DSP Design

Pipelining
&
Parallel Processing
cont.

Steffen Malkowsky

Repetition

- **DSP algorithms are non-terminating** = repeatedly execute same code
- ***Iteration*** = all operations are executed once
- ***Iteration period*** = time to perform one iteration
- ***Sampling rate*** (***throughput***) = number of samples per second
- ***Latency*** = time difference between output sample and corresponding input sample (how long before producing output samples)

Block Diagram - close to actual hardware – interconnected functional blocks, potentially with delay elements between blocks

DFG - Capture the data-driven nature of a DSP system, intra-iteration and inter-iteration constraints, nodes are computations (functions, subtasks), edges are data paths, very general description. Difference from block diagram: Hardware not allocated, scheduled in DFG.

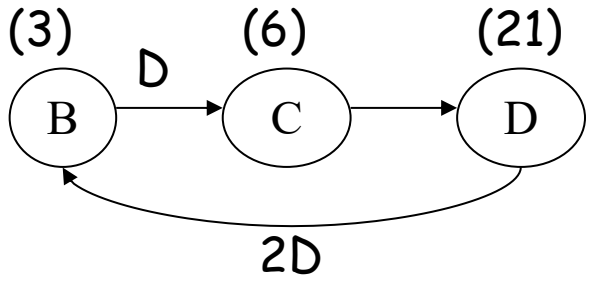
Repetition

- **Critical path** - the combinational path with maximum total execution time
- **Loop (=cycle)** - a path beginning and ending at same node
- **Loop bound for loop**

$$\frac{T_j}{W_j}$$

T_j loop computation time

W_j number of delays in loop



- **Iteration Bound** - maximum of all loop bounds

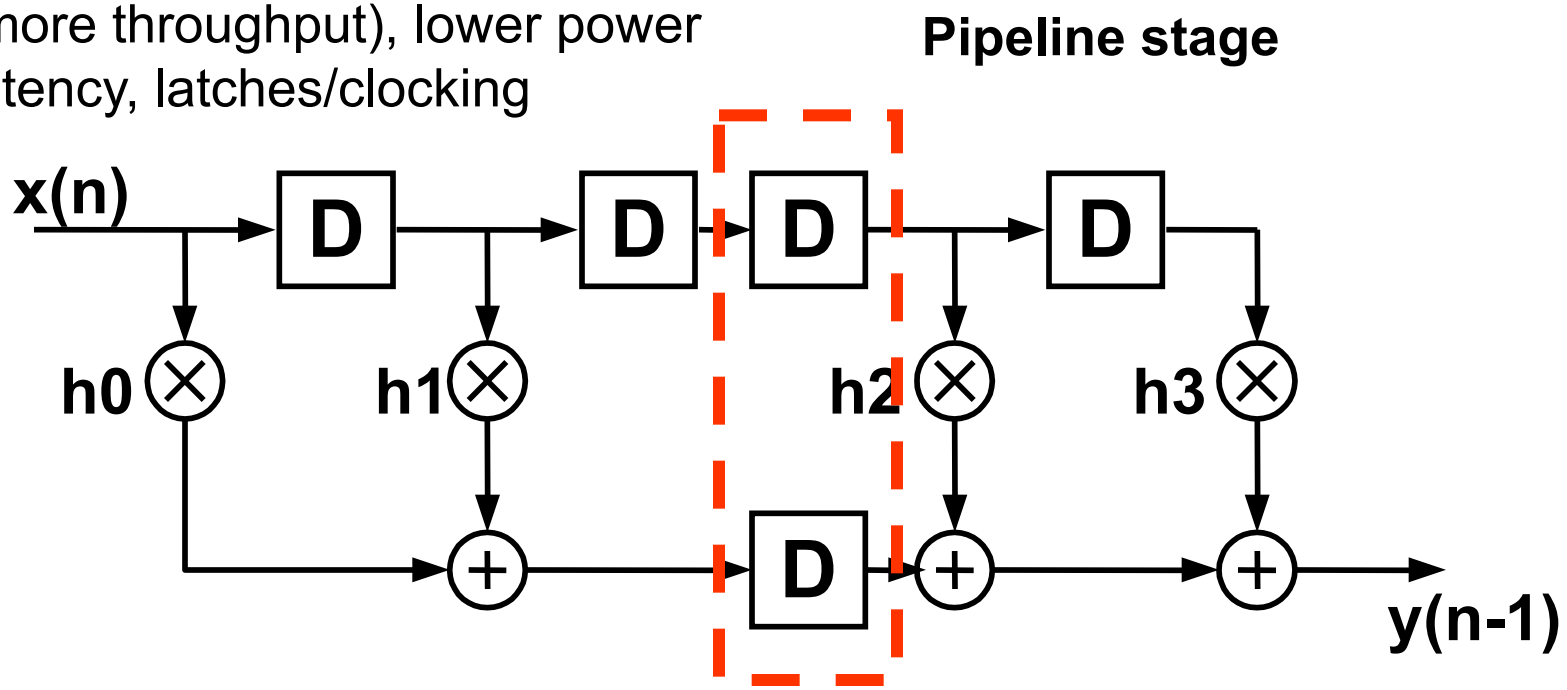
$$T_\infty = \max_{l \in L} \left\{ \frac{t_l}{w_l} \right\}$$

It is the lower bound on execution time for DFG (assuming only pipelining, retiming, unfolding)

Repetition

Pipelining - insert delay elements to reduce critical path length, 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

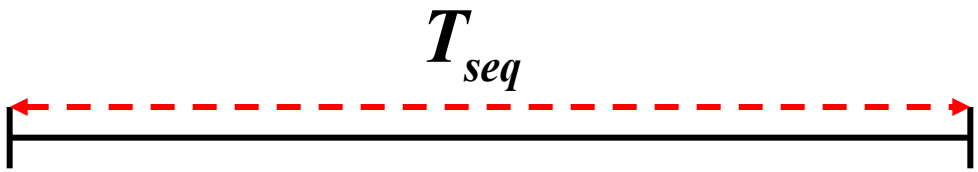
+ Faster (more throughput), lower power
- Added latency, latches/clocking



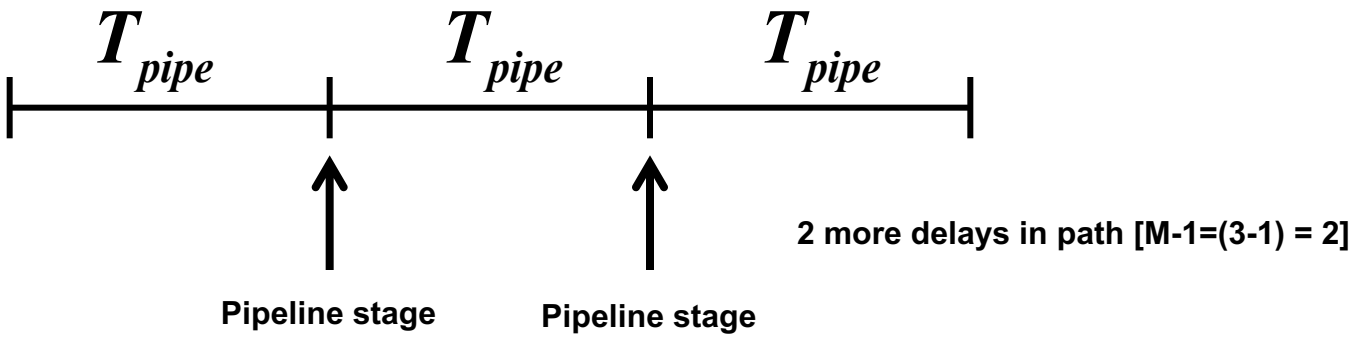
Repetition

- **Pipelining** - insert delay elements to reduce critical path length. 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

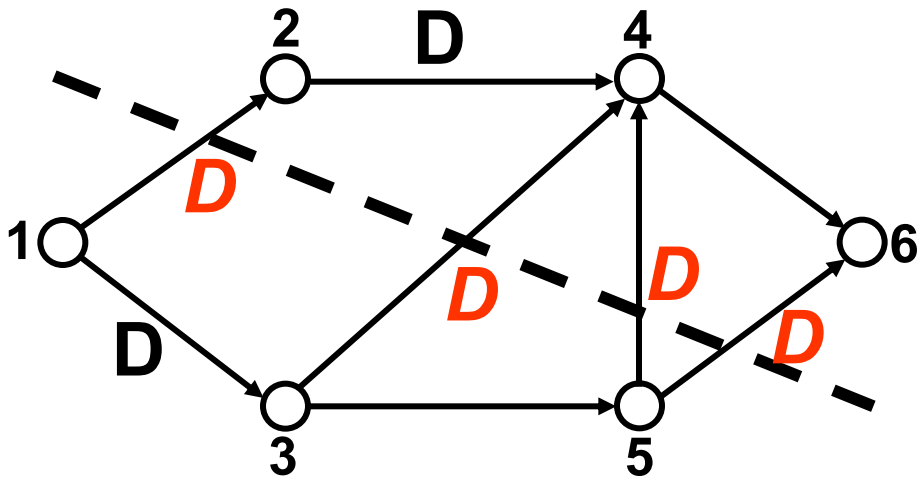
Sequential (critical path):



Pipelined: (critical path when M=3)



Repetition: Feedforward Cutsets

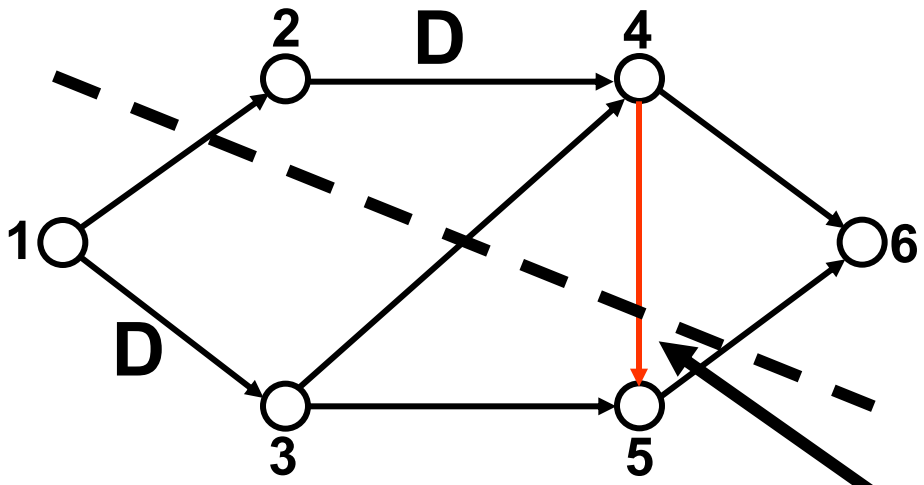


$T_{critical}$ = 3⇨5⇨4⇨6

Feedforward cutset

Must place delays on all edges in the cutset

Repetition: Feedforward Cutsets



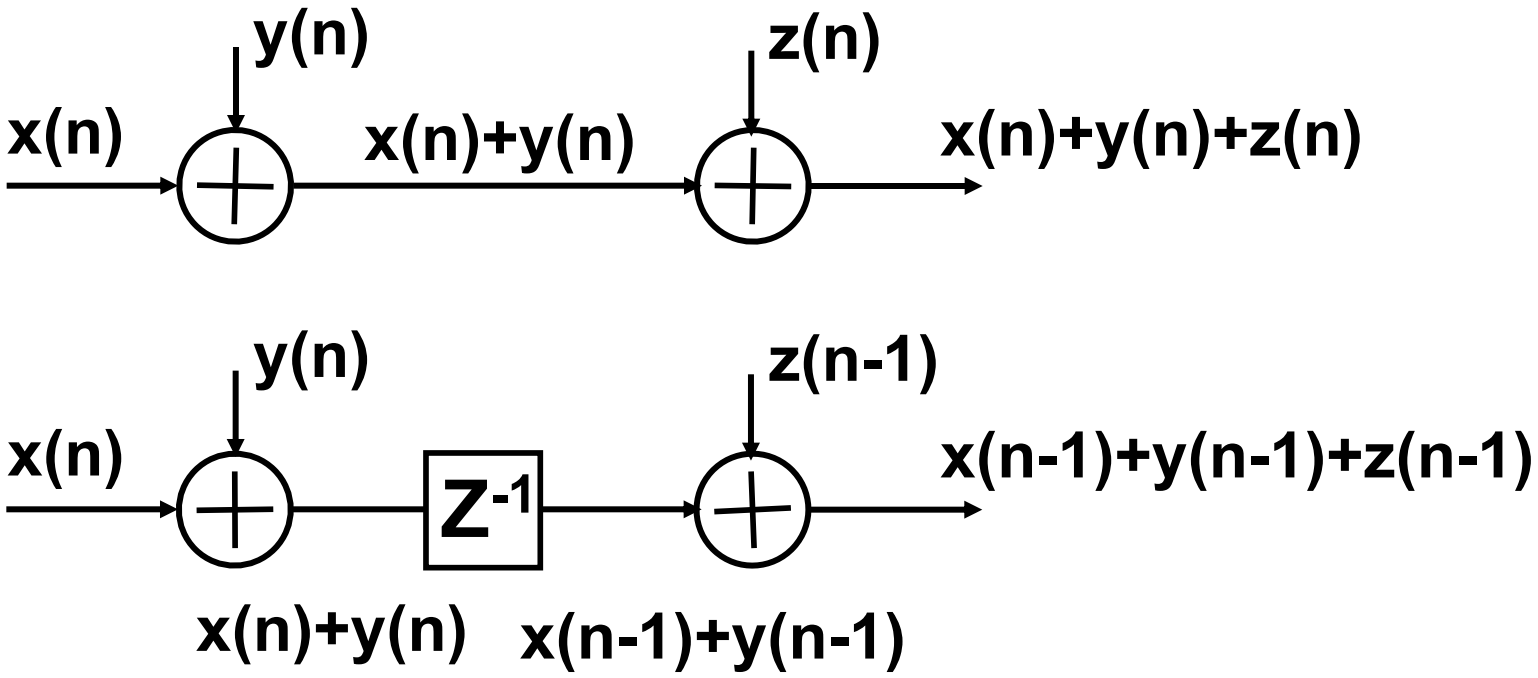
$T_{critical}$ = 3⇨4⇨5⇨6

Not a Feedforward cutset

Pipelining not possible

Let’s continue…

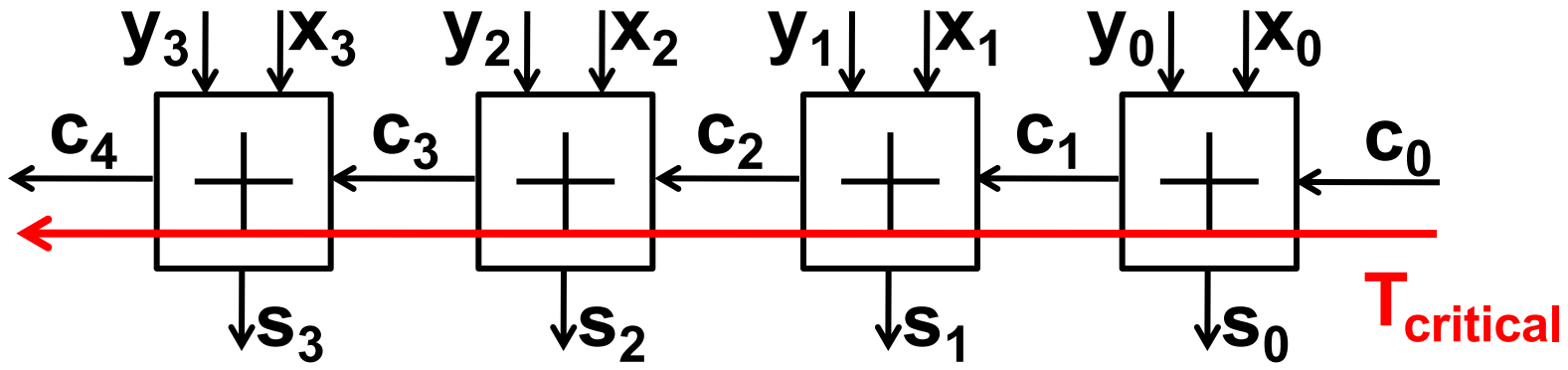
Pipelining (cont.)



We said that the critical path could be cut in half by introducing pipelining.

Is that always true??

