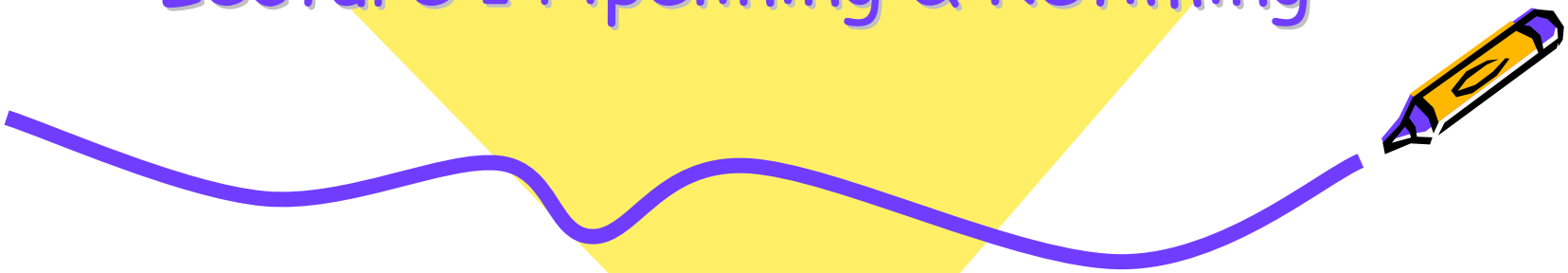
VLSI Signal Processing

Lecture 1 Pipelining & Retiming



Introduction

- DSP System
  - Real time requirement
  - Data driven – synchronized by data
  - Programmable processor
  - ASIC or custom DSP DSPs
  - What to use depends on requirements
    - Throughput
    - Power, Energy
    - Area
    - Wordlength – precision
    - Flexibility
    - Time-to-market
    - Time-in-market

DSP Representations

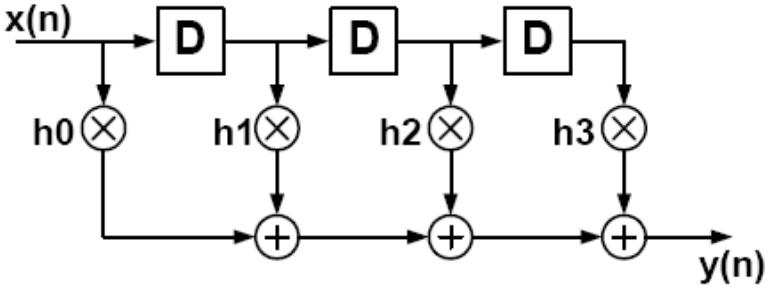
- DSP Algorithm are non-terminating and data-intensive

- Iteration period: the time for one iteration, one execution of the loop within DSP algorithm
- Critical path: longest path without delay
  - Critical path sets bound on clock frequency, i.e. $T_{clk} \geq T_{critical}$
- Sampling rate: number of samples processed / second
- Latency: time difference between output and corresponding input (combinational logic = gate delays, sequential logic = number of clock cycles)
- The clock rate (or frequency) of the DSP system is not the same as its sampling rate

Block Diagram

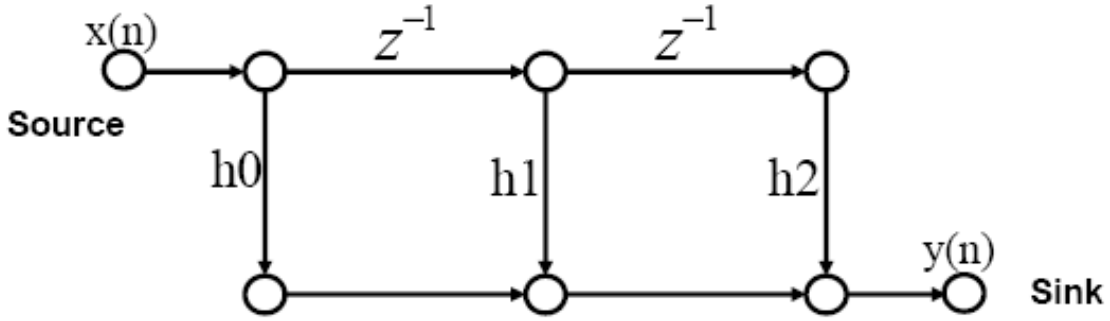
The graphical representations of DSP algorithm exhibit all the *parallelism* (concurrence) and *data-driven properties* (dependence) of the system, as well as provide insight into space and time tradeoffs.

- Example $y[n] = h_0 x[n] + h_1 x[n-1] + h_2 x[n-2] + h_3 x[n-3]$
- Consists of functional blocks connected with directed edges, which represent data flow from its input block to its output block



Signal-Flow Graph (SFG)

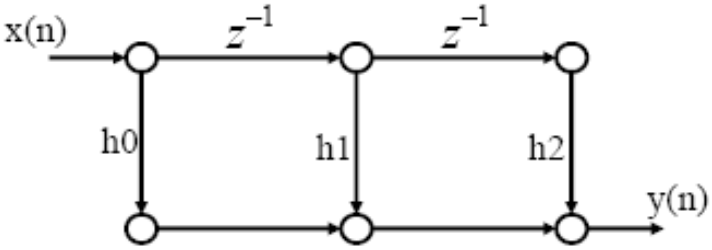
- Nodes: represent computations and/or task
- Directed edge (j,k): denote a linear transformation from the input signal at node j to the output signal at node k
- Source: no entering edge
- Sink: only entering edges

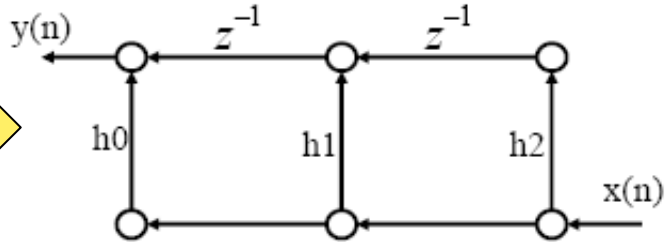


$$y[n] = h_0 x[n] + h_1 x[n-1] + h_2 x[n-2]$$

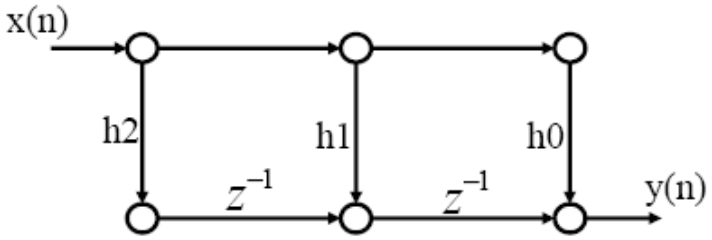
Transposition of an SFG

- Applicable to linear SISO systems
- Flow graph reversal
  - Reserve the direction of all edges
  - Exchange input and output



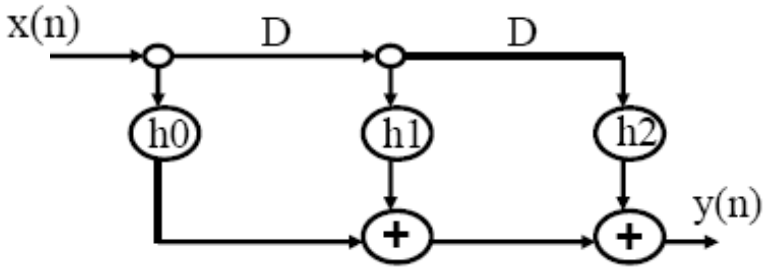


$$y[n] = h_0 x[n] + h_1 x[n-1] + h_2 x[n-2]$$



Data-Flow Graph (DFG)

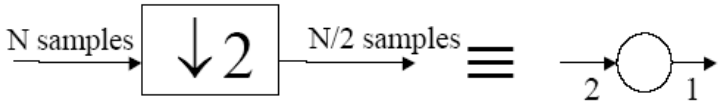
- DFG captures the data-driven property of the DSP algorithm
- Nodes represent computations, functions, or tasks
- Directed edges represents data flow with non-negative number of delays
- A node can execute whenever all its input data are available
- Each edge describes a precedence constraint between two nodes in DFG
  - Intra iteration precedence constraint: if no delay
  - Inter-iteration precedence constraint: if existing one or more delays



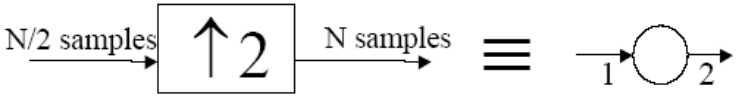
$$y[n] = h_0 x[n] + h_1 x[n-1] + h_2 x[n-2]$$

Examples of DFGs

- DFGs and Block diagrams can be used to describe both linear single-rate and nonlinear multi-rate DSP systems.
- Synchronous DFG (SDFG): the number of input and output samples are specified at each node in each execution.

