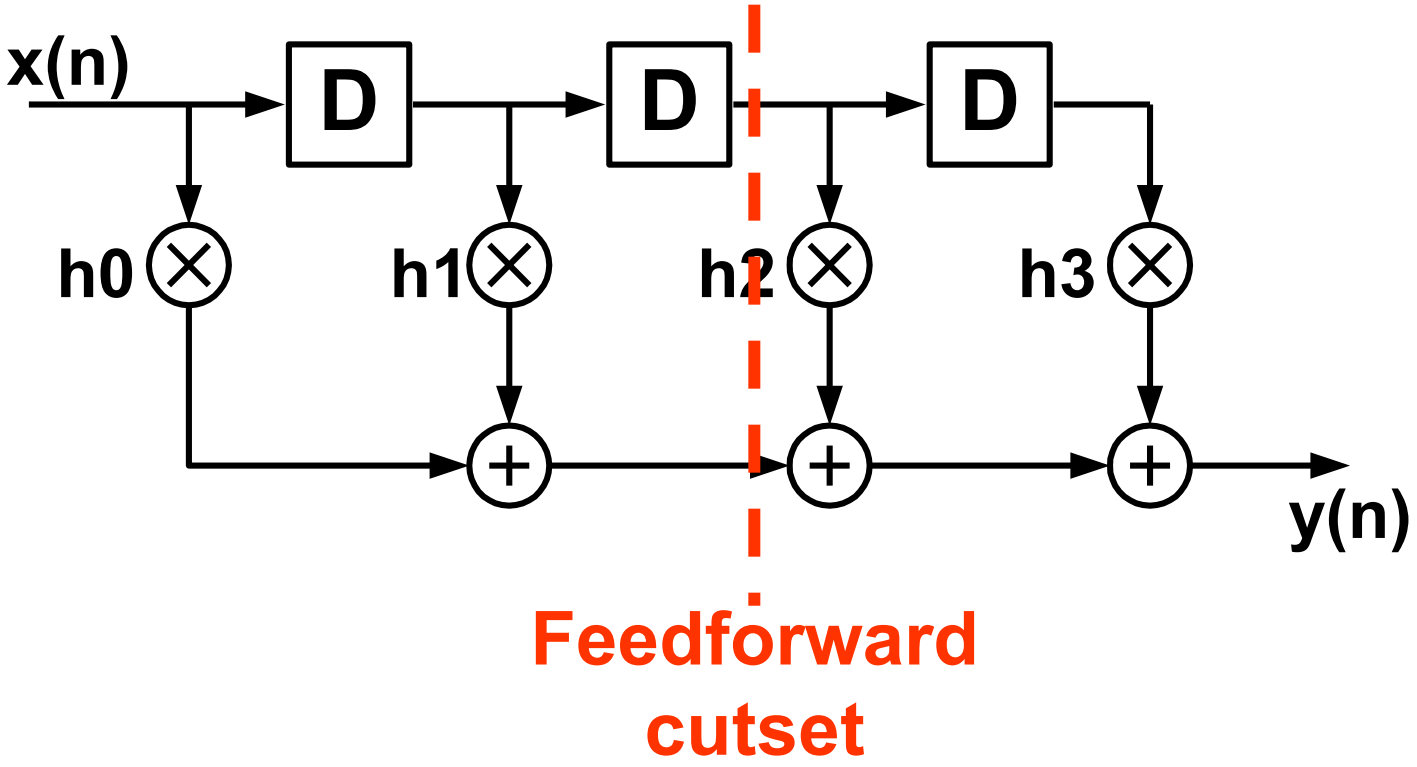


Feedback Cutset: data in both directions.