Example: Pipelining when ripple-carry adders



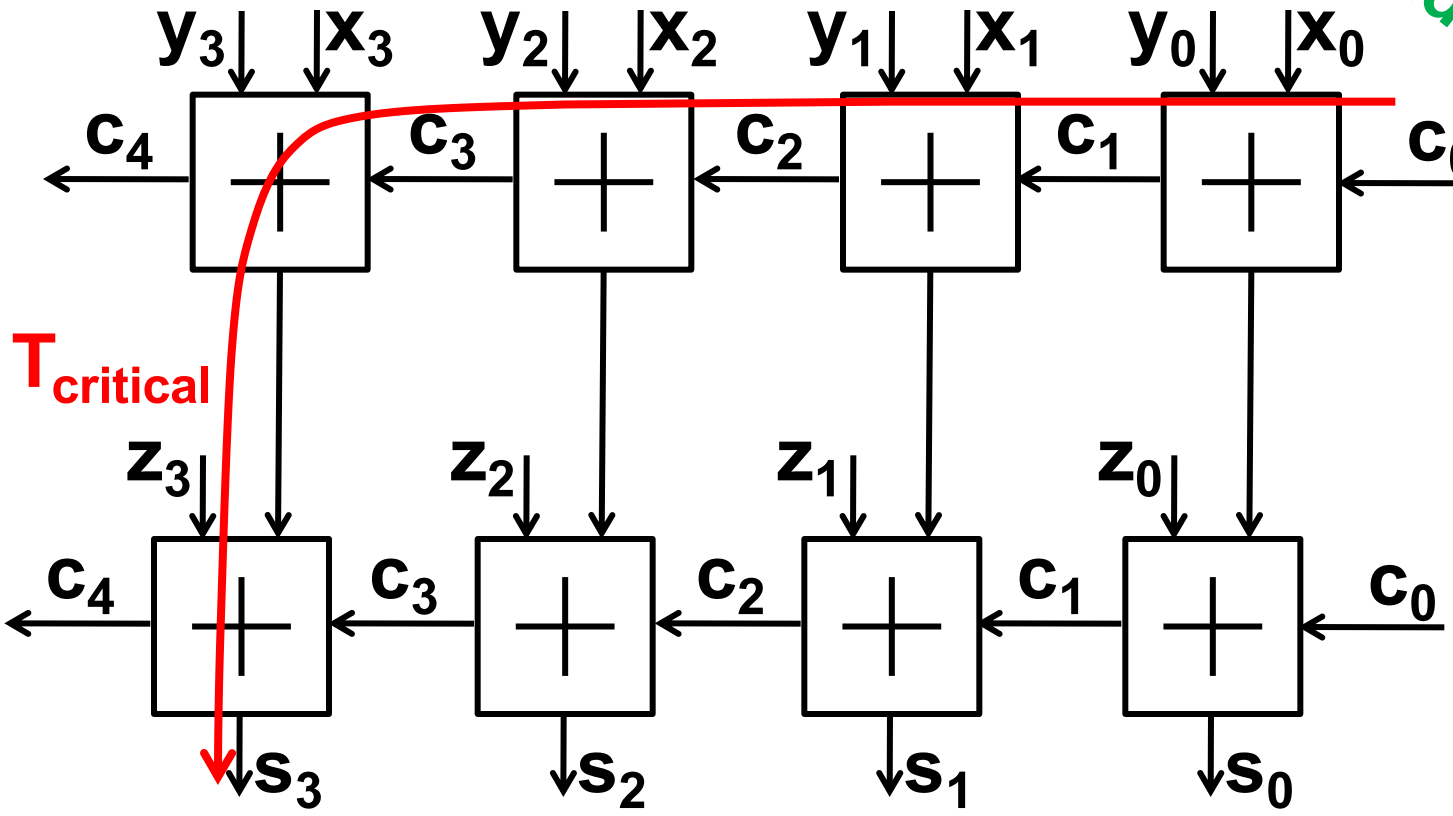
Assume the delay in calculating the sum, s_{delay}, is equal to calculating the carry, c_{delay}.

What is the critical path?

4 x c_{delay}

Example: Pipelining when ripple-carry adders

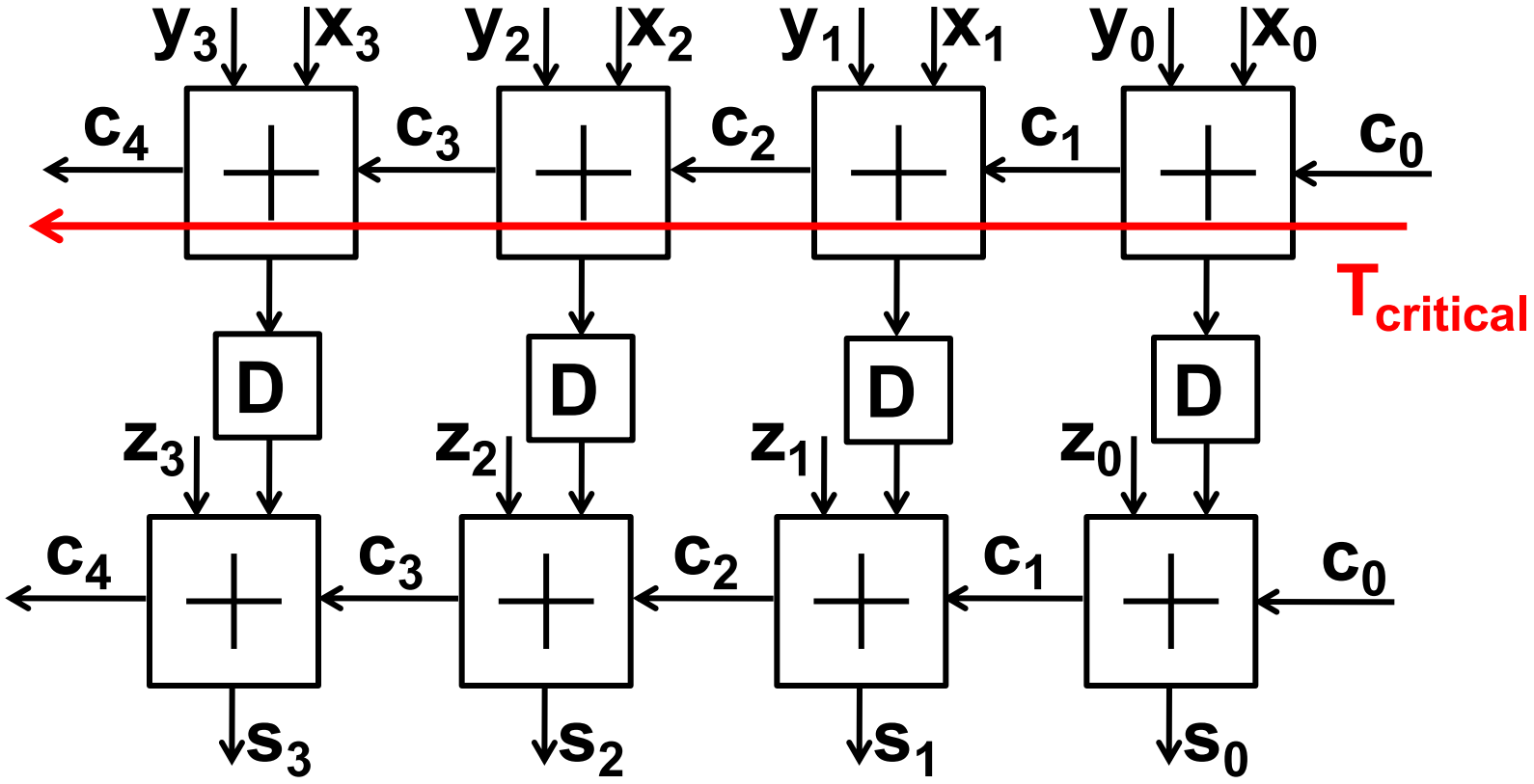
Let's add a stage



What is the critical path now?

$$4 \times c_{delay} + s_{delay} \approx 5 \times c_{delay}$$

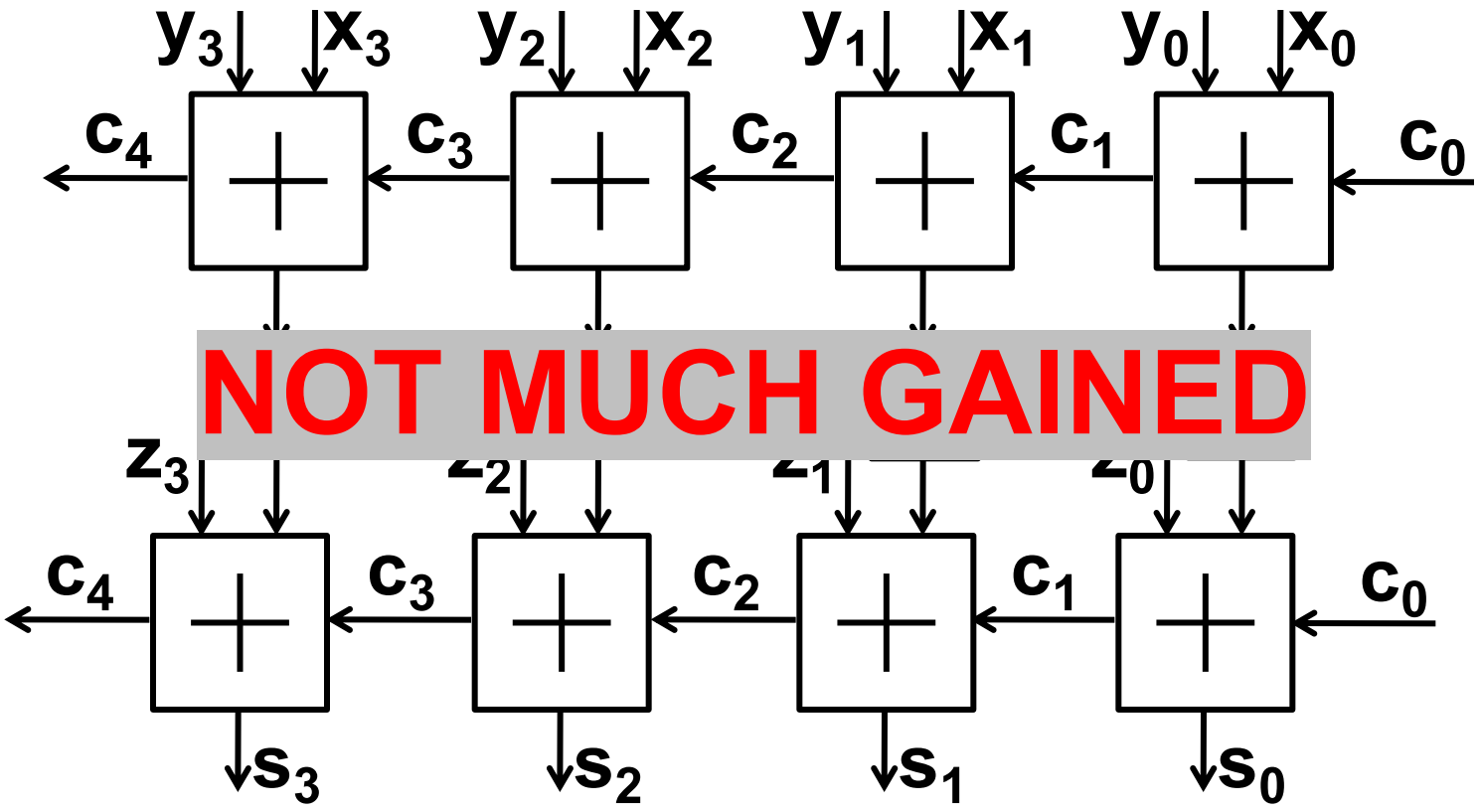
Example: Pipelining when ripple-carry adders



What is the critical path now?

4 x c_{delay}

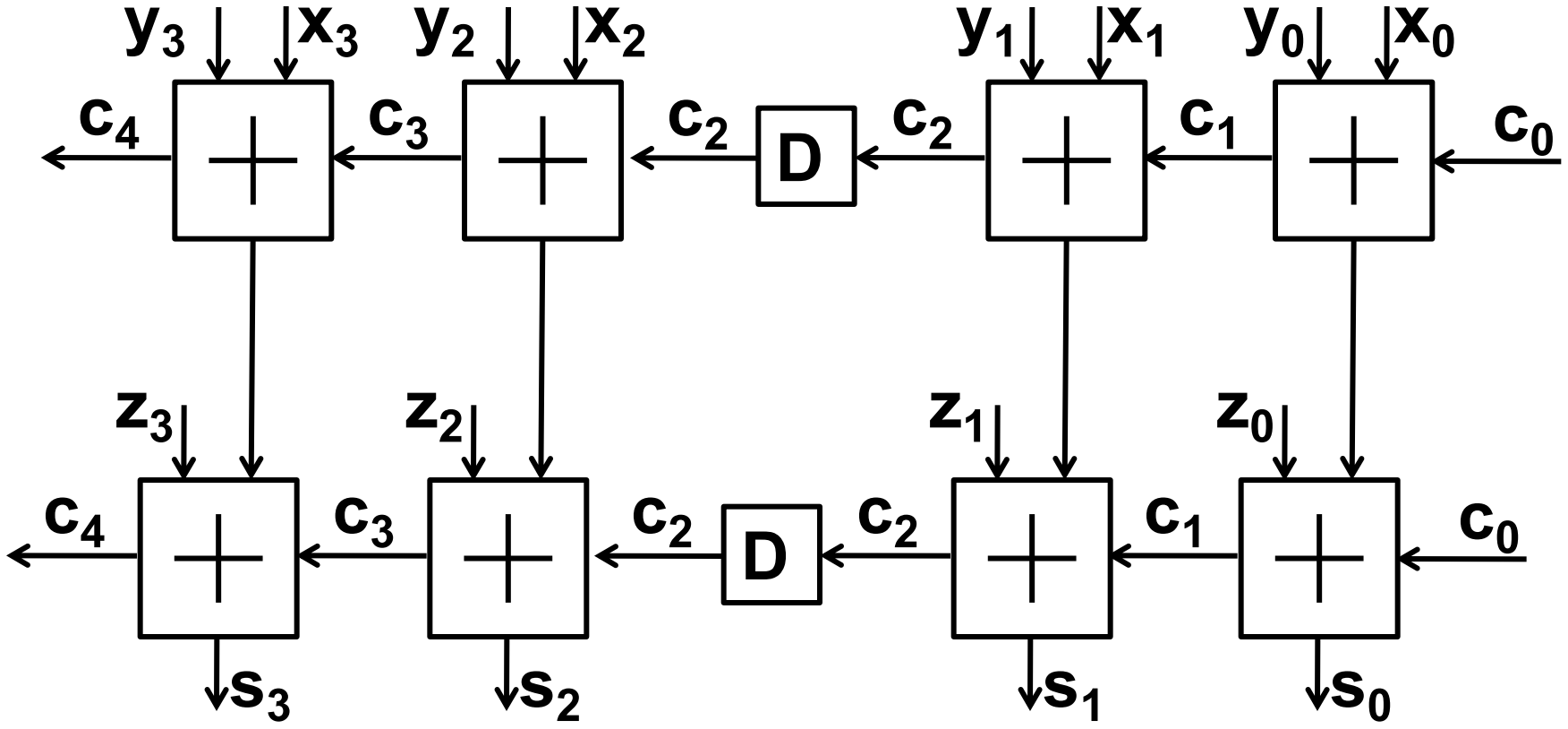
Example: Pipelining when ripple-carry adders



What is the critical path now?

4 x c_{delay}

Example: Pipelining when ripple-carry adders



What is the critical path now?

Possible??

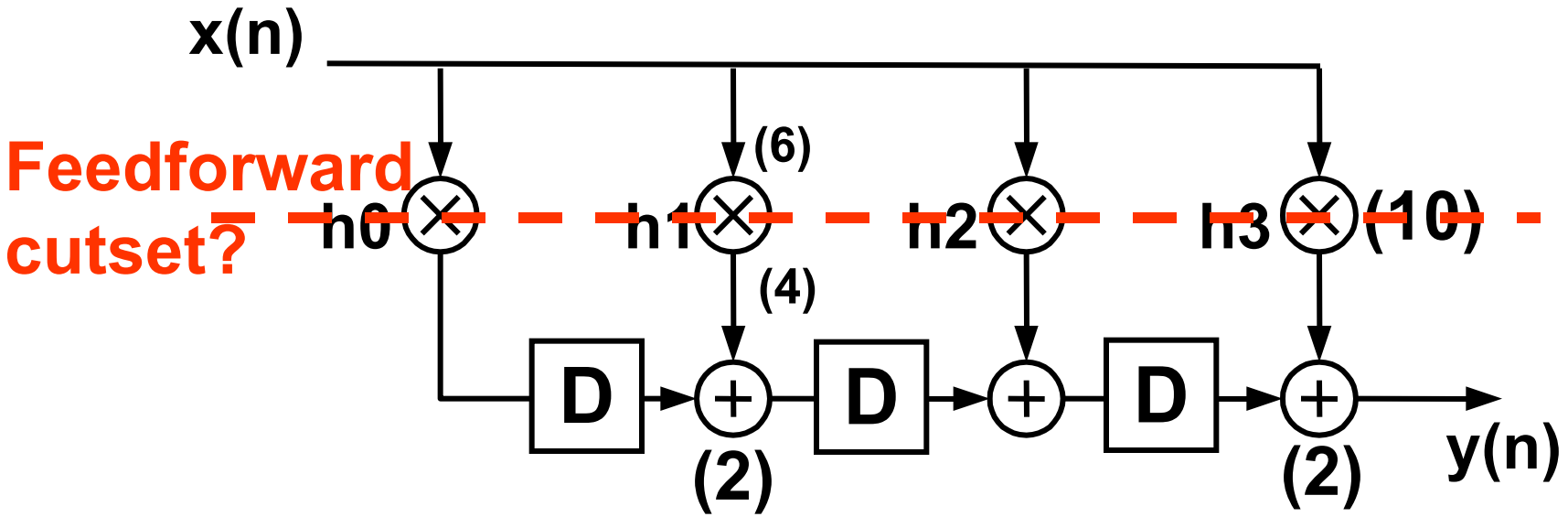
Conclusion

The result depends on the structure of the used blocks, e.g. type of adder (ripple-carry, carry save, carry look-ahead,...).

We have to understand how the blocks work and if the critical paths are independent or if there is a relationship.

