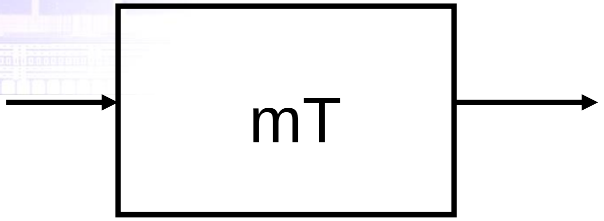
VLSI Architecture Design ABC

Shao-Yi Chien

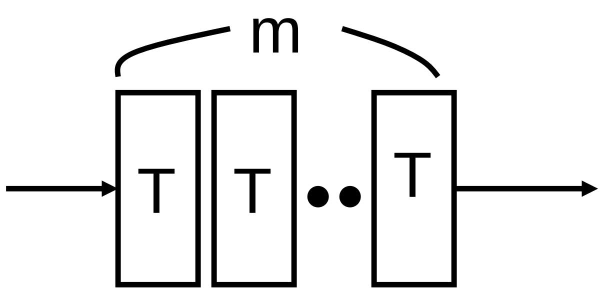
Outline

- Pipeline
- Parallel
- Folding
- Unfolding
- Important concepts:
  - scheduling/resource allocation/design space exploration

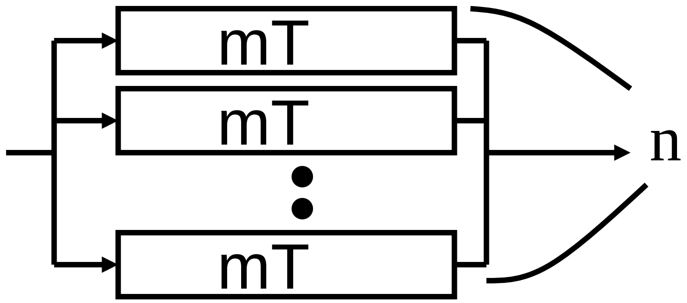
An Example: Car Production Line



Cost: 1
Throughput: 1/mT
Latency: mT



Cost: >1
Throughput: 1/T
Latency: >mT



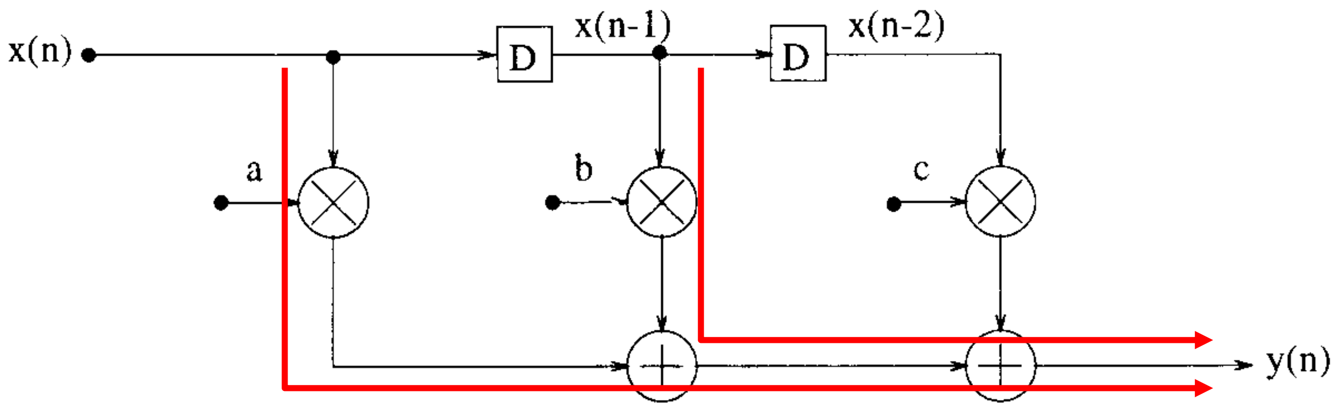
Cost: >n
Throughput: (1/mT)*n
Latency: >mT

Throughput: how many cars produced in one hour
Latency: how long will it take to produce one car

Pipelining of Digital Filters (1/3)

- FIR filter

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

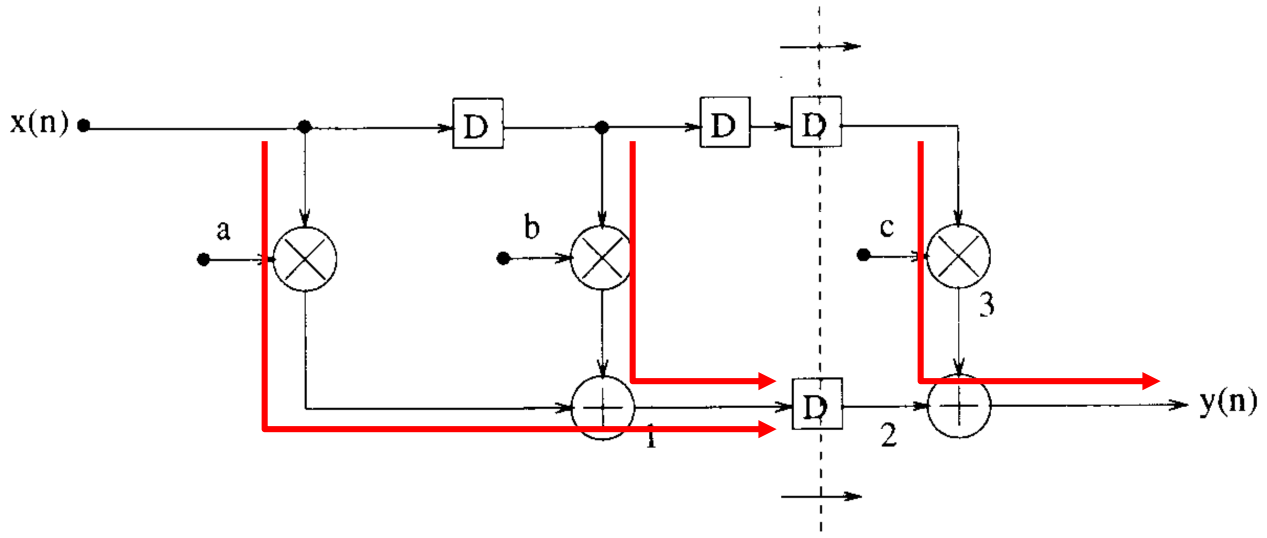


  - Critical path: T_M+2T_A

$$T_{sample} \geq T_M + 2T_A \qquad f_{sample} \leq \frac{1}{T_M + 2T_A}$$

Pipelining of Digital Filters (2/3)