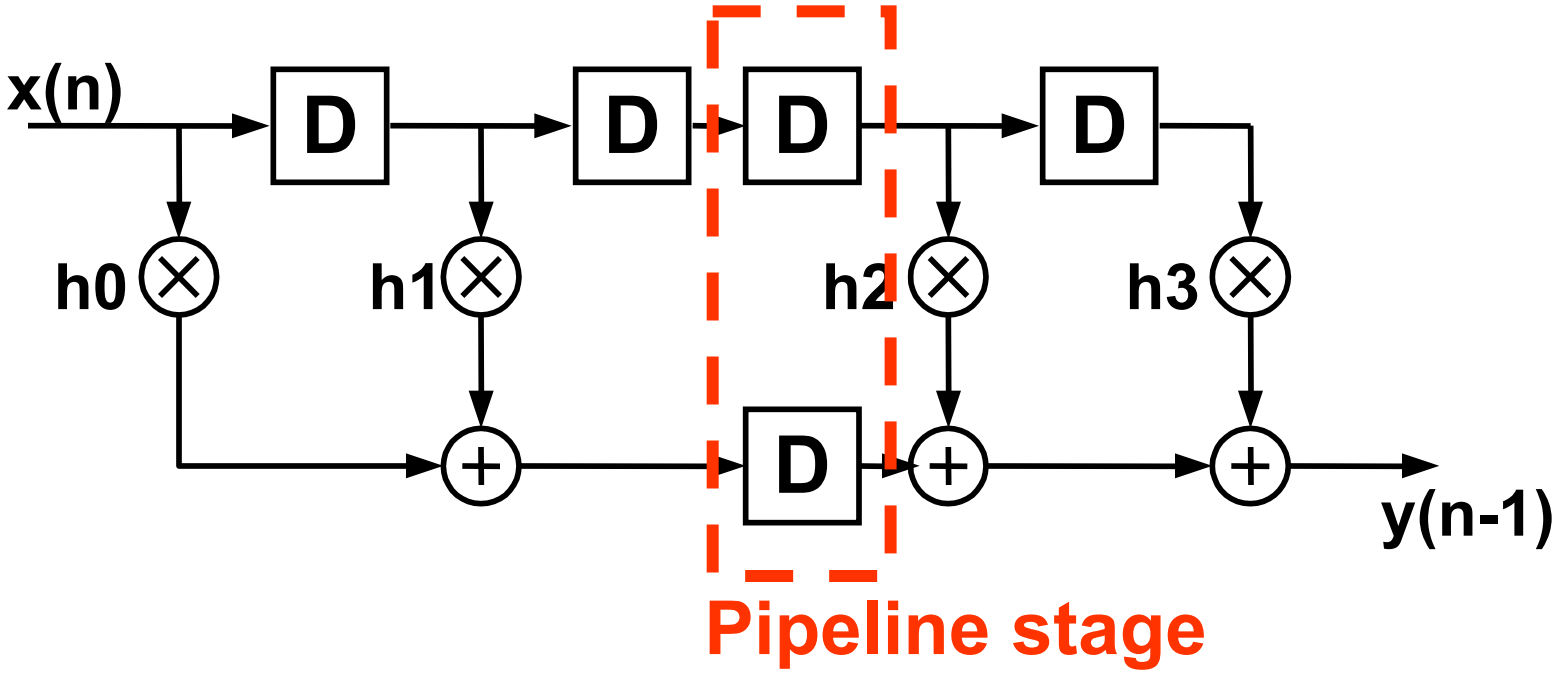
Pipelining

Pipelining: Placing delays at feedforward cutsets.



Pipelining

Pipelining: Placing delays at feedforward cutsets.



Pipelining will not affect functionality but introduce latency!

Pipelining

- In a pipelined system:
  - In an M-level pipelined system, the number of delay elements in any path from input to output is (M-1) greater than that in the same path in the original sequential circuit
  - Pipelining reduces the critical path but leads to increased latency

    ** Latency: the difference in the availability of the first output data in the pipelined system and the sequential system*

- Two main drawback
  - Increase in the number of latches.
  - Increase system latency.

End of Lecture 3