

*Lecture*

# 4

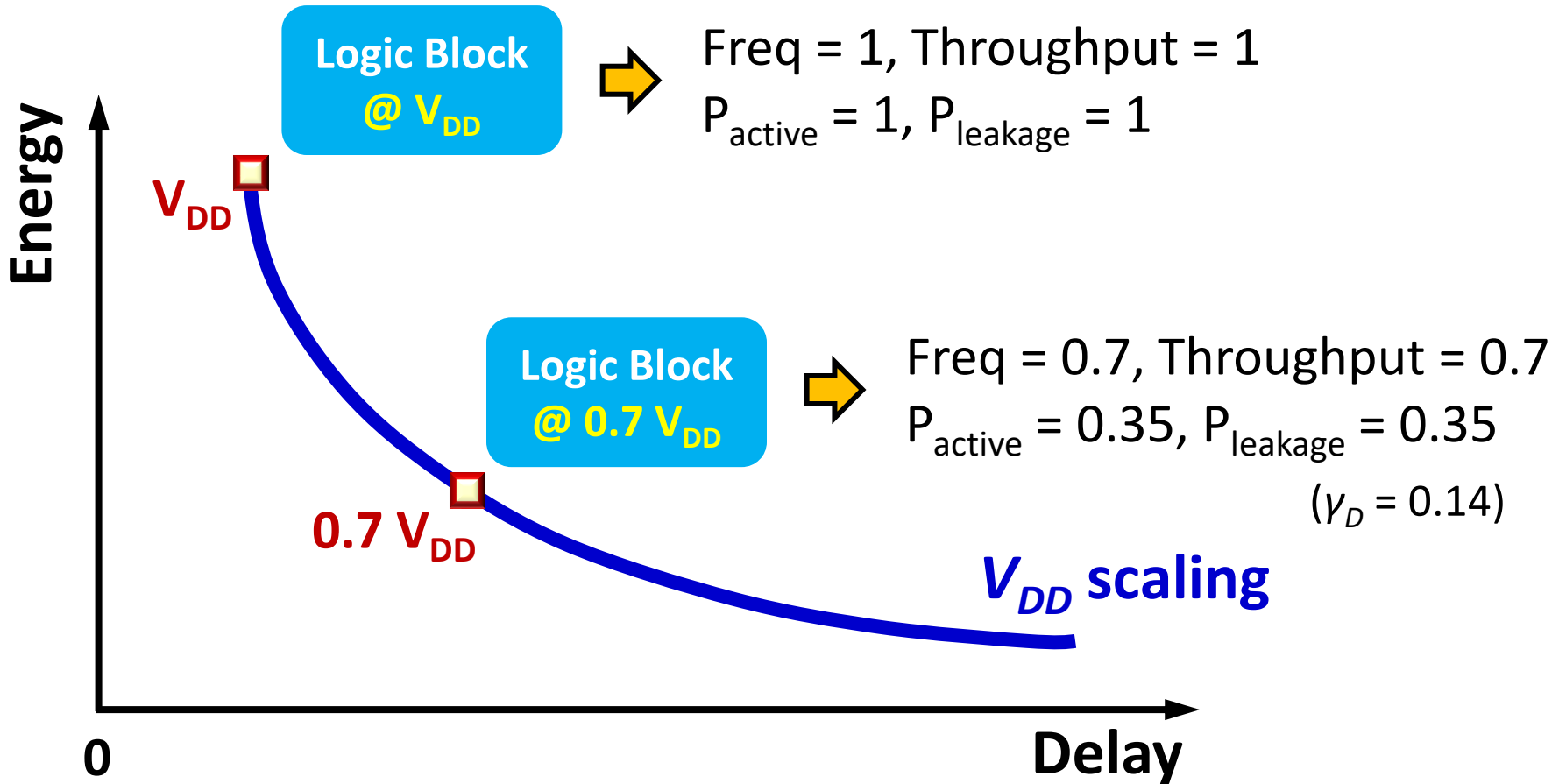
ECE216B

# Architectural Techniques

**Prof. Dejan Marković**

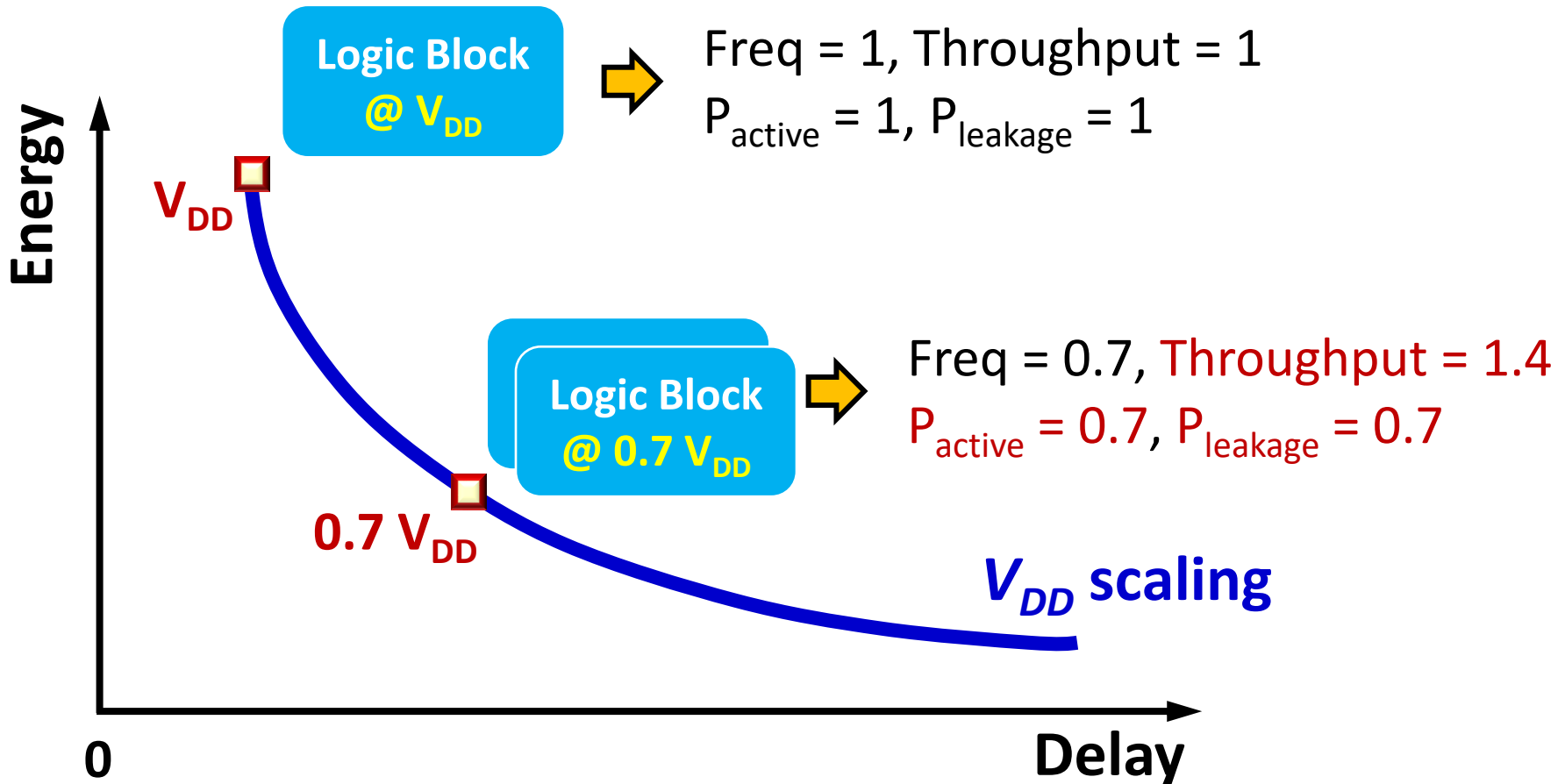
ee216b@gmail.com

# Energy-Delay Tradeoff for a Logic Block



# Parallelism: Two Logic Blocks

Lower  $V_{DD}$  & parallelism: higher throughput, lower power



# Agenda

---

- **Understanding of main architectural techniques**
  - Feed-forward algorithms
  - Recursive algorithms
- **Architecture Exploration**
  - **Energy-Delay:** circuit
  - **Energy-Area:** architecture
- **Design Guidelines**

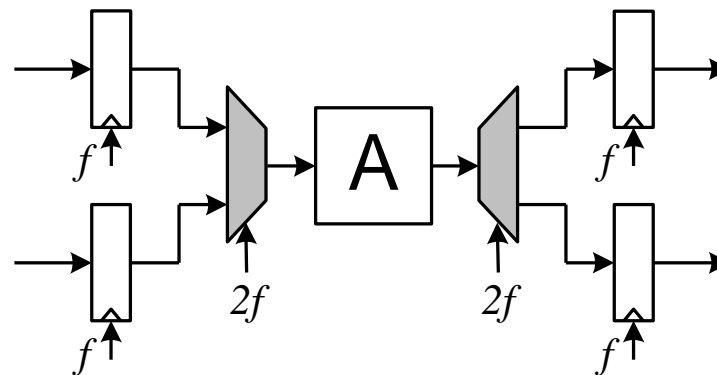
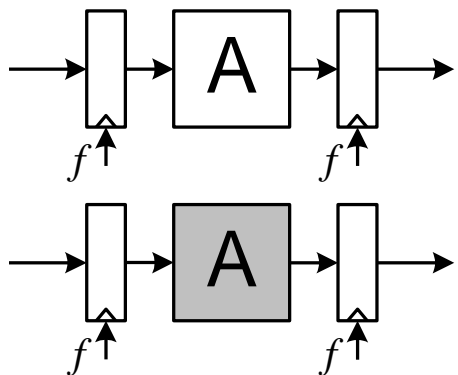
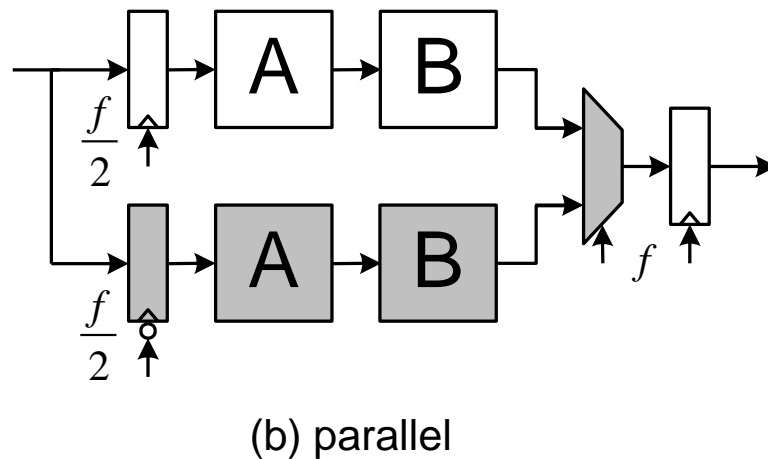
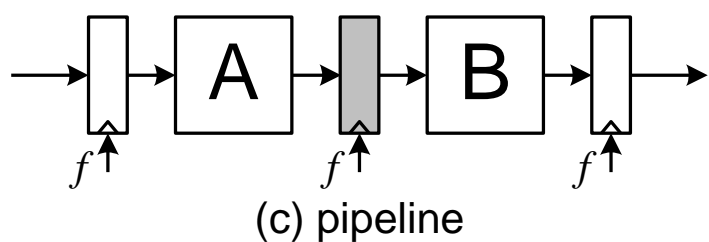
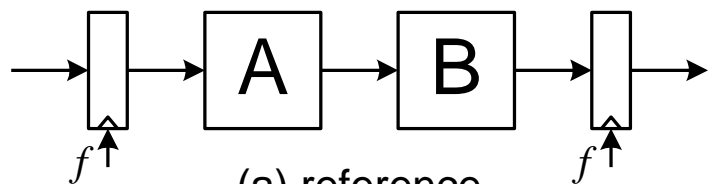
# Architectural Techniques

---

- ➔ • **Feed-forward algorithms**
  - Parallelism
  - Pipelining
  - Time multiplexing
- **Recursive algorithms**
  - Interleaving
  - Folding

# Parallelism, Pipelining, Time Multiplexing

Shaded blocks represent overhead

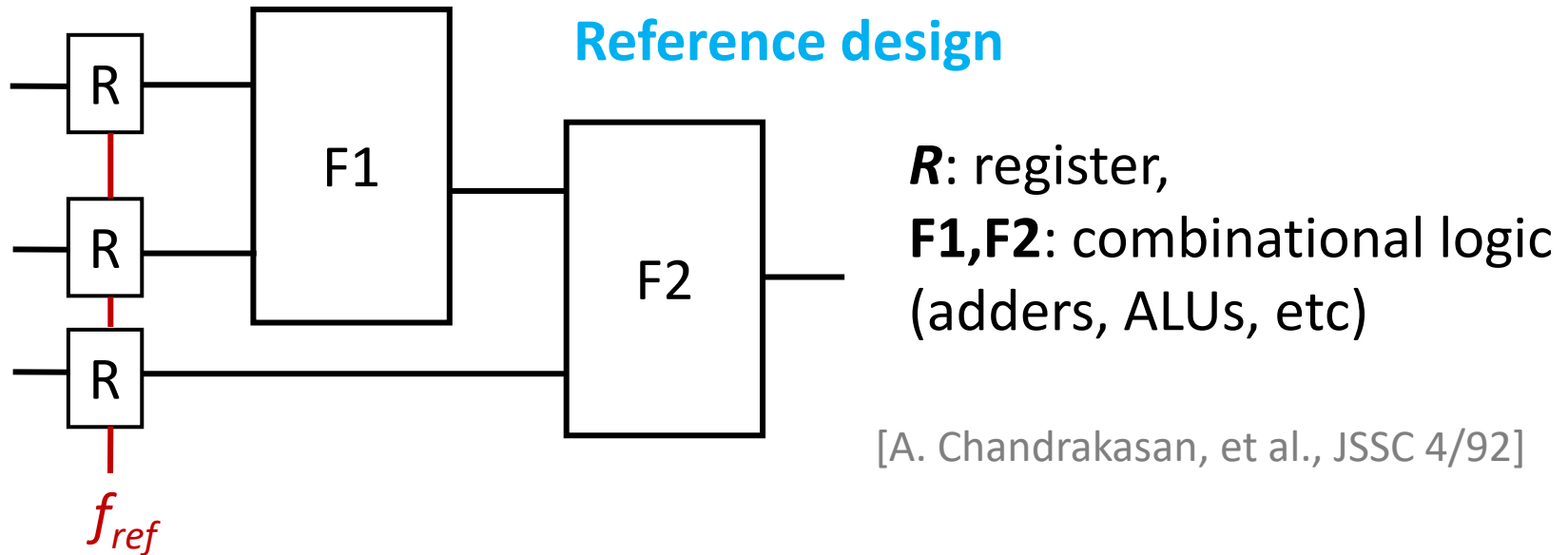


# Reducing the Supply Voltage

(while maintaining performance)