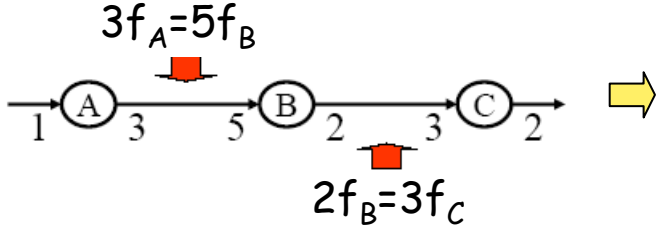


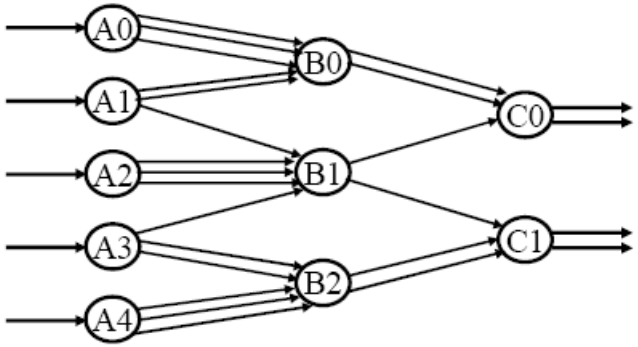
Decimator (down-sampling)



Expander (up-sampling)

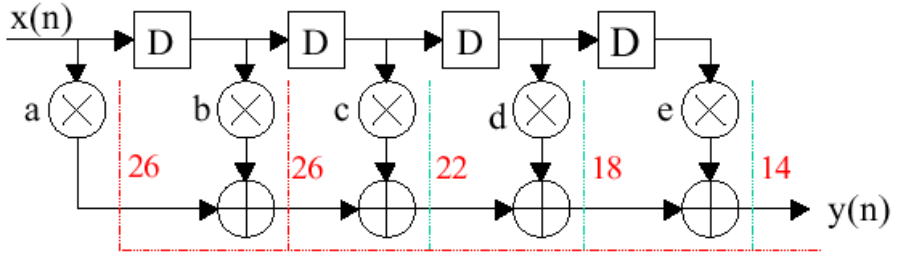
- Multi-rate SDFG





Critical Path – fundamental limit

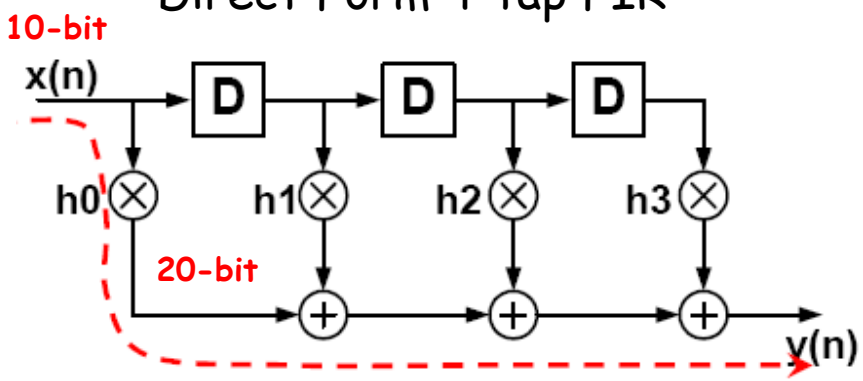
- 4 possible paths
  - Input nodes to delay element
  - Input node to output node
  - Delay element to delay element
  - Delay element to output
- Critical path: the path with the longest computation time among all paths without delay elements
- Clock period is bounded by the computation time of the critical path
- Example: 5-tap FIR filter and assume T_A=4ns, T_M=10ns



critical paths = 26ns

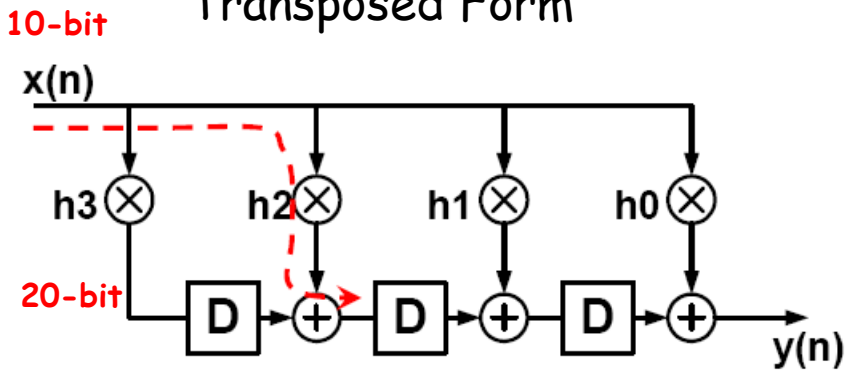
Remarks

Direct Form 4-tap FIR



1. $T_{Critical}=T_M+(N-1)\,T_A$, where N is the number of taps
2. Short word-length delay elements

Transposed Form



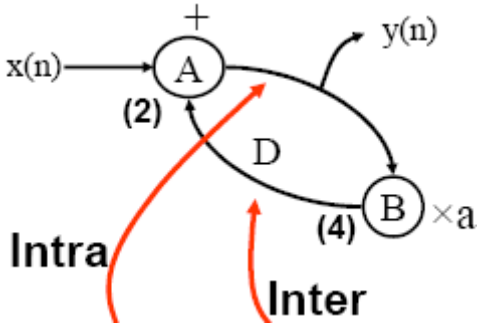
1. $T_{Critical}=T_M+T_A$
2. High fan out causes high driving force
3. Long word-length delay elements

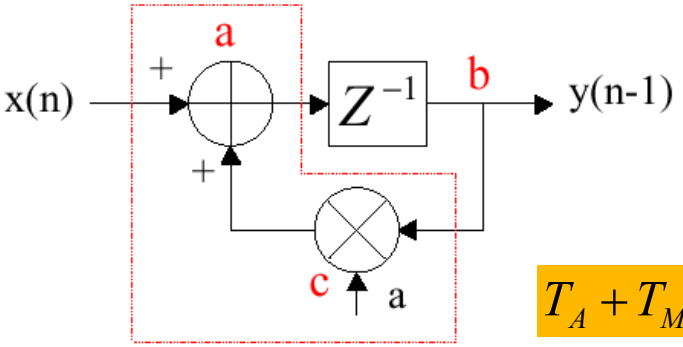
Iteration Period –fundamental limit

- Iteration: execution of all computations of an algorithm once
- Iteration period: the time required for execution of on iteration

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

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





$T_A + T_M$

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

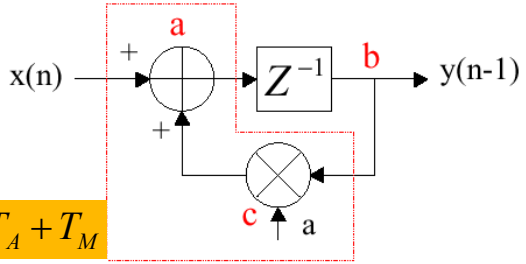
- Iteration rate: the number iterations executed per second
- Sample rate (throughput): the number of samples processed per second

Loop Bound

- Loop: a directed path that begins and ends at the same node
- Loop bound of the loop j: T_j / W_j, where T_j is the loop computation time and W_j is the number of delays in the loop

Examples:

- assume T_A+T_M =6ns,
  a, b, c, a is a loop with loop bound = 6ns



$T_s \geq T_A + T_M$

- assume T_A+T_M =6ns, the loop bound = 3ns

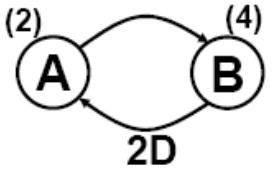
$$A_0 \rightarrow B_0 \Rightarrow A_2 \rightarrow B_2 \Rightarrow A_4 \ldots$$

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

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

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

$y(3) = x(3) + a\ y(1)$ …

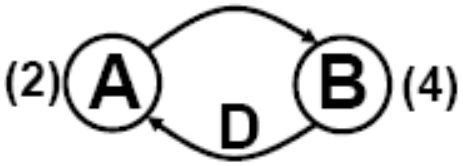


y(n)=ay(n-2)+x(n)

$T_s \geq T_A + T_M / 2$

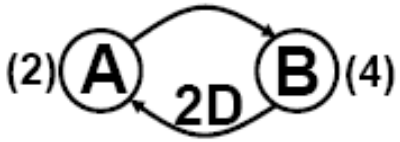
Remark

- The delay elements within a loop determine the loop bound



Critical path = 6
Loop bound = 6

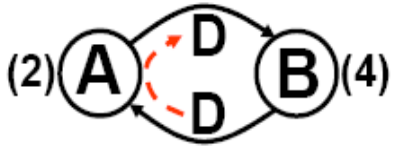
$A_0 \rightarrow B_0 \Rightarrow A_1$



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

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

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



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

$A_N \Rightarrow B_{N+1} \Rightarrow A_{N+2} \Rightarrow B_{N+3} \cdots$

Same loop bound
Different critical path

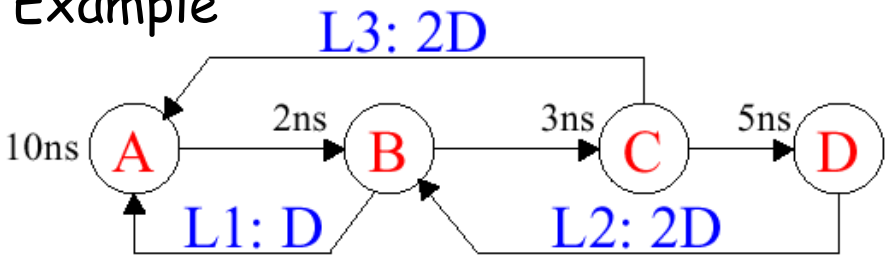
Iteration Bound – fundamental limit

- Critical loop: the loop with the maximum loop bound
- Iteration bound: the loop bound of the critical loop
- Not possible to achieve iteration period lower than iteration bound even with infinite processing power
- Definition

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

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

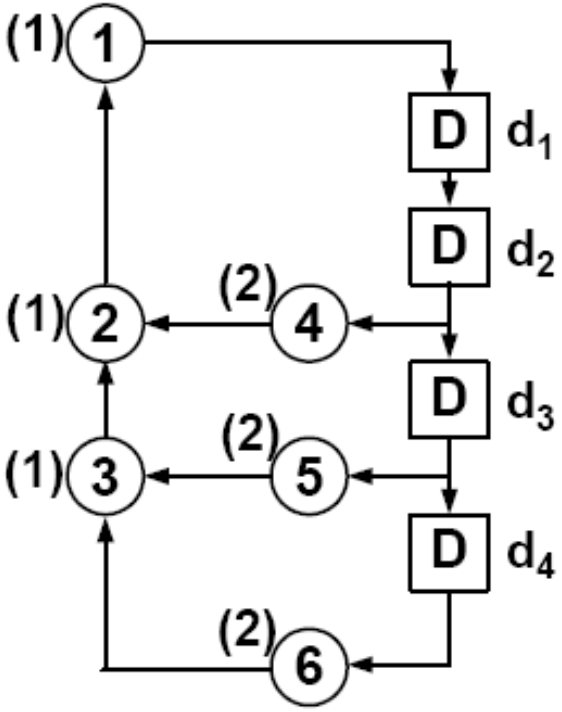


$$T_{L1} = (10+2)/1 = 12ns$$
$$T_{L2} = (2+3+5)/2 = 5ns$$
$$T_{L3} = (10+2+3)/2 = 7.5ns$$
$$T_\infty = \max_{l \in L}\{12,\ 5,\ 7.5\}$$