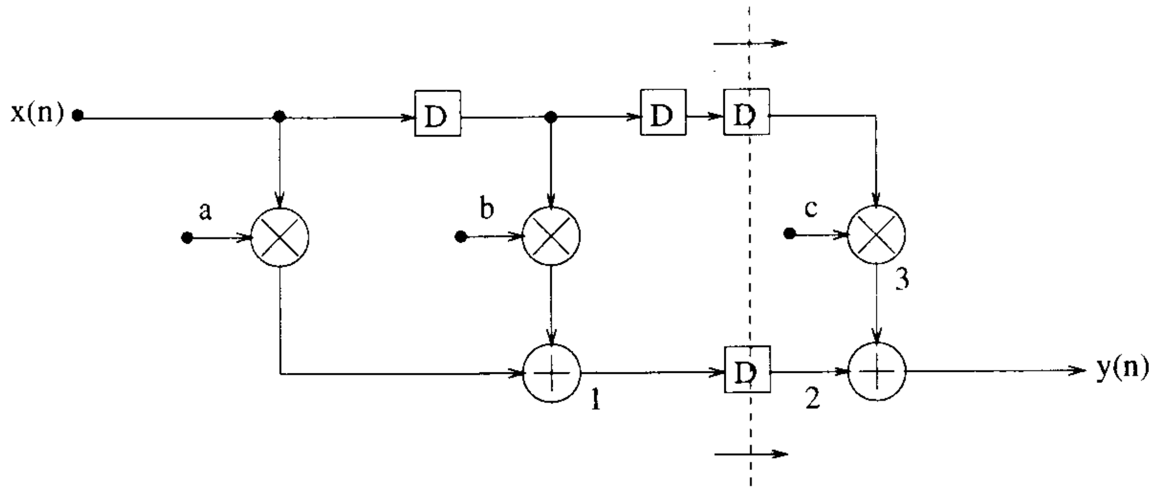
- Pipelined FIR filter



$$T_{sample} \geq T_M + T_A \qquad f_{sample} \leq \frac{1}{T_M + T_A}$$

Pipelining of Digital Filters (3/3)

- Schedule



Clock	Input	Node 1	Node 2	Node 3	Output
0	$x(0)$	$ax(0)+bx(-1)$	$-$	$-$	$-$
1	$x(1)$	$ax(1)+bx(0)$	$ax(0)+bx(-1)$	$cx(-2)$	$y(0)$
2	$x(2)$	$ax(2)+bx(1)$	$ax(1)+bx(0)$	$cx(-1)$	$y(1)$
3	$x(3)$	$ax(3)+bx(2)$	$ax(2)+bx(1)$	$cx(0)$	$y(2)$

Pipeline

- Can reduce the critical path to increase the working frequency and sample rate
  - $T_M+2T_A \rightarrow T_M+T_A$

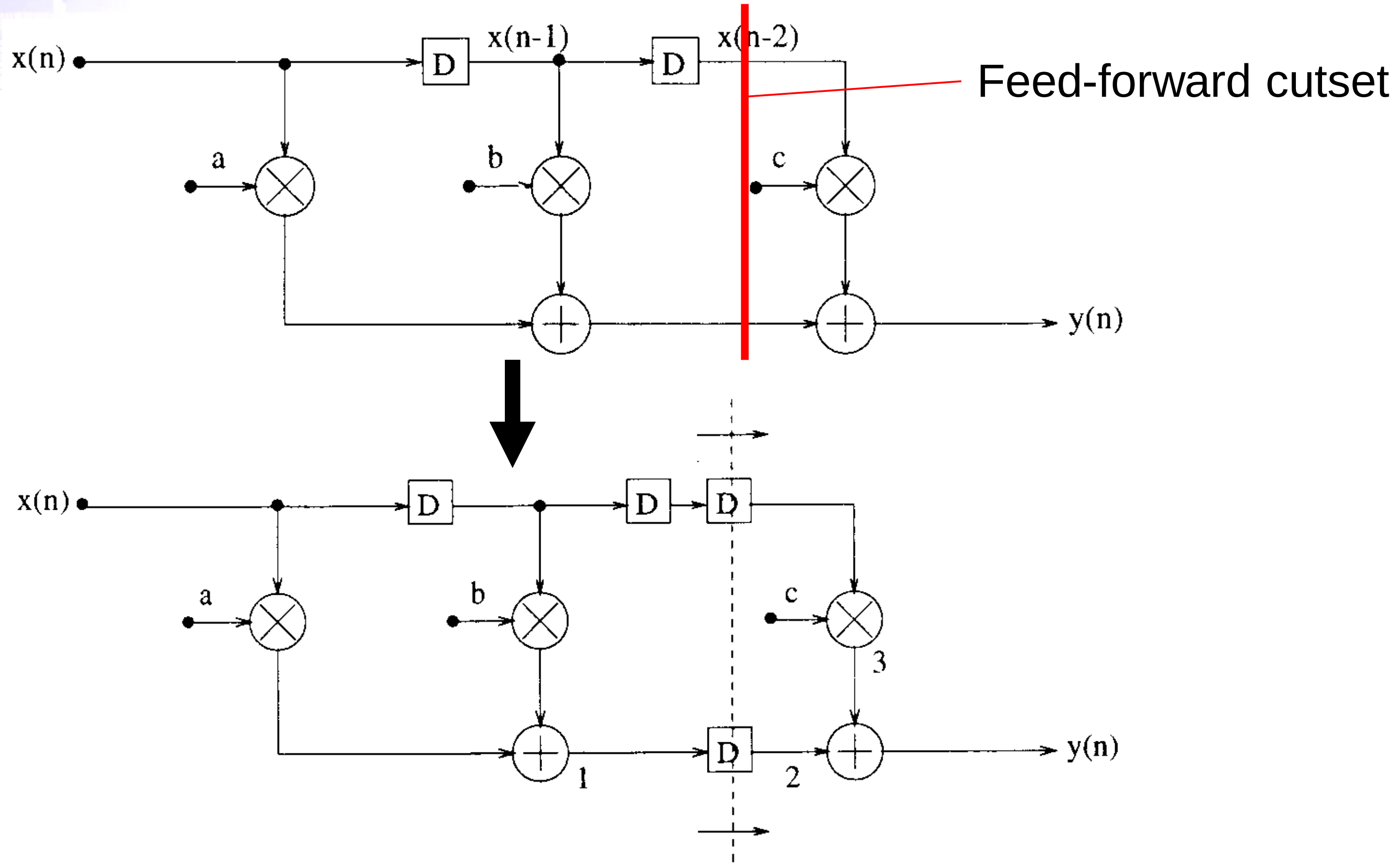
Drawbacks of Pipelining

- Increasing latency (in cycle)
  - For M-level pipelined system, the number of delay elements in any path from input to output is (M-1) greater than the origin one
- Increase the number of latches (registers)