## Concurrency:

trading off clock frequency versus area to reduce power

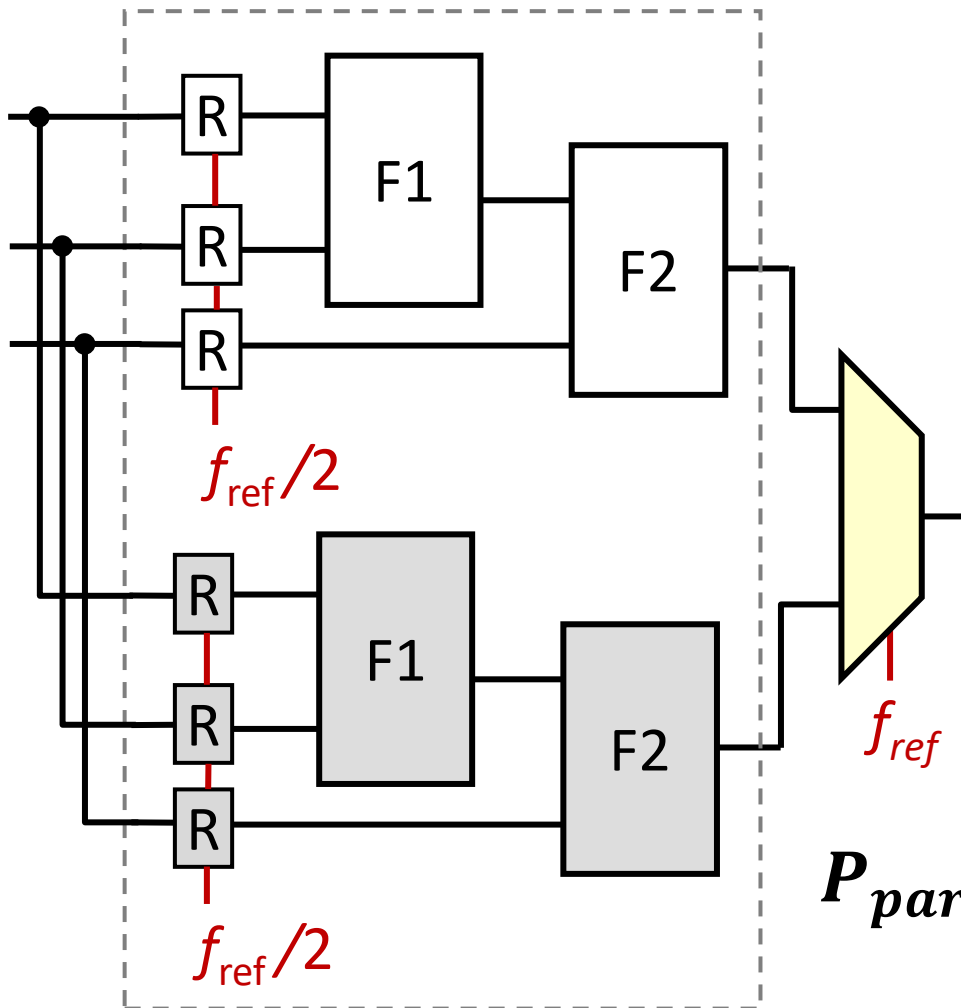


$$P_{ref} = C_{ref} \cdot V_{dd,ref}^2 \cdot f_{ref}$$

$C_{ref}$ : average switching capacitance

# A Parallel Implementation

- Slower logic = **lower**  $V_{DD}$  = lower power



$$f_{par} = \frac{f_{ref}}{2}$$

$$C_{par} = (2 + \alpha v_{par}) \cdot C_{ref}$$

$$V_{DD,par} = \epsilon_{par} \cdot V_{DD,ref}$$

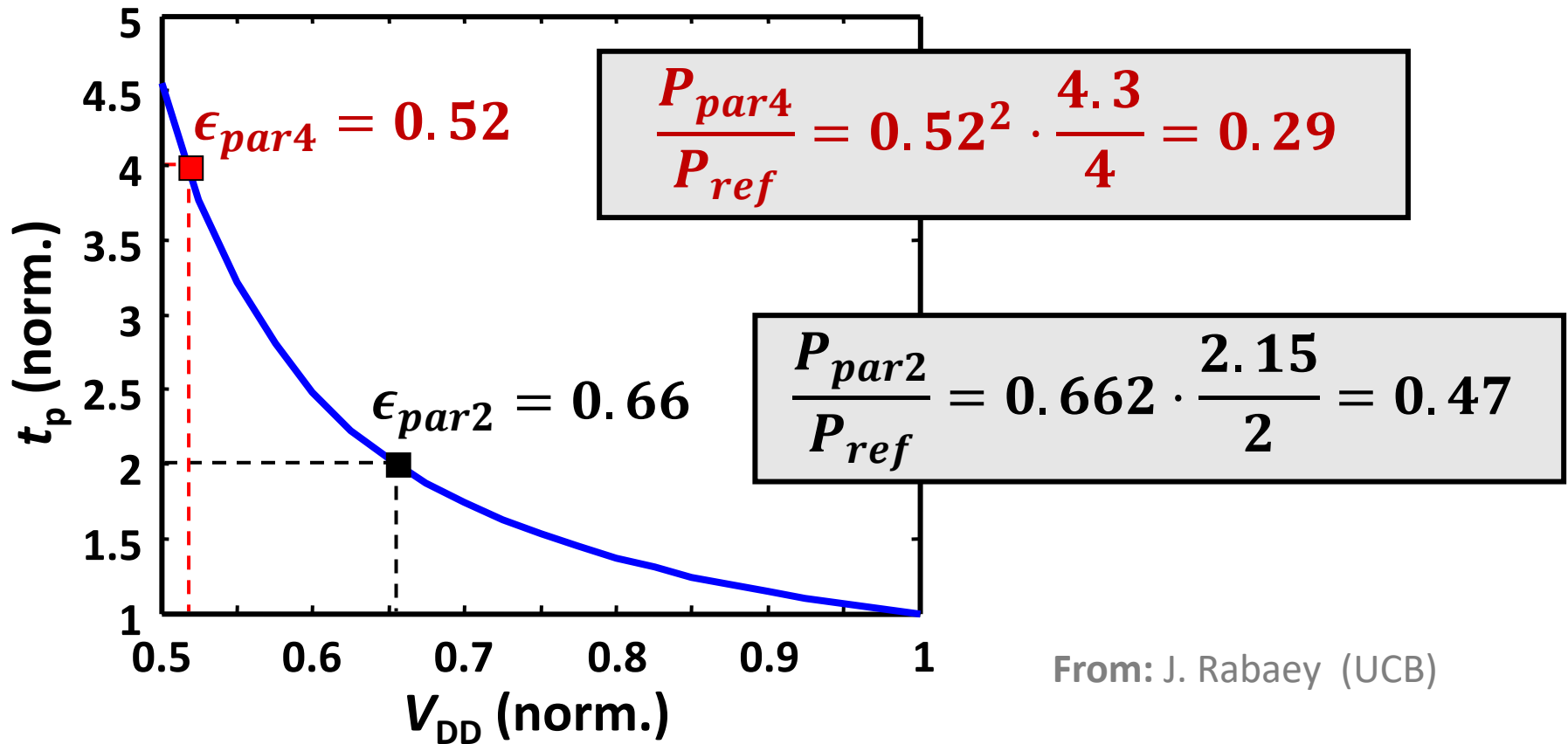
Power saving      Almost cancels

$$P_{par} = \epsilon_{par}^2 \cdot \frac{2 + \alpha v_{par}}{2} \cdot P_{ref}$$



# Parallelism Example (90nm CMOS)

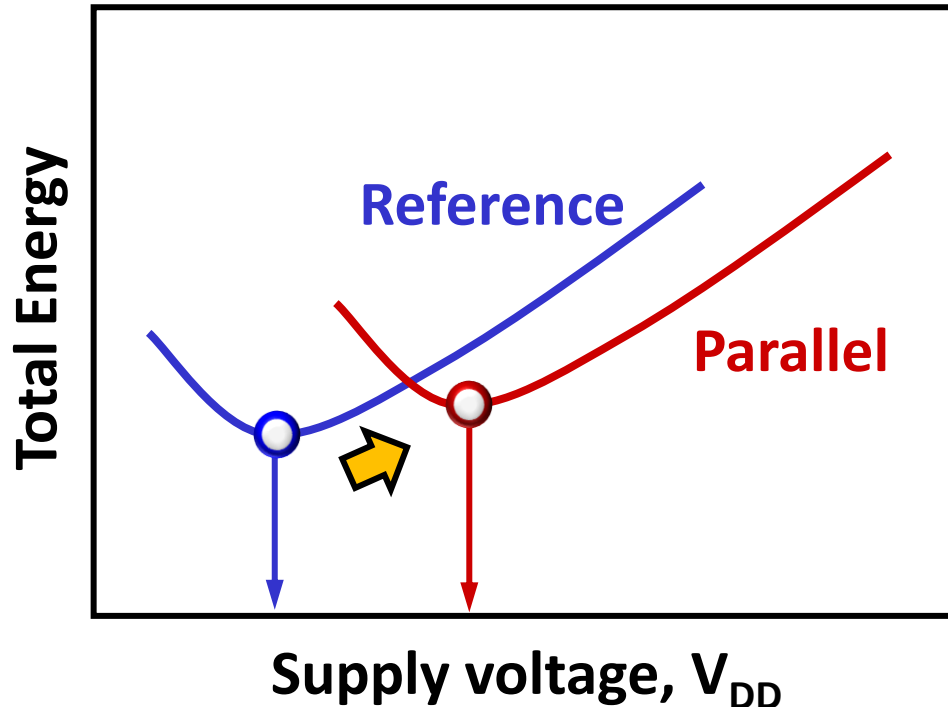
Power (assuming  $ov_{par} = 7.5\%$ )



## How many levels of parallelism?

# More Parallelism (P): Not Always Better

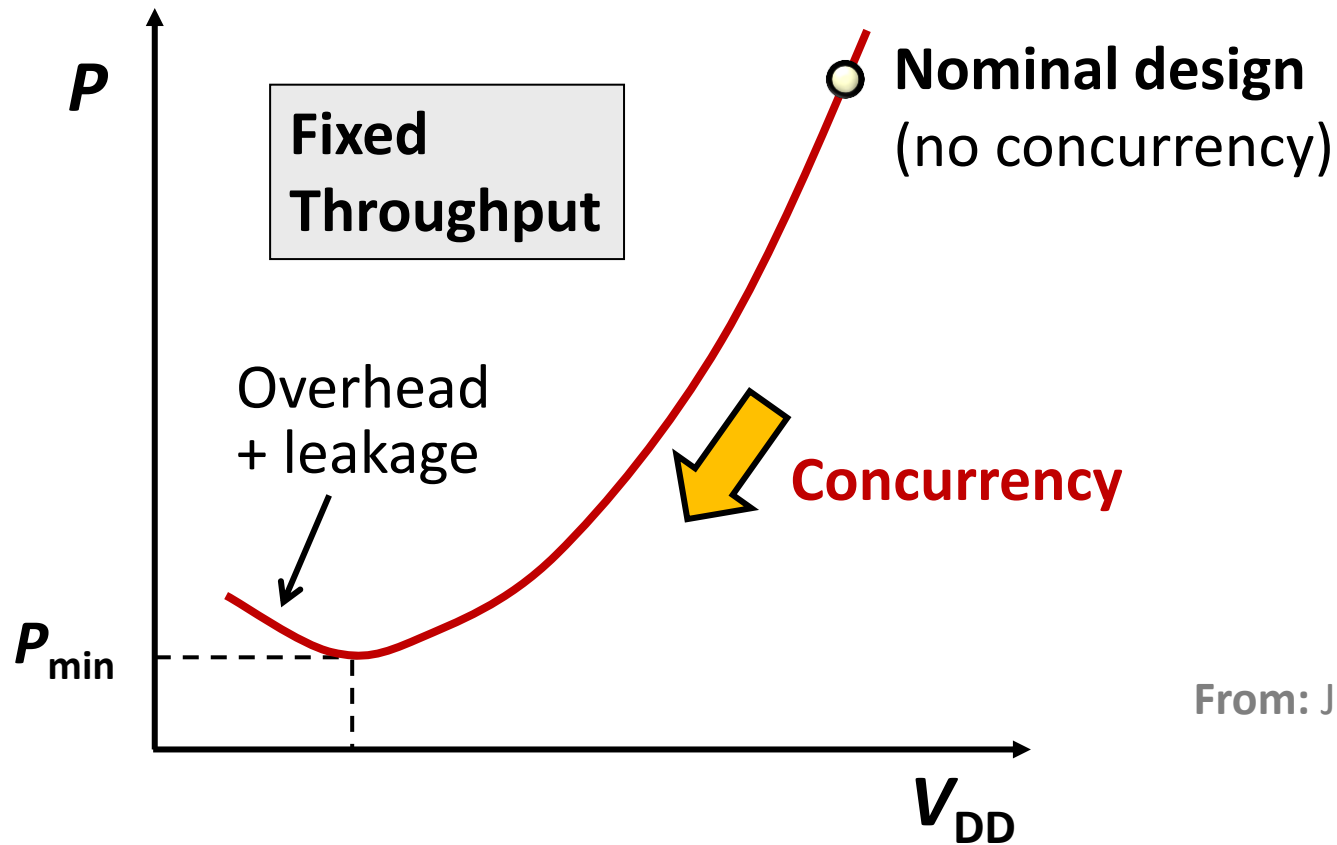
$$E_{tot} = E_{sw} + P \cdot E_{lk} + E_{overhad}$$



From:  
J. Rabaey  
(UCB)

- Leakage and overhead start to dominate at high P
- Optimal  $V_{DD}$  and min E increase with parallelism

# Increasing use of Concurrency Saturates

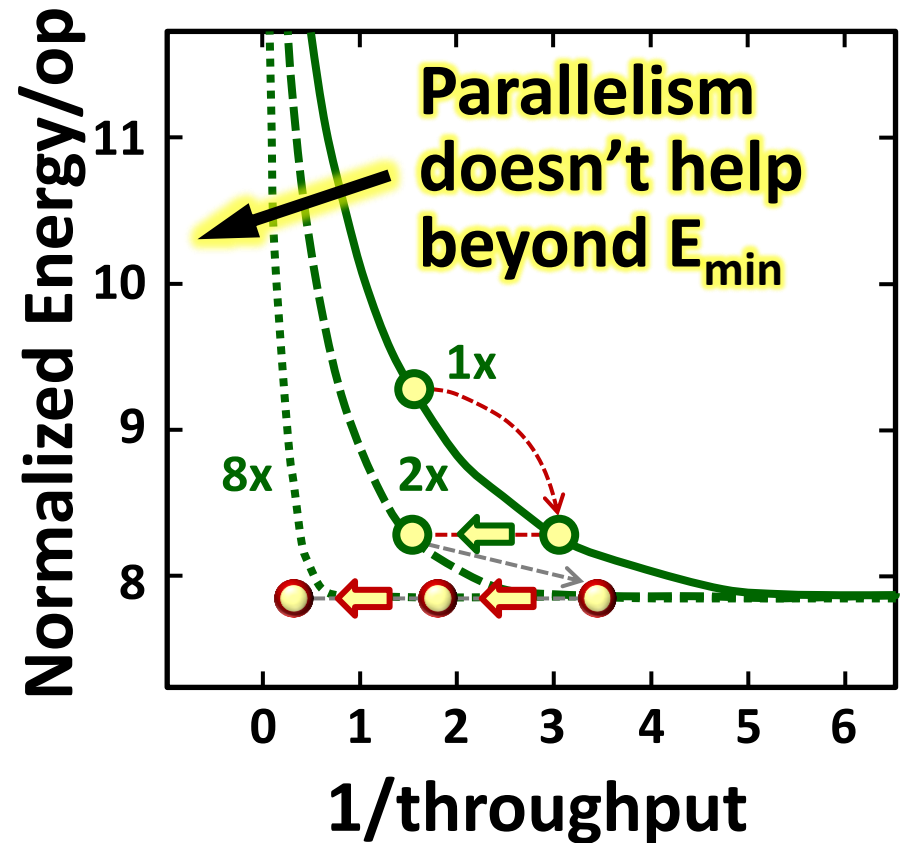
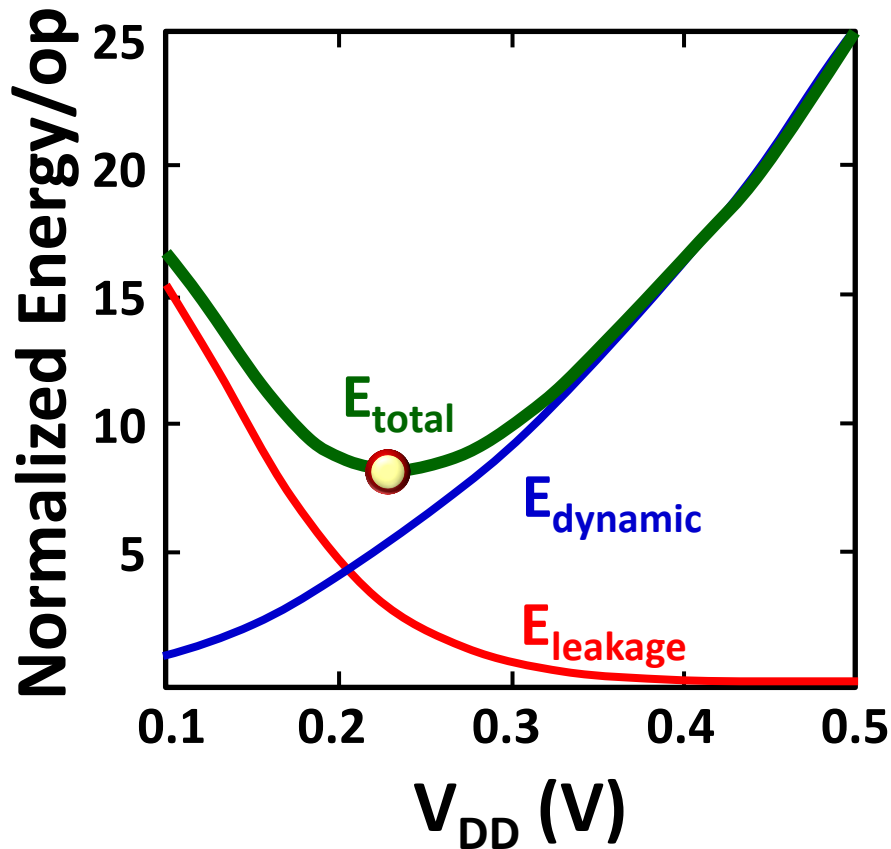