Algorithms for Computing Iteration Bound

- Longest Path Matrix (LPM) Algorithm
- Minimum Cycle Mean (MCM) Algorithm

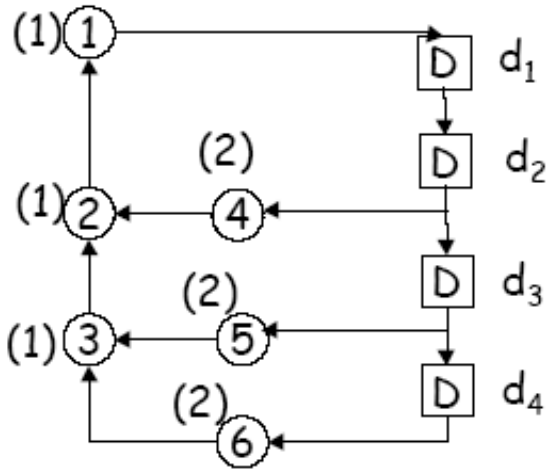
LPM Algorithm (1/2)



- d: number of delays, d_i: ith delay element
- Construct matrices $L^{(m)}$, $1 \le m \le d$, where each entry $\ell_{i,j}^{(m)}$ is the longest path from d_i to d_j with m-1 delays
- $\ell_{i,j}^{(m)} = -1$ if no such path
- $L^{(m+1)}$ can be obtained from $L^{(1)}$ and $L^{(m)}$ recursively by, if there is k such that $\ell_{i,j}^{(m+1)} = \ell_{i,k}^{(1)} + \ell_{k,j}^{(m)}$, otherwise $\ell_{i,j}^{(m)} = -1$
- Note that the diagonal element represents the longest computation time of all loops with m delays contains d_i
- Then the iteration bound can be calculated by $T_\infty = \max\{\ell_{i,i}^{(m)} / m\}$, for $1 \le i,m \le d$

LPM Algorithm (2/2)

- Example :



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

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

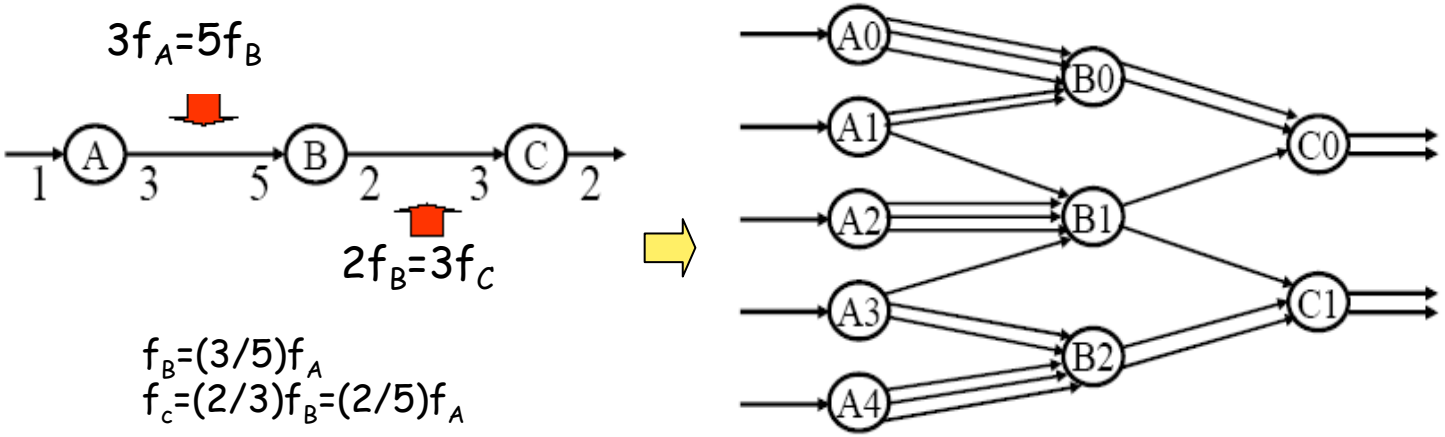
$$L^{(3)} = \begin{bmatrix} 5 & 4 & -1 & 0 \\ 8 & 5 & 4 & -1 \\ 9 & 5 & 5 & -1 \\ 9 & -1 & 5 & -1 \end{bmatrix}$$

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

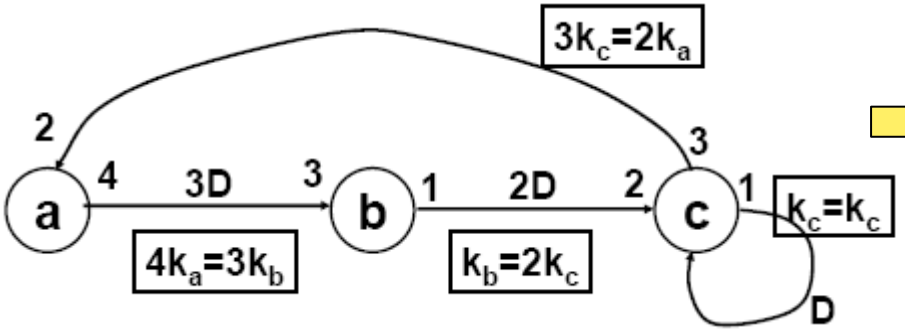
$T_\infty = \max\{4/2, 4/2, 5/3, 5/3, 5/3, 8/4, 8/4, 5/4, 5/4\} = 2.$

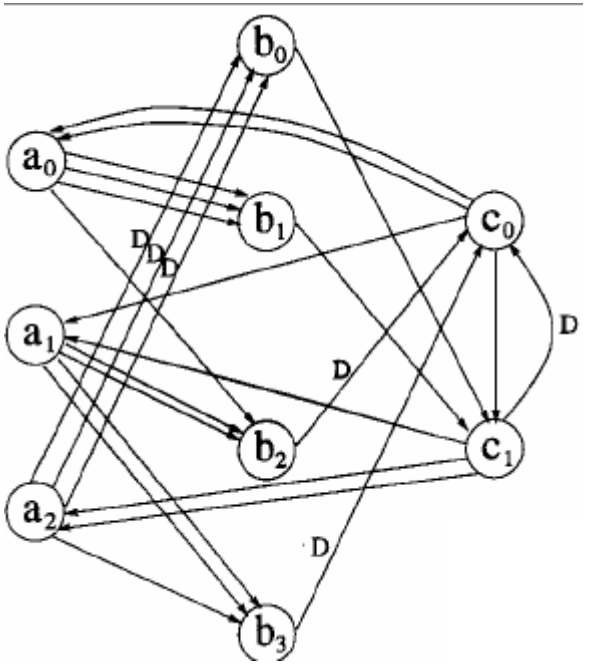
T_∞ of Multirate DFGs (MRDFGs)

- Construct the equivalent single-rate DFG (SRDFG)
- Compute the iteration bound of the equivalent SRDFG (unrolling or unfolding)
- The iteration bound of the MRDFG is the same as the iteration bound of the equivalent SRDFG.



Transformation of MRDFG





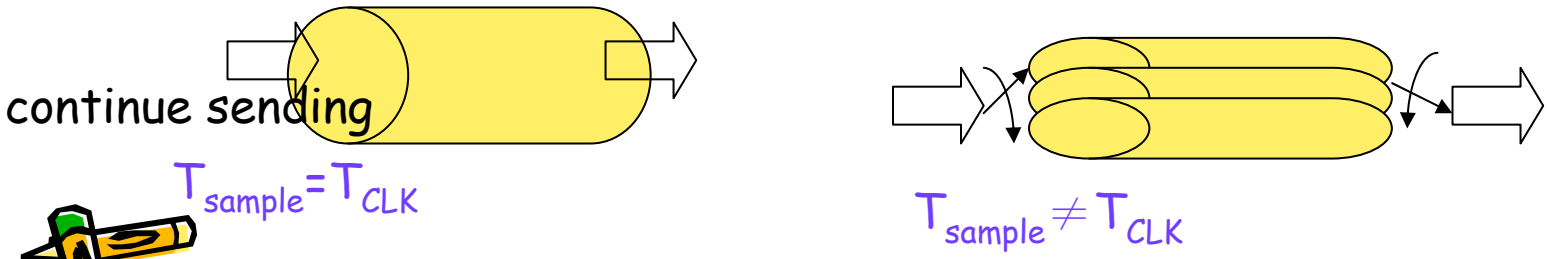
k_x: the number of nodes related to "x"

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

Note: same number of delay elements

Pipelining & Parallel Processing

- Pipelining processing
  - By using pipelining latches to reduce critical path
  - Either to increase the clock speed or sample speed, or to reduce the power consumption at same speed
- Parallel processing
  - By using replicating hardware to process multiple samples in parallel in a clock period
  - The effective sampling speed is increased by the level of parallelism.
  - Can be used to the reduction of power consumption



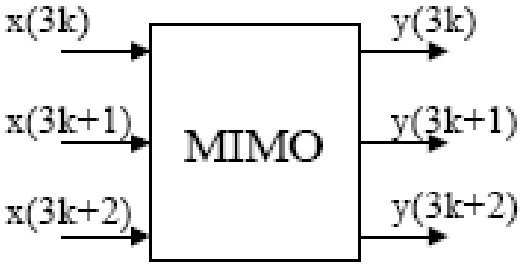
Parallel Processing

- Parallel processing and pipelining processing are duals each other
- Parallel processing is just the block processing.
- The number of inputs processed in a clock cycle is referred to as the block size.

delay

$$y(n) = ax(n) + bx(n-1) + cx(n-2)$$

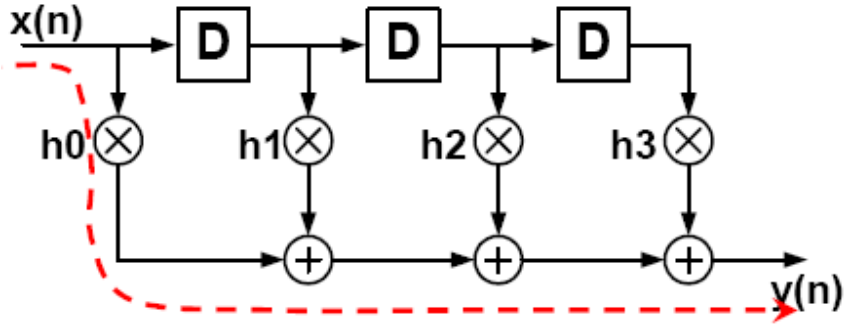
Block delay



$$\begin{cases} y(3k) &= ax(3k) + bx(3k-1) + cx(3k-2) \\ y(3k+1) &= ax(3k+1) + bx(3k) + cx(3k-1) \\ y(3k+2) &= ax(3k+2) + bx(3k+1) + cx(3k) \end{cases}$$