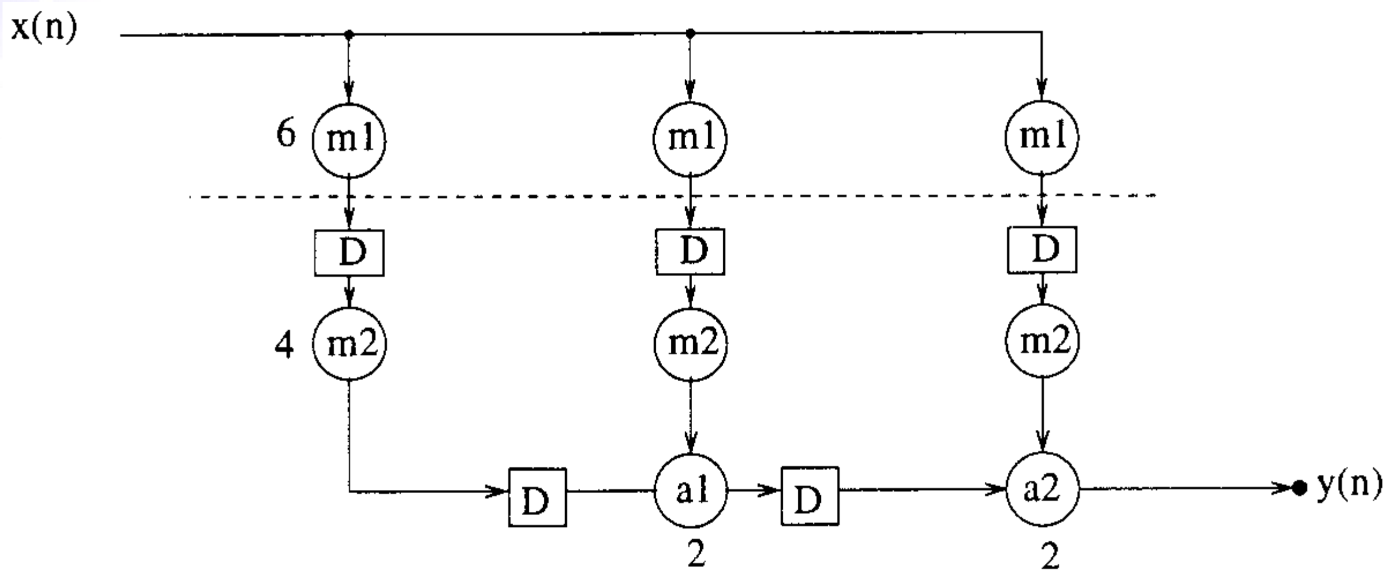
How to Do Pipelining?

- Put pipelining latches across any feed-forward cutset of the graph
- Cutset
  - A cutset is 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 the edges of the cutset

How to Do Pipelining?



Fine-Grain Pipelining



- Critical path (T_M=10, T_A=2)
  - $T_M+2T_A=14$
  - $T_M+T_A=12$
  - $T_{M1}=6$ or $T_{M2}+T_A=6$

Notes for Pipelining (1/2)

- Pipelining is a very simple design technique which can maintain the input output data configuration and sampling frequency
- $T_{clk}=T_{sample}$
- Supported in many EDA tools
- Still has some limitations
  - Pipeline bubbles
  - Has some problems for recursive system
  - Introduces large hardware cost for 2-D or 3-D data
  - Communication bound

Notes for Pipelining (2/2)

- Effective pipelining
  - Put pipelining registers on the critical path
  - Balance pipelining
    - 10➔(2+8): critical path=8
    - 10➔(5+5): critical path=5

Parallel of Digital Filters (1/5)

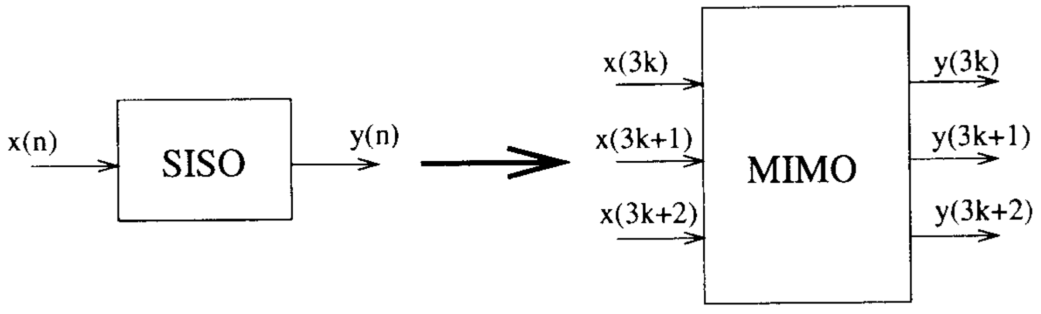
- Single-input single-output (SISO) system

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

- Multiple-input multiple-output (MIMO) system

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

3-Parallel System!

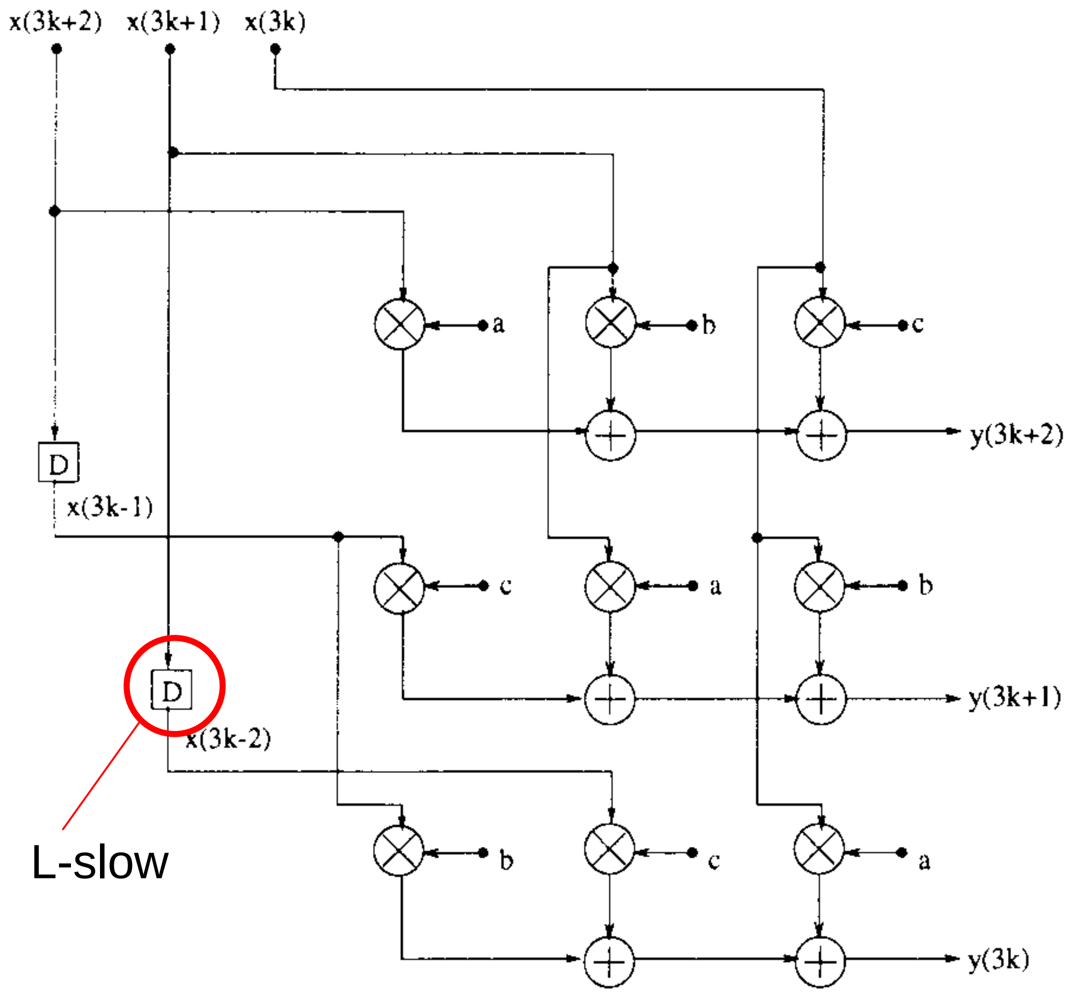


Sequential System 3-Parallel System

Parallel of Digital Filters (2/5)