From: J. Rabaey (UCB)

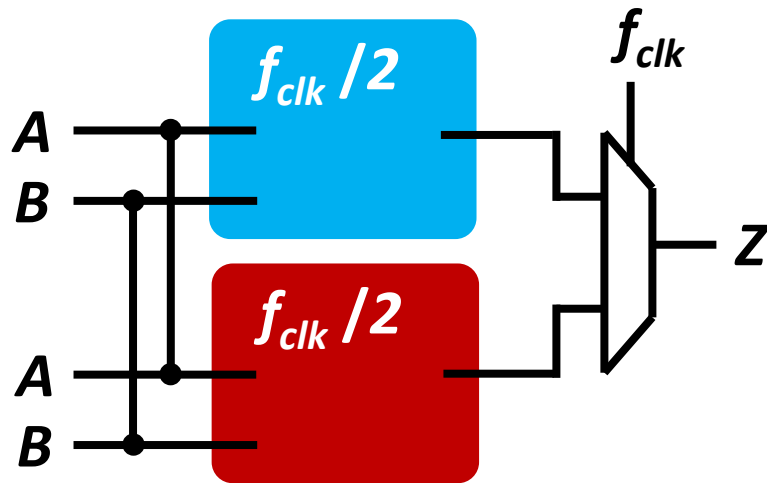
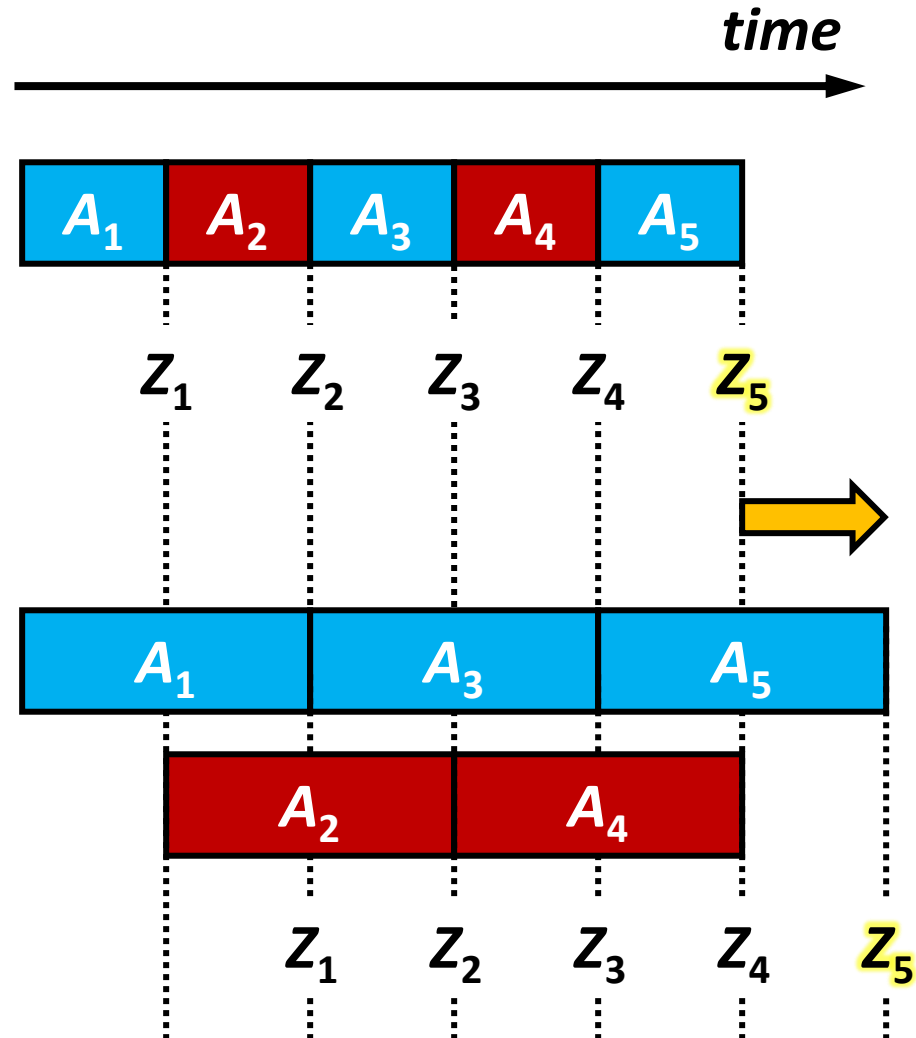
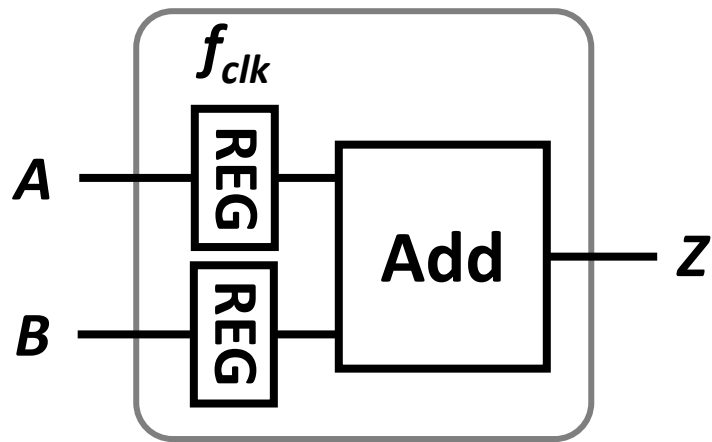
# Impact on Multi-Core: **Limited Energy/Op**

- Leakage defines minimum Energy/op for CMOS

- Cannot reduce Energy/op if operating at min E/op

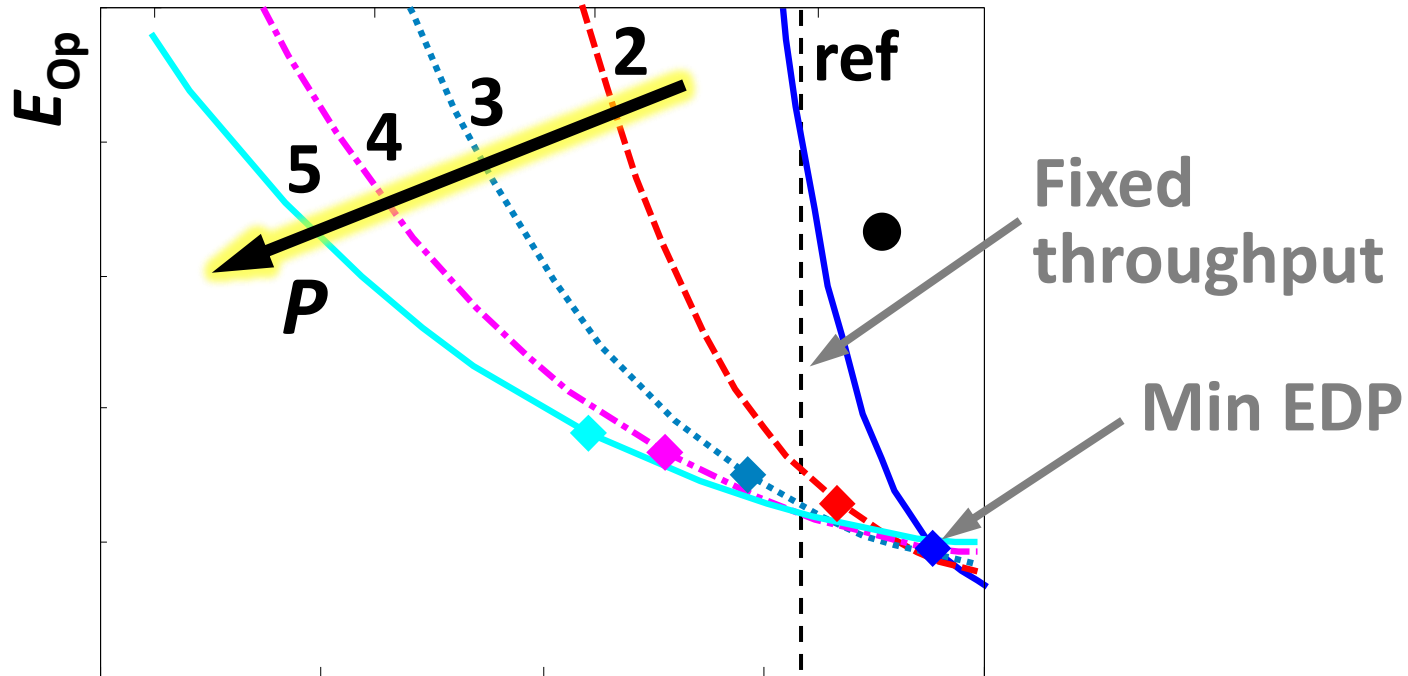


# Beware: Parallelism Adds Latency



Level of parallelism  $P = 2$

# When to Add Parallelism to Minimize E/Op?



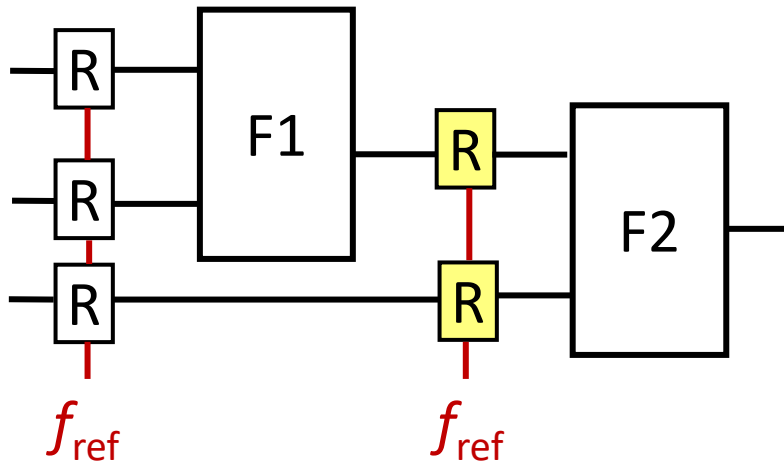
[D. Marković *et al.*, JSSC 8/04]

Delay = 1/Throughput

- Level of concurrency depends on target performance
- **Rule of thumb:** if speed exceeds MEDP, add parallelism

# A Pipelined Implementation

Shallower logic reduces required supply voltage



$$f_{pipe} = f_{ref}$$

$$C_{pipe} = (1 + ov_{pipe}) \cdot C_{ref}$$

$$V_{DD,pipe} = \epsilon_{pipe} \cdot V_{DD,ref}$$

Power  
saving

Typically  
small

$$P_{pipe} = \epsilon_{pipe}^2 \cdot (1 + ov_{pipe}) \cdot P_{ref}$$

# Comparison of Par/Pipe @ Same $V_{DD}$

---

**Parallel**  
( $ov_{par} = 7.5\%$ )

$$\frac{P_{par4}}{P_{ref}} = 0.52^2 \cdot \frac{4.3}{4} = \boxed{0.29}$$

$$\frac{P_{par2}}{P_{ref}} = 0.662 \cdot \frac{2.15}{4} = \boxed{0.47}$$

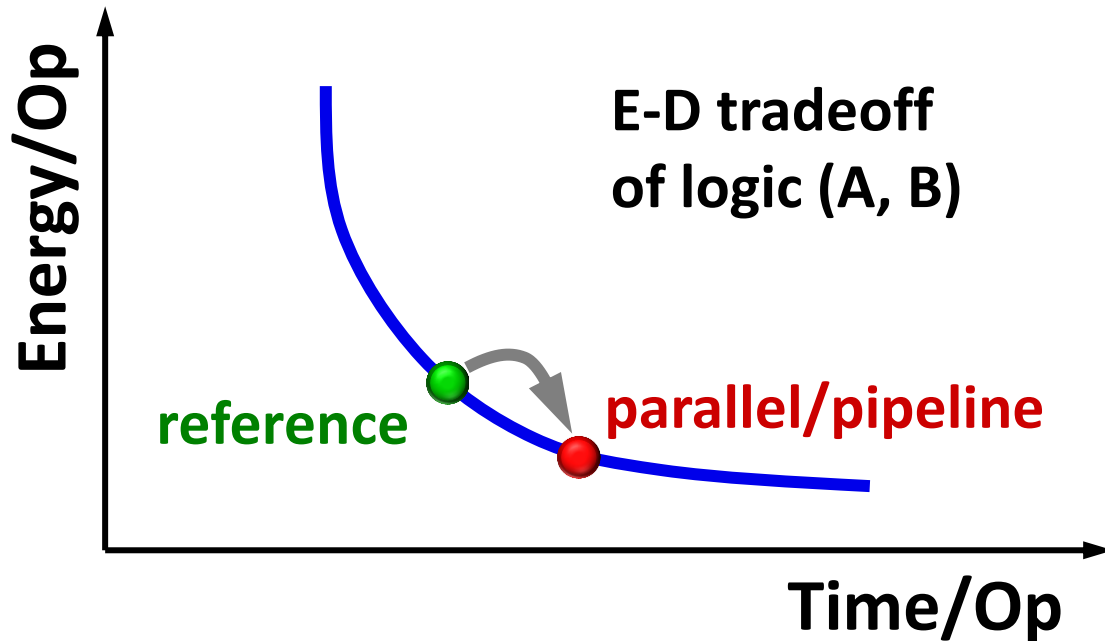
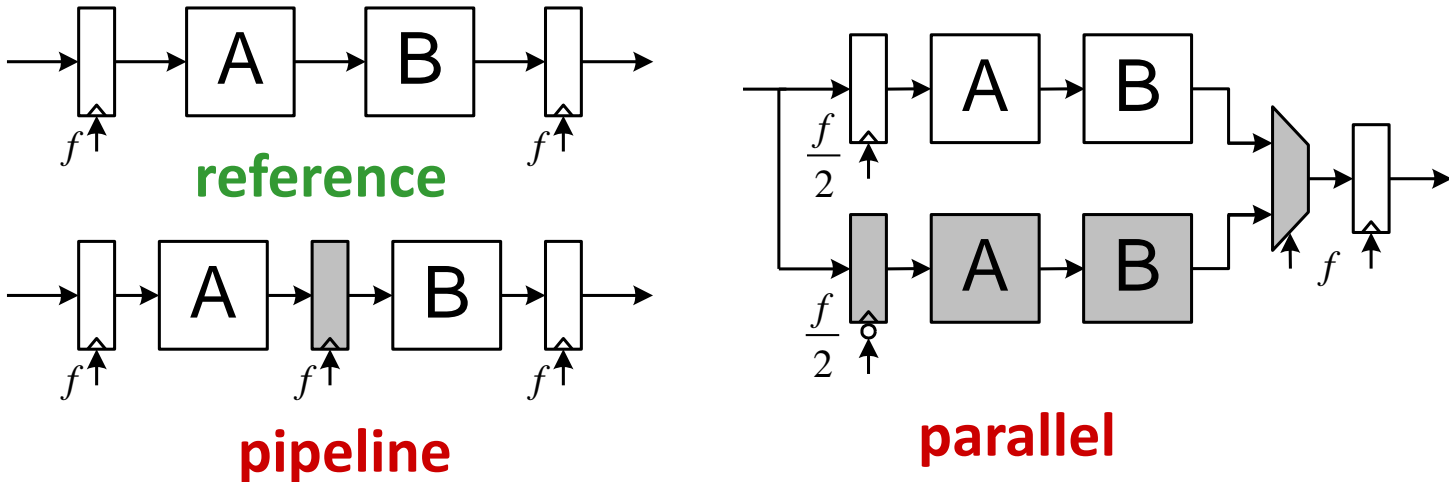
**Pipeline**  
( $ov_{pipe} = 10\%$ )

$$\frac{P_{pipe4}}{P_{ref}} = 0.52^2 \cdot 1.1 = \boxed{0.29}$$

$$\frac{P_{pipe2}}{P_{ref}} = 0.662 \cdot 1.1 = \boxed{0.48}$$



# Parallelism and Pipelining in E-D Space



# Architecture Summary: Simple Datapath

---

- **Pipelining and parallelism relax datapath performance, which allows  $V_{DD}$  reduction and results in power savings**
- **Pipelining has less area overhead than parallelism, but is generally harder to implement (finding logic cut-sets)**