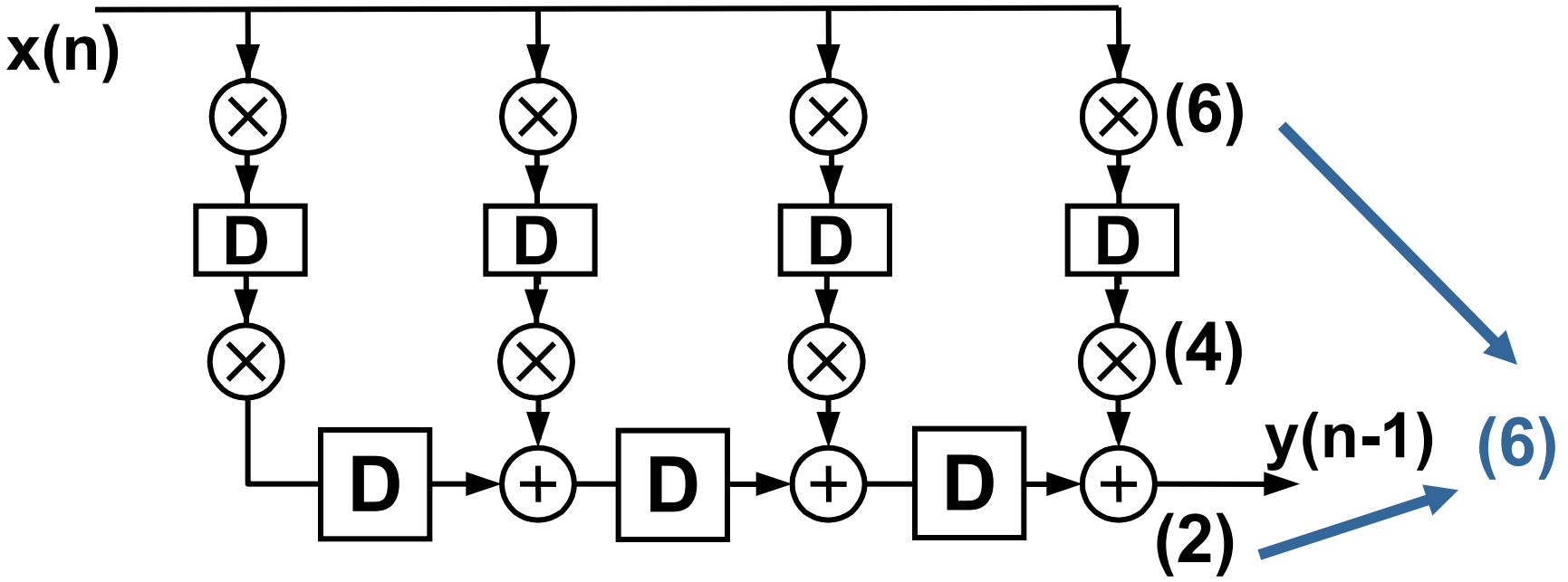
Fine-Grain pielining

– Let T_M=10 units and T_A=2 units. If the multiplier is broken into 2 smaller units with processing times of 6 units and 4 units, respectively (by placing the latches on the horizontal cutset across the multiplier), then the desired clock period can be achieved as $(T_M+T_A)/2$



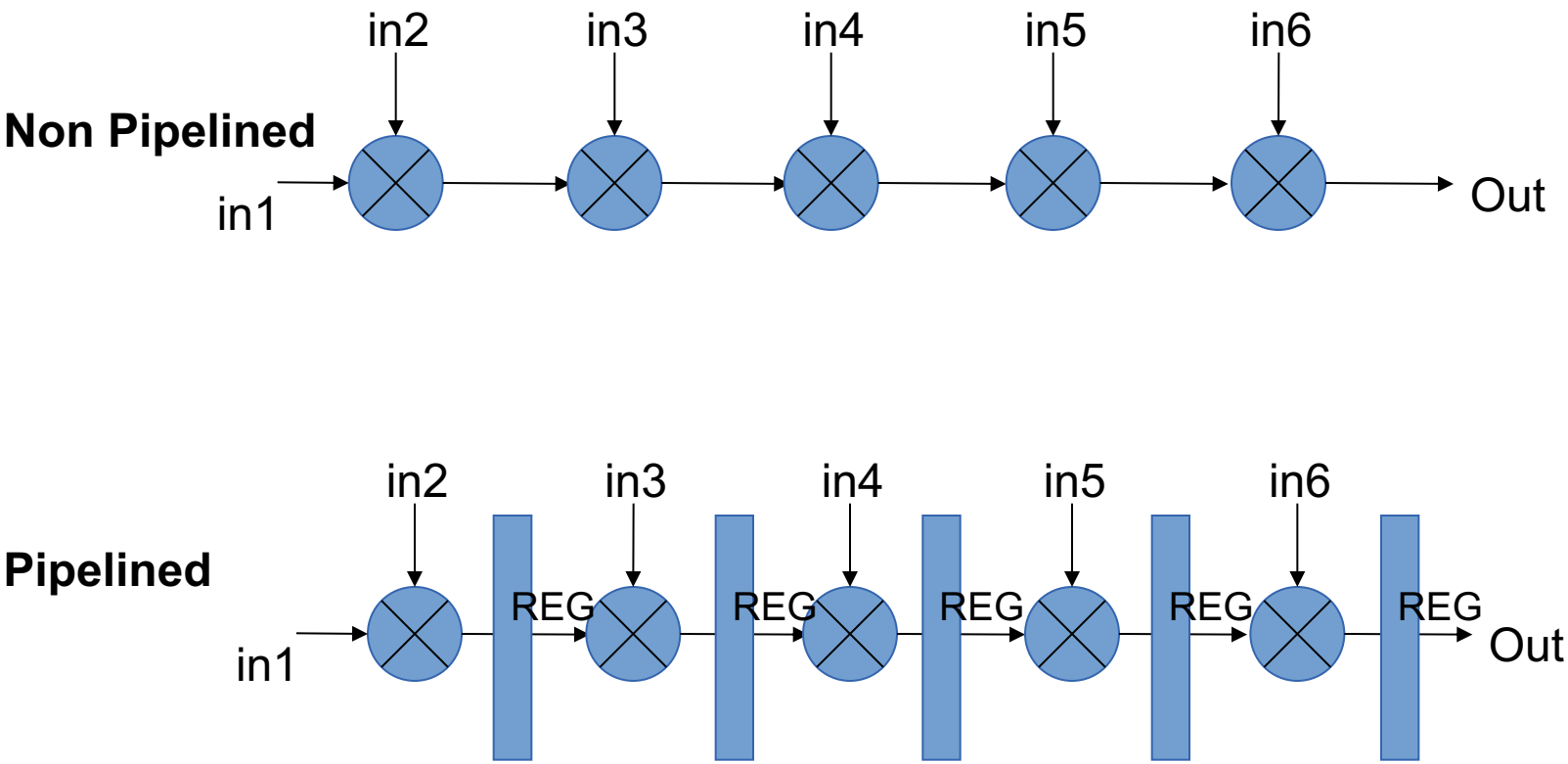
Fine-Grain pielining

- Equal paths. We get a more balanced design.
- A fine-grain pipelined version of the 3-tap data-broadcast FIR filter is shown below.



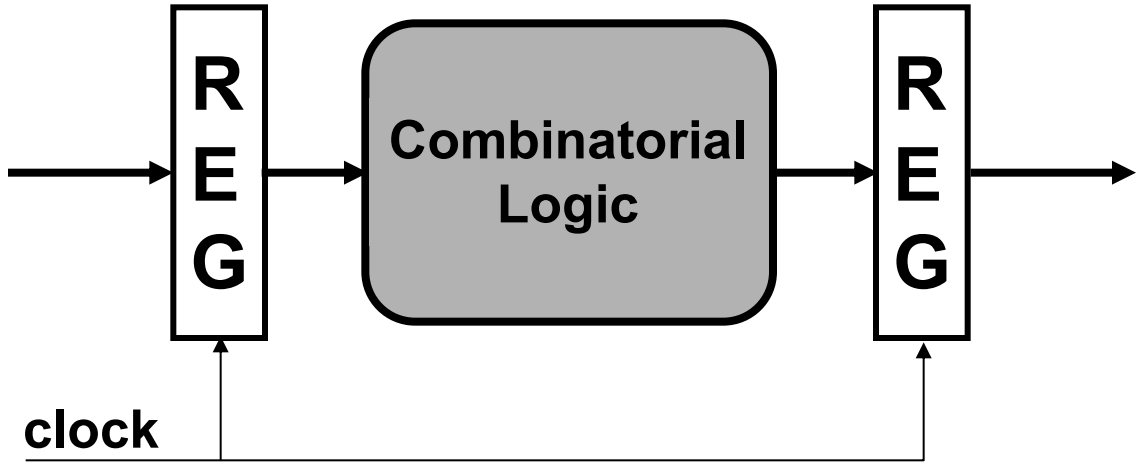
Pipelining to Reduce Switching Activity

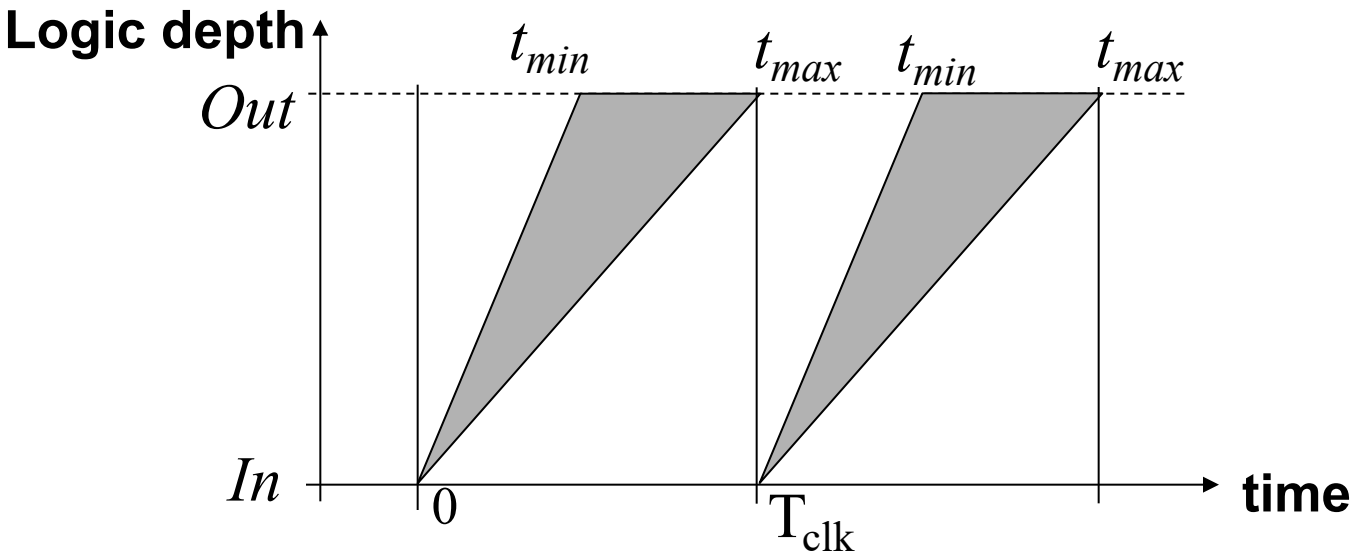
In chained operations there will be spurious transitions.



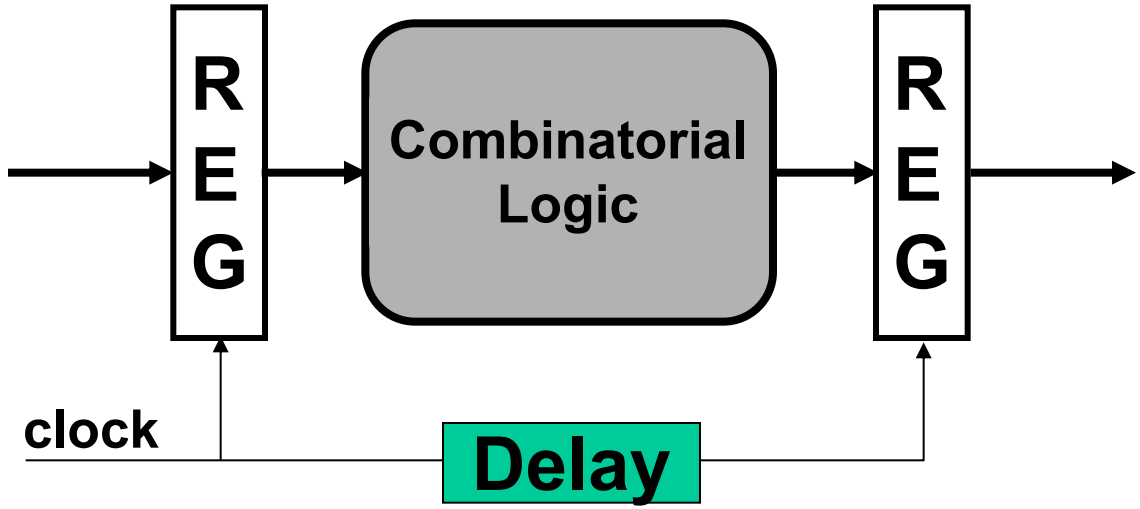
Results from PhD student Rakesh Gangarajaiah

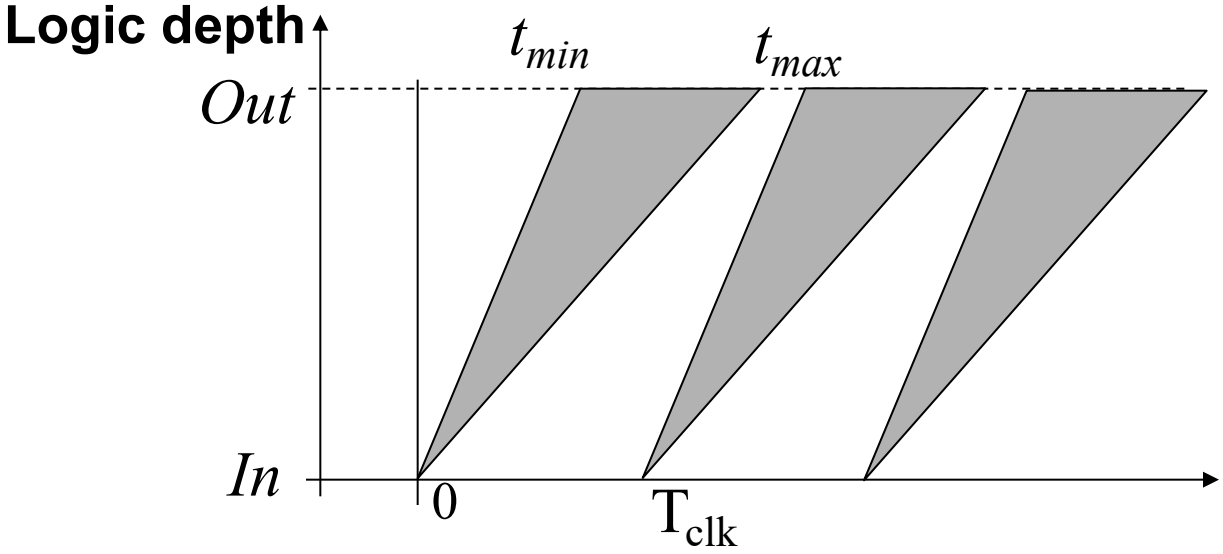
Synchronous Pipelining





Wave Pipelining





New input data is applied before the previous computation is done.

$T_{clk} < T_{critical\ path}$

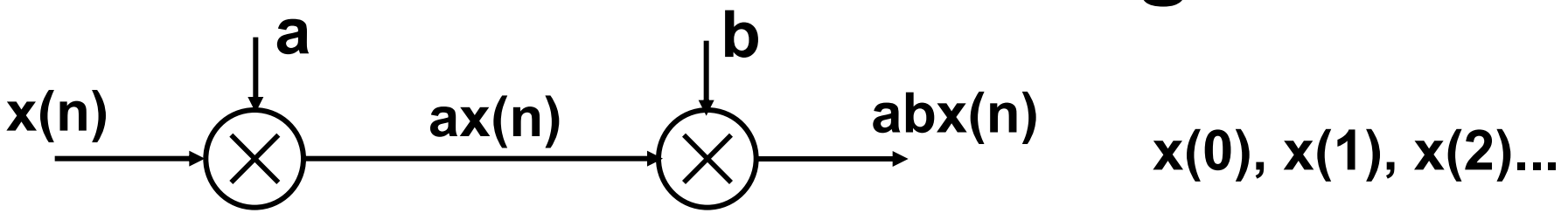
Wave pipelining: Pros and cons?

+ shorter T_{clk}

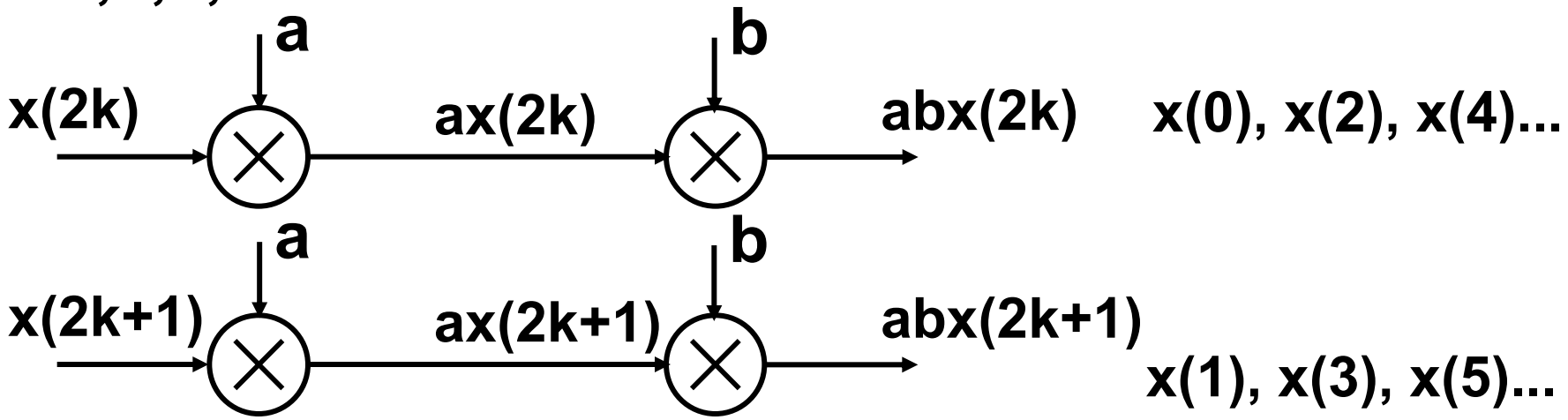
- extensive simulation
- tedious design
- hard to verify
- lack of tools

Parallel Processing

Parallel Processing



k=0,1,2,3...