- Parallel processing, block processing
- Block size (L): the number of data to be processed at the same time
- Block delay (L-slow)
  - A latch is equivalent to L clock cycles at the sample rate

Parallel of Digital Filters (3/5)



$$T_{clk} \geq T_M + 2T_A$$

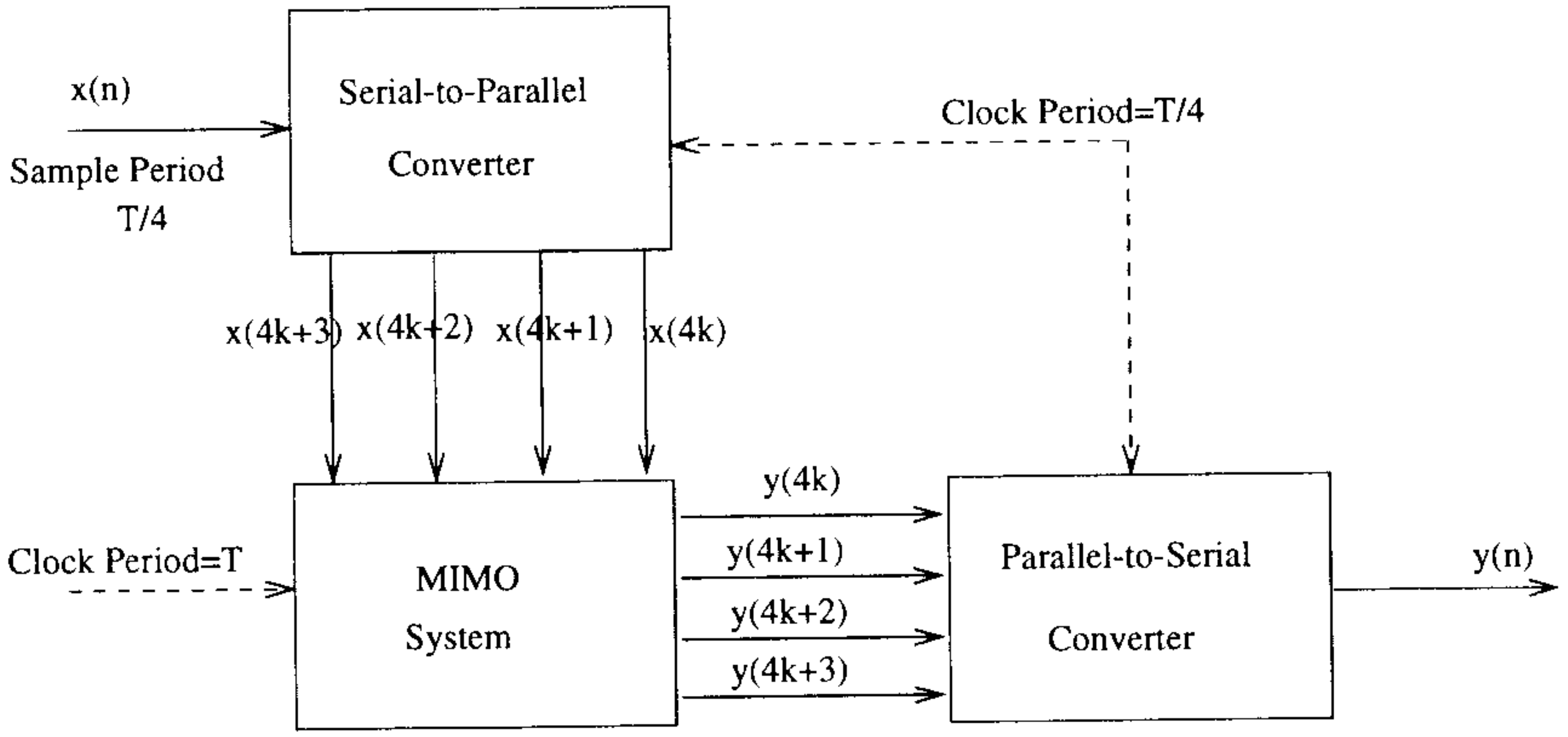
The critical path is the same

$$T_{iter} = T_{sample} = \frac{1}{L} T_{clk} \geq \frac{1}{3}(T_M + 2T_A)$$

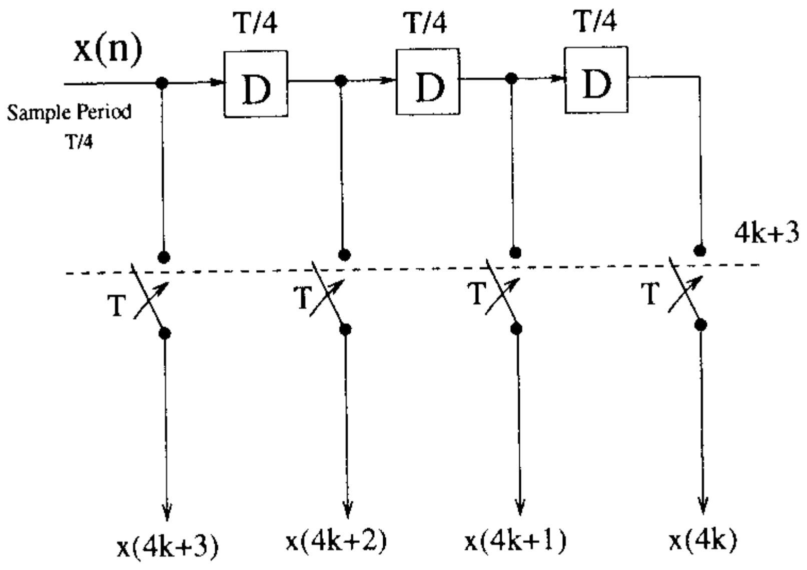
T_{clk} is not equal to T_{sample}

Parallel of Digital Filters (4/5)

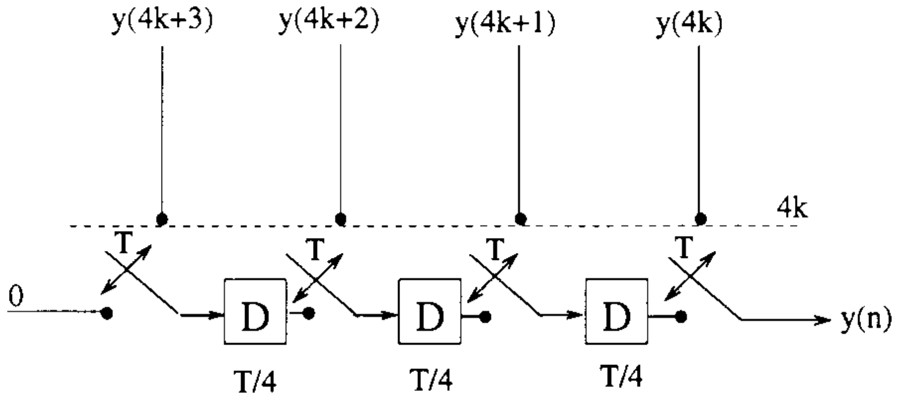
- Whole system



Parallel of Digital Filters (5/5)