Results from [1]

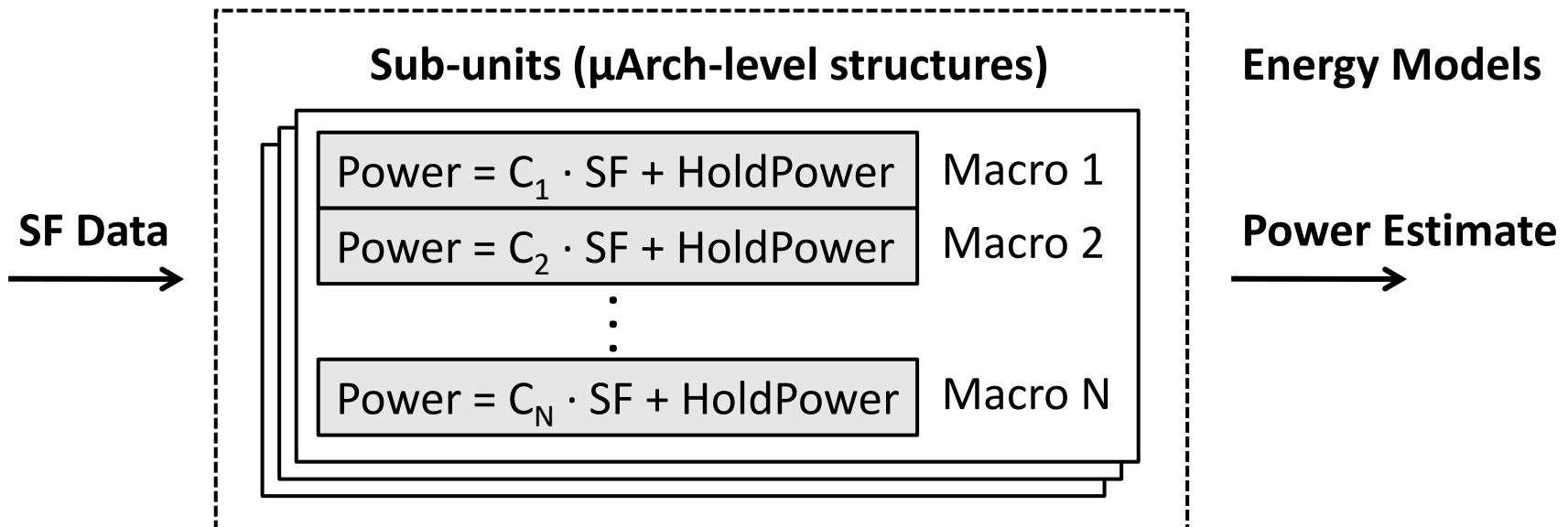
Architecture type	Voltage	Area	Power
Reference datapath	5 V	1	1
Pipelined datapath	2.9 V	1.3	0.39
Parallel datapath	2.9 V	3.4	0.36
Pipeline-Parallel	2.0 V	3.7	0.2

---

[1] A.P. Chandrakasan, S. Sheng, and R.W. Brodersen, "Low-Power CMOS Digital Design," *IEEE J. Solid-State Circuits*, vol. 27, no. 4, pp. 473-484, Apr. 1992.

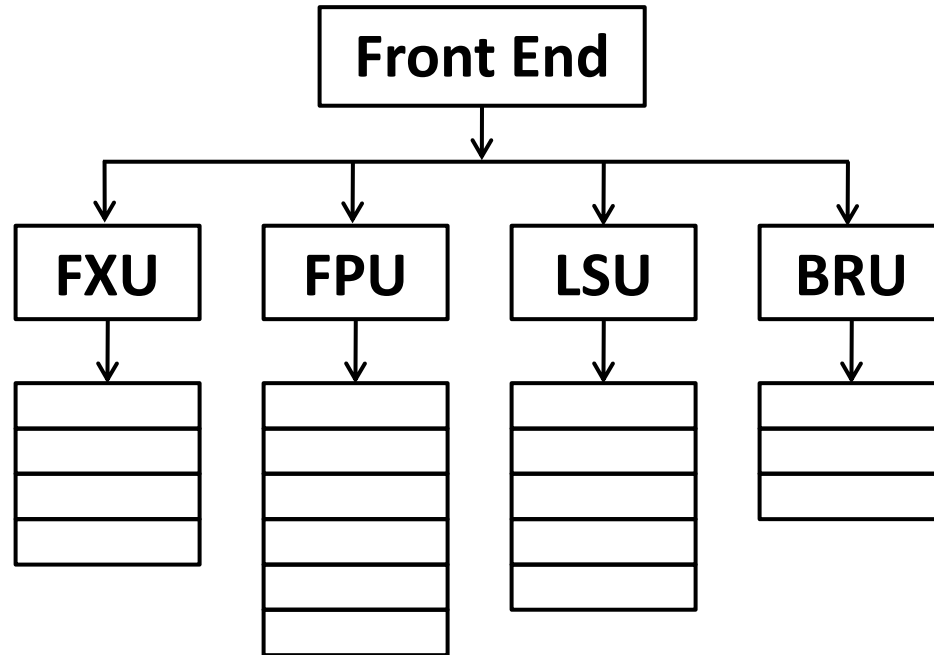
# Example: Microprocessor Pipelining

- **Superscalar processor**
  - Determine optimal pipeline depth and target frequency
- **Power model**
  - PowerTimer toolset developed at IBM T.J. Watson
  - Methodology to build energy models based on results of circuit-level power analysis tool



# Timing: Analytical Pipeline Model

- Time per stage of pipeline:  $T_i = t_i/s_i + c_i$



<b>Stages:</b>	$s_1$	$s_2$	$s_3$	$s_4$
<b>Logic delay:</b>	$t_1$	$t_2$	$t_3$	$t_4$
<b>Latch delay/stage:</b>	$c_1$	$c_2$	$c_3$	$c_4$

# Timing: Analytical Pipeline Model

- Time to complete FXU operation in presence of stalls

$$T_{fxu} = T_1 + Stall_{fxu-fxu} \cdot T_1 + Stall_{fxu-fpu} \cdot T_2 + \dots + Stall_{fxu-bru} \cdot T_4$$

$$Stall_{fxu-fxu} = f_1 \cdot (s_1 - 1) + f_2 \cdot (s_1 - 2) + \dots$$

$f_i$  is conditional probability that an FXU instruction  $m$  depends on FXU instruction  $(m - i)$

$$Throughput = u_1/T_{fxu} + u_2/T_{fpu} + u_3/T_{lsu} + u_4/T_{bru} \quad [3]$$

$u_i$  fraction of time pipe  $i$  has instructions arriving from FE of the machine  $u_i = 0$  unutilized pipe,  $u_i = 1$  fully utilized

[3] V. Srinivasan *et al.*, "Optimizing Pipelines for Power and Performance," in *Proc. IEEE/ACM Int. Symp. on Microarchitecture*, Nov. 2002, pp. 333-344.

# Simulation Results

---

- **Optimal pipeline depth for two applications (SPEC 2000, TPC-C) under different optimization metrics**
  - **Performance-aware optimization:** maximize BIPS
  - **Power-aware optimization:** maximize  $\text{BIPS}^3/\text{W}$
- **More pipeline stages for low power ( $\text{BIPS}^3/\text{W}$ )**

Application	Max BIPS	Max $\text{BIPS}^3/\text{W}$
Spec 2000	10 FO4	18 FO4
TPC-C	10 FO4	25 FO4

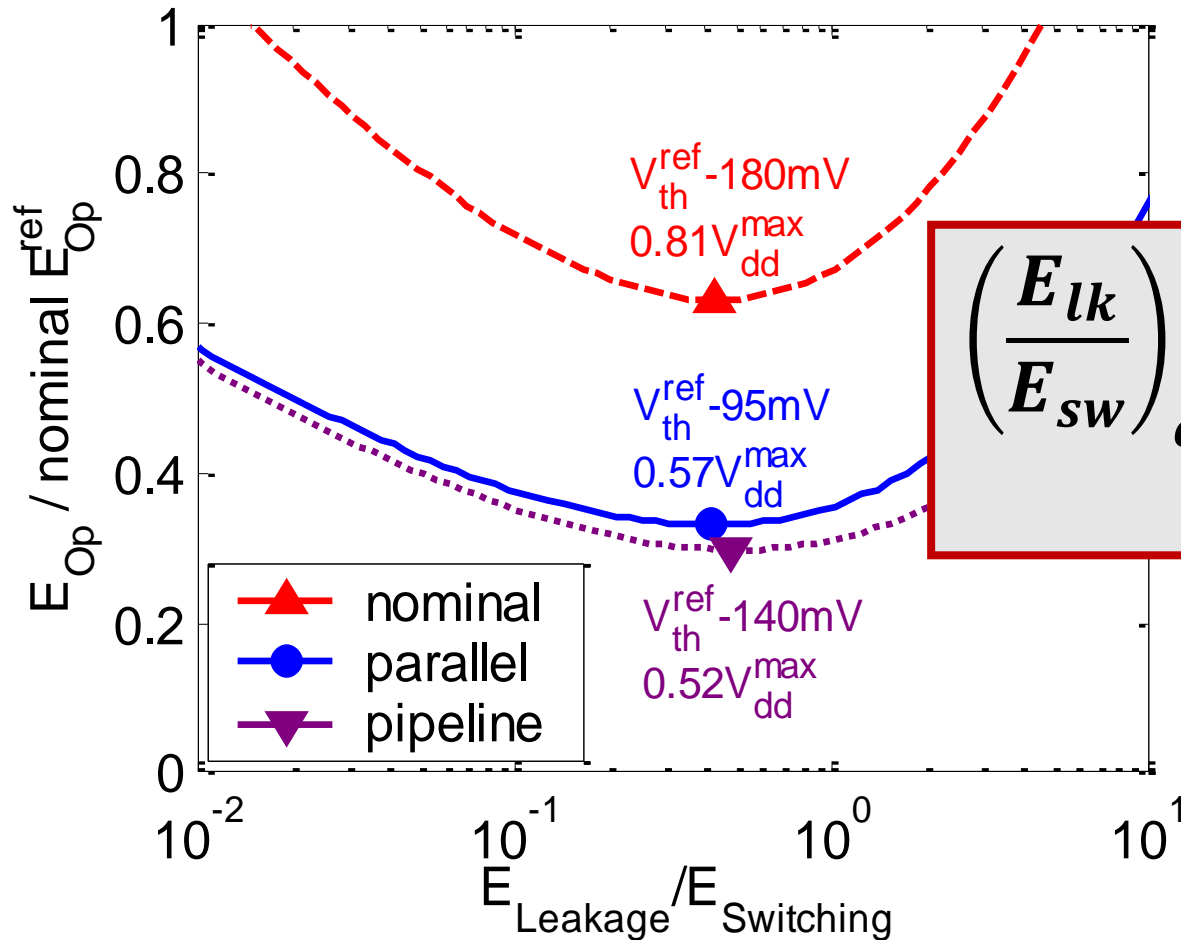
- **Choice of pipeline register also impacts BIPS**
  - A 20% better BIPS performance by using a register with 2 FO4 delay as compared to a register with 5 FO4 delay

# Minimum Energy:

$$E_{Lk} / E_{sw}?$$

# Minimum Energy: $E_{Lk}/E_{Sw} \approx 0.5$

Set  $V_T$ , find  $V_{DD}$  to min  $E_{Op}$  for a fixed performance



- Large  $(E_{Lk}/E_{Sw})^{opt}$
- Flat  $E_{Op}$  minimum

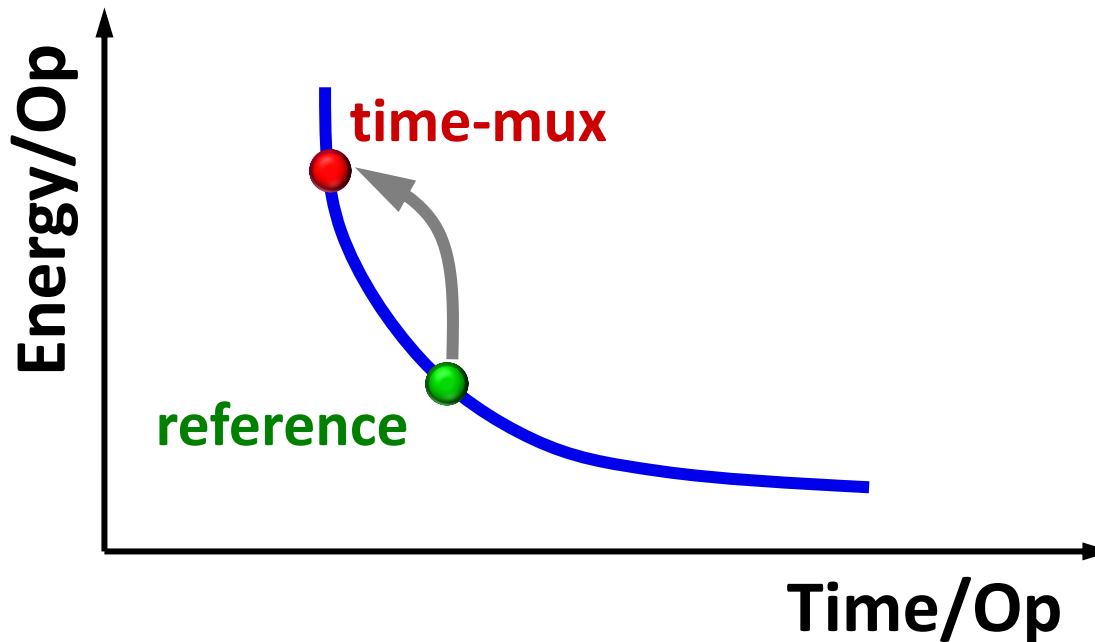
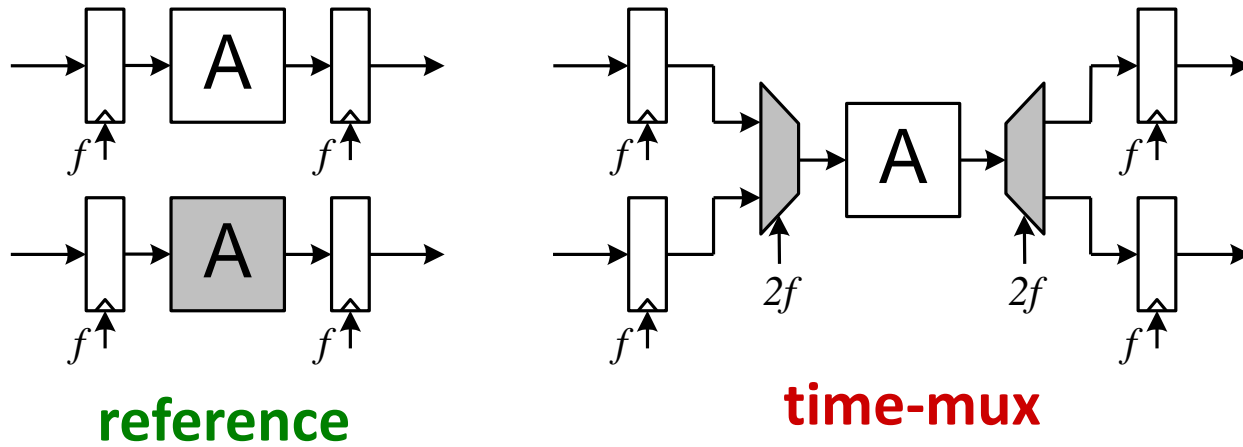
$$\left(\frac{E_{lk}}{E_{sw}}\right)_{opt} = \frac{2}{\ln\left(\frac{L_d}{\alpha_{avg}}\right) - K}$$