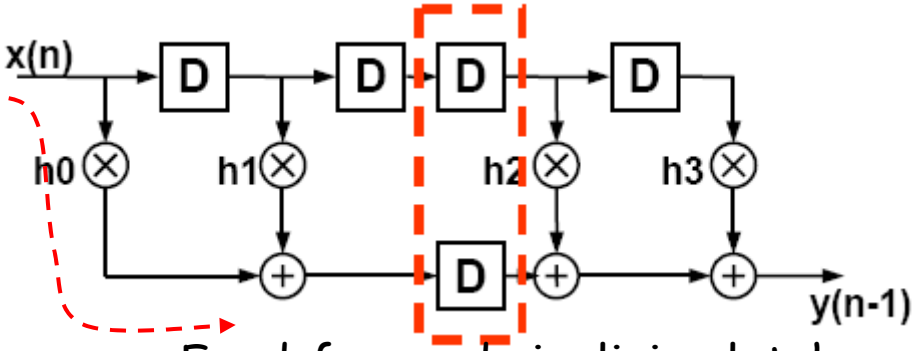
Pipelining Latches

Direct Form 4-tap FIR



1. $T_{Critical}=T_M+(N-1)\,T_A$, where N is the number of taps
2. Clock speed limited by critical path

2-level pipelined architecture



Feed-forward pipelining latches

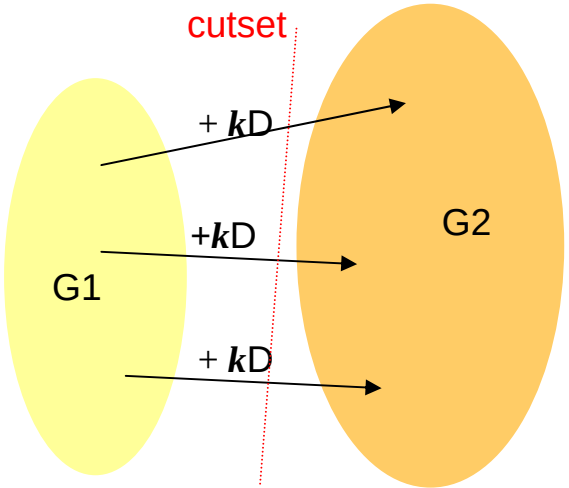
1. $T_{Critical}=T_M+T_A$
2. Pipelining will not affect functionality but introduce latency

Cutset Pipelining (1/2)

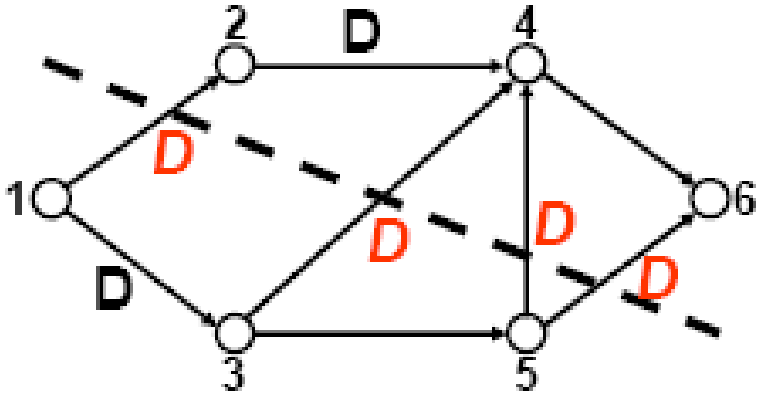
- Cutset: a set of edges of a graph such that if these edges are removed from the graph, the graph becomes disjoint
- Feed-forward cutset: the data move in the forward direction on all edges of the cutset
- In a synchronous DFG (SDFG),
  - communications occurs on the edges only
  - data flow through the functional units directly and the operations are synchronized with the arranged data arrival times (via delay insertion)

Cutset Pipelining (2/2)

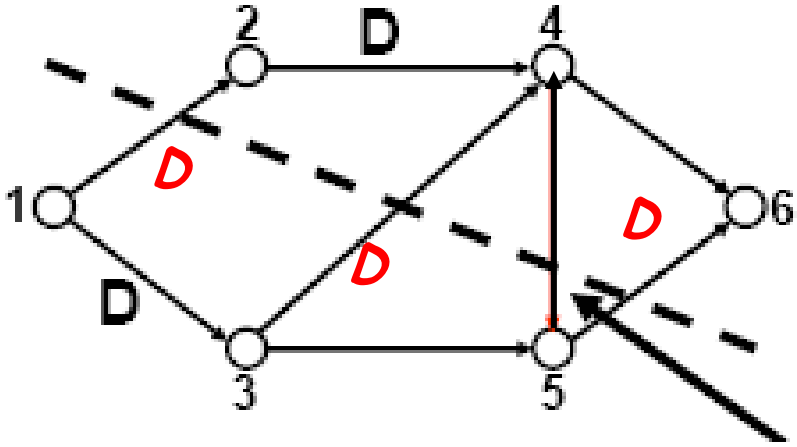
- Cutset pipelining preserves architecture's behavior
  - the data movement between the two disjoint sub-graphs only occurs on the feed-forward cutset, delaying or advancing the data movement along all edges on the cutset by the same amount of time do not change the architecture's behavior
- Two main drwaback
  - Increase in the number of latches
  - System latency



Feed-forward Cutset



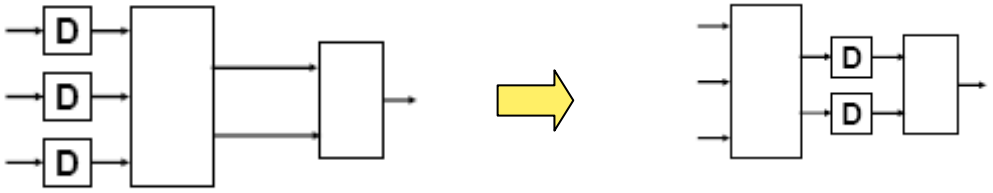
Must place delays on all edges in the cutset

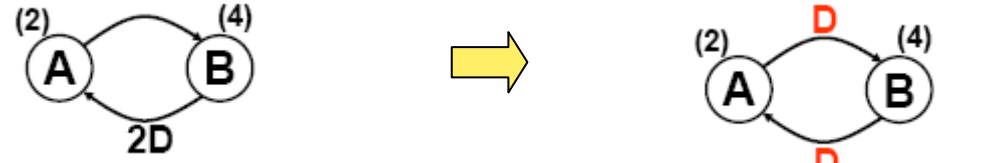


Not a valid pipelining

Retiming

- Retiming is to move around existing delays s.t.
  - Does not alter the latency of the system
  - Changes (Reduces) the critical path of the system
  - Reduces the number of register



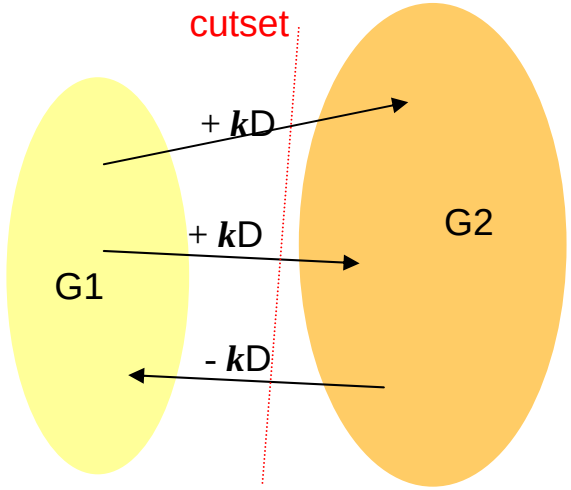


Critical path = 6
Loop bound = 6/2 = 3

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

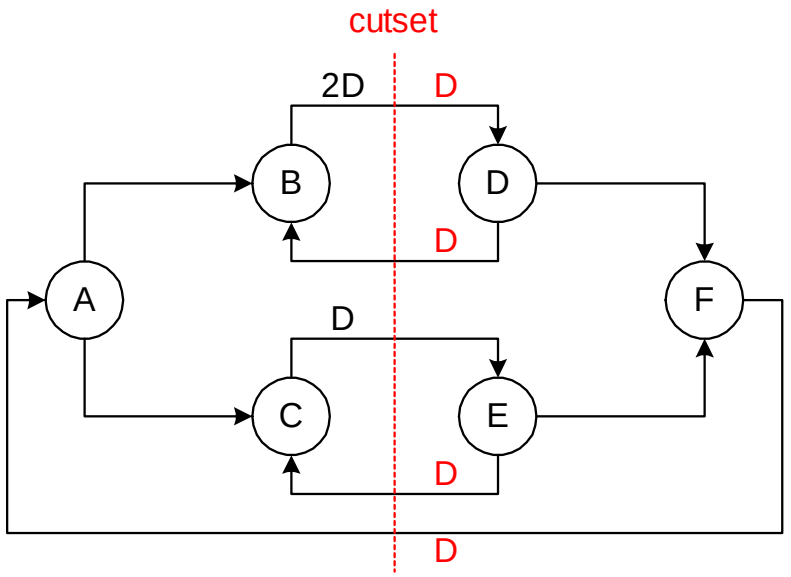
- Pipelining is equivalent to introducing many delays at the input followed by retiming