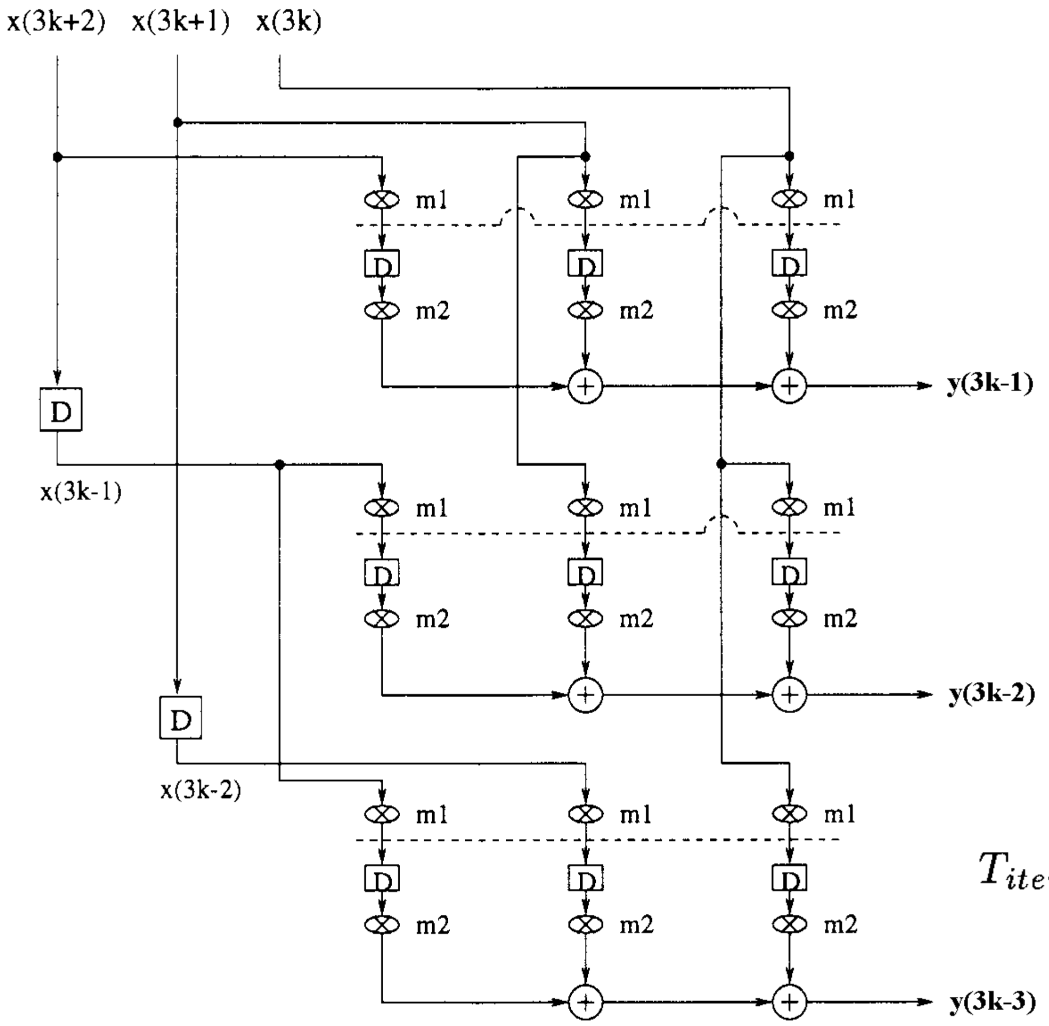


Serial-to-parallel converter



Parallel-to-serial converter

Pipelining-Parallel Architecture



$$T_{iter} = T_{sample} = \frac{1}{LM}T_{clk} = \frac{1}{6}(T_M + 2T_A)$$

Notes for Parallel Processing

- The input/output data access scheme should be carefully designed, it will cost a lot sometimes
- $T_{clk} > T_{sample}$, $f_{clk} < f_{sample}$
- Large hardware cost
- Combined with pipelining processing

Low Power Issues

- Pipelining and parallel processing are also beneficial for low power design
- Propagation delay

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

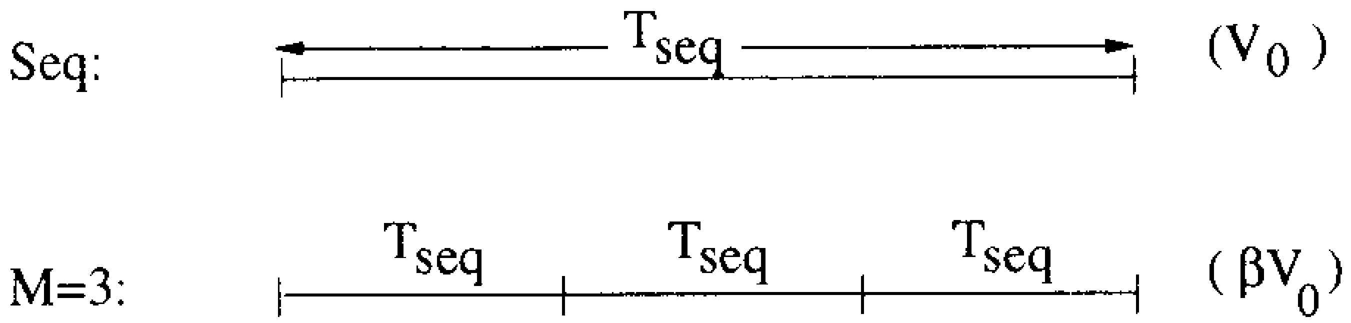
- Power consumption

$$P = C_{total} V_0^2 f \qquad f = \frac{1}{T_{seq}}$$

- Assume the sampling frequency is the same

Pipelining for Low Power (1/2)

- For M-level pipelining
  - Critical path is reduced to 1/M
  - The capacitance is also reduced to C_{charge}/M
  - The supply voltage can be reduced to βV_0, and the propagation delay remains unchanged



Pipelining for Low Power (2/2)

- Power consumption:

$$P_{pip} = C_{total}\beta^2 V_0^2 f = \beta^2 P_{seq}$$

- How about the parameter β ?