Topology	Inv	Add	Dec
$(E_{Lk}/E_{Sw})^{opt}$	0.8	0.5	0.2

V. Stojanović *et al.*,  
ESSCIRC 2002, pp. 211-214.




# Time Multiplexing



# Architectural Techniques

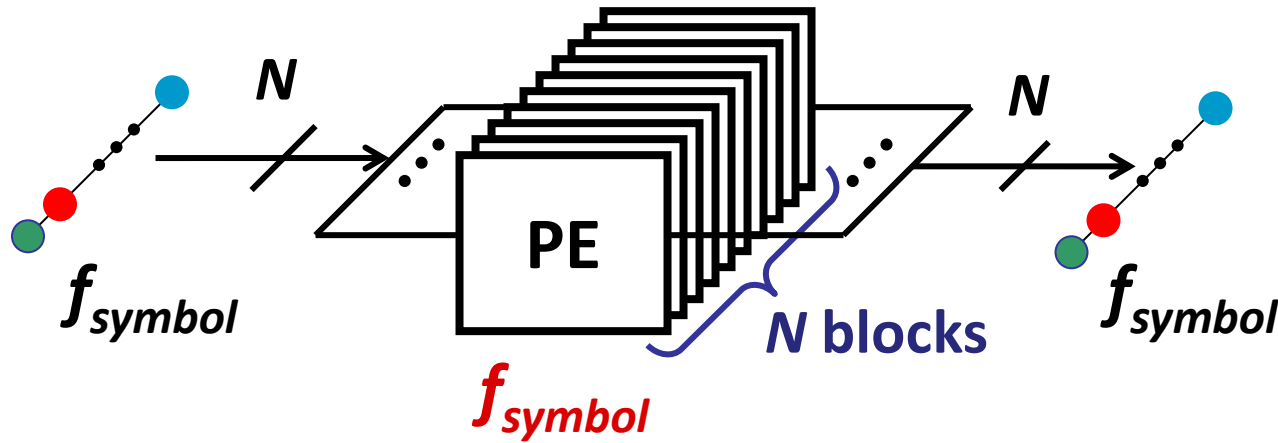
---

- **Feed-forward algorithms**
  - Parallelism
  - Pipelining
  - Time multiplexing