Two samples are processed in parallel

- double throughput

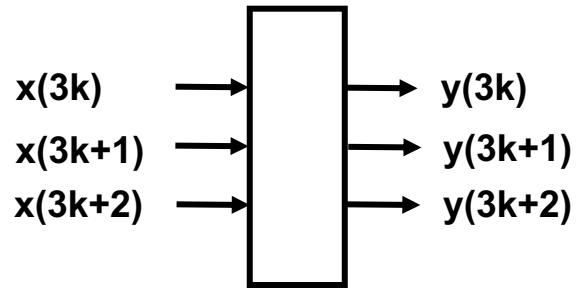
or

- lower power consumption due to reduced V_{DD}

Parallel Processing

- **Parallel processing and pipelining techniques are duals of each other: if a computation can be pipelined, it can also be processed in parallel. Both of them exploit concurrency available in the computation in different ways.**
- **How to design a Parallel FIR system?**
  - **Consider a single-input single-output (SISO) FIR filter:**
    - **$y(n)=ax(n)+bx(n-1)+cx(n-2)$**
  - **Convert the SISO system into an MIMO (multiple-input multiple-output) system in order to obtain a parallel processing structure**
    - **To get a parallel system with 3 inputs per clock cycle**

$$y(3k)=ax(3k)+bx(3k-1)+cx(3k-2)$$
$$y(3k+1)=ax(3k+1)+bx(3k)+cx(3k-1)$$
$$y(3k+2)=ax(3k+2)+bx(3k+1)+cx(3k)$$

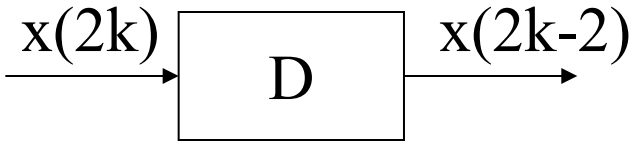


Parallel processing system is also called *block processing*, and the number of inputs processed in a clock cycle is referred to as the *block size*

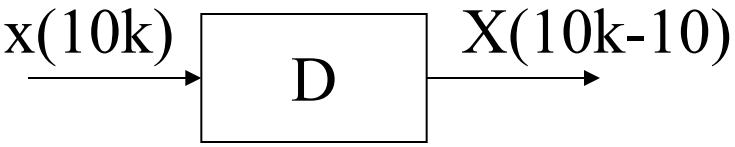
Parallel Processing (cont'd)

– **For example:**

When block size is 2, 1 delay element = 2 sampling delays



When block size is 10, 1 delay element = 10 sampling delays



Ex. Parallel 3-Tap FIR

HW x 2

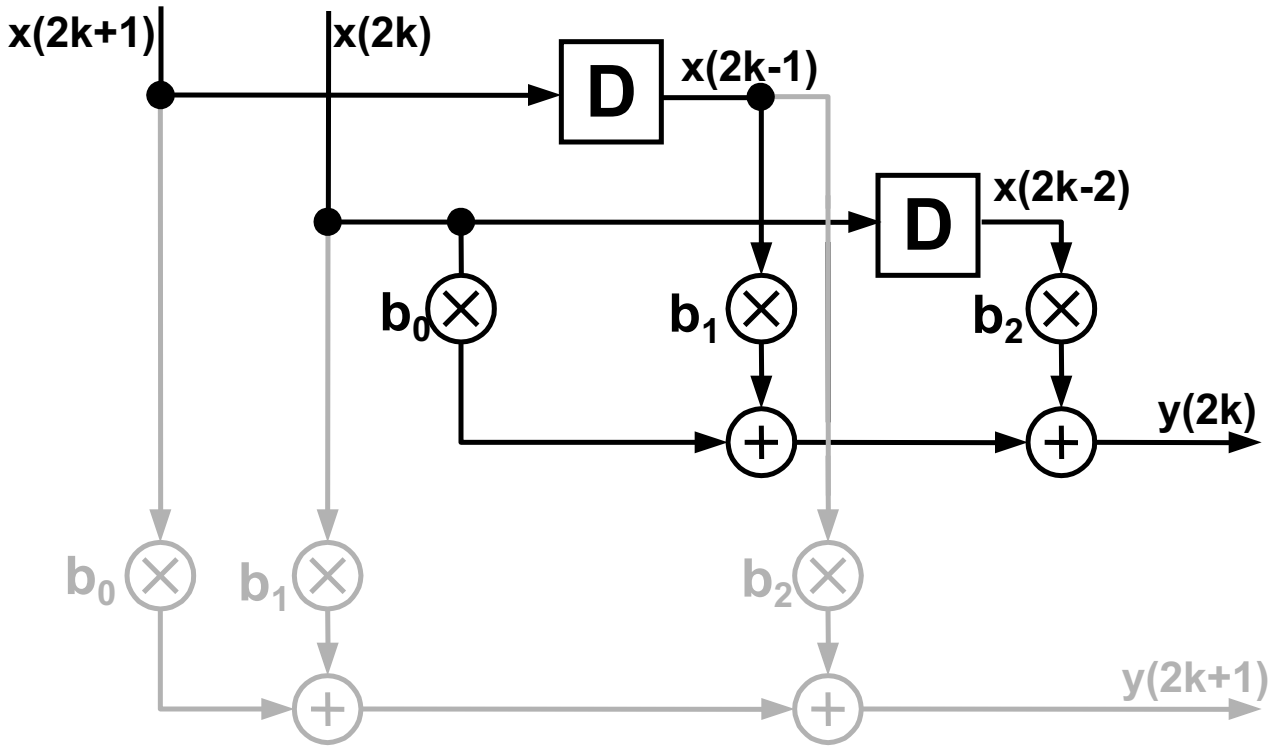
$$\begin{cases} y(n) = b_0 x(n) + b_1 x(n-1) + b_2 x(n-2) \\ y(n+1) = b_0 x(n+1) + b_1 x(n) + b_2 x(n-1) \end{cases}$$

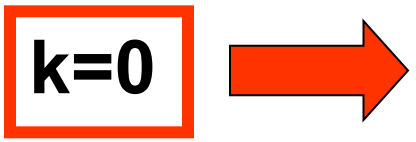
n changed to $2k$

$$\begin{cases} y(2k) = b_0 x(2k) + b_1 x(2k-1) + b_2 x(2k-2) \\ y(2k+1) = b_0 x(2k+1) + b_1 x(2k) + b_2 x(2k-1) \end{cases}$$

Parallel 3-Tap (2)

$$\begin{cases} y(2k) = b_0 x(2k) + b_1 x(2k-1) + b_2 x(2k-2) \\ y(2k+1) = b_0 x(2k+1) + b_1 x(2k) + b_2 x(2k-1) \end{cases}$$

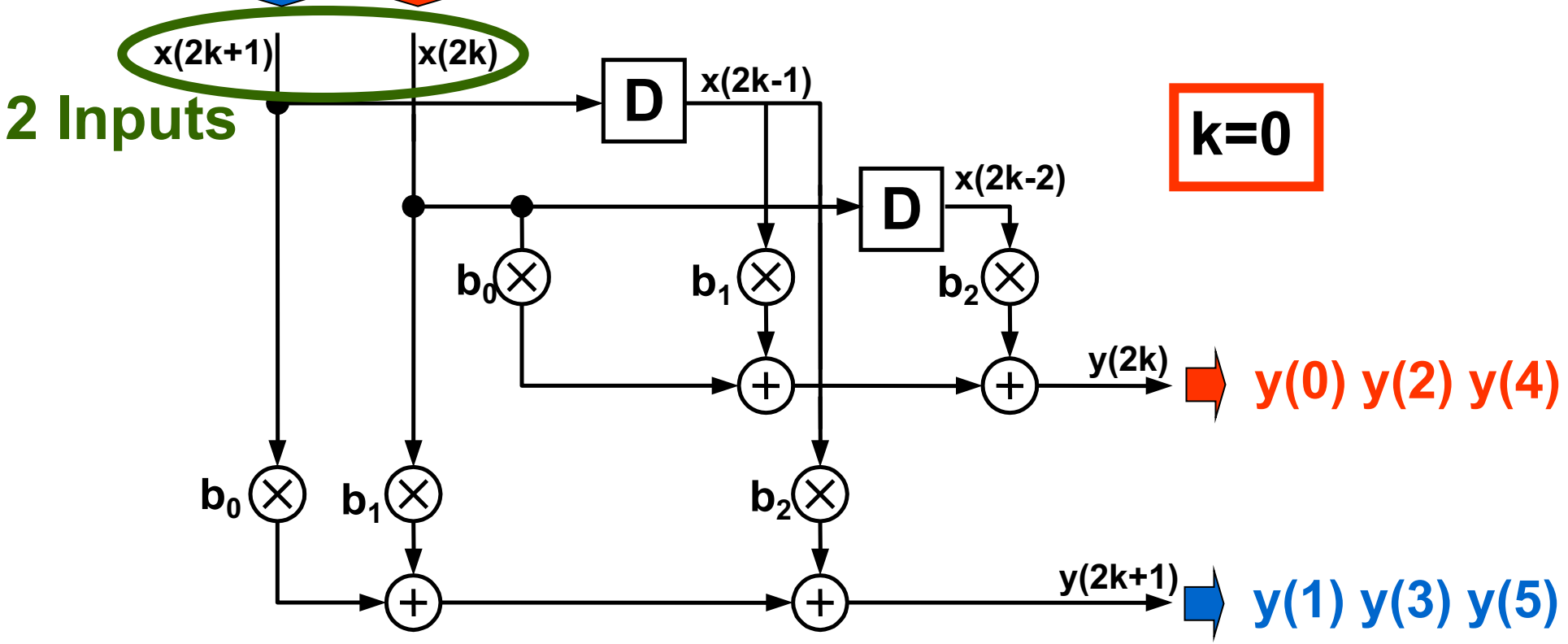




Parallel 3-Tap (3)

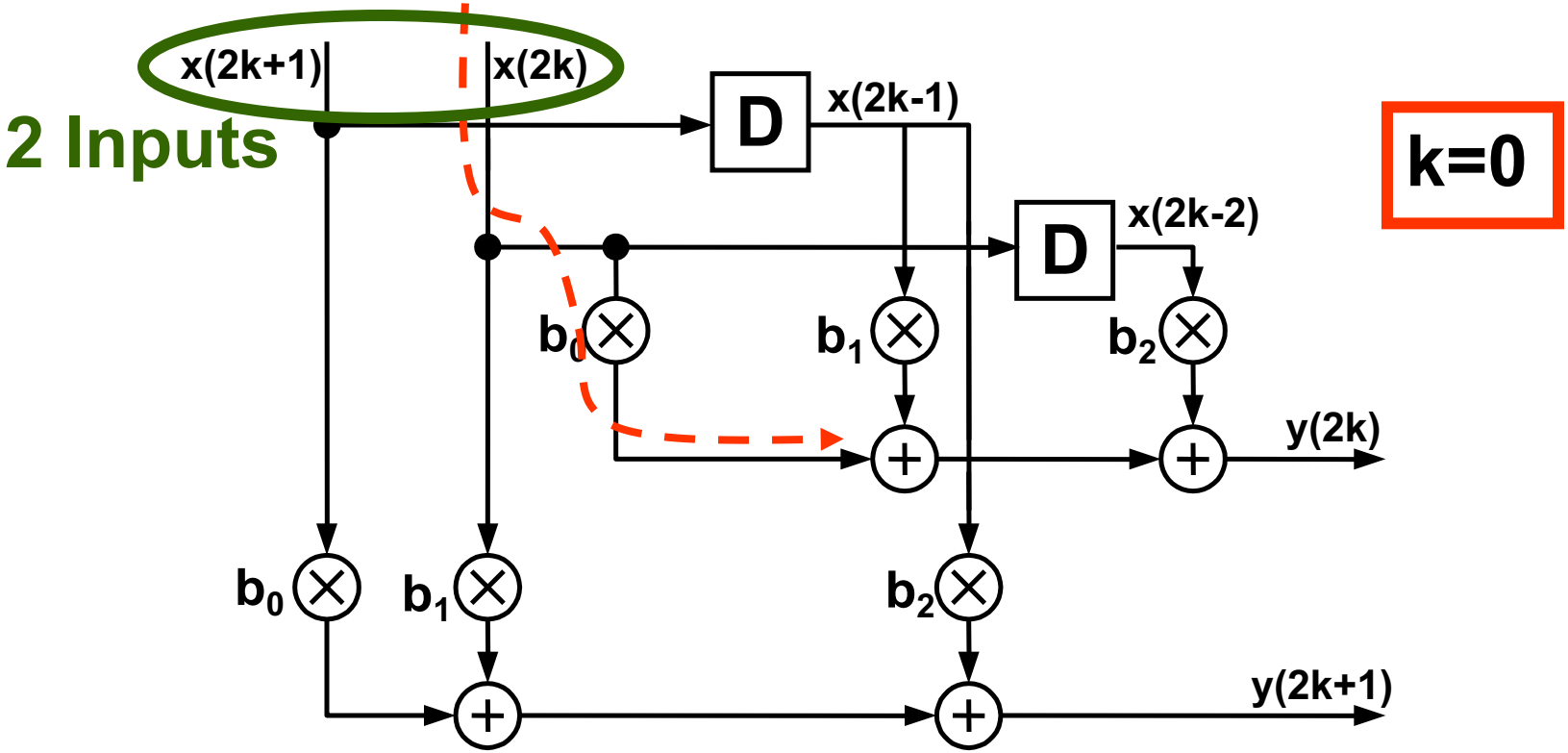
x(5) x(4)
x(3) x(2)
x(1) x(0)

$$\begin{cases} y(0) = b_0 x(0) + b_1 x(-1) + b_2 x(-2) \\ y(1) = b_0 x(1) + b_1 x(0) + b_2 x(-1) \end{cases}$$



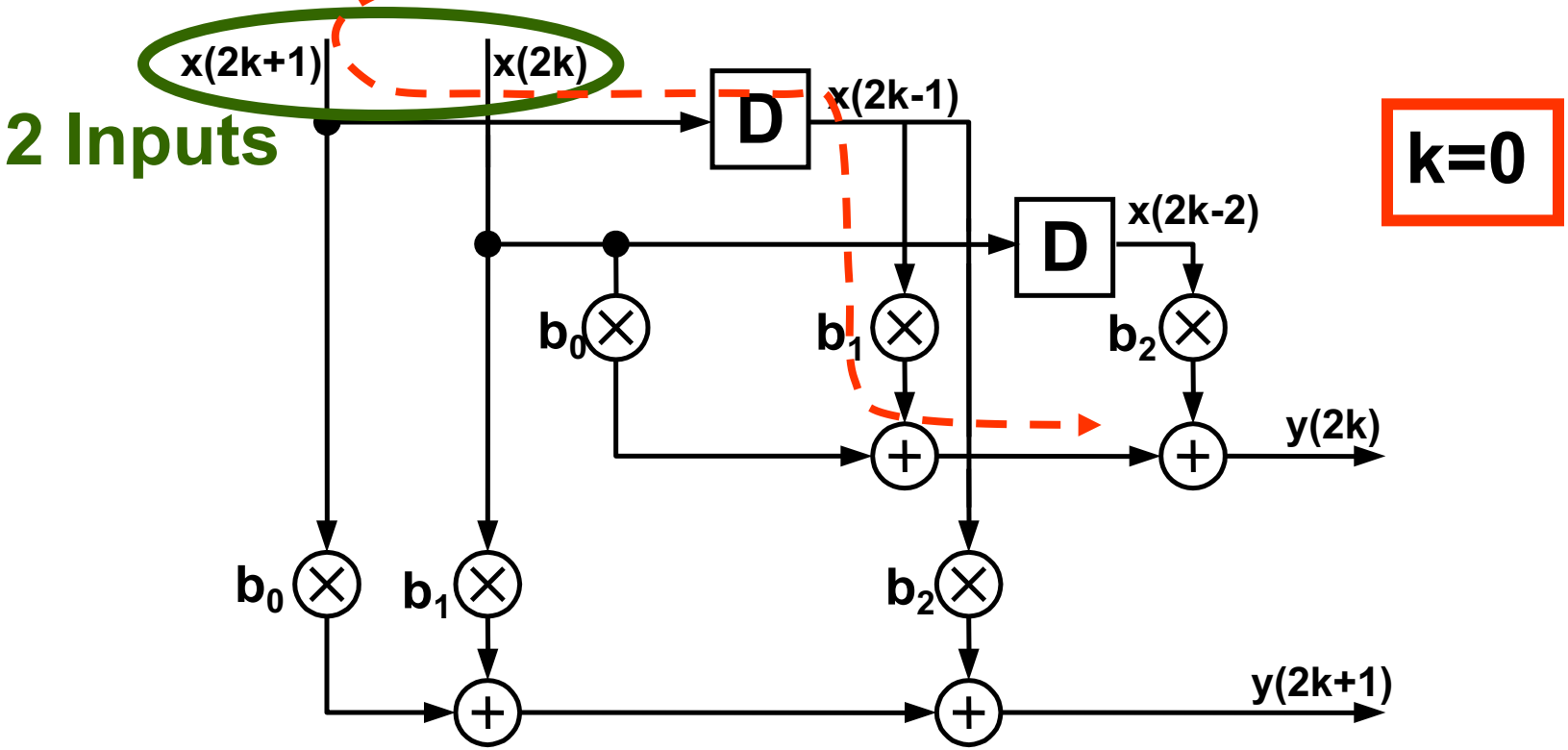
Parallel 3-Tap (4)

$$\begin{cases} y(0) = b_0 x(0) + b_1 x(-1) + b_2 x(-2) \\ y(1) = b_0 x(1) + b_1 x(0) + b_2 x(-1) \end{cases}$$