Example -- Cutset Retiming



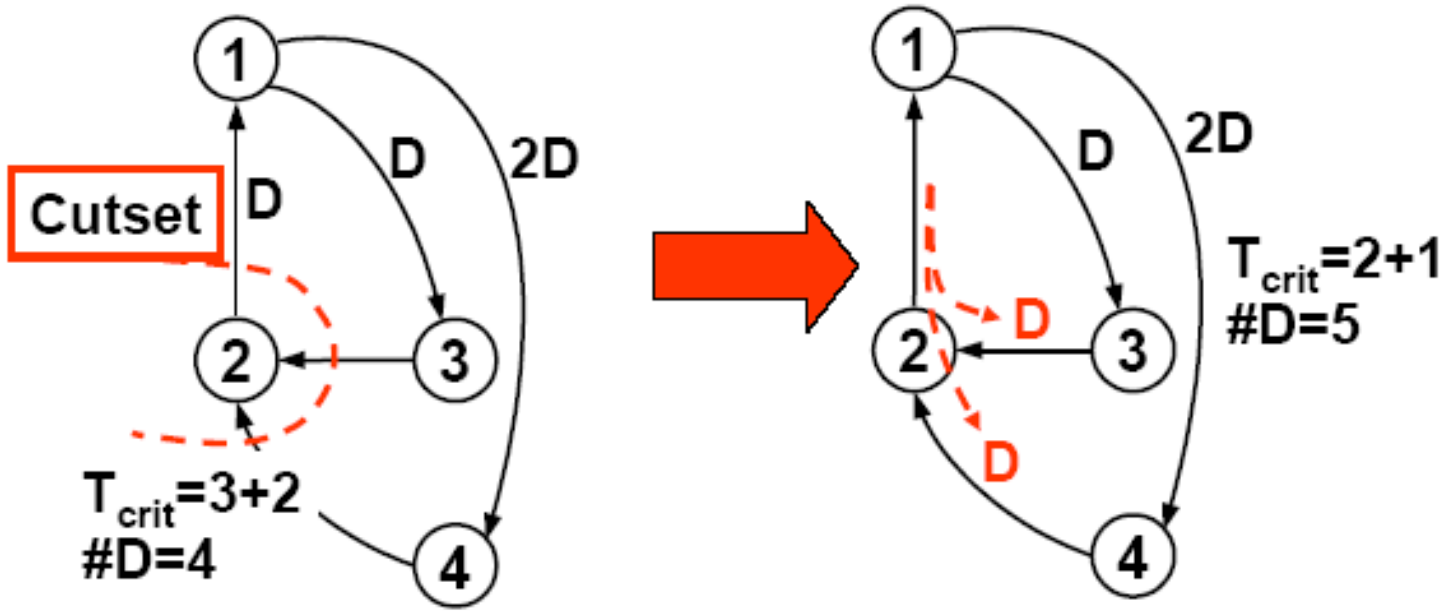
Add delays to edges going one way and remove from ones going the other

Example



Example – Node Retiming

To cutset around one specified node



Node Retiming

- The cutset can be selected such that one of the disjoint sub-graphs contains only one node
- Retiming value r(V): defined as the number of delays moved from each output edge of the node V to each of its input edges
- Feasibility constraints: for each directed edge UV of the retimed SDFG, the number of delay elements must be non-negative

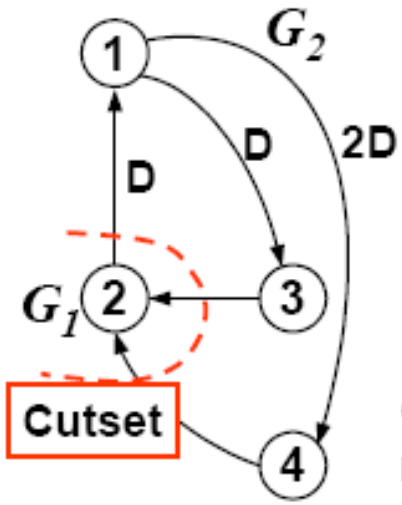
$$w_r(e) = w(e) + r(V) - r(U) \geq 0$$
$$r(U) - r(V) \leq w(e)$$

where $w_r(e)$ and $w(e)$ are the number of delay elements on the edge $\overrightarrow{UV}$ after and before retiming respectively

- For any SDFG, there is a retiming design space defined by the system of inequalities (each edge contributes an inequality)

- Solving systems of inequalities

Node Retiming with Formation



Original weights

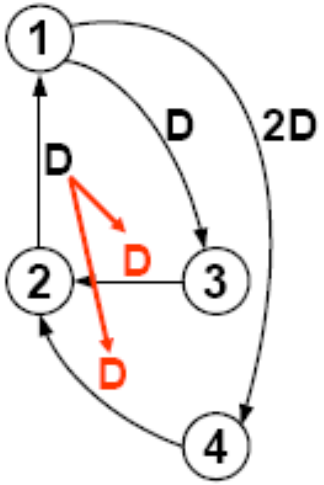
$$1 \rightarrow 3 = 1$$
$$1 \rightarrow 4 = 2$$
$$2 \rightarrow 1 = 1$$
$$3 \rightarrow 2 = 0$$
$$4 \rightarrow 2 = 0$$

Choose retiming values
r(1)=0, r(2)=1, r(3)=0 and r(4)=0,
G_2=0* and *G_1=1

$$\omega_r(e) = \omega(e) + r(V) - r(U)$$
Receive Send

$$1 \rightarrow 3 = 1+0-0 = 1$$
$$1 \rightarrow 4 = 2+0-0 = 2$$
$$2 \rightarrow 1 = 1+0-1 = 0$$
$$3 \rightarrow 2 = 0+1-0 = 1$$
$$4 \rightarrow 2 = 0+1-0 = 1$$

Result

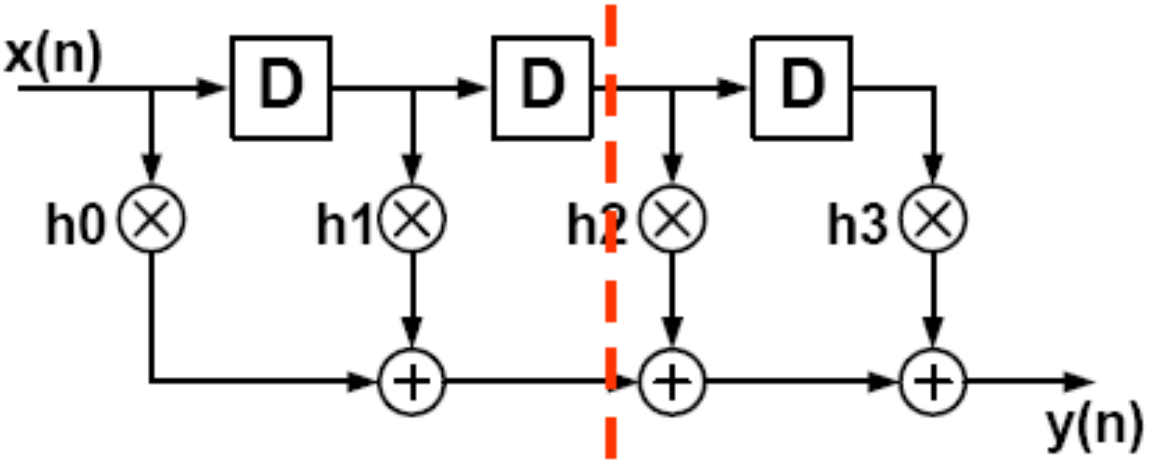


Properties of Retiming

- P.1: The weight of a retimed path $p = V_0, V_1, \ldots, V_k$ is given by $w_r(p) = w(p)+r(V_k)-r(V_0)$
- P.2: Retiming does not change the number of delays in a cycle (special case of P.1 with $V_k=V_0$)
- P.3: Retiming does not alter the iteration bound in a DFG as the number of delays in a cycle does not change
- P.4: Adding a constant value j to the retiming value of each node does not change the mapping from G to G_r

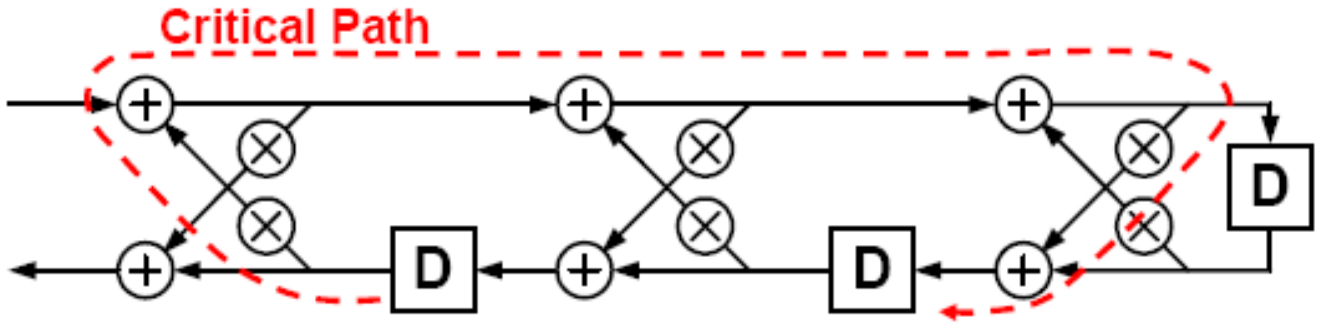
Pipelining – Feedforward Cutset

- A special case of cutset retiming by placing delays at feedforward cutset



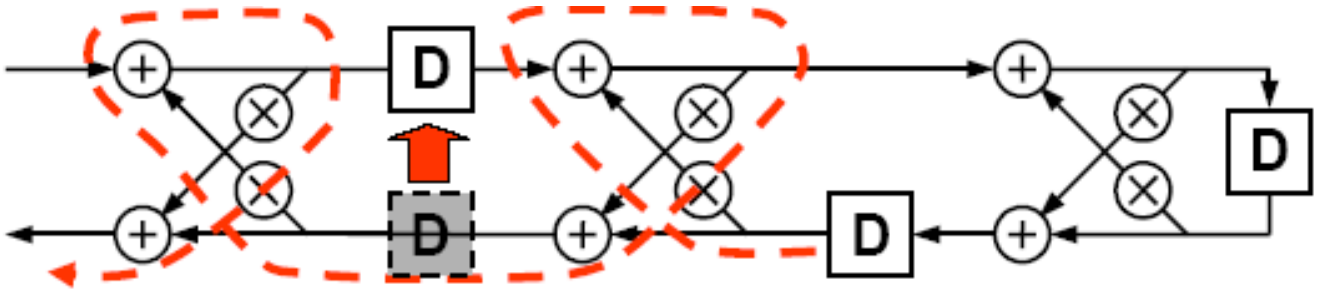
Remarks

3-stage Lattice Filter



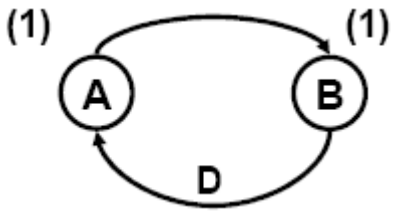
$T_{Critical}=2T_M+(N-1)\,T_A$, where N is the number of taps

Cutset retiming



$T_{Critical}=4T_M+4T_A$, a constant, which is independent of N

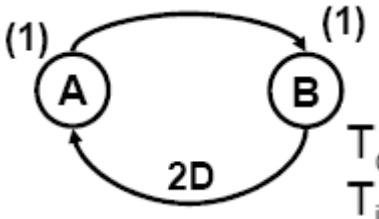
K Slow-down and Retiming



T_{clk}= 2t.u.
T_{iter}= 2t.u.