-  • **Recursive algorithms**
  - Interleaving
  - Folding

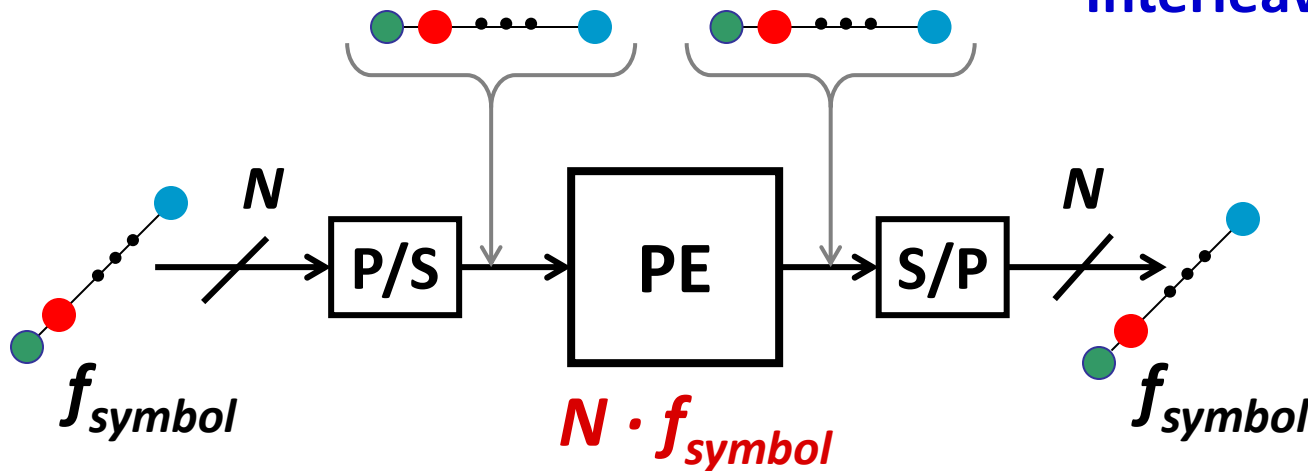
# Data-Stream Interleaving

PE = recursive operation



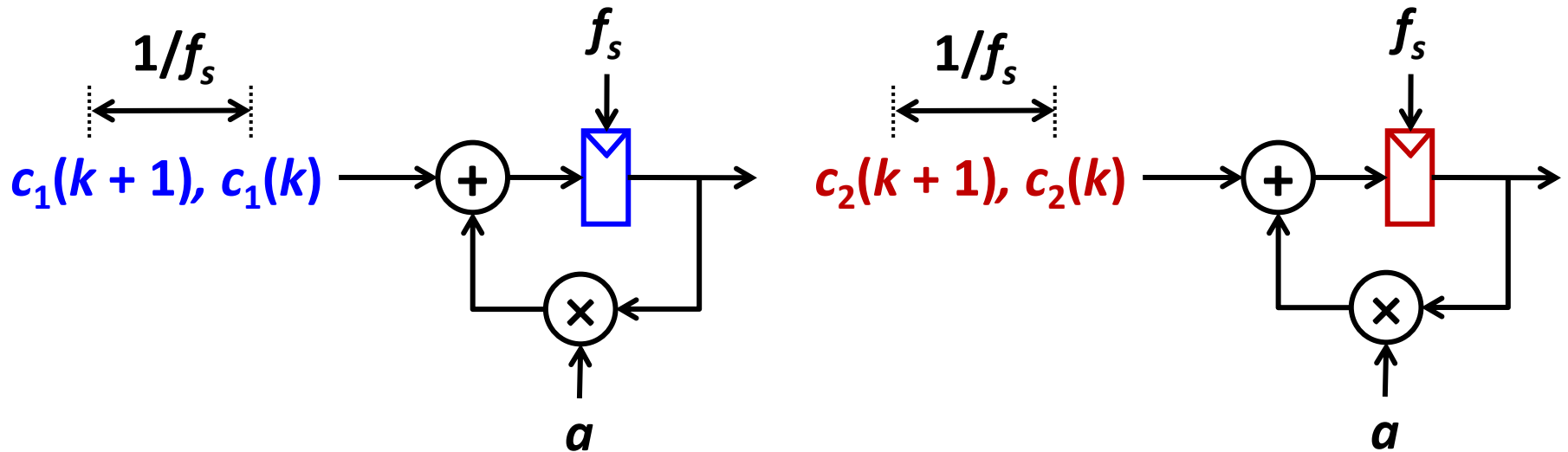
- PE too fast
- Large area

## Interleaved Architecture

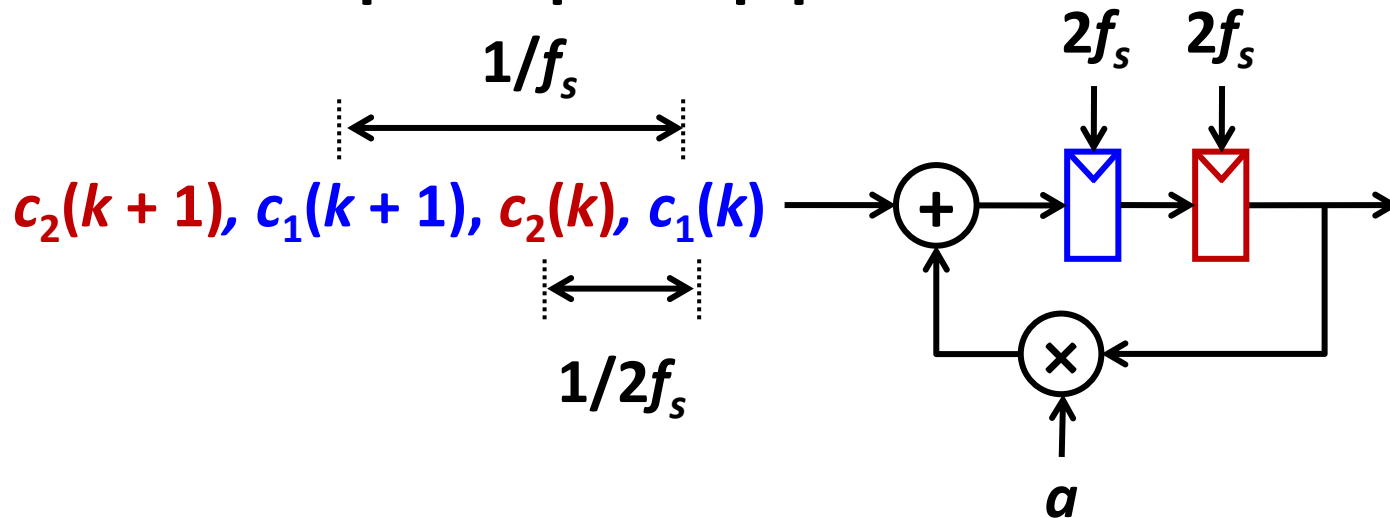


- Reduced area
- P/S overhead
- Pipelined

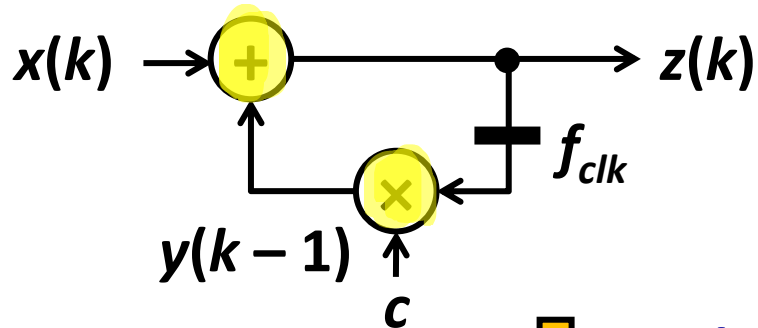
# PE Performs Recursive Operation



- Interleave = up-sample & pipeline

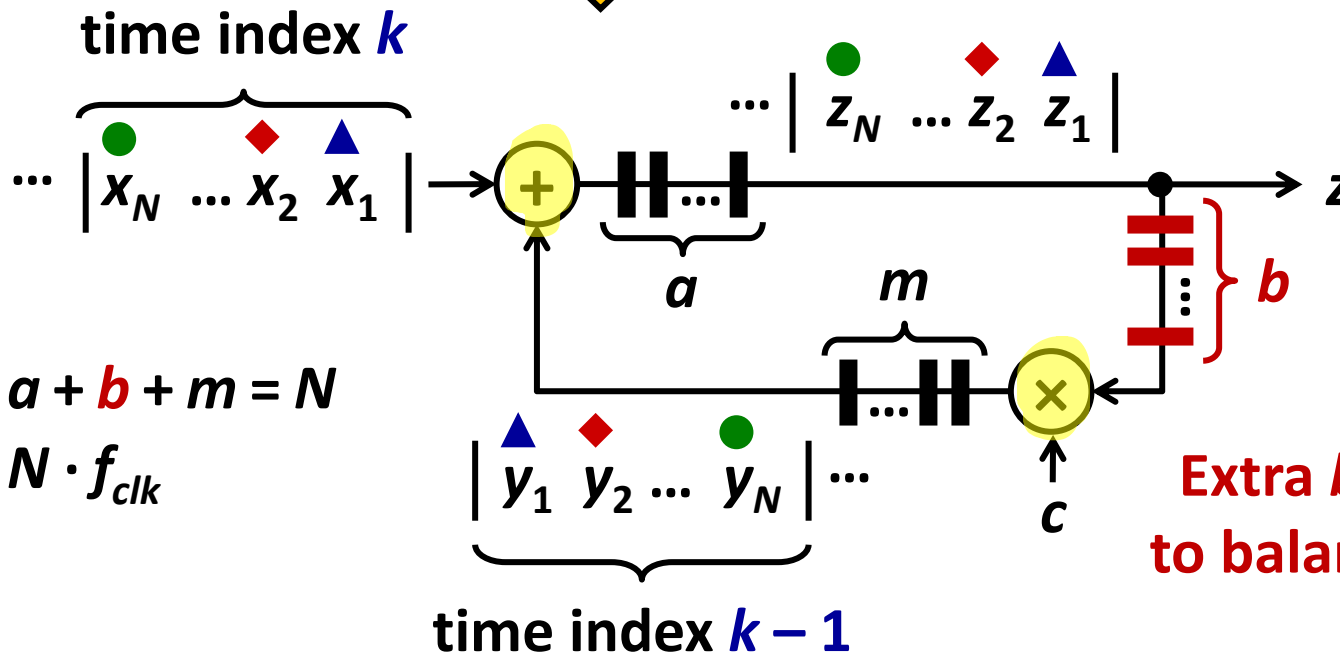


# Data-Stream Interleaving Example

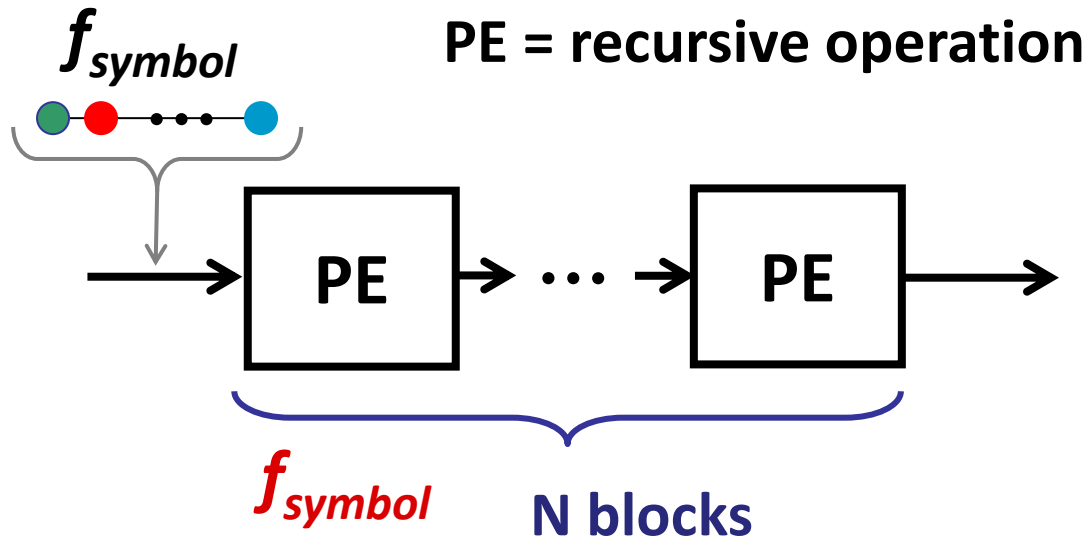


Recursive operation:  
 $z(k) = x(k) + c \cdot z(k-1)$

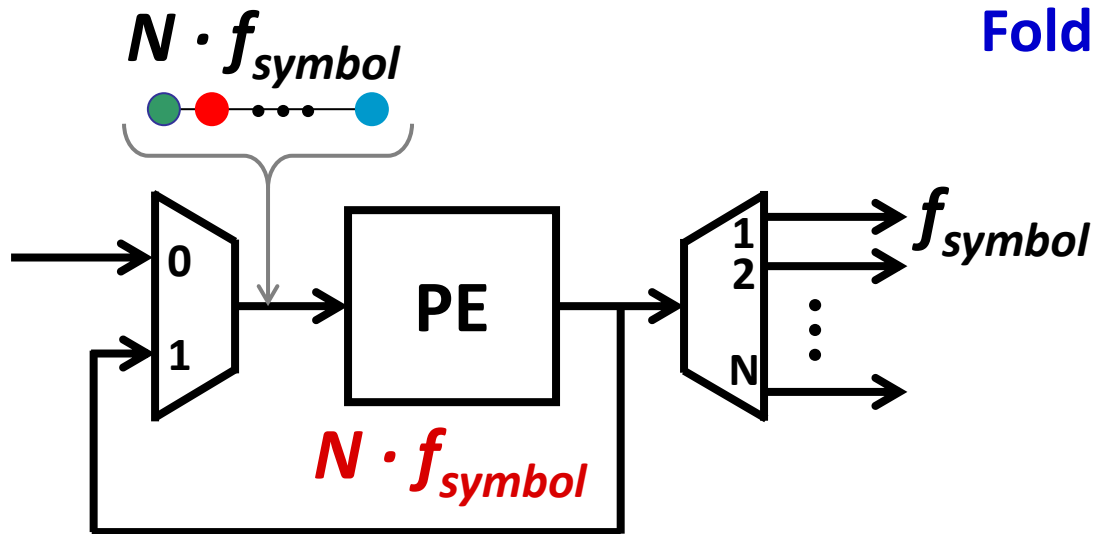
N data streams:  $x_1, x_2, \dots, x_N$



# Folding



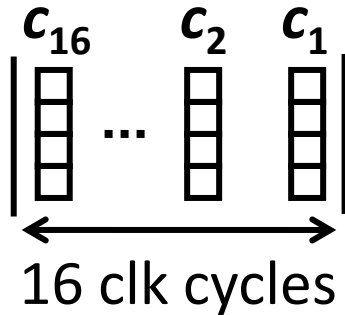
- PE too fast
- Large area



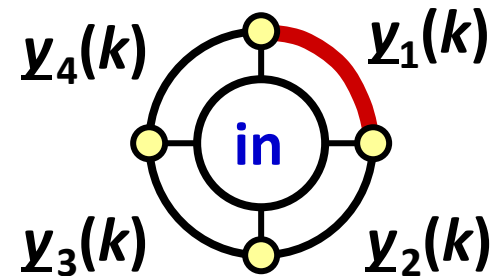
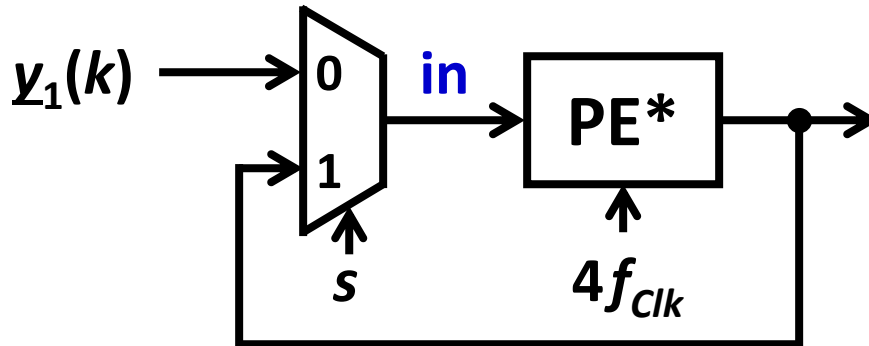
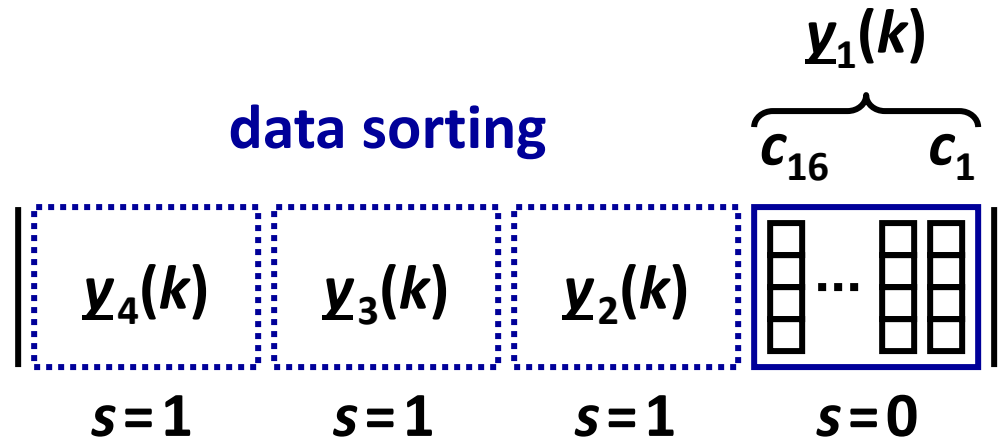
- Reduced area
- Highly pipelined

# Folding Example

16 data streams



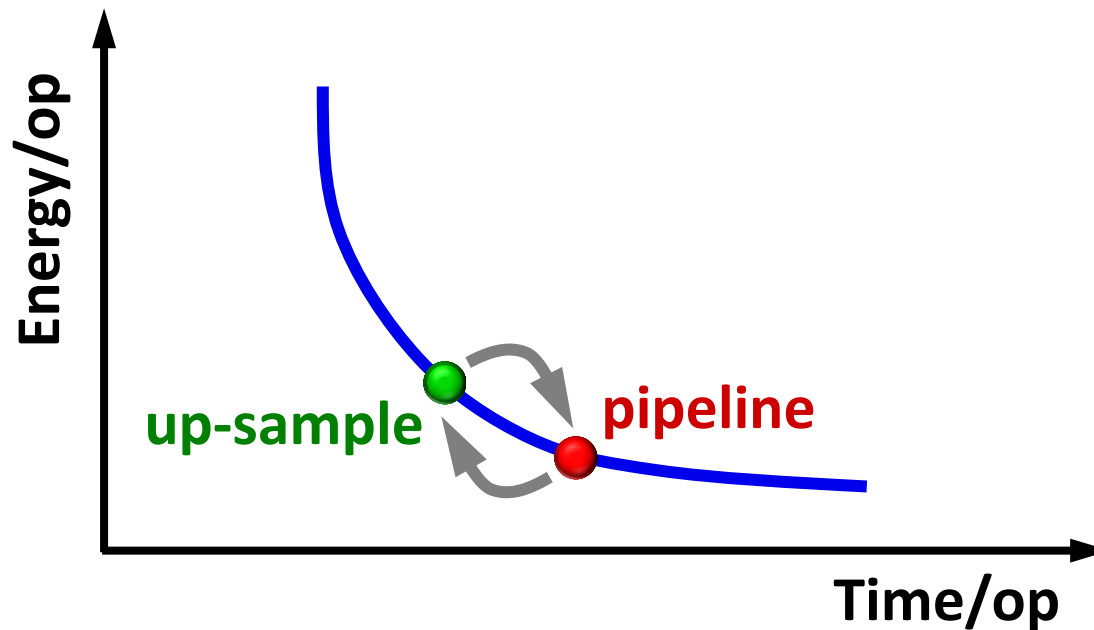
data sorting



- **Folding = up-sampling & pipelining**
  - Reduced area (shared datapath logic)

# Area Benefit of Interleaving and Folding

- Area:  $A = A_{logic} + A_{registers}$
- Interleaving or folding of level  $N$ 
  - $A = A_{logic} / N + A_{registers}$
- Timing and energy stay the same

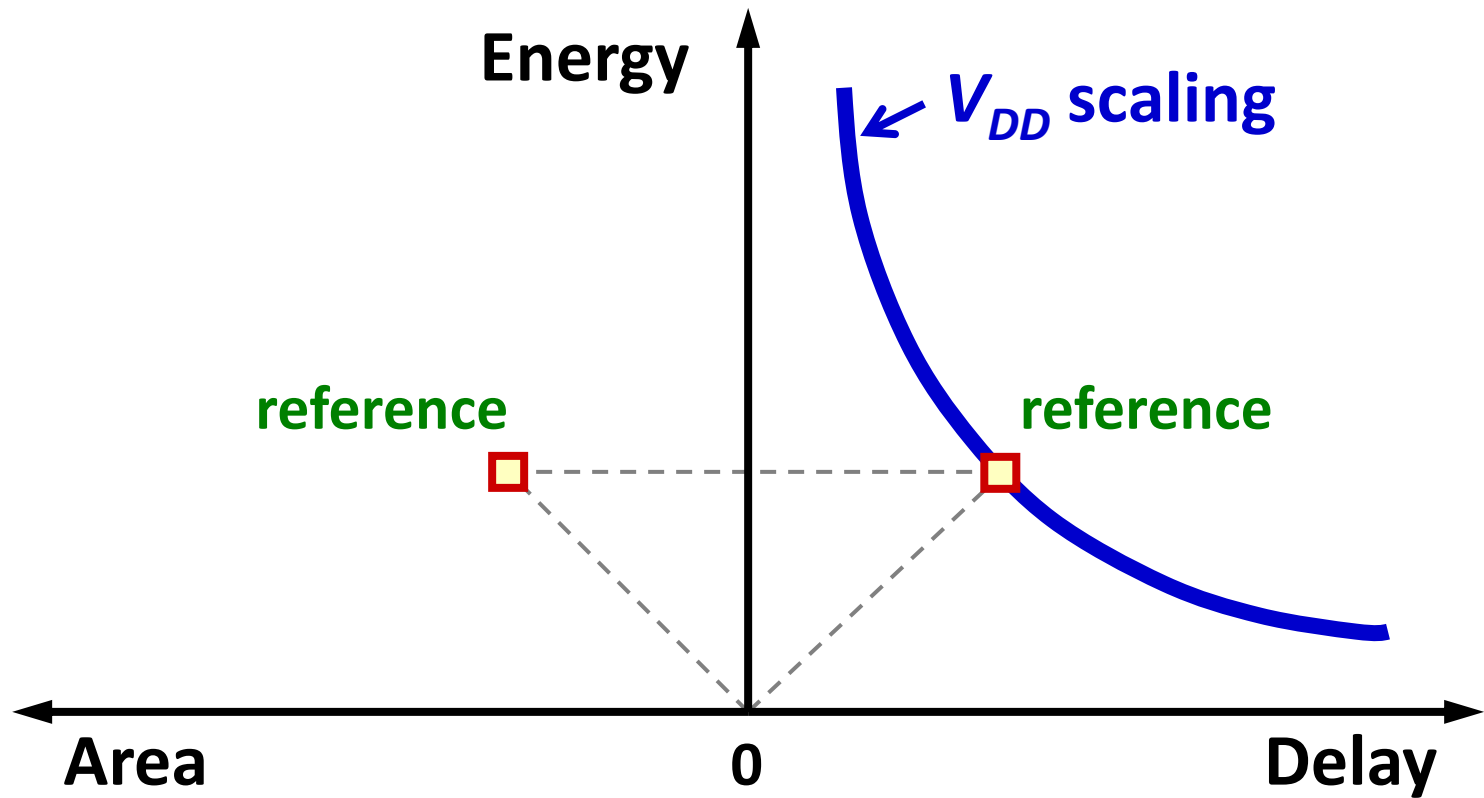




# **Architectural Techniques:** **Energy-Area-Delay Space**

# Architectural Transformations

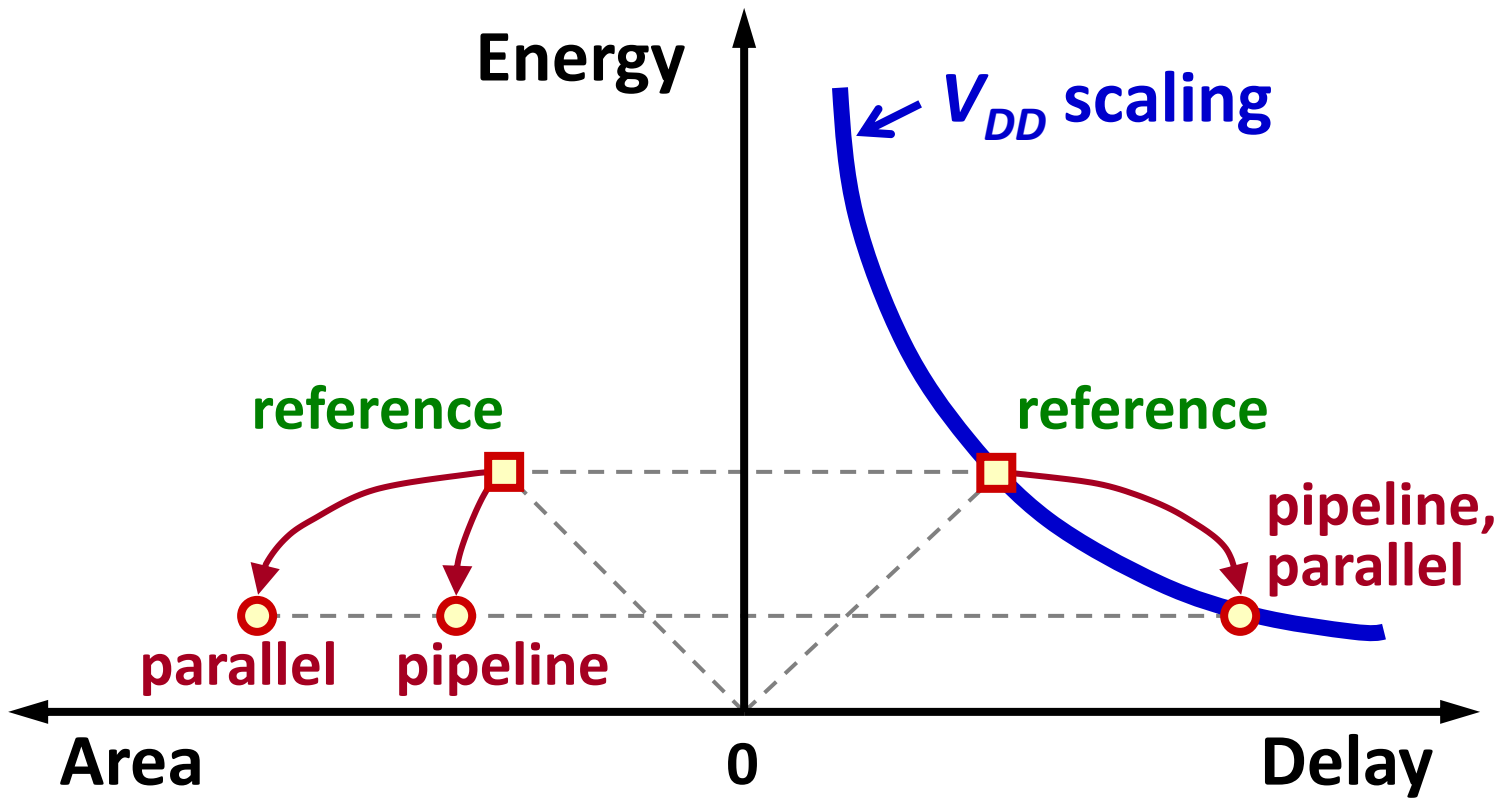
**Goal:** move toward desired E-D point while minimizing area