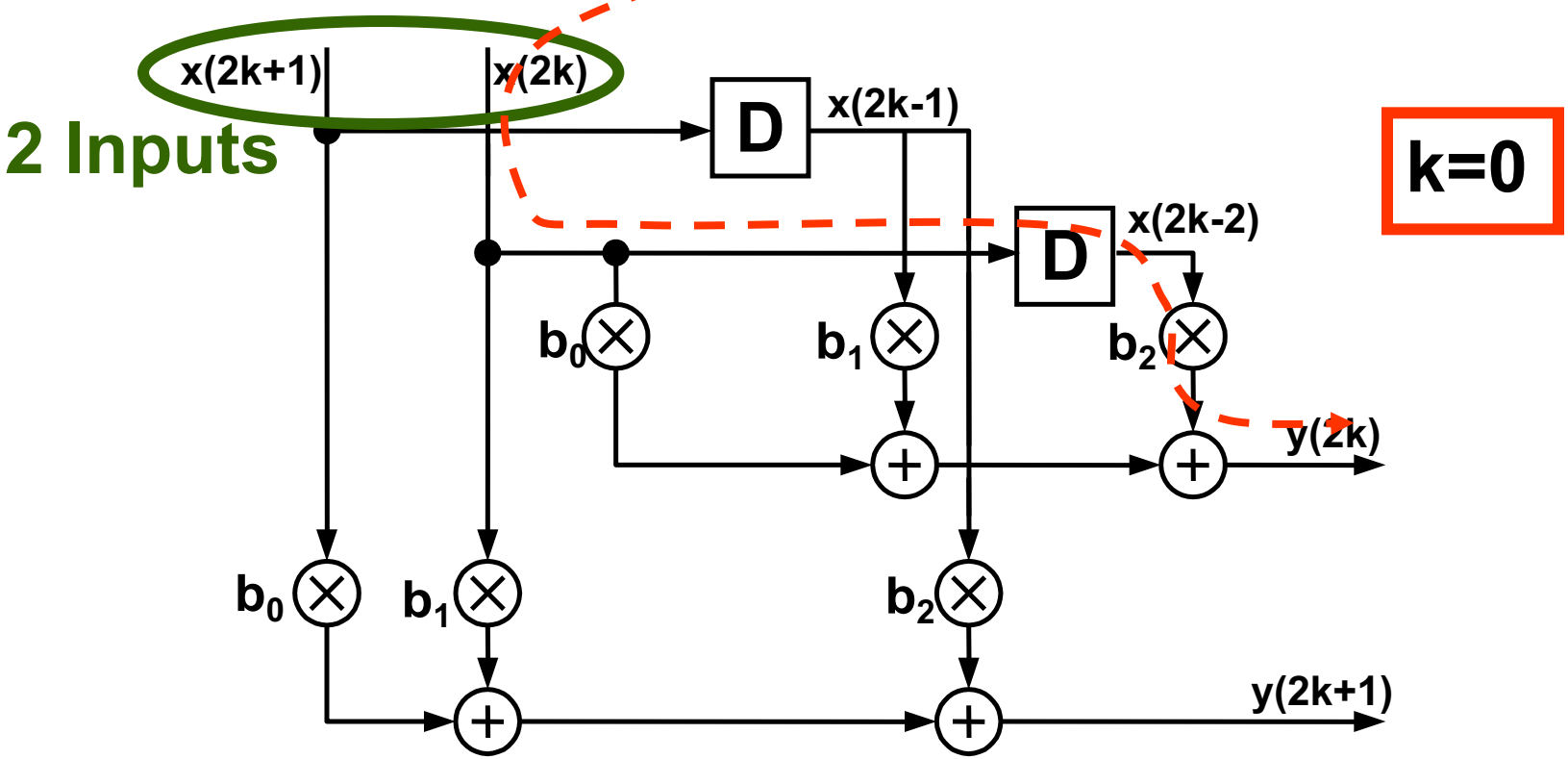
Parallel 3-Tap (4)

$$\begin{cases} y(0) = b_0 x(0) + b_1 x(-1) + b_2 x(-2) \\ y(1) = b_0 x(1) + b_1 x(0) + b_2 x(-1) \end{cases}$$



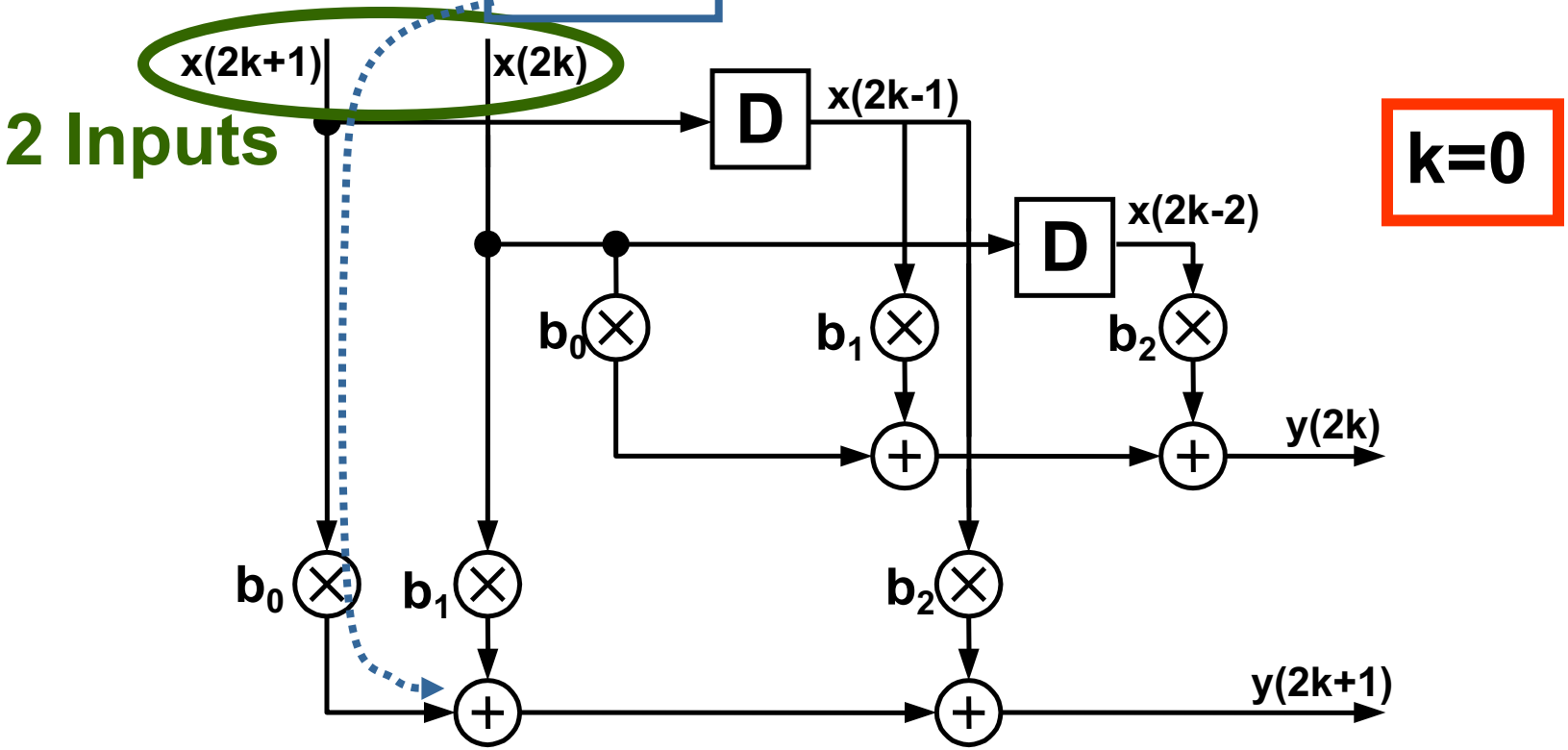
Parallel 3-Tap (4)

$$\begin{cases} y(0) = b_0 x(0) + b_1 x(-1) + b_2 x(-2) \\ y(1) = b_0 x(1) + b_1 x(0) + b_2 x(-1) \end{cases}$$



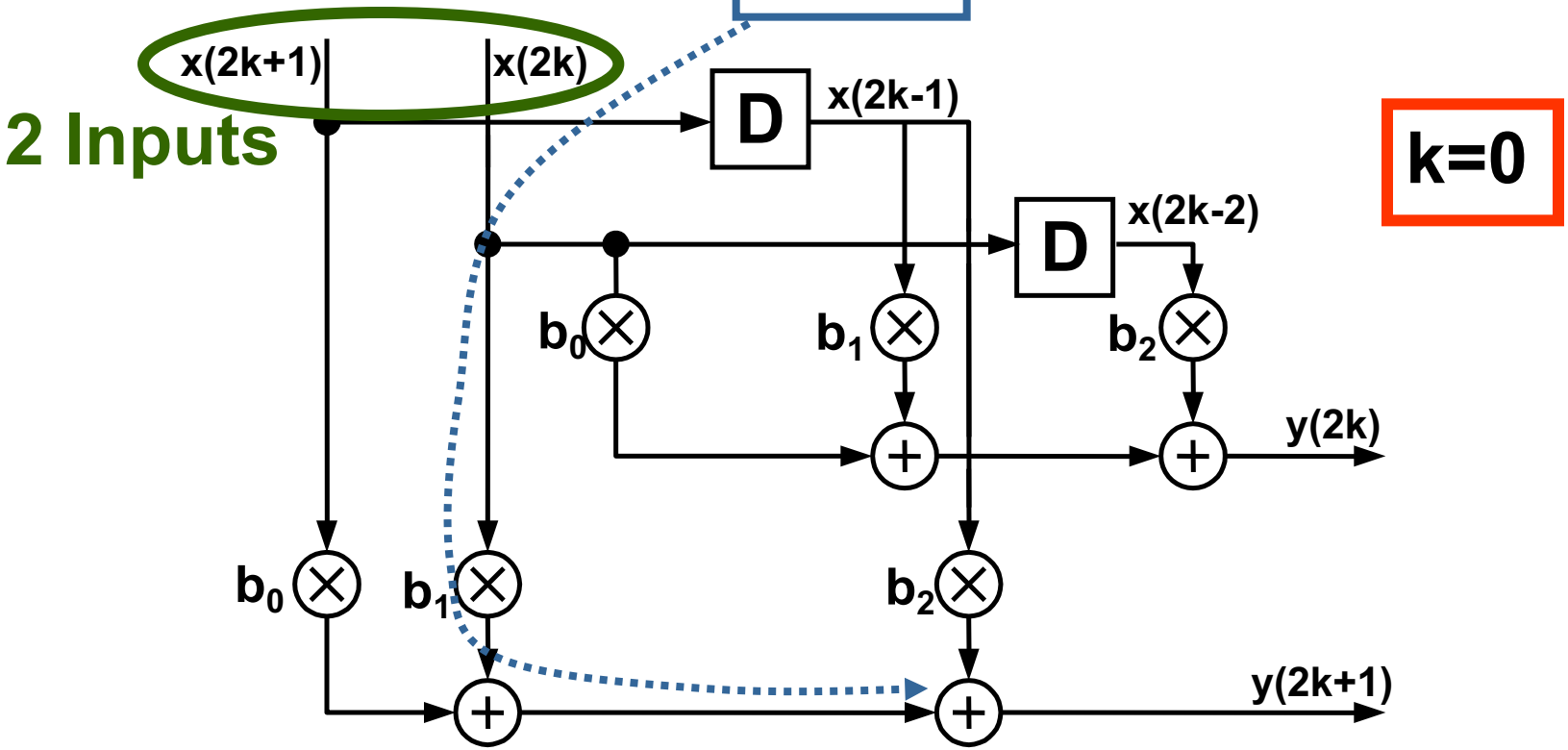
Parallel 3-Tap (5)

$$\begin{cases} y(0) = b_0 x(0) + b_1 x(-1) + b_2 x(-2) \\ y(1) = b_0 x(1) + b_1 x(0) + b_2 x(-1) \end{cases}$$



Parallel 3-Tap (5)

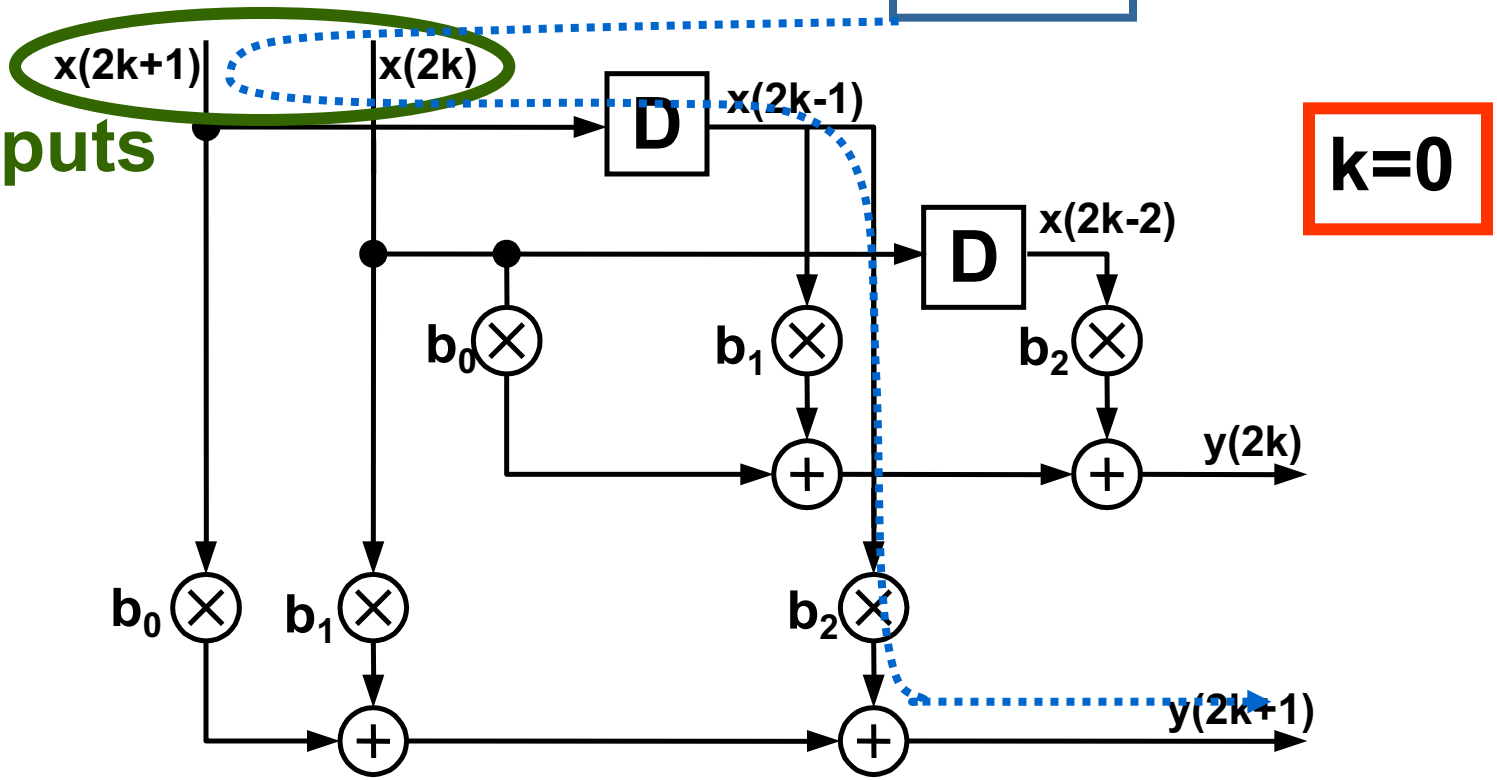
$$\begin{cases} y(0) = b_0 x(0) + b_1 x(-1) + b_2 x(-2) \\ y(1) = b_0 x(1) + b_1 x(0) + b_2 x(-1) \end{cases}$$



Parallel 3-Tap (5)

$$\begin{cases} y(0) = b_0 x(0) + b_1 x(-1) + b_2 x(-2) \\ y(1) = b_0 x(1) + b_1 x(0) + b_2 x(-1) \end{cases}$$

2 Inputs



k=0

Parallel Processing (cont'd)

- Note: The critical path of the block (or parallel) processing system remains unchanged. But since *L* samples are processed in 1 clock cycle, the iteration (or sample) period is given by the following equations:

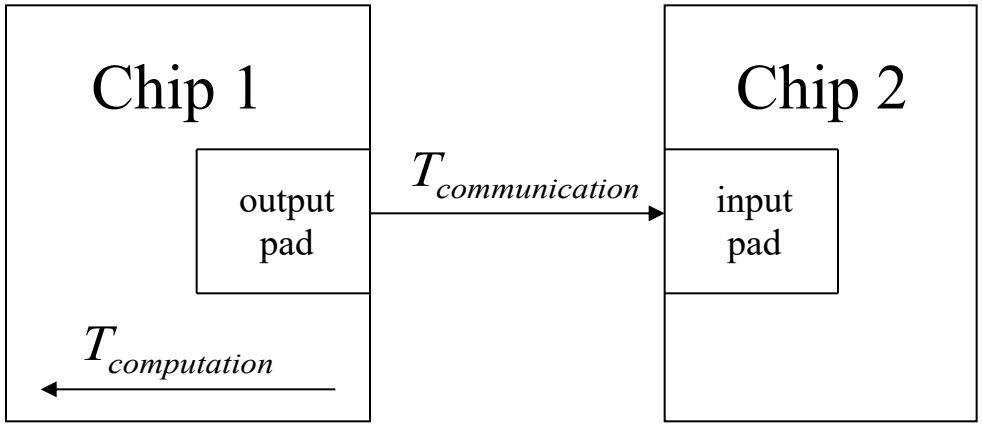
$$T_{clock} \geq T_M + 2T_A$$ for a 3-tap FIR filter

$$T_{iteration} = T_{sample} = \frac{T_{clock}}{L}$$

  - So, it is important to understand that in a parallel system $T_{sample} \neq T_{clock}$, whereas in a pipelined system $T_{sample} = T_{clock}$

Parallel Processing (cont'd)

- **Why use parallel processing when pipelining can be used equally well?**
  - **Consider the following chip set: when the critical path is less than the I/O bound (output-pad delay plus input-pad delay and the wire delay between the two chips), we say this system is *communication bounded***
  - **So, we know that pipelining can be used only to the extent such that the critical path computation time is limited by the communication (or I/O) bound. Once this is reached, pipelining can no longer increase the speed**



Parallel Processing (cont'd)