2-slow transformation

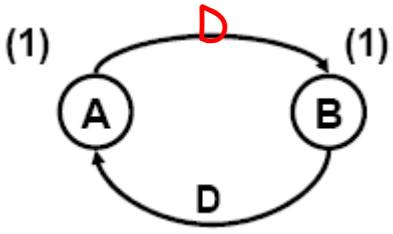
Replace each D by KD



T_{clk}= 2t.u.
T_{iter}=2×2t.u.
=4t.u.

1. K-1 null operation must be inserted
2. Hardware utilization is reduced (e.g. 50%)

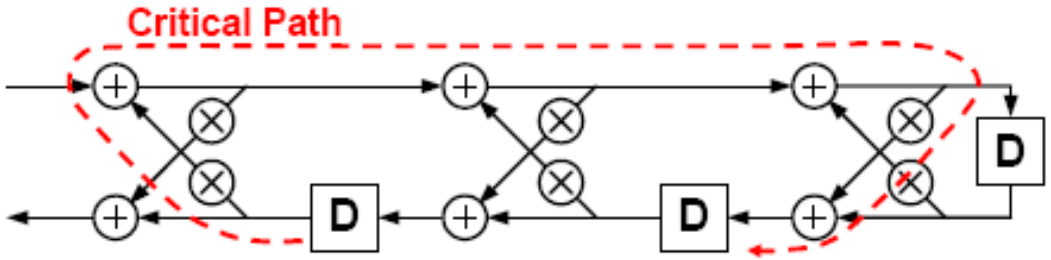
Retiming
2-slow version



Hardware can be fully utilized if two independent operations are available

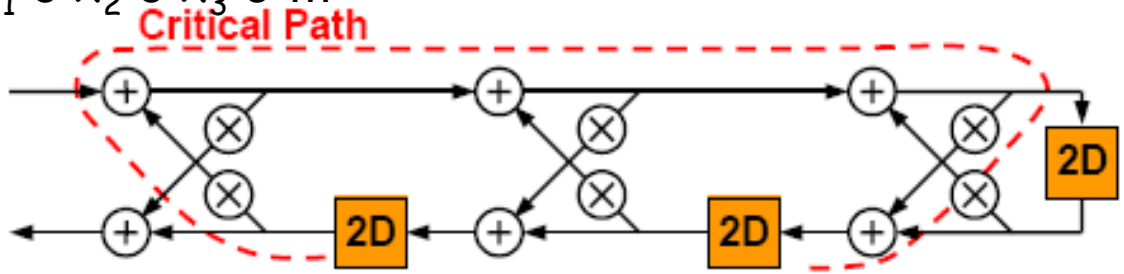
T_{clk}=1 t.u., T_{iter}=2 t.u.

Cutset Retiming 2-slow Lattice Filter

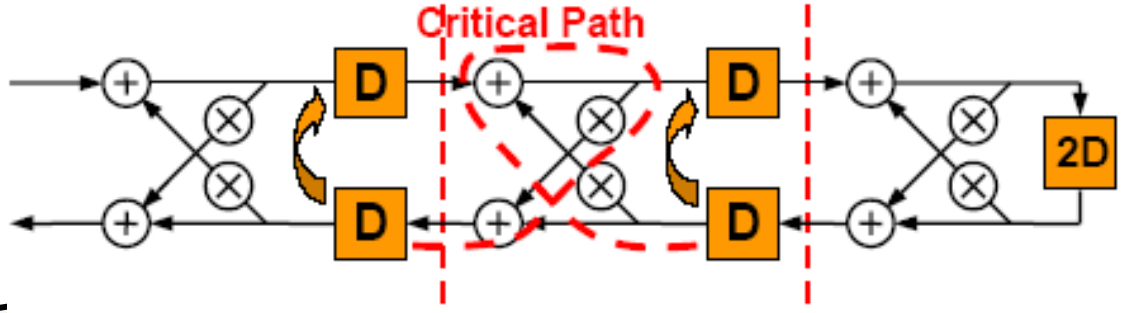


$T_{Critical}=2T_M+(N-1)\,T_A$, where N is tap number

$x_0\ 0\ x_1\ 0\ x_2\ 0\ x_3\ 0\ \ldots$



Inserting of "0" to preserve behavior !!



$T_{sample} = 2\ T_{critical}$

2-slow

Retiming for Min Clock Period (1/3)

- $\Phi(G)$: the minimum feasible clock period is defined as

$$\Phi(G) = \max\{t(p) : w(p) = 0\}$$

w(p) denotes the delays of the path
t(p) denotes the computation time of the path

- ***In the retiming design space (defined by the system of inequalities), find a retimed SDFG with the minimum*** $\Phi(G)$
- Definition:
  - ***W(U,V)***: the minimum number of registers on any path from node U to node V

$$W(U,V) = \min\{w(p) : U \xrightarrow{p} V\}$$

  - ***D(U,V)***: the maximum computation time among all paths from U to V with weight W(U,V)

$$D(U,V) = \max\{t(p) : U \xrightarrow{p} V \text{ and } w(p) = W(U,V)\}$$

Retiming for Min Clock Period (2/3)

Step 1: Compute W(U,V) and D(U,V)

- Let $M = t_{max}n$, where t_{max} is the maximum computation time of the nodes in G and n is the number of nodes in G
- Form a new graph G', which is the same as G except the edge weights are replaced by $w'(e) = Mw(e) - t(u)$ for all edges from U to V
- Solve the all-pair shortest path problem on G' (using ***Floyd-Warshall algorithm***). Let S'_{UV} be the shortest path from U to V
- If U = V,
  - W(U,V) = 0 and D(U,V) = t(U)

  else,
  - W(U,V) = ceiling (S'_{UV}/M) and $D(U,V) = MW(U,V) - S'_{UV} + t(V)$

Retiming for Min Clock Period (3/3)

Step 2: Given a clock period c, find a feasible solution such that $\Phi(G) \le c$

- The retimed design space is now defined by
  - feasibility constraints

$$r(U) - r(V) \le w(e)$$

r(U)-r(V) ≤ W(U,V)

  - critical path constraints

$$r(U) - r(V) \le W(U,V) - 1$$

for all nodes U and V in G, such that $D(U,V) > c$

i.e.

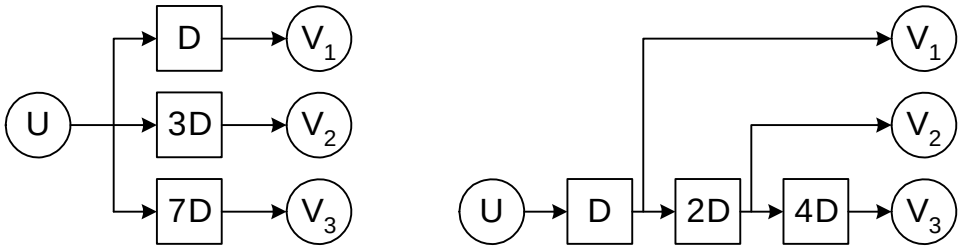
$$\text{if } D(U,V) > c,\ r(U) - r(V) \le W(U,V) - 1$$

$$\Leftrightarrow \text{if } W(U,V) + r(V) - r(U) < 1,\ D(U,V) \le c$$

- If the system of inequalities is not ϕ, try a smaller c

Retiming for Register Minimization

- ***Find a feasible solution with minimum number of registers in the design space constrained by feasibility and critical-path constraints (the refined retimed space with minimum clock period)***
- Multiple fan-out problem



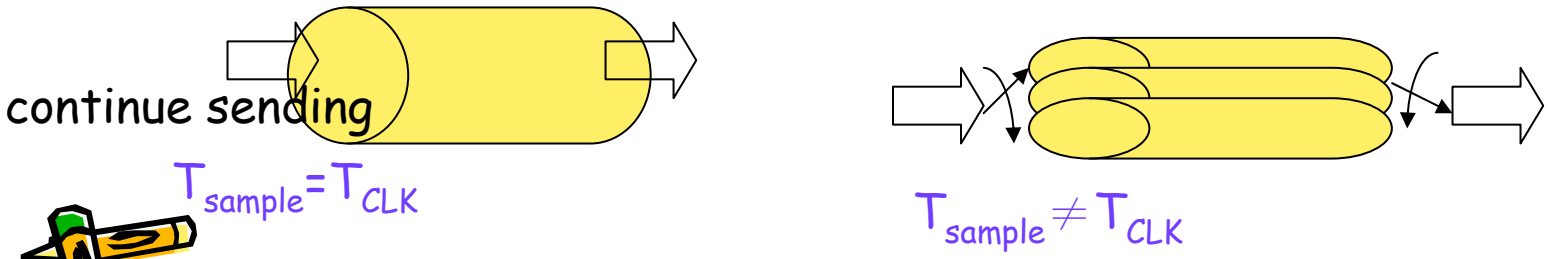
- Modeled as an ***integer linear programming*** (ILP) problem
  Minimize $\sum q(U)$, while satisfying
  - longest fanout constraints $w(e)+r(V)-r(U)\le q(U)$
  - feasibility constraints $r(U)-r(V)\le w(e)$
  - critical path constraints $r(U)-r(V)\le W(U,V)-1$

    for all nodes U and V in G, such that $D(U,V)>c$

Note solving the ILP model is NP-hard, and a gadget model can be found in the literature.