D. Marković, Ph.D. Thesis, University of California, Berkeley, 2006.

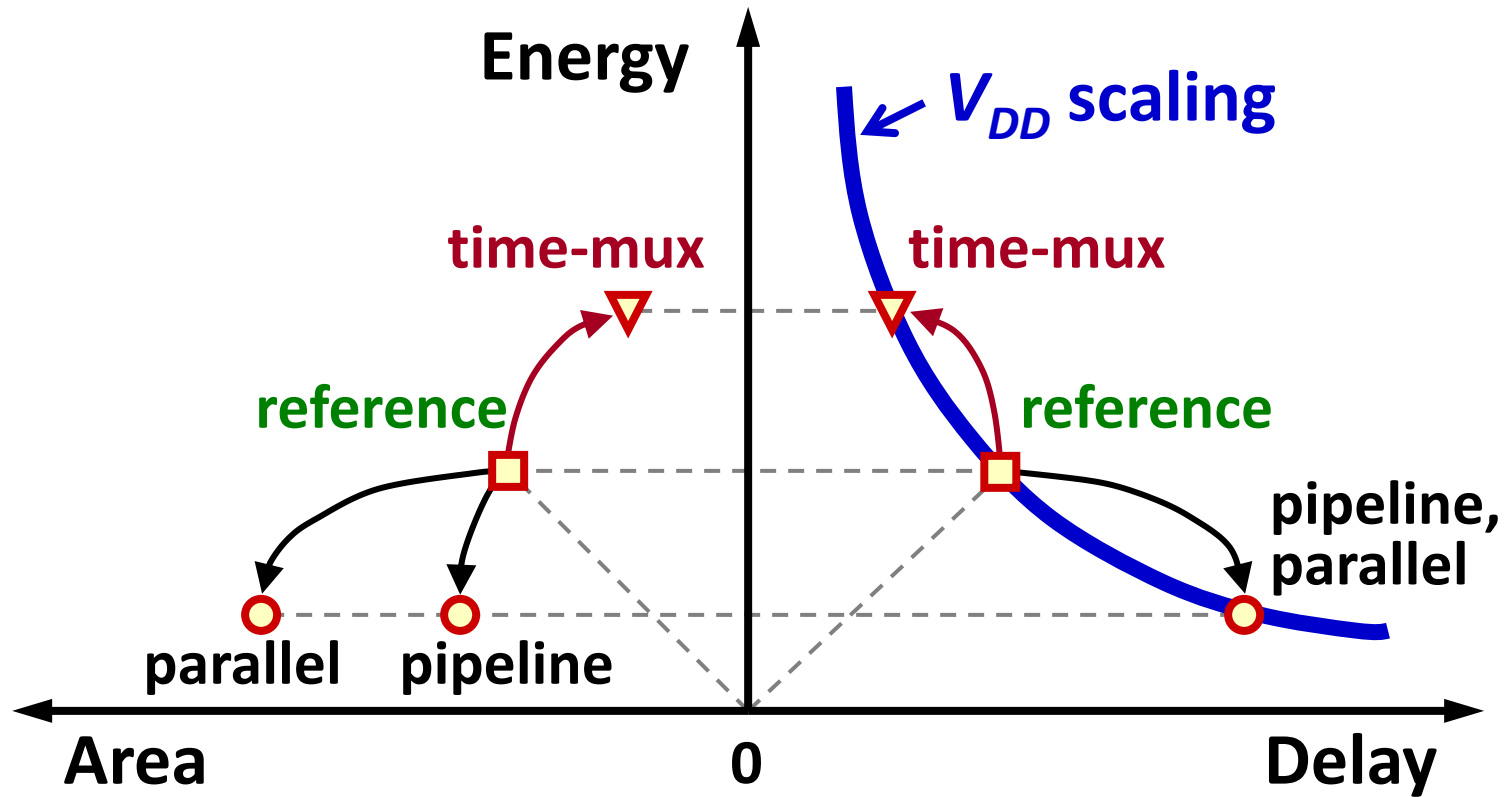
# Parallelism & Pipelining

Energy ↓, Area ↑



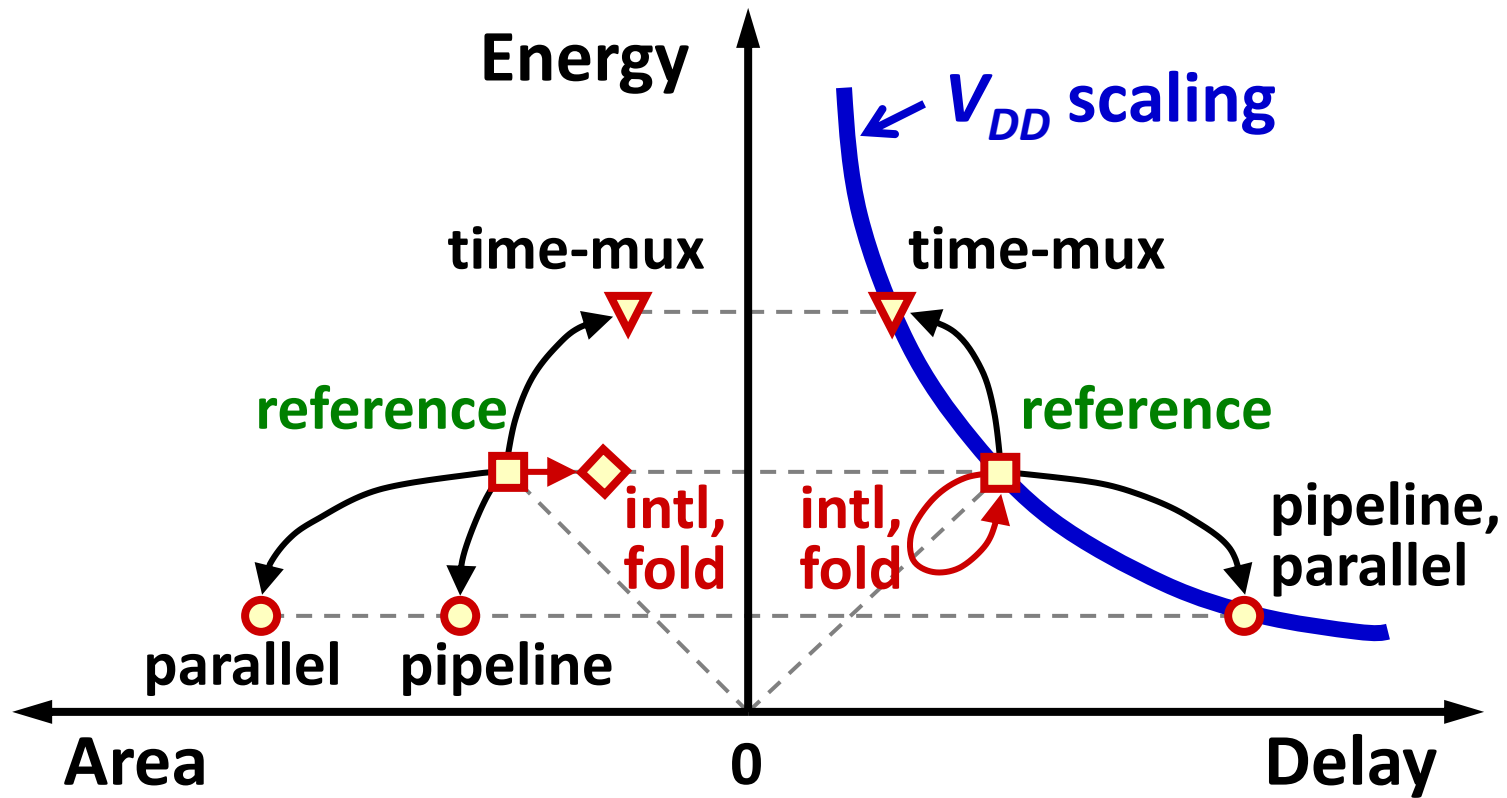
# Time Multiplexing

Energy  $\uparrow$ , Area  $\downarrow$

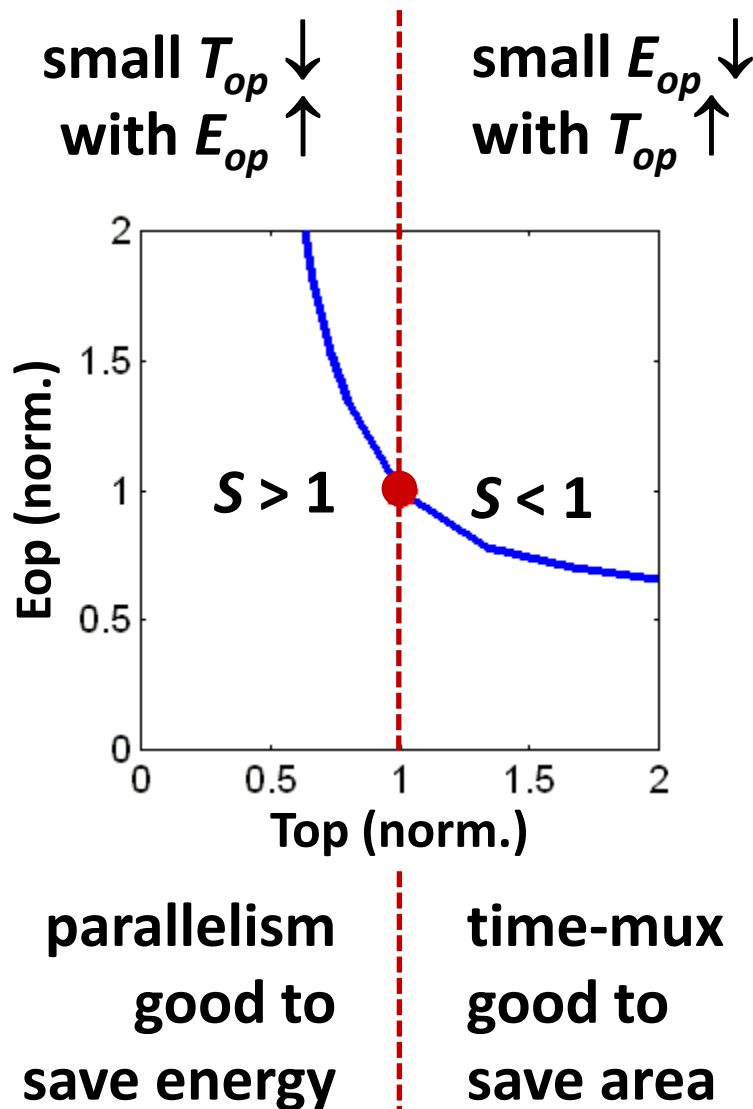


# Interleaving & Folding

Energy  $\approx$  const, Area  $\downarrow$

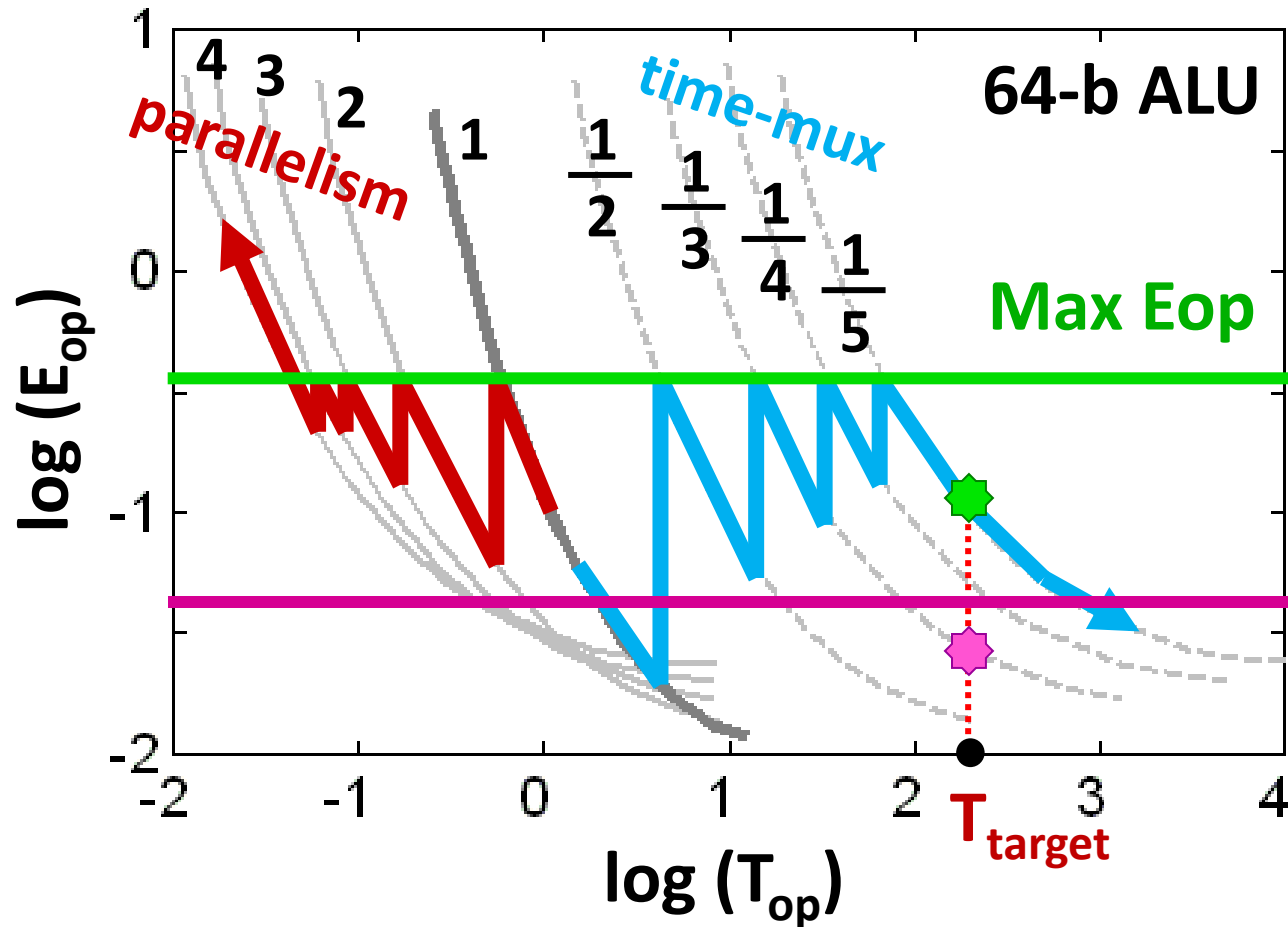


# Back to Sensitivity Analysis



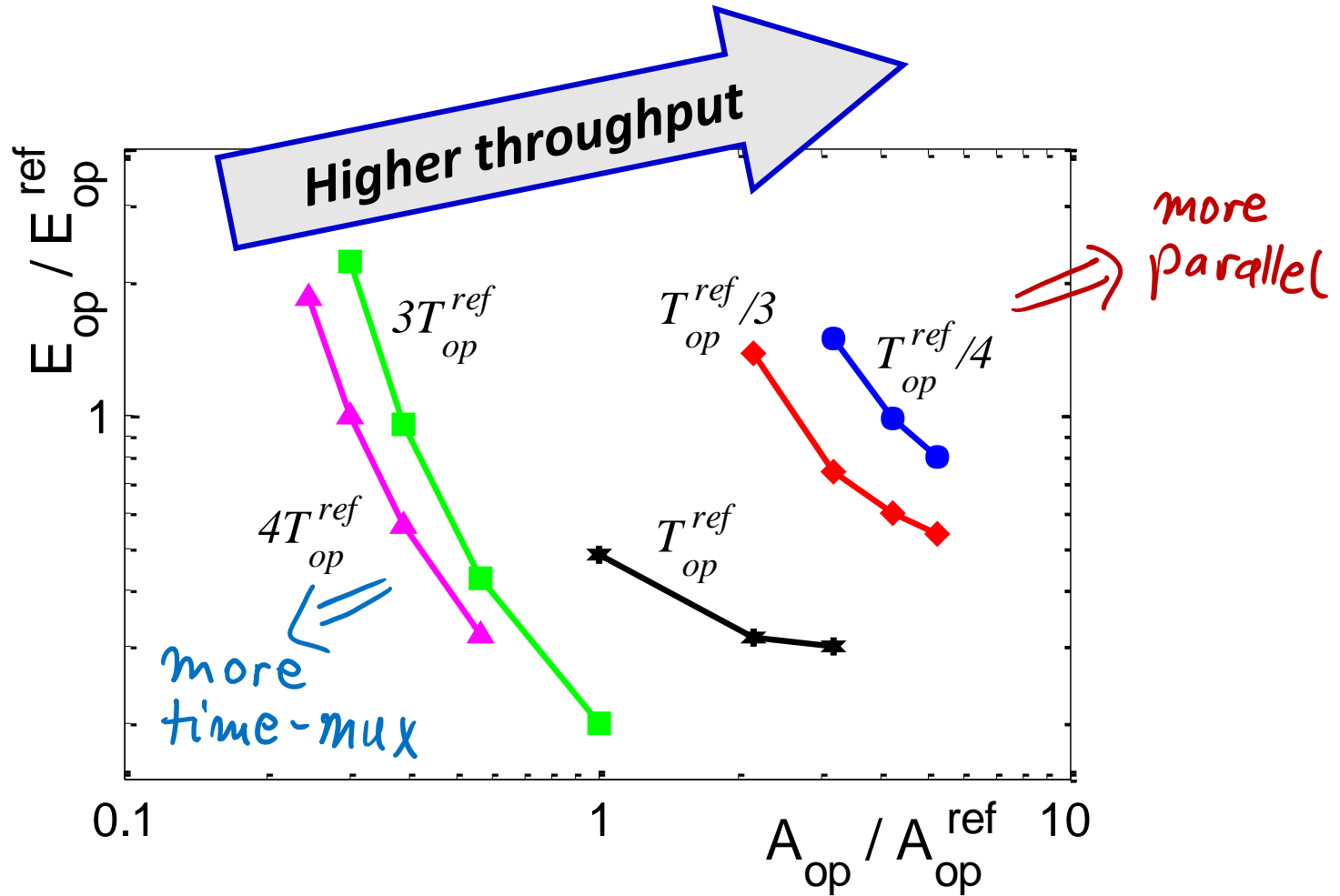
# Energy-Area Tradeoff...

High throughput: Parallelism = Large Area



Low throughput: Time-Mux = Small Area

# ...Time-Space Tradeoff

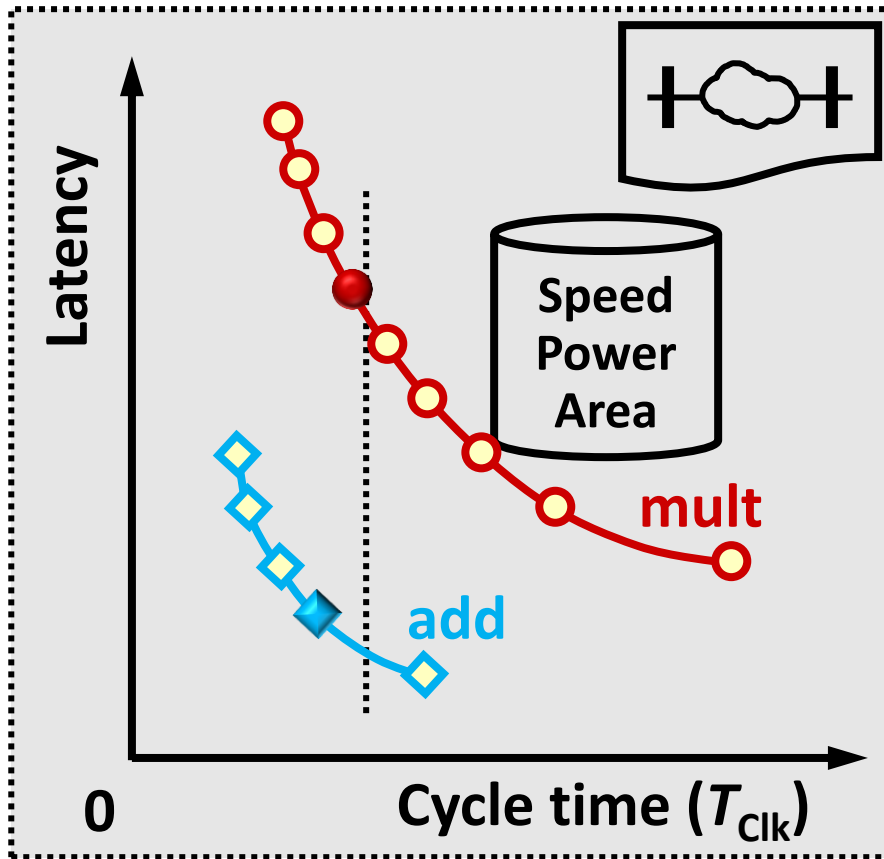




# Architecture + Circuits: Design Guidelines

# Issue: Block Latency / Retiming

Goal: balance logic depth within a block



## *Micro-Architecture*

- Select block latency to achieve target  $T_{clk}$ 
  - Balances pipeline logic depth
- Apply  $W$  and  $V_{DD}$  scaling to this logic

# Including Supply Voltage Scaling

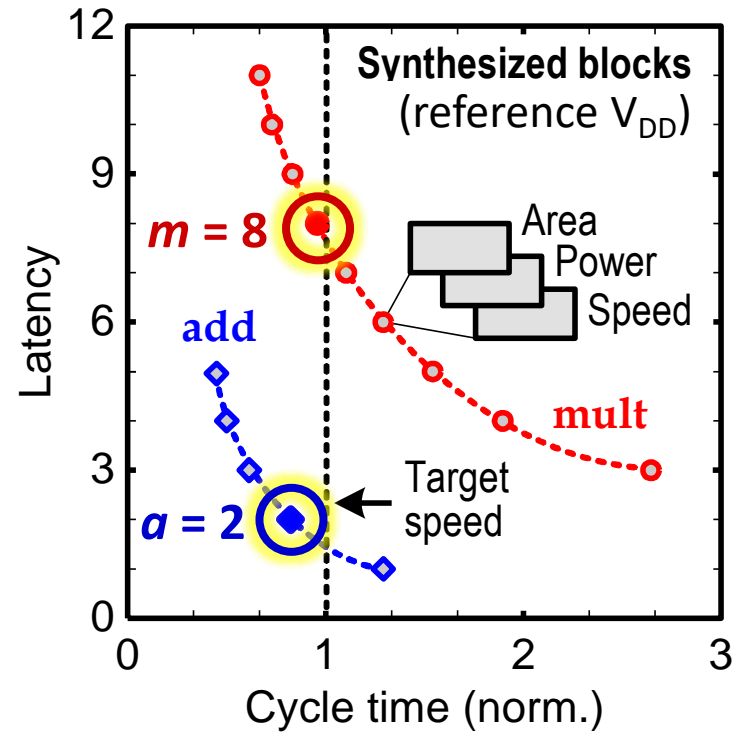
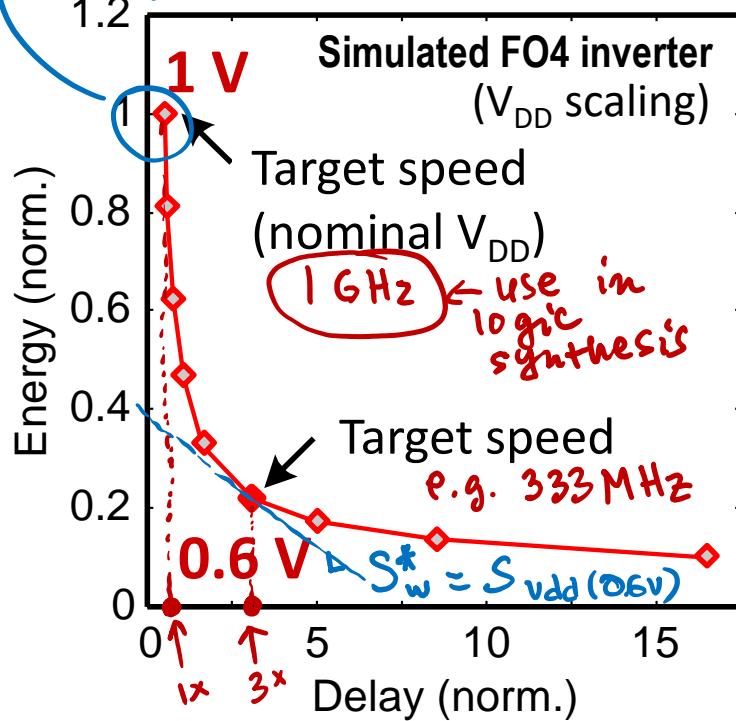
For a block with predetermined wordlength

*incremental compile to match  $S_w^*$*

Translate timing to target  $V_{DD}$



Choose latency for a given  $T_{CLK}$

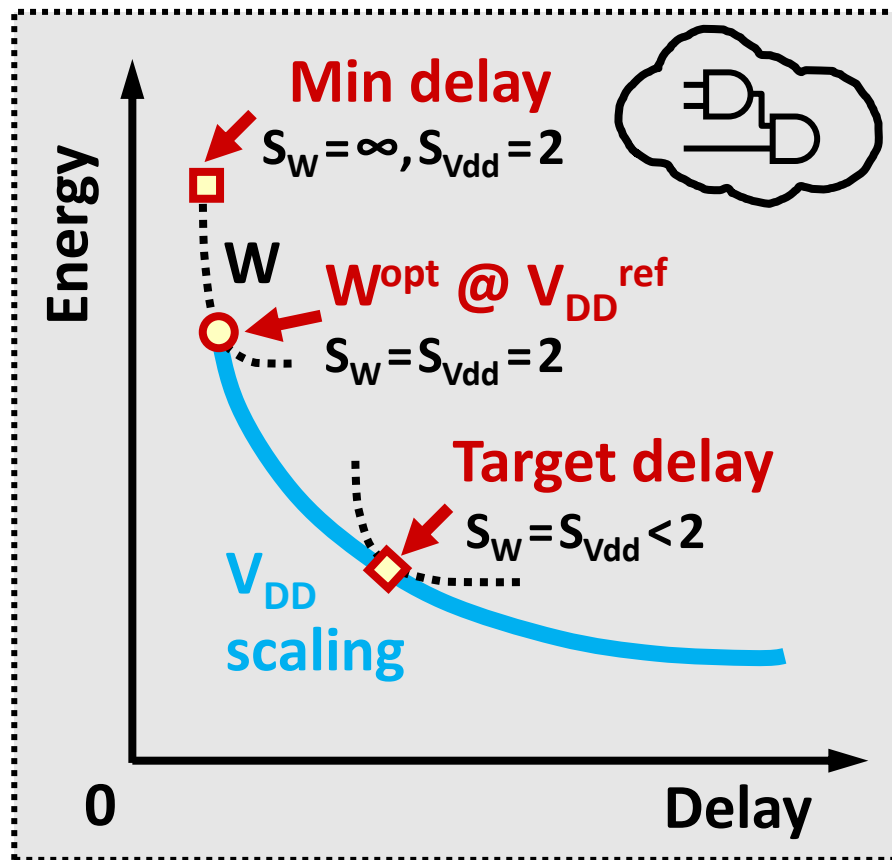
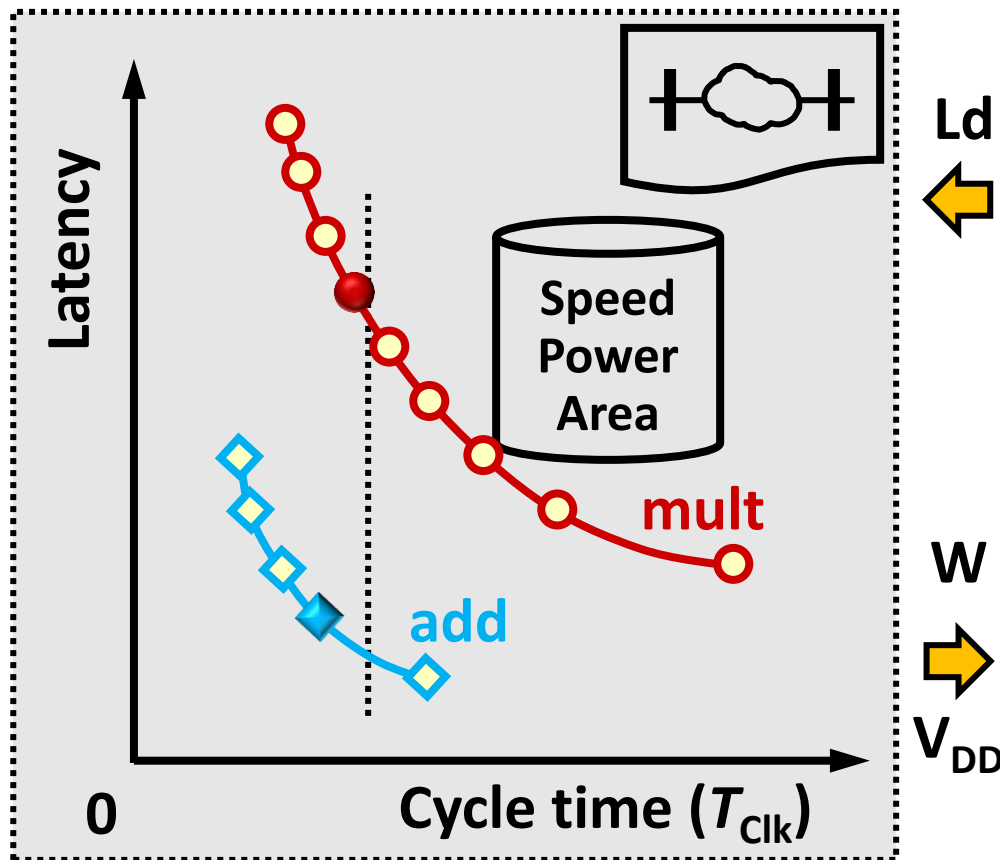


D. Marković, Ph.D. Thesis, University of California, Berkeley, 2006.