- So, in such cases, pipelining can be combined with parallel processing to further increase the speed of the DSP system

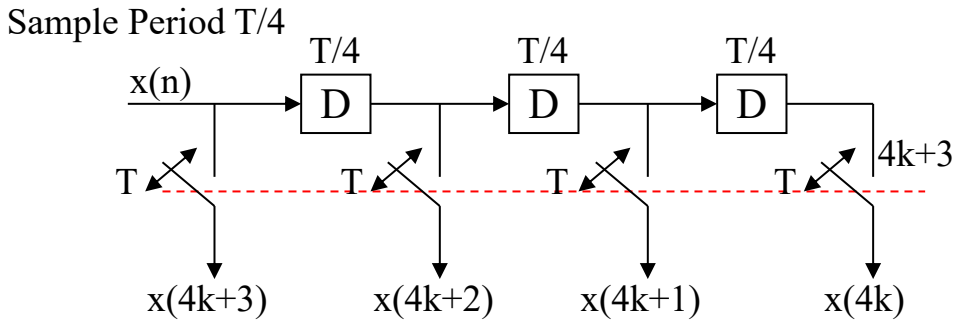
- By combining parallel processing (block size: L) and pipelining (pipelining stage: M), the sample period can be reduced to:

$$T_{iteration} = T_{sample} = \frac{T_{clock}}{L \cdot M}$$

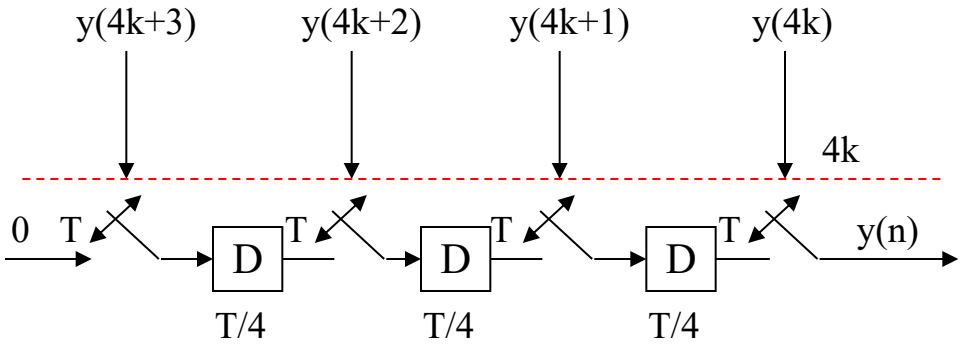
- Parallel processing can also be used for reduction of power consumption while using slow clocks

Parallel Processing- Converters

- **A serial-to-parallel converter**



- **A parallel-to-serial converter**



Pipelining and Parallel Processing for Low Power

Power Dissipation

Two measures are important

- **Peak power (sets dimensions)**

$$P_{peak} = V_{DD} \times I_{DDmax}$$

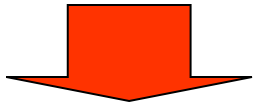
- **Average power (battery and cooling)**

$$P_{av} = \frac{V_{DD}}{T} \int_0^T i_{DD}(t)\, dt$$

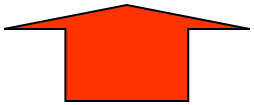
or rather ”the energy”.

Power consumption in CMOS

Gaining more importance with technology scaling



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



The one we will look at and reduce with pipelining and parallelism

CMOS Power Consumption

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

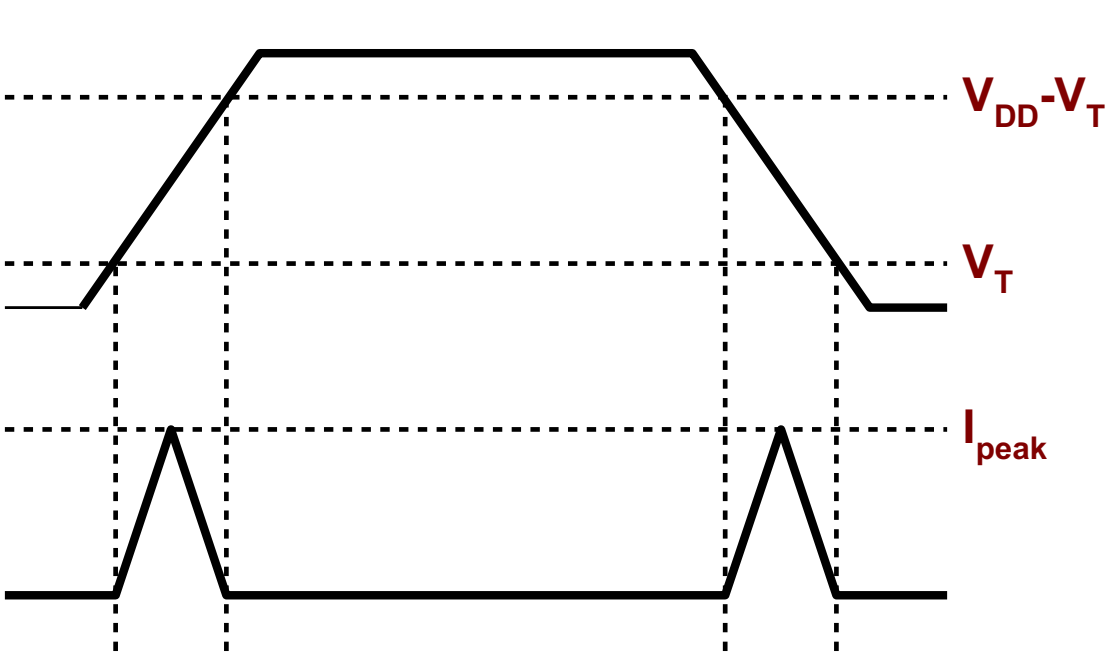
$$= \alpha f C_L V_{DD}^2 + V_{DD} I_{sc} + I_{leakage} V_{DD}$$

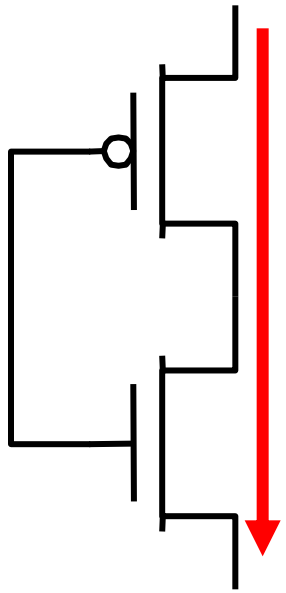
α = probability for switching

$C_L = C_{charge}$

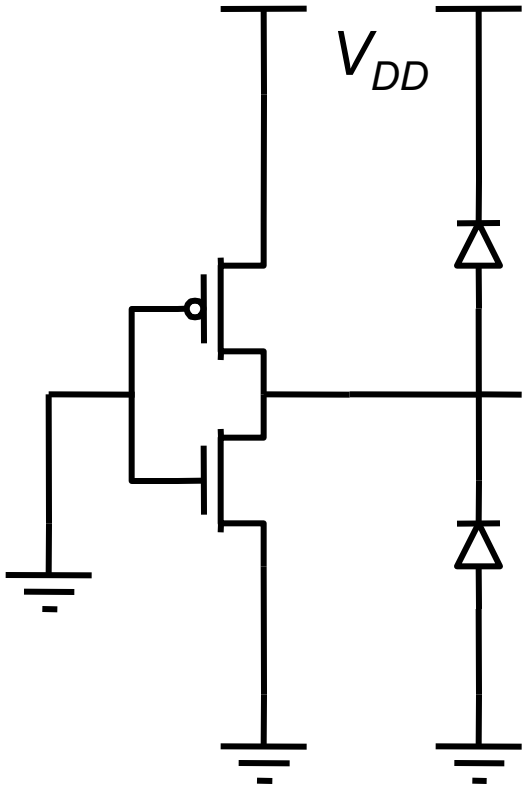
Short Circuit - Current Spikes

Current peak when both N- and PMOS are open





Static Power Consumption
due to leakage current



$$P_{stat} = I_{leakage} \times V_{DD}$$

Drain Leakage
$I_{leakage}$

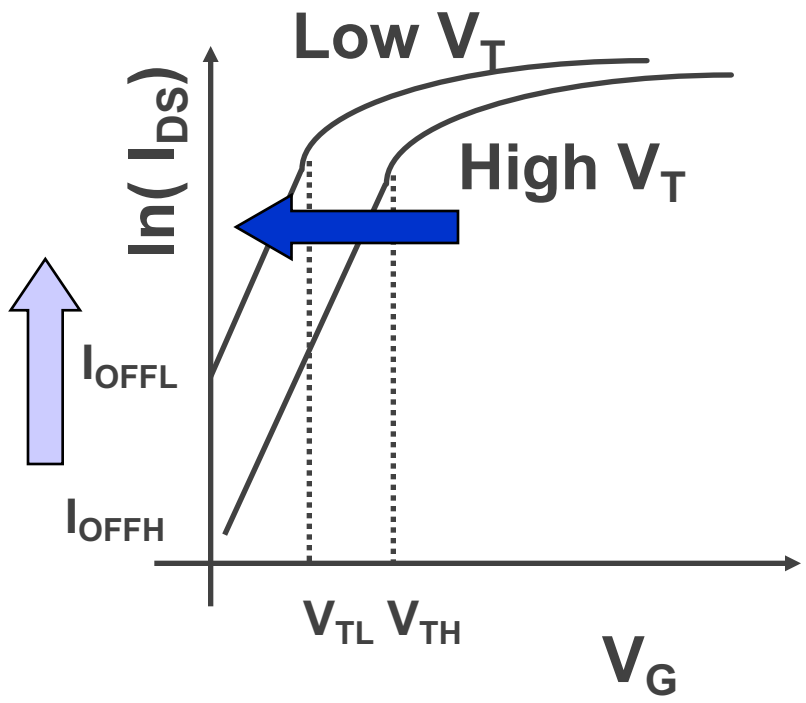
Subthreshold Current

$I_{leakage}$ increases with decreasing V_T

V_T Scaling: V_T and I_{OFF} Trade-off

Performance vs Leakage:

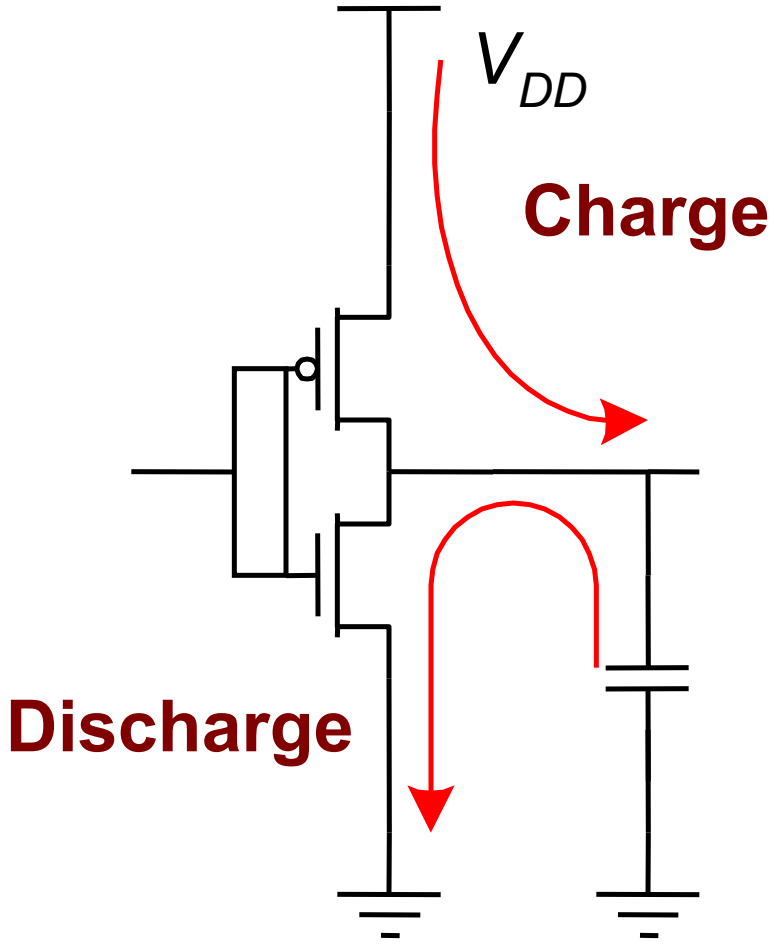
$V_T \downarrow \Rightarrow I_{OFF} \uparrow$



⇨ As V_T decreases, sub-threshold leakage increases

Dynamic Power Consumption

Assumed that $V_{swing}=V_{DD}$



Energy charged in a capacitor

$$E_C = CV^2/2 = C_L V_{DD}^2/2$$

Energy E_c is also discharged, i.e.

$$E_{tot} = C_L V_{DD}^2$$

Power consumption

$$P = C_L V_{DD}^2 f \ (=C_{tot} V_0^2 f)$$

Since squared most efficient to reduce P

Reduce...

Capacitances

- **Transistor/Gate C**
- **Load C**
- **Interconnects, more and more important**
- **External**

Activity

Frequency

Power supply – squared so most efficient

...without reducing performance?.

Propagation Delay in CMOS

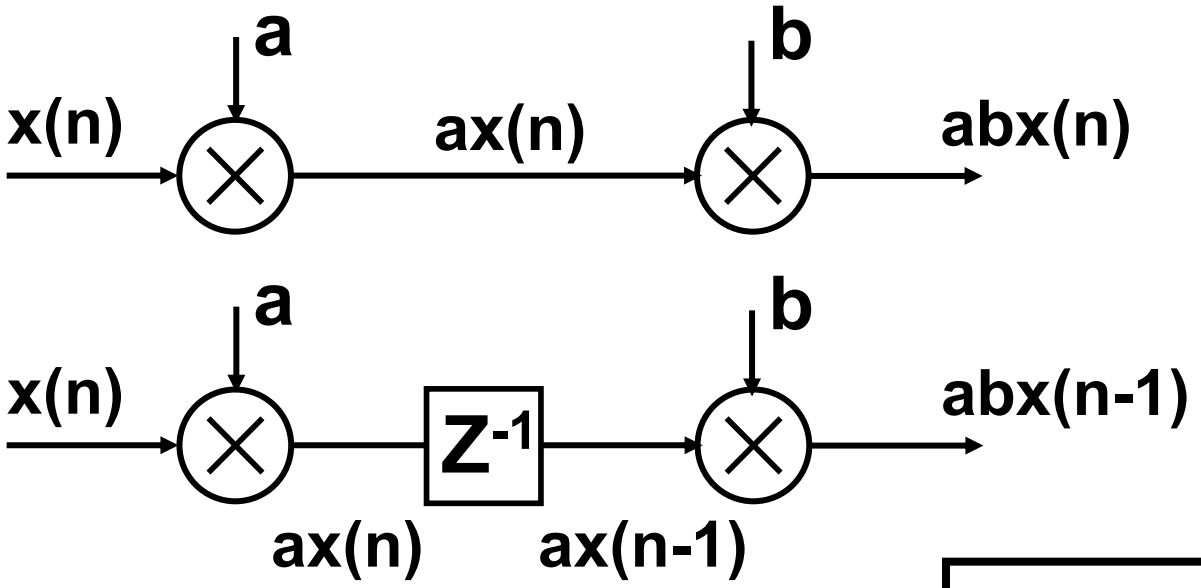
$$T_{pd} = \frac{C_L \cdot V_{DD}}{k(V_{DD} - V_T)^2}, \quad k \propto \mu, \frac{W}{L}, C_{ox}$$

if $V_{DD} >> V_T$

$$T_{pd} = \frac{C_L}{kV_{DD}} = \frac{1}{f}$$

f proportional to V_{DD}

Pipelining, *power consumption*



$$P = f\, C\, V_{DD}^{\,2}$$

Critical path cut in half

- double f ⇨ double throughput

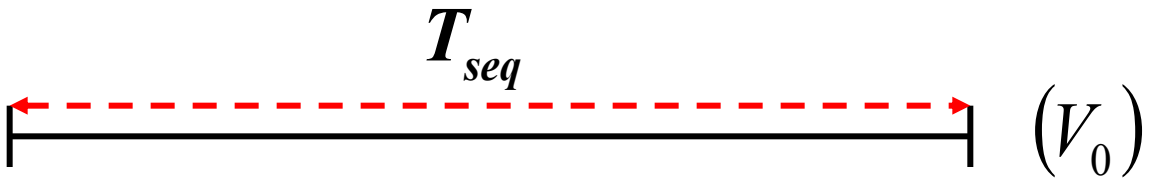
or

- same f ⇨ double time for mult ⇨ reduced V_{DD}

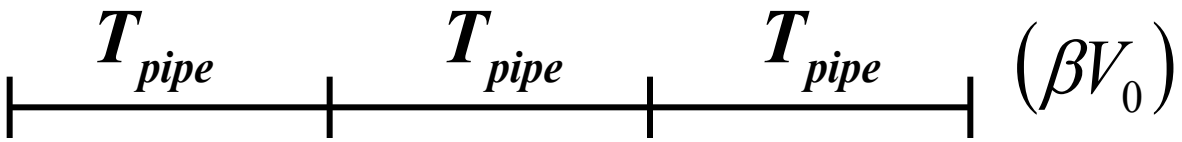
Reduction of Critical Path

Propagation delay of the original filter and the pipelined filter

Sequential (critical path):



Pipelined: (critical path when M=3)



Pipelining, *power consumption*

– The power consumption in original architecture

$$P_{seq} = f\, C_L\, V_{DD}^{\,2}$$

– The supply voltage can be reduced to βV_{DD} , $(0<\beta<1)$.
Hence, the power consumption of the pipelined filter is:

$$P_{pipe} = f\, C_L \left(\beta V_{DD}\right)^2 = \beta^2 P_{seq}$$

Pipelining

Propagation delays of the sequential and the pipelined architecture:

$$T_{seq} = \frac{C_L \cdot V_{DD}}{k(V_{DD} - V_T)^2}, \quad T_{pip} = \frac{(C_L / M) \cdot \beta V_{DD}}{k(\beta V_{DD} - V_T)^2}$$

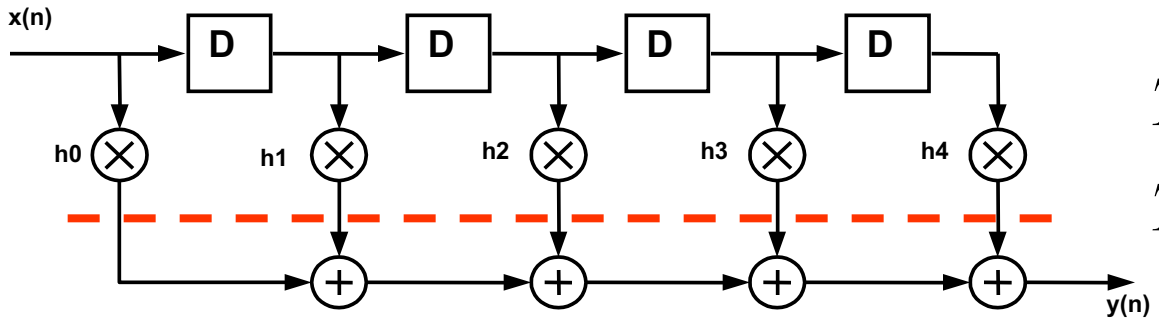
The capacitance in each stage has been reduced.