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

$$\|$$

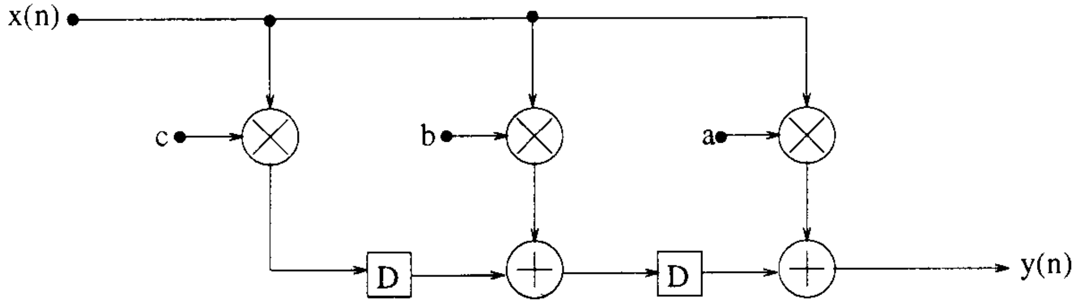
$$T_{pip} = \frac{\frac{C_{charge}}{M}\beta V_0}{k(\beta V_0 - V_t)^2}$$

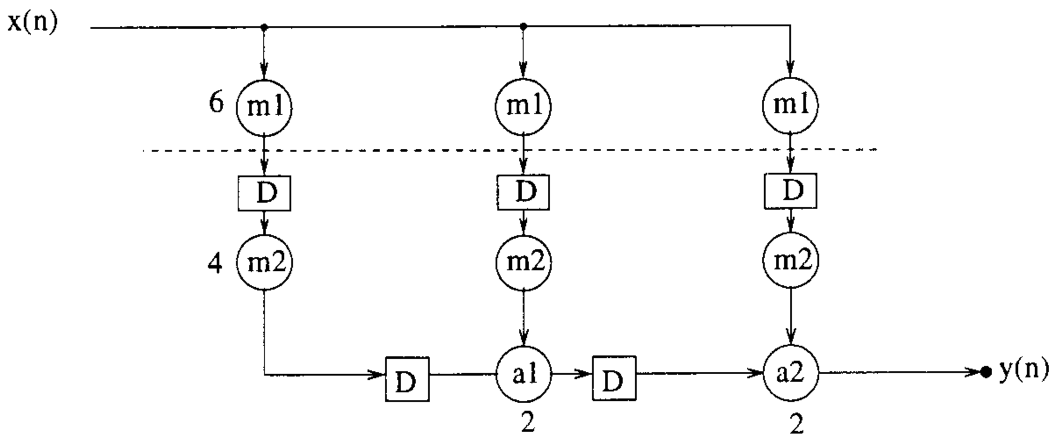
$$\longrightarrow \quad M(\beta V_0 - V_t)^2 = \beta(V_0 - V_t)^2$$

Solve this equation to get β

Example

- Parameters
  - T_M=10 u.t.
  - T_A=2 u.t.
  - T_{m1}=6 u.t.
  - T_{m2}=4 u.t.
  - C_M=5C_A
  - V_t=0.6V
  - Normal V_{cc}=5V
- (a) New supply voltage?
- (b) Power saving percentage?





Answer

(a)

Origin system: $C_{charge} = C_M + C_A = 6C_A$

Pipelined system: $C_{charge} = C_{m1} = C_{m2} + C_A = 3C_A$

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

$$\beta = 0.6033, \ or \ \boxed{\beta = 0.0239}$$

Invalid value, less than threshold voltage

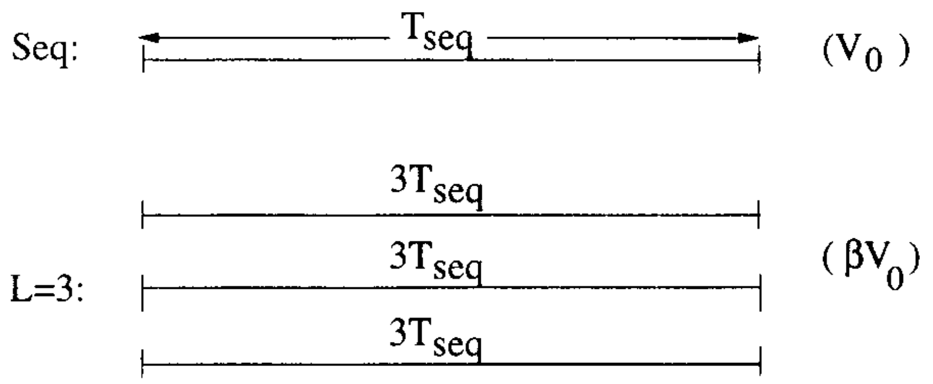
$$V_{pip} = \beta V_0 = 3.0165\ V.$$

(b)

$$Ratio = \beta^2 = 36.4\%.$$

Parallel Processing for Low Power (1/2)

- For L-parallel system
  - Clock period: $T_{seq} \rightarrow LT_{seq}$
  - C_{charge} remains unchanged
  - $C_{totol} \rightarrow LC_{total}$
  - Have more time to charge the capacitance, the supply voltage can be lower βV_0



Parallel Processing for Low Power (2/2)

- Power consumption

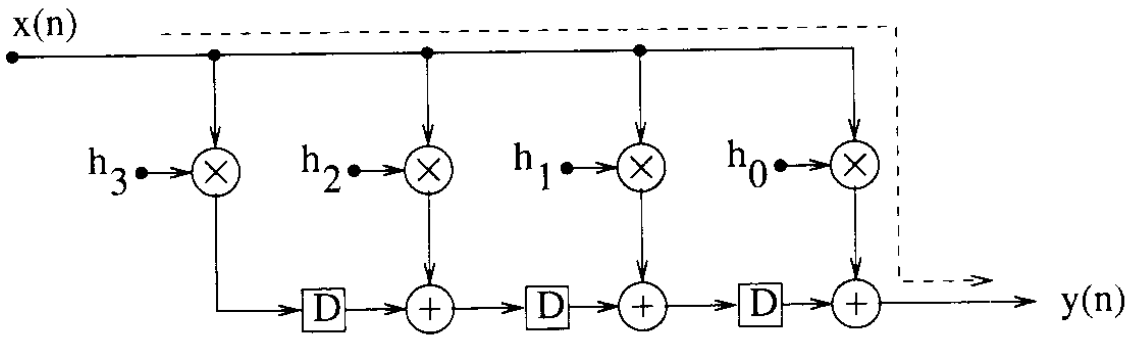
$$
\begin{aligned}
P_{par} &= (LC_{total})(\beta V_0)^2 \frac{f}{L} \\
&= \beta^2 C_{total} V_0^2 f \\
&= \beta^2 P_{seq}
\end{aligned}
$$