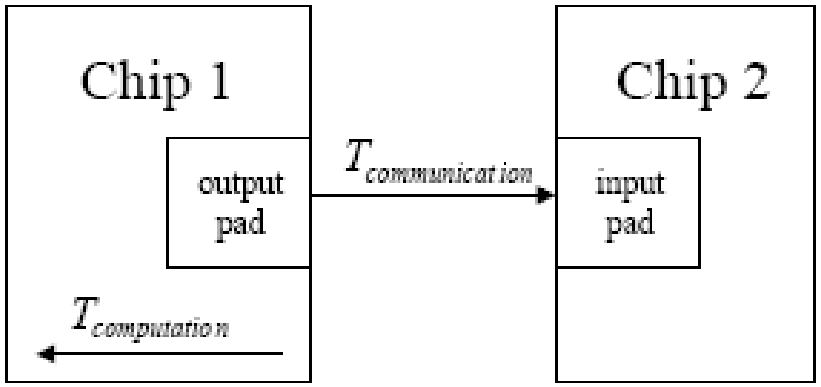
Pipelining & Parallel Processing

- Pipelining processing
  - By using pipelining latches to reduce critical path
  - Either to increase the clock speed or sample speed, or to reduce the power consumption at same speed
- Parallel processing
  - By using replicating hardware to process multiple samples in parallel in a clock period
  - The effective sampling speed is increased by the level of parallelism.
  - Can be used to the reduction of power consumption



Pipelining & Parallel Processing (2/1)

- Communication bounded
  - When the critical path is less than that the I/O bound, i.e. output-pad delay plus input-pad delay and the wire delay between two chips, then the system is communication bounded.
  - Pipelining can be used only to the extent such that the critical path is limited by the communication bound.
  - Once this is reached, pipelining can no longer increase the speed



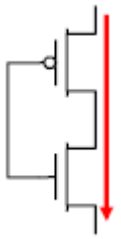
Pipelining & Parallel Processing (2/2)

- M-level pipelining processing
  - The critical path is reduced to 1/M
- L-level parallel processing
  - The critical path of the block processing system remains unchanged.
  - L samples are processed in 1 (not L) clock cycle, the iteration period is $T_{iteration}=T_{sample}=T_{clk}/L$, where $T_{clk} \geq T_{critical}$
- By combining M-level pipelining and L-level parallel processing, then
  - $T_{iteration}=T_{sample}=(1/LM)T_{clk}$

Power Consumption in CMOS

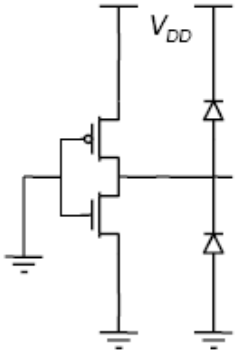
- $P_{total}=P_{dynamic}+P_{short\text{-}circuit}+P_{static}$

- Short circuit Power Consumption: current spikes



$V_{DD}\ I_{sc}$

- Static Power Consumption: due to leakage current

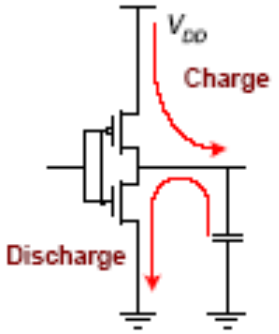


$I_{leakage}\ V_{DD}$

CMOS Power Consumption

- (Dynamic) power dissipation

$$P = C_{total} \cdot V_0^2 \cdot f$$



  - C_{total}: the total capacitance of the CMOS circuit
- Propagation delay

$$T_{\text{pd}} = \frac{C_{\text{charge}} \cdot V_0}{k(V_0 - V_t)^2}$$

  - C_{charge}: the capacitance to be charged or discharged in a single clock cycle (along the critical path)
  - V_0: the supply voltage; V_t: the threshold voltage
  - K: technology parameter

Reduce…

- Capacitances
  - Transistor/Gate C
  - Load C
  - Interconnects
  - External
- Activity
- Frequency
- Power supply
- Some comments
  - Off-chip connections have high capacitive load
  - System integration

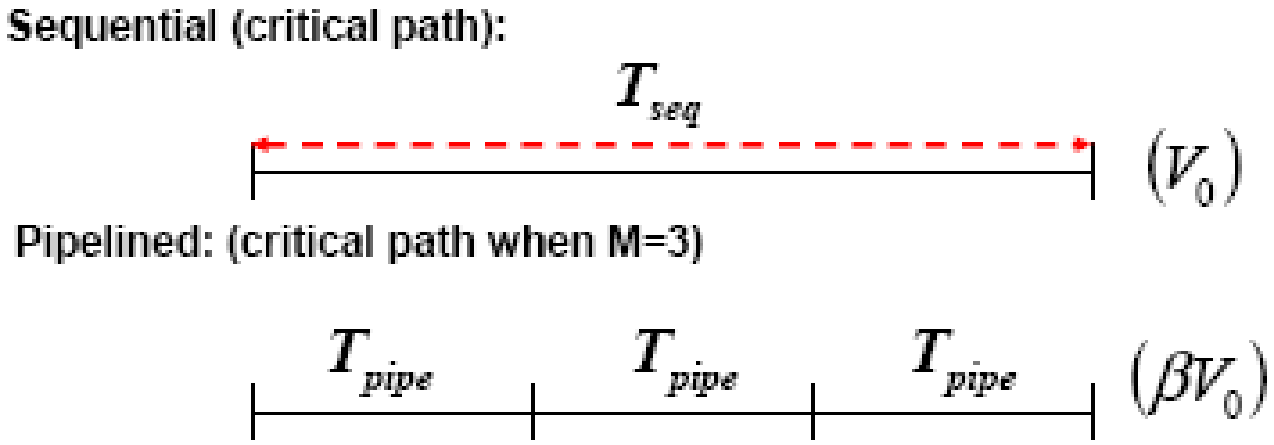
Pipelining for Low Power (1/2)

- For an *M-level* pipelined architecture, the critical path is reduced to *1/M* and the capacitance to be charged or discharged in a single cycle (i.e. C_{charge}) is also reduced to *1/M*
- If the same clock speed is maintained (i.e. $f = 1/T_{pd}$), only *1/M* of the non-pipelined capacitance is required to be charged or discharged, which suggests voltage reduction
- Suppose the voltage can be reduced to $\beta \cdot V_0$, the power consumption of a pipelined architecture becomes

$$P_{pipelined} = C_{total} \cdot (\beta \cdot V_0)^2 \cdot f$$
$$= \beta^2 \cdot P_{non-pipelined}$$

Remarks

Propagation delay of the original filter and the pipelined filter



Pipelining for Low Power (2/2)

- Solving
  - propagation delay of the original architecture

$$T_{\text{non-pipelined}} = \frac{C_{\text{charge}} \cdot V_0}{k(V_0 - V_t)^2}$$

  - propagation delay of the pipelined architecture

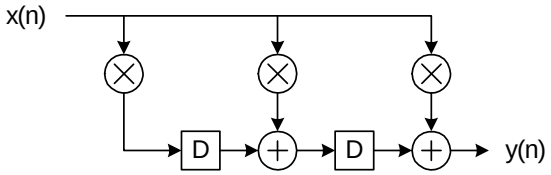
$$T_{\text{pipelined}} = \frac{\frac{C_{\text{charge}}}{M} \cdot \beta \cdot V_0}{k(\beta \cdot V_0 - V_t)^2}$$

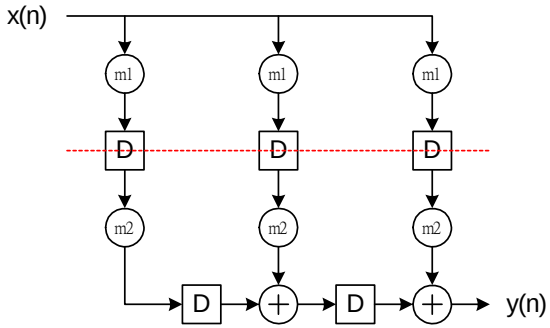
  - setting the above two equations equal, the following quadratic equation can be obtained to solve β

$$M(\beta \cdot V_0 - V_t)^2 = \beta \cdot (V_0 - V_t)^2$$

Example: Reduce Power by Pipelining

- Consider the following two FIR filters.





- Assume
  - the multiplication and the addition take ***10 u.t.*** and ***2 u.t.*** respectively
  - the capacitance of the multiplier is ***5*** times that of an adder
  - In the fine-grain pipelined filter, the multiplier is broken into ***m1***and ***m2***, with ***6*** and ***4*** u.t. computation delay and ***3*** and ***2*** times capacitance of an adder
  - supply voltage V_0 and threshold voltage V_t are ***5*** and ***0.6*** respectively
- Problems:
  - What is the supply voltage of the pipelined architecture if the clock periods are identical?
  - What is the relative power consumption?

Solution

- Calculate C_{charge}: let C_M, C_A, C_{m1} and C_{m2} be the capacitance of the multiplier, the adder, and the two fine-grain pipelining parts of the pipelined FIR filter respectively
  - Non-pipelined: $C_{charge} = C_M + C_A = 6C_A$
  - Pipelined: $C_{charge} = C_{m1} = C_{m2} + C_A = 3C_A$

  Notice the pipelining level $M = 2$ and the charging capacitance of the pipelined filter is one-half of that of the original one
- Solve

$$2 \cdot (\beta \cdot 5 - 0.6)^2 = \beta \cdot (5 - 0.6)^2$$

$$50 \cdot \beta^2 - 31.36 \cdot \beta + 0.72 = 0$$

$$\beta = 0.6033 \text{ or } 0.0239 \text{ (invalid, } \because \beta \cdot V_0 = 0.1195 < V_t)$$

- The supply voltage of the pipelined filter is

$$V_{pipelined} = \beta \cdot V_0 = 3.0165$$

- Power consumption ratio is $\beta^2 = 36.4\%$

Parallel Processing for Low Power (1/2)