Since the same f is maintained ⇨ $T_{seq} = T_{pipe}$

$$M(\beta V_{DD} - V_T)^2 = \beta(V_{DD} - V_T)^2$$

Pipelining

$$M(\beta V_{DD} - V_T)^2 = \beta(V_{DD} - V_T)^2$$



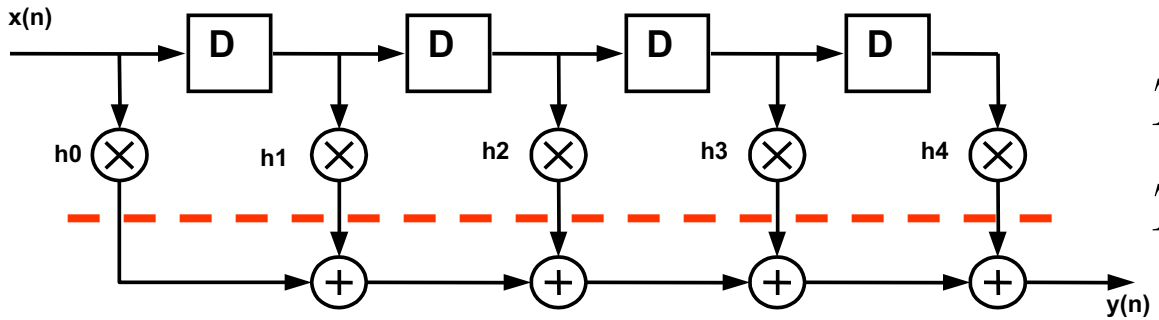
$$Tadd = 1$$

$$Tmult = 3$$

$$M = ?$$

Pipelining

$$M(\beta V_{DD} - V_T)^2 = \beta(V_{DD} - V_T)^2$$



$$Tadd = 1$$

$$Tmult = 3$$

$$T_{original} = T_{mult} + 4T_{add} = 7t.u.$$

$$T_{Pipelined} = 4T_{add} = 4t.u.$$

$$\Rightarrow \quad M = \frac{7}{4}$$

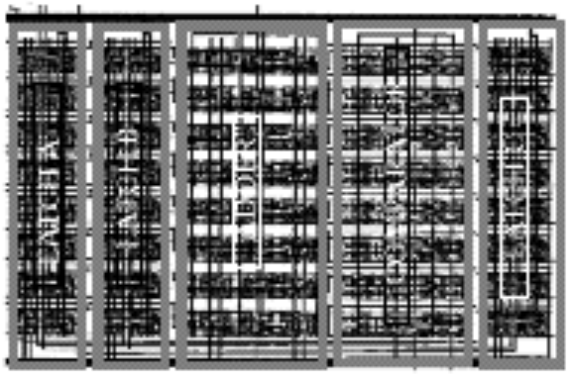
If M doesn't divide the critical path evenly you have to use the "real M",

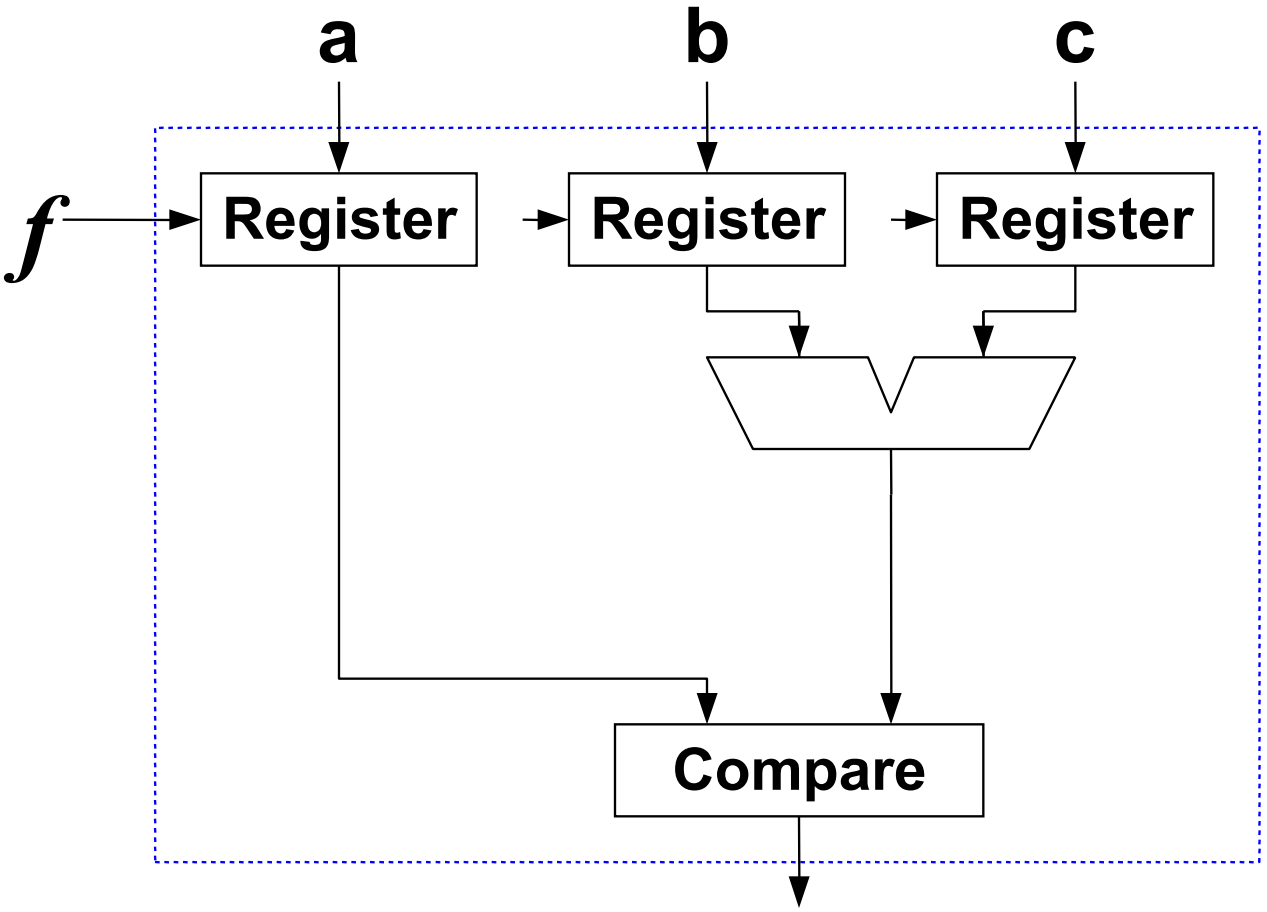
i.e. how big is the actual reduction of the critical path.

Example: simple datapath

$$P = f\, C\, V^2$$

C = the total switched capacitance

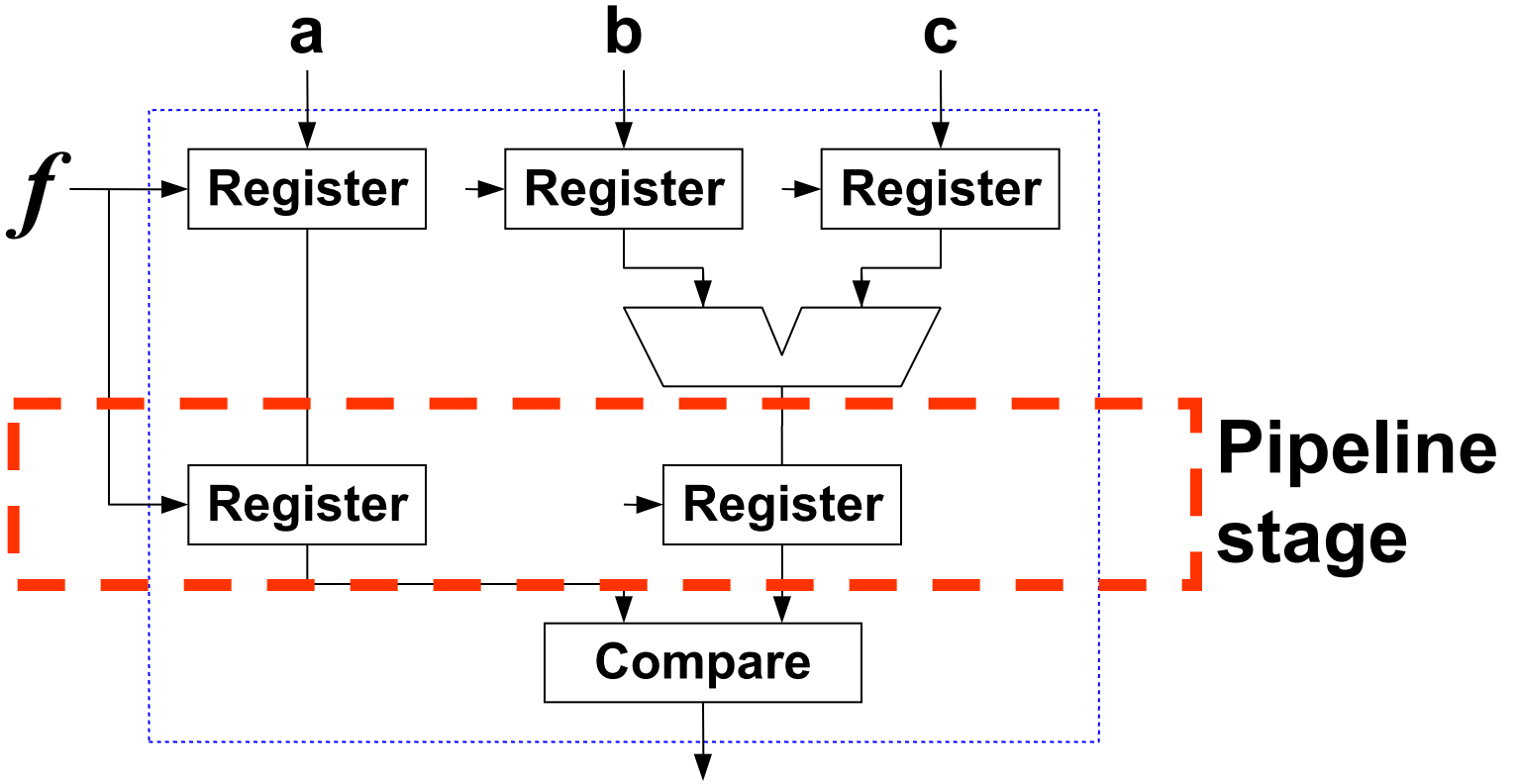




Pipelining

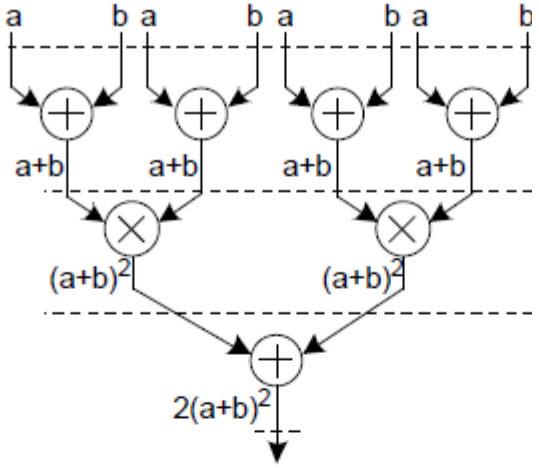
Increased C due to register

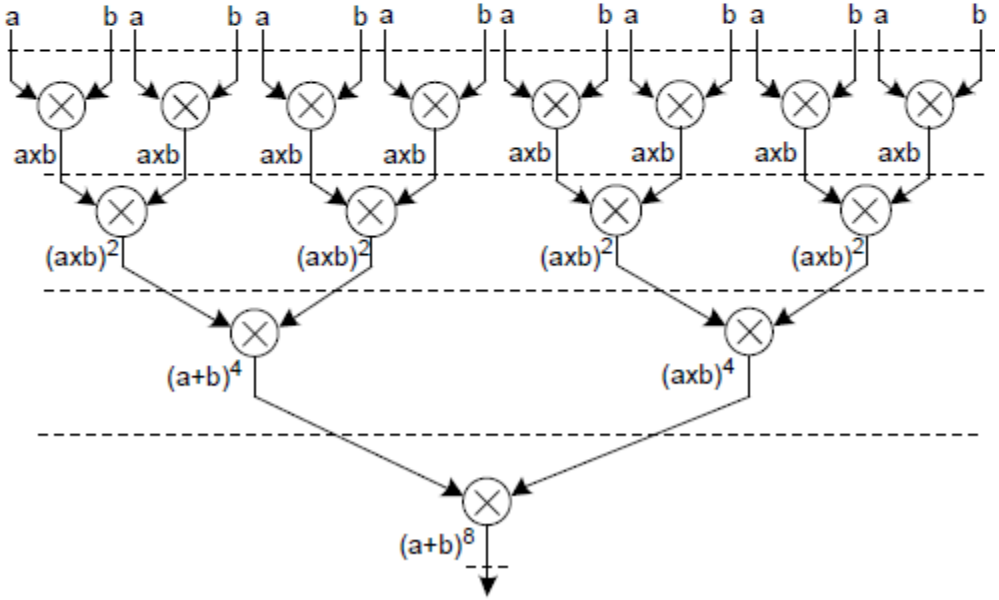
$$P_{pipe} = f \times 1.1C \times (0.58V)^2 = 0.37P$$