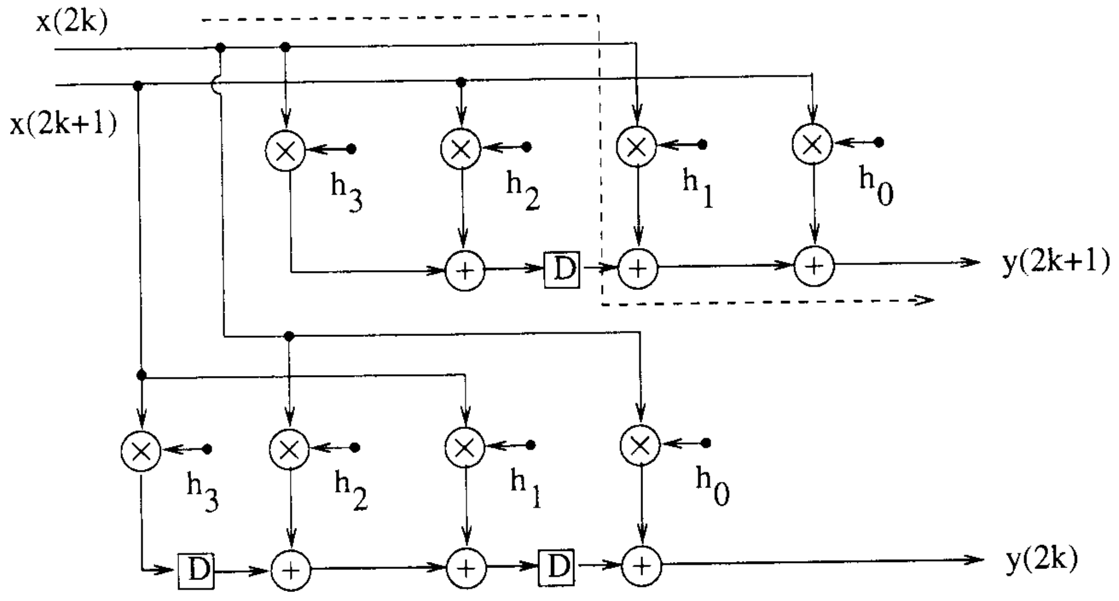
- To derive the parameter β

$$
\begin{aligned}
T_{seq} &= \frac{C_{charge} V_0}{k(V_0 - V_t)^2} \\
&\downarrow \\
LT_{seq} &= \frac{C_{charge} \beta V_0}{k(\beta V_0 - V_t)^2}
\end{aligned}
\longrightarrow L(\beta V_0 - V_t)^2 = \beta (V_0 - V_t)^2
$$

Example

- Parameters
  - T_M=8 u.t.
  - T_A=1 u.t.
  - T_{sample}=9 u.t.
  - C_M=8C_A
  - V_t=0.45V
  - Normal V_{cc}=3.3V
- (a) New supply voltage?
- (b) Power saving percentage?





Answer

(a)

Origin: $C_{charge} = C_M + C_A = 9C_A$

2-parallel system: $C_{charge} = C_M + 2C_A = 10C_A$

$$T_{seq} = \frac{9C_A V_0}{k(V_0 - V_t)^2},$$

$$T_{par} = \frac{10C_A \beta V_0}{k(\beta V_0 - V_t)^2}.$$

$$T_{par} = 2T_{sample} = 2T_{seq}$$

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

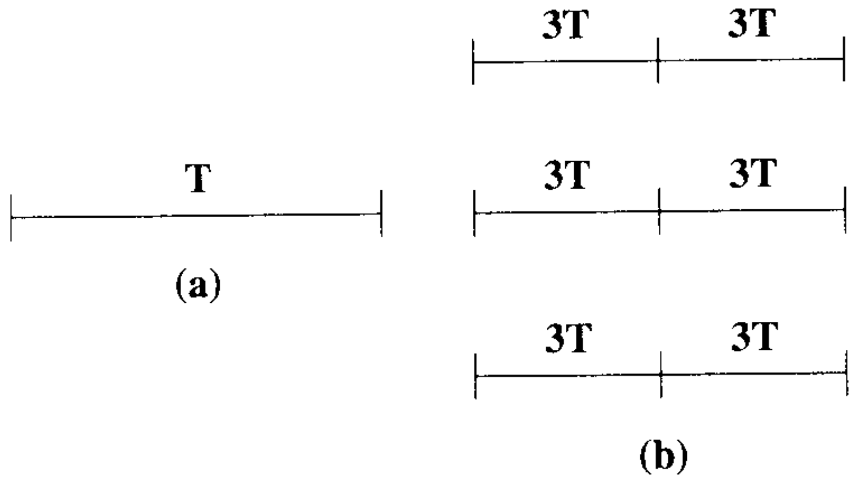
$$\beta = 0.6589, \ or \ \boxed{\beta = 0.0282}.$$

Invalid value, less than threshold voltage

(b)

$$Ratio = \beta^2 = 43.41\%.$$

Pipelining-Parallel for Low Power



$$LT_{pd} = \frac{(C_{charge}/M)\beta V_o}{k(\beta V_0 - V_t)^2} = \frac{LC_{charge}V_0}{k(V_0 - V_t)^2}.$$

$$ML(\beta V_0 - V_t)^2 = \beta(V_0 - V_t)^2$$

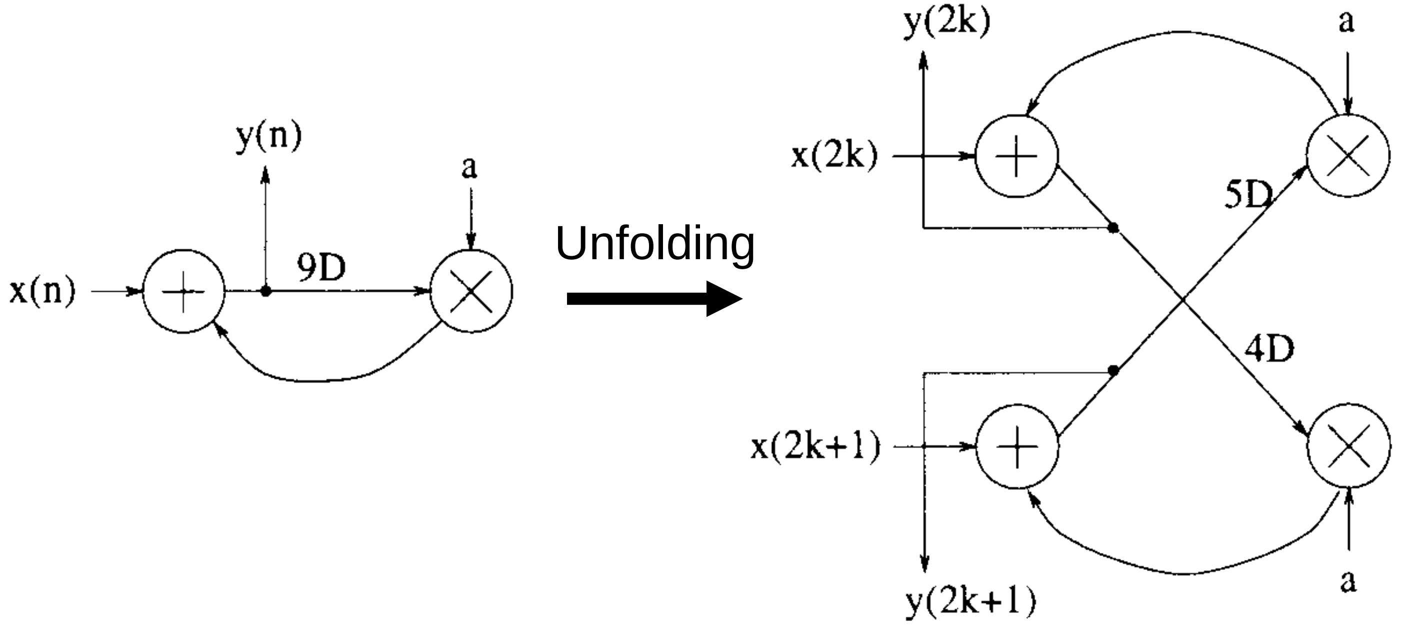
Unfolding (1/4)

- **Unfolding** is a transformation technique that can be applied to a DSP program to create a new program describing more than one iterations of the original program
- Unfolding factor J: J consecutive iterations
- Also called as loop unrolling

Unfolding (2/4)

- For the DSP algorithm
  - $y(n)=ay(n-9)+x(n)$
- Replace n with 2k and 2k+1
  - $y(2k)=ay(2k-9)+x(2k)$
  - $y(2k+1)=ay(2k-8)+x(2k+1)$
- It is an unfolded algorithm with J=2!