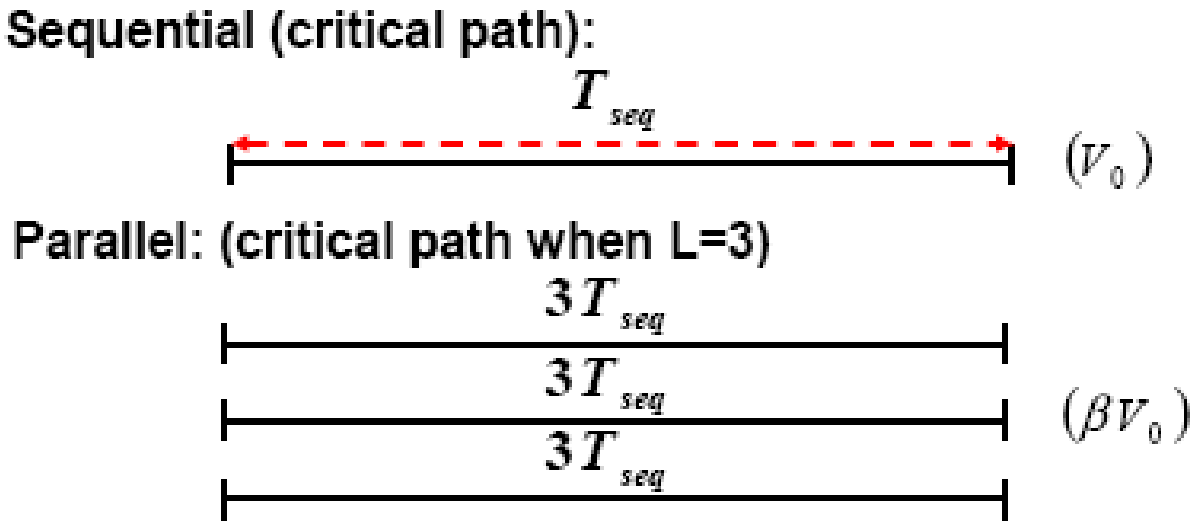
- For an *L-parallel* architecture, the charge capacitance remains the same, but the total capacitance (i.e. C_{total}) is increased *L* times
- The clock speed of the *L-parallel* architecture is reduced to *1/L* (i.e. $f = 1/L\,T_{pd}$) to maintain the same sample rate, which means the C_{charge} is charged or discharged *L* times longer.
- The supply voltage can be reduced to $\beta \cdot V_0$, since more time is allowed to charged or discharged the same capacitance. Thus the power consumption of the parallel architecture becomes

$$P_{parallel} = (L \cdot C_{total}) \cdot (\beta \cdot V_0)^2 \cdot \frac{f}{L}$$

$$= \beta^2 \cdot P_{non-parallel}$$

Remarks

Propagation delay of the original filter and the parallel filter



Parallel Processing for Low Power (2/2)

- Solving β
  - propagation delay of the original architecture

$$T_{\text{non-parallel}} = \frac{C_{\text{charge}} \cdot V_0}{k(V_0 - V_t)^2}$$

  - propagation delay of the parallel architecture

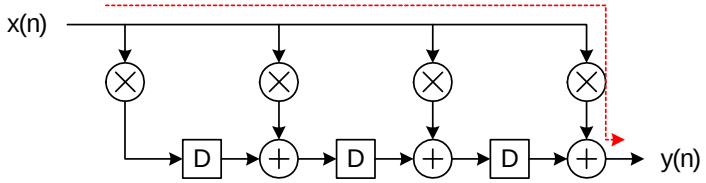
$$T_{\text{parallel}} = \frac{C_{\text{charge}} \cdot \beta \cdot V_0}{k(\beta \cdot V_0 - V_t)^2} = T_{\text{non-parallel}} \cdot L$$

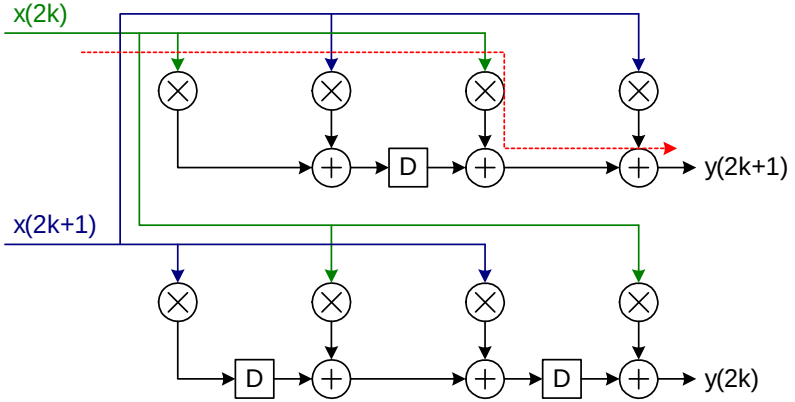
  - setting these two propagation delays equal, the following quadratic equation can be obtained to solve β

$$L(\beta \cdot V_0 - V_t)^2 = \beta \cdot (V_0 - V_t)^2$$

Example: Reduce Power by Parallel

- Consider the following two FIR filters, with critical paths denoted in dash lines respectively





- Assume
  - the multiplication and the addition take ***8 u.t.*** and ***1 u.t.*** respectively
  - the capacitance of the multiplier is ***8*** times that of an adder
  - both architectures operate at the sample period of ***9 u.t.***
  - supply voltage V_0 and threshold voltage V_t are ***3.3*** and ***0.45*** respectively
- What is the supply voltage of the parallel architecture?
- What is the relative power consumption?

Solution

- Calculate C_{charge}: let C_M and C_A be the capacitance of the multiplier and the adder respectively
  - Non-parallel: $C_{charge} = C_M + C_A = 9C_A$
  - Parallel: $C_M + 2C_A = 10C_A$

  Notice the charging capacitance of the two architectures are not equal, and we cannot use the equation directly
- Solve β

$$T_{non-parallel} = \frac{9C_A \cdot V_0}{k(V_0 - V_t)^2} \text{ and } T_{parallel} = \frac{10C_A \cdot \beta \cdot V_0}{k(\beta \cdot V_0 - V_t)^2}$$

$$\because T_{parallel} = 2 \cdot T_{non-parallel}$$

$$5 \cdot \beta \cdot (V_0 - V_t)^2 = 9 \cdot (\beta \cdot V_0 - V_t)^2$$

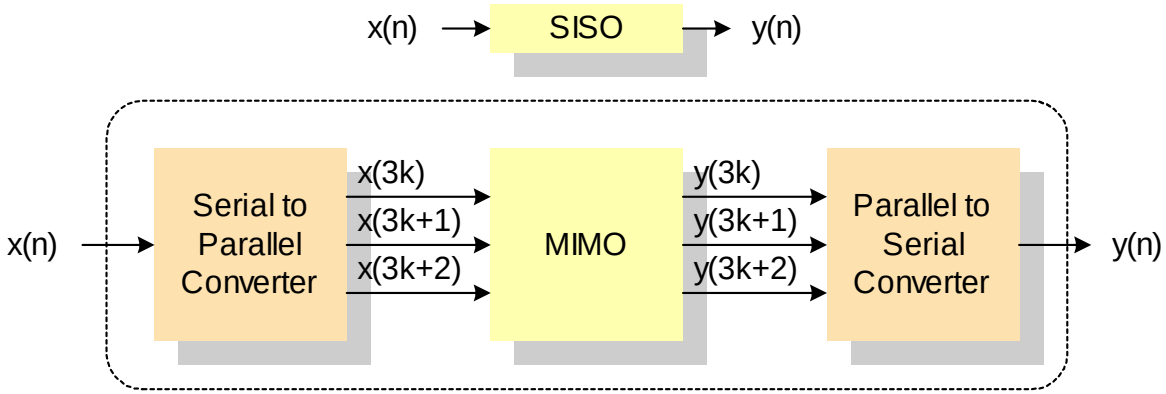
$$98.01 \cdot \beta^2 - 67.3425 \cdot \beta + 1.8225 = 0$$

$$\beta = 0.6589 \text{ or } 0.0282 \text{ (invalid, } \because \beta \cdot V_0 = 0.09306 < V_t)$$

- The supply voltage of the parallel filter is $V_{pipelined} = \beta \cdot V_0 = 2.17437$
- Power consumption ratio is $\beta^2 = 43.41\%$

Parallel Processing

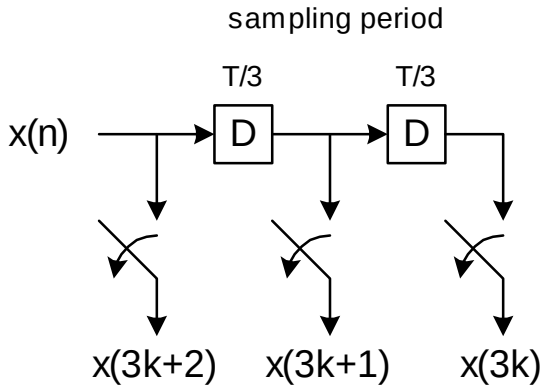
- Block processing
  - the number of inputs processed in a clock cycle is referred to as the ***block size***



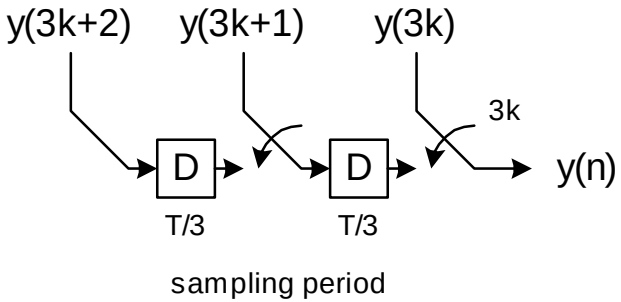
  - at the ***k-th*** clock cycle, three inputs x(3k), x(3k+1), and x(3k+2) are processed simultaneously to generate y(3k), y(3k+1), and y(3k+2)

I/O Conversion

- Serial to parallel converter



- Parallel to serial converter



Parallel + Pipelining

- I

$$T_{\text{pipelined}} = \frac{\frac{C_{\text{charge}}}{M} \cdot \beta \cdot V_0}{k(\beta \cdot V_0 - V_t)^2} = T_{\text{non-pipelined}}$$

- II

$$T_{\text{parallel}} = \frac{C_{\text{charge}} \cdot \beta \cdot V_0}{k(\beta \cdot V_0 - V_t)^2} = T_{\text{non-parallel}} \cdot L$$

- III

$$T_{\text{L-parallel+M-pipeline}} = T_{\text{M-pipeline}} \cdot L$$

$$LM(\beta \cdot V_0 - V_t)^2 = \beta \cdot (V_0 - V_t)^2$$