Power in sub-V_T with Pipelining

AMA = Addition-Multiplication-Addition





MT = Multiplication Tree

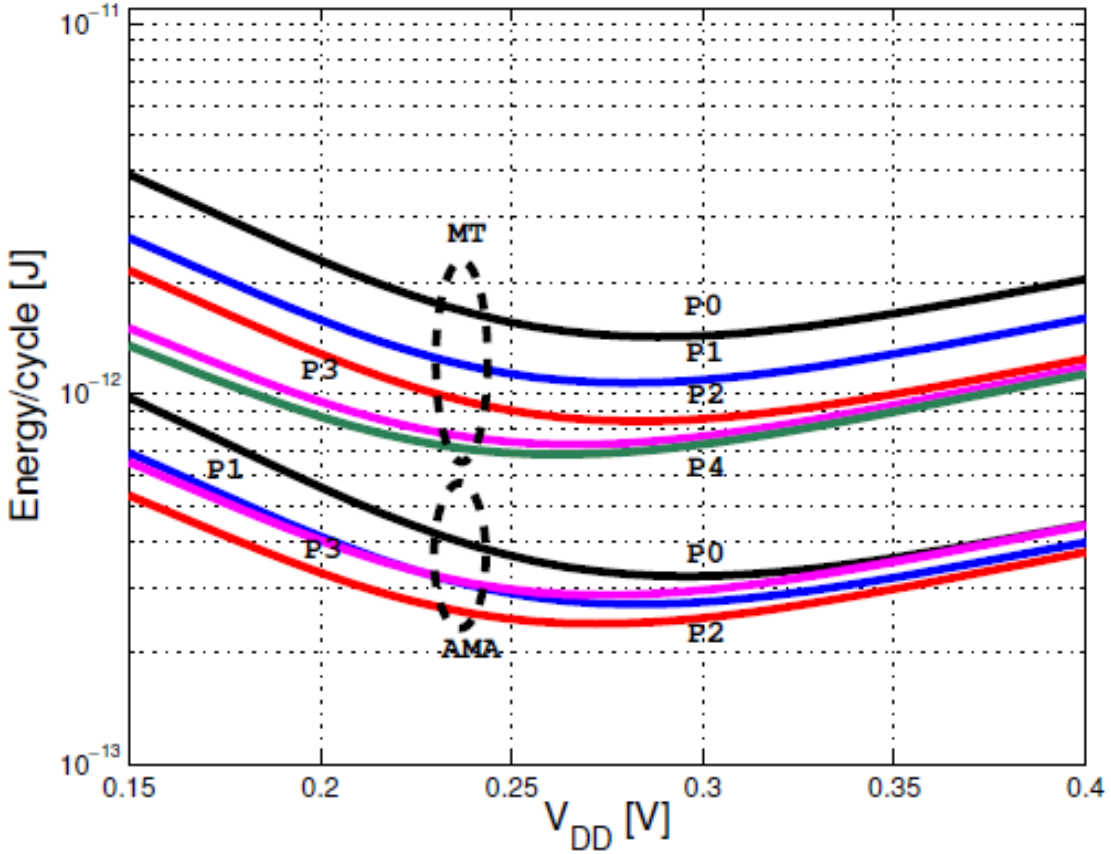
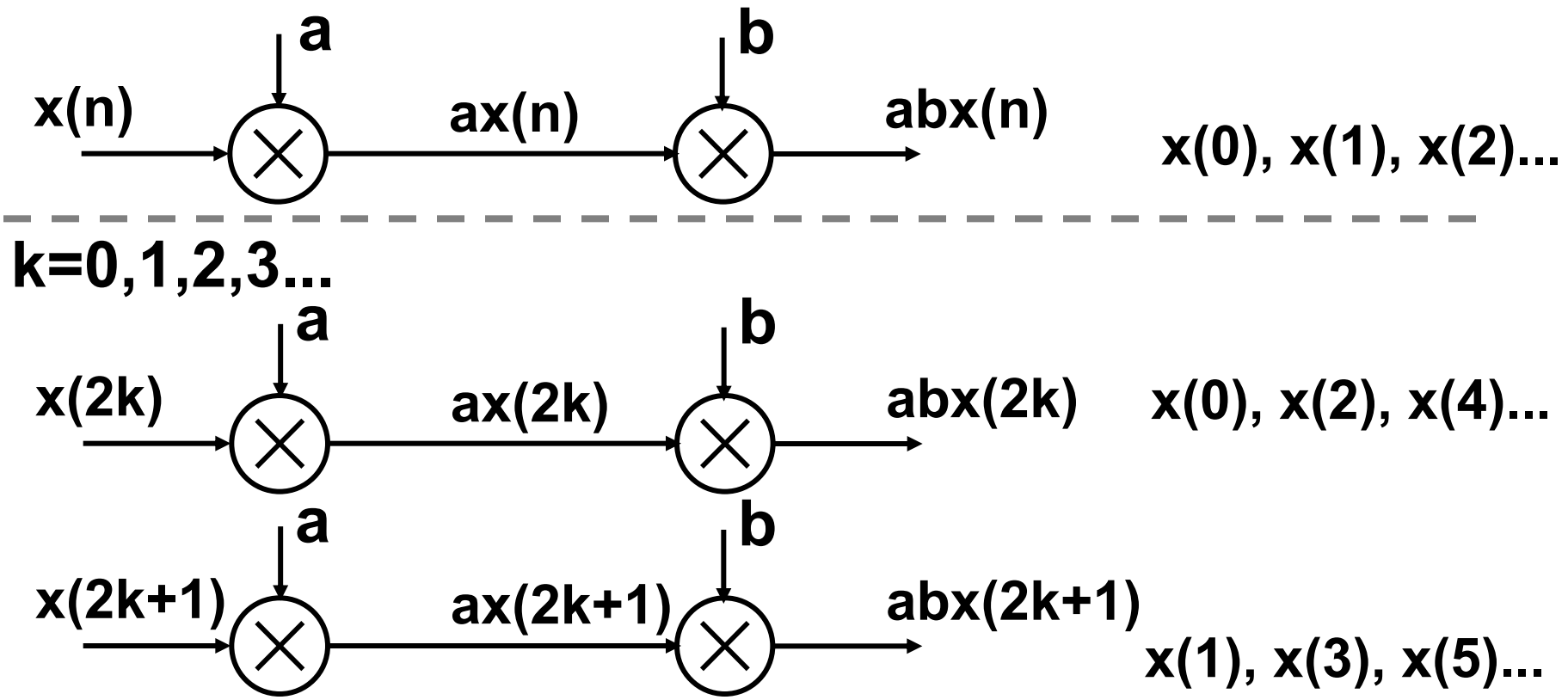


Figure 6.6.: Energy per cycle vs V_{DD} @max freq. for the two designs corresponding to the pipelines

Power decreases up to a point and then increase due to increased overhead.

From PhD thesis *Design Space Exploration of Digital Circuits for Ultra-low Energy Dissipation* by Yasser Sherazi, January 2014

Parallel Processing



Two samples are processed in parallel

- same f but two samples ⇨ double throughput

or

- reduce f and two samples ⇨ same throughput and reduced V_{DD}

Parallel Processing for Low Power

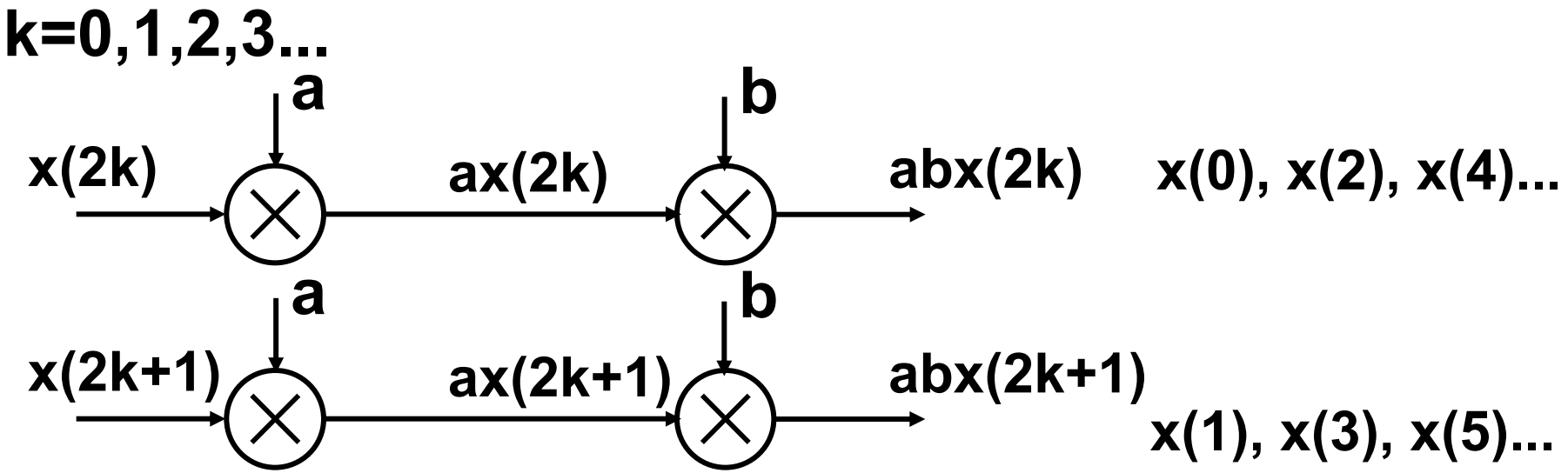
– Total capacitance, C, is increased by L

– To maintain the same sample rate

f is reduced by $1/L$

– f reduced ⇨ V_{DD} can be reduced

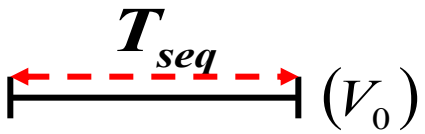
Parallel Processing



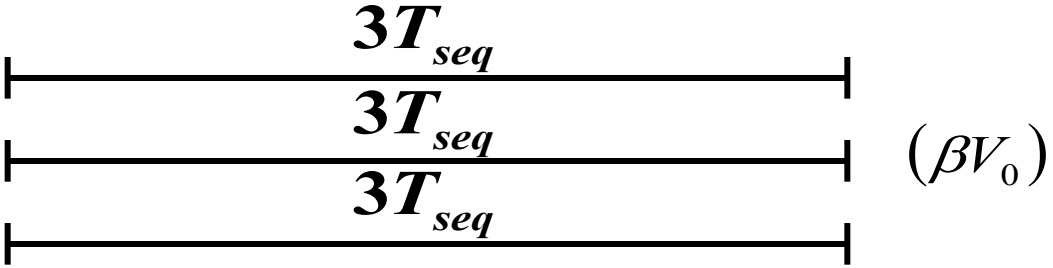
$$P_{para} = \cancel{L} C_L \frac{f}{\cancel{L}} (\beta V_{DD})^2 =$$

Parallel Processing for Low Power

Sequential (critical path):



Parallel: (critical path when L=3)



Propagation delay of the L-parallel system is given by

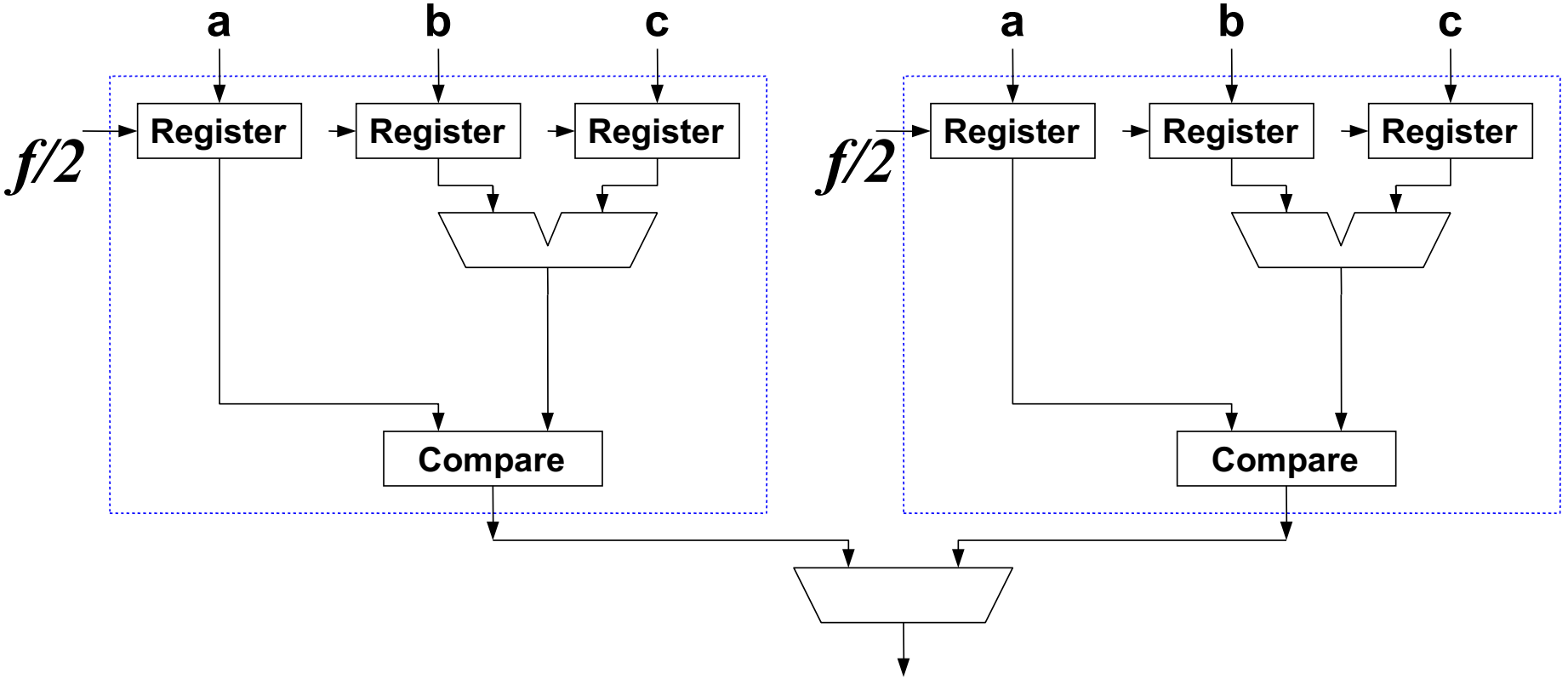
$$T_{par} = L \cdot T_{seq} \Rightarrow \frac{C_L \cdot \beta V_{DD}}{k(\beta V_{DD} - V_T)^2} = L \frac{C_L \cdot V_{DD}}{k(V_{DD} - V_T)^2}$$

$$L(\beta V_{DD} - V_T)^2 = \beta (V_{DD} - V_T)^2$$

Parallel Processing

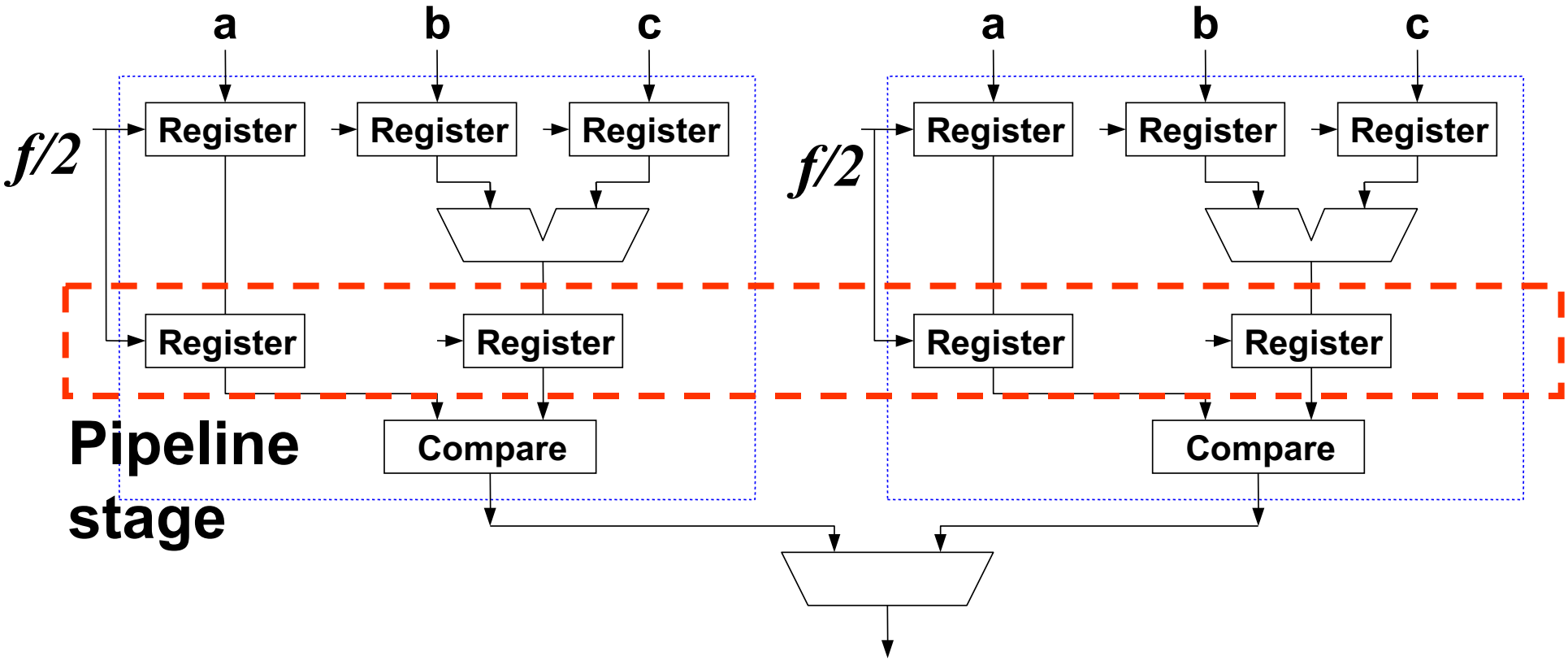
2.15 due to mux β

$$P_{par} = 0.5f \times 2.15C \times (0.58V)^2 = 0.36P$$



Parallel Processing and Pipelining

$$P_{par,pipe} = 0.5f \times 2.35C \times (0.4V)^2 = 0.19P$$



Summary - The Effect of Pipelining and Parallelization

We have seen that the power can be significantly reduced in a system using pipelining and parallelization.

Pipelining only

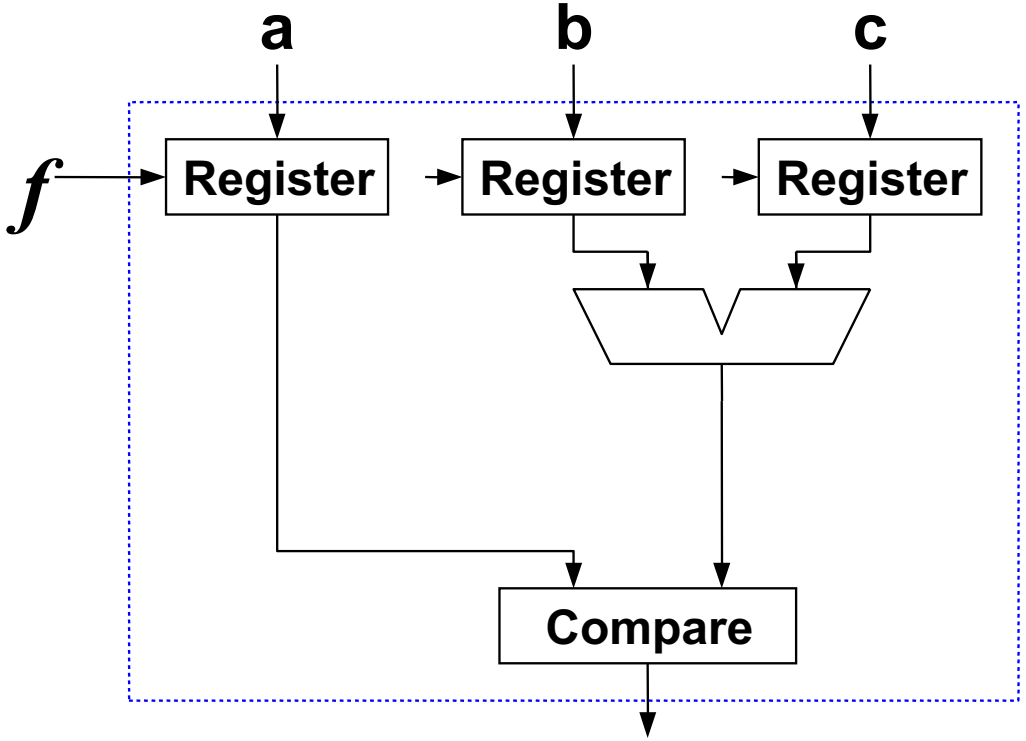
$$P_{pipe} = f \times 1.1C \times (0.58V)^2 = 0.37P$$

Parallelization only

$$P_{par} = 0.5f \times 2.15C \times (0.58V)^2 = 0.36P$$

Pipe- and Parallelization

$$P_{par,pipe} = 0.5f \times 2.35C \times (0.4V)^2 = 0.19P$$



End of Lecture 4