# Putting it All Together

Adjust latency to meet  $T_{\text{clk}}$   
→ Balanced logic depth (Ld)

Optimize  $V_{\text{DD}}$  &  $W$  of logic  
( $T_{\text{clk}} \approx \text{Ld} \cdot \text{Delay}$ )



# Summary: Architecture Techniques

---

- **Performance-centric designs (that maximize BIPS) require shorter (fewer FO4 stages) logic pipelines**
- **Energy-performance optimal designs have roughly equal leakage and switching components of energy**
  - Otherwise, one can be traded for another for further energy reduction
- **Techniques for direct (parallelism, pipelining) and recursive (interleaving, folding) systems can be compared in area-energy-performance space**
  - Latency (# of pipeline stages) is dictated by  $T_{\text{clk}}$

# Summary: Design Guidelines

---

- **For maximum performance**
  - Maximize use of concurrency at the cost of area
- **For given performance**
  - Optimal amount of concurrency for minimum energy
- **For given energy**
  - Least amount of concurrency that meets performance
- **For minimum energy**
  - Solution with minimum overhead  
(direct mapping between function and architecture)

# System Perspective

---

- **Optimizations at higher abstraction levels have greater potential impact**
- **While circuit techniques may yield improvements in the 10-50% range**
- **Architecture and algorithm optimizations can reach orders of magnitude power reduction**

