# The BubbleWrap Many-Core:
## Popping Cores for Sequential Acceleration

**Ulya Karpuzcu**, Brian Greskamp, Josep Torrellas
University of Illinois

http://iacoma.cs.uiuc.edu/

University of Illinois

i-acoma group

# Problem: The Power Wall

# Problem: The Power Wall

## Ideal Scaling

# Problem: The Power Wall

## Ideal Scaling

- Dynamic (switching) power density remains constant

# Problem: The Power Wall

## Ideal Scaling

- Dynamic (switching) power density remains constant

$P_{DYN}$ = #Devices x

# Problem: The Power Wall

## Ideal Scaling

- Dynamic (switching) power density remains constant

$P_{DYN}$ = #Devices x Frequency of switching x

# Problem: The Power Wall

## Ideal Scaling

- Dynamic (switching) power density remains constant

$P_{DYN}$ = #Devices x Frequency of switching x Energy per switching

# Problem: The Power Wall

## Ideal Scaling

- Dynamic (switching) power density remains constant

$P_{DYN}$ = #Devices x Frequency of switching x Energy per switching

# Problem: The Power Wall

## Ideal Scaling

- Dynamic (switching) power density remains constant

$P_{DYN}$ = #Devices x Frequency of switching x Energy per switching

# Problem: The Power Wall

## Ideal Scaling

- Dynamic (switching) power density remains constant

$P_{DYN}$ = #Devices x Frequency of switching x Energy per switching

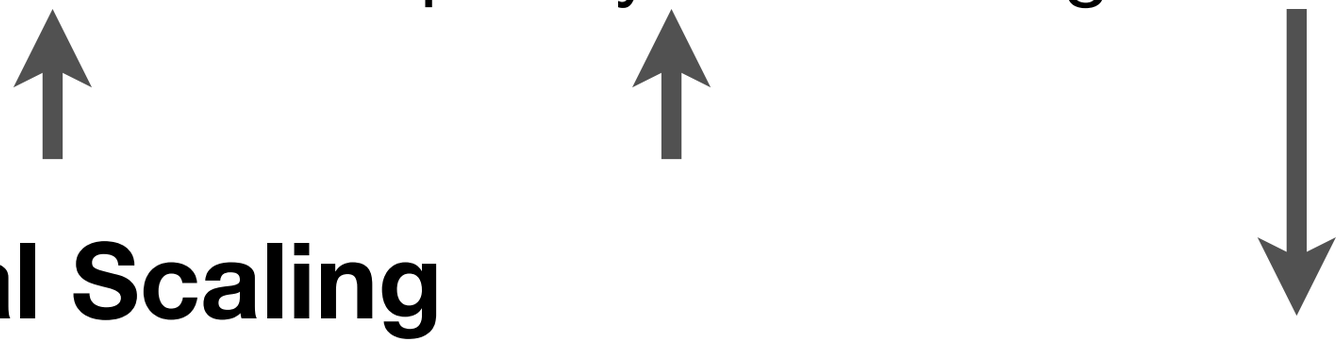
$\propto Vdd^2$

# Problem: The Power Wall

## Ideal Scaling

- Dynamic (switching) power density remains constant

$P_{DYN}$ = #Devices x Frequency of switching x Energy per switching



$\propto Vdd^2$

## Practical Scaling

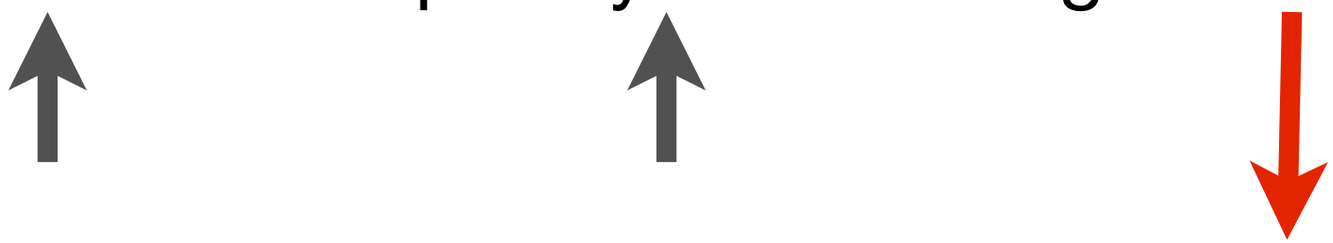
- Vdd has been scaling down slower than ideally

# Problem: The Power Wall

## Ideal Scaling

- Dynamic (switching) power density remains constant

$P_{DYN}$ = #Devices x Frequency of switching x Energy per switching




## Practical Scaling

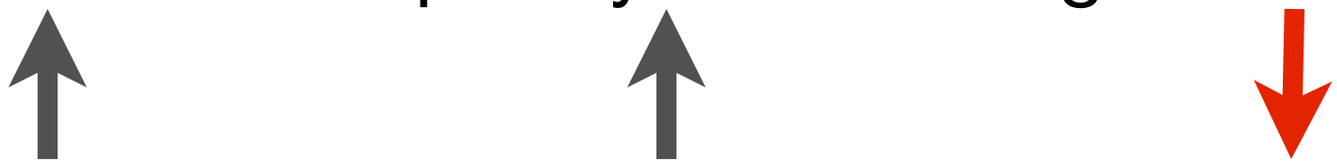
- Vdd has been scaling down slower than ideally
- Historically: Higher Vdd ≡ Higher performance

# Problem: The Power Wall

## Ideal Scaling

- Dynamic (switching) power density remains constant

$P_{DYN}$ = #Devices x Frequency of switching x Energy per switching




## Practical Scaling

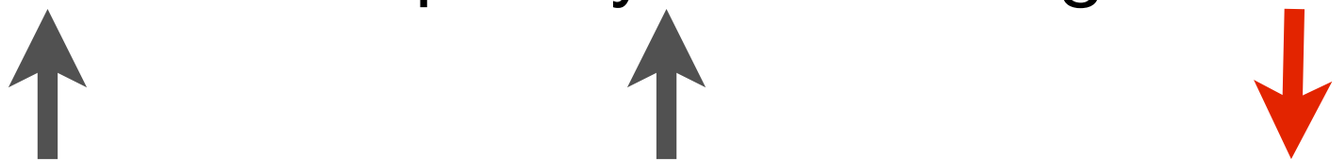
- Vdd has been scaling down slower than ideally
- Historically: Higher Vdd ≡ Higher performance
- Recently: Practically stagnated Vth scaling to control static power

# Problem: The Power Wall

## Ideal Scaling

- Dynamic (switching) power density remains constant

$P_{DYN}$ = #Devices x Frequency of switching x Energy per switching



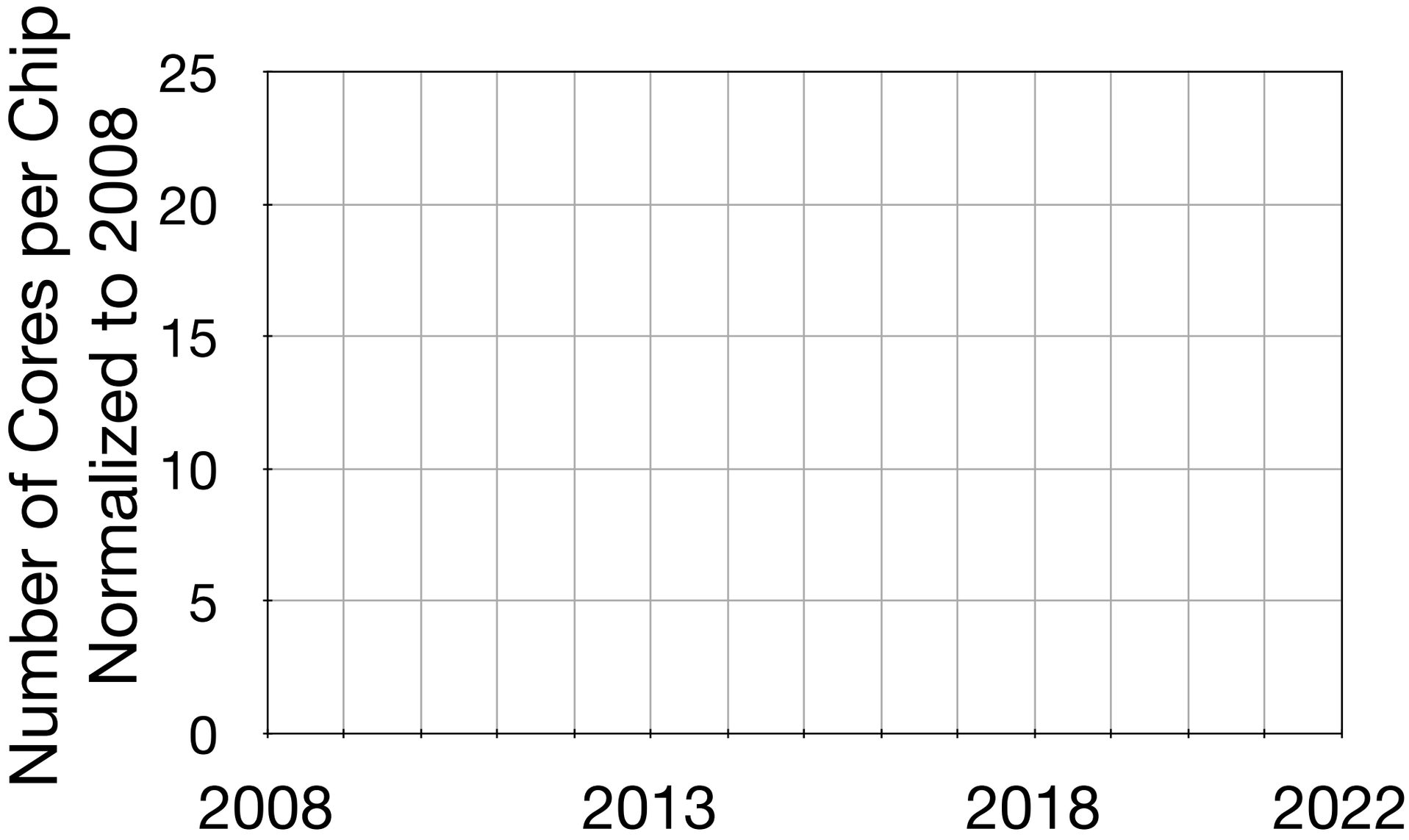

## Practical Scaling

- Vdd has been scaling down slower than ideally
- Historically: Higher Vdd ≡ Higher performance
- Recently: Practically stagnated Vth scaling to control static power

**Dynamic Power Density is increasing**

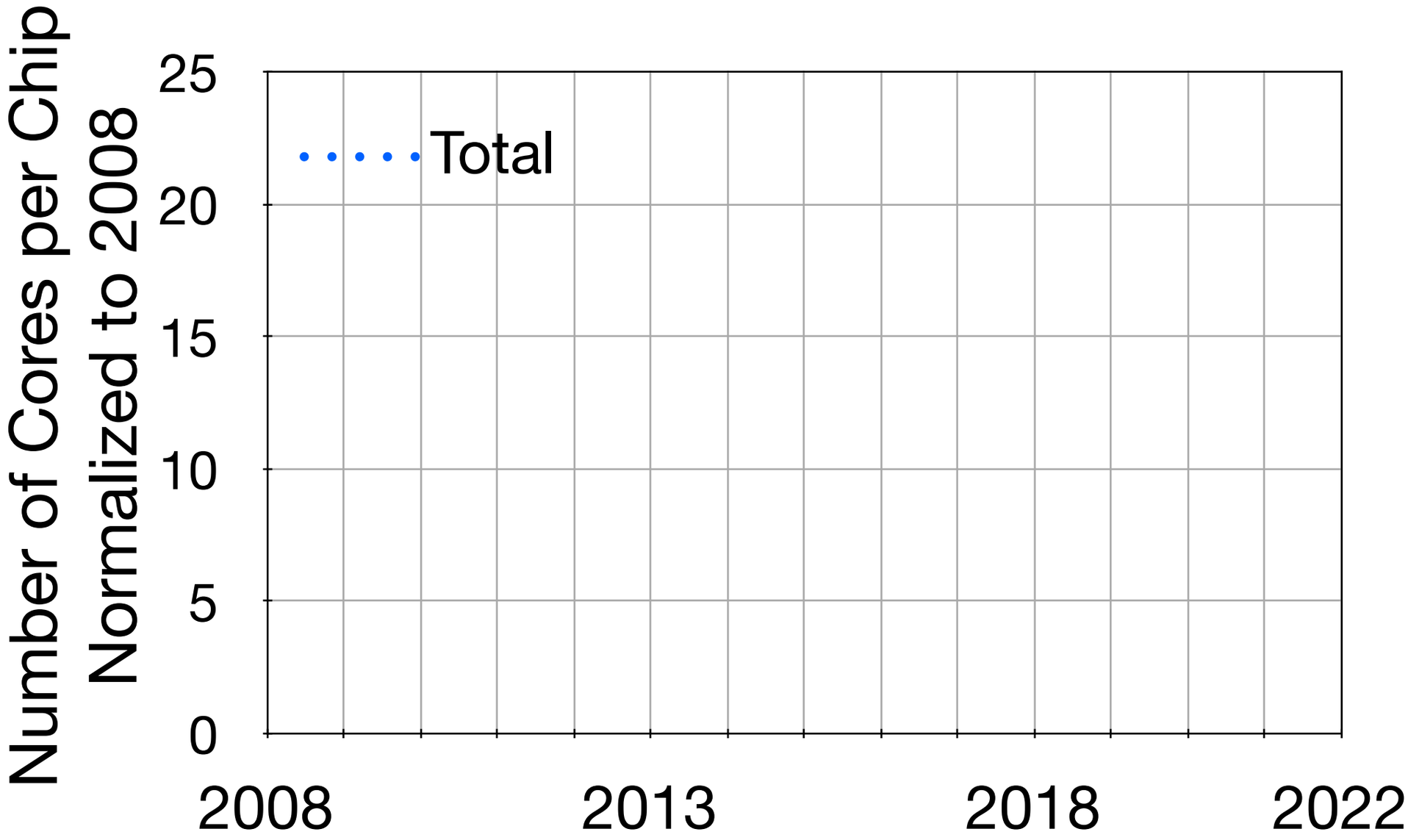
# The Power Wall

# The Power Wall