Unfolding (3/4)



- Note that, in unfolded systems, each delay is J-slow

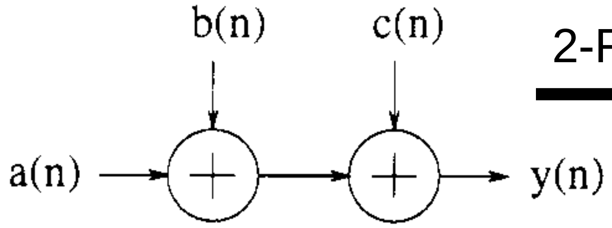
Unfolding (4/4)

- Applications of unfolding
  - To reveal hidden concurrent so that the program can be scheduled to a smaller iteration period
  - To design parallel architecture

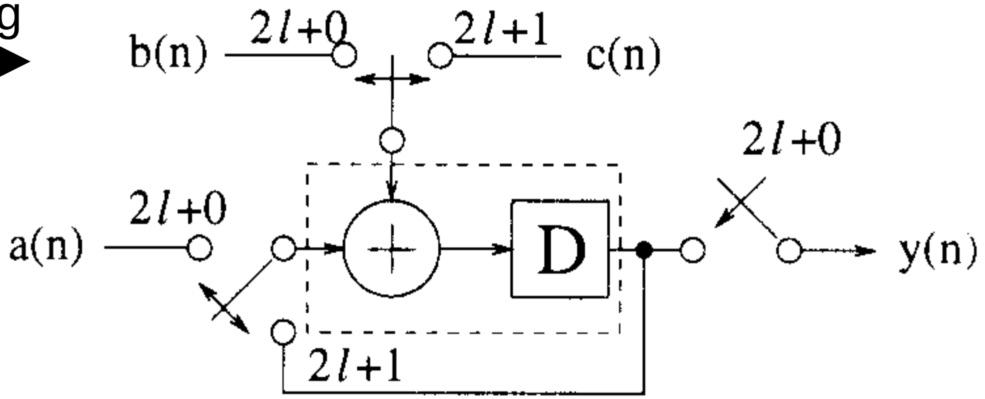
Folding (1/2)

- Folding transform is used to systematically determine the control circuits in DSP architectures where multiple algorithm operations are **time-multiplexed** to a single functional unit
  - Trading area for time in a DSP architecture
  - Reducing the number of hardware functional units by a factor of N at the expense of increasing the computation time by a factor of N

Folding (2/2)



2-Folding



Scheduling

Cycle	Adder Input (left)	Adder Input (top)	System Output
0	$a(0)$	$b(0)$	–
1	$a(0)+b(0)$	$c(0)$	–
2	$a(1)$	$b(1)$	$a(0)+b(0)+c(0)$
3	$a(1)+b(1)$	$c(1)$	–
4	$a(2)$	$b(2)$	$a(1)+b(1)+c(1)$
5	$a(2)+b(2)$	$c(2)$	–

Scheduling and Resource Allocation

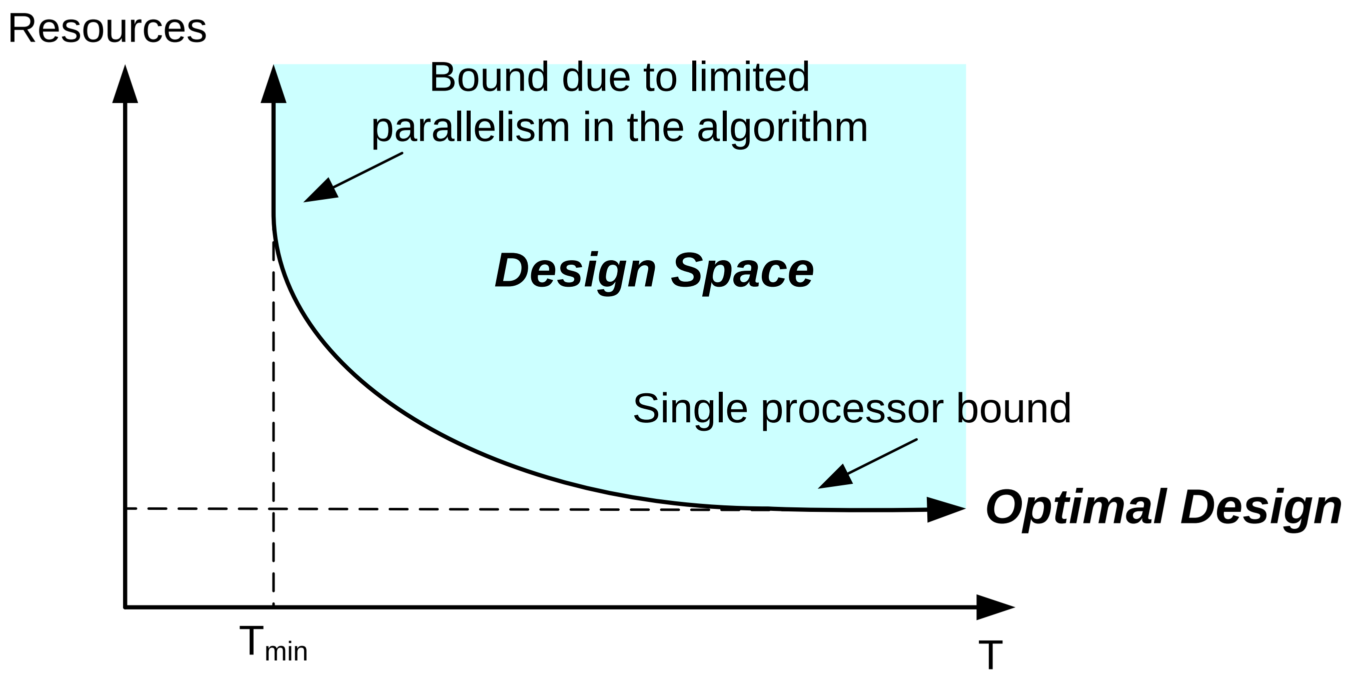
- **Scheduling** and **resource allocation** are two important tasks in hardware or software synthesis of DSP systems
- Scheduling – **when** to do the process?
  - Assign every node of the DFG to a control time step, the fundamental sequencing units in synchronous systems and correspond to clock cycles
- Resource allocation – **who** to execute the process?
  - Assign operations to hardware with the goal of minimizing the amount of hardware required to implement the desired behavior

Scheduling

- Static scheduling
  - If all processes are known in advance
  - Perform scheduling before run time
  - Most DSP algorithms are amenable to static scheduling
- Dynamic scheduling
  - When the process characteristics are not completely known
  - Decide dynamically at run time by scheduler that runs concurrently with the program

Design Space Exploration

- Resource v.s. sample rate for different schedules


Resources
Bound due to limited parallelism in the algorithm
Design Space
Single processor bound
Optimal Design
T_{min}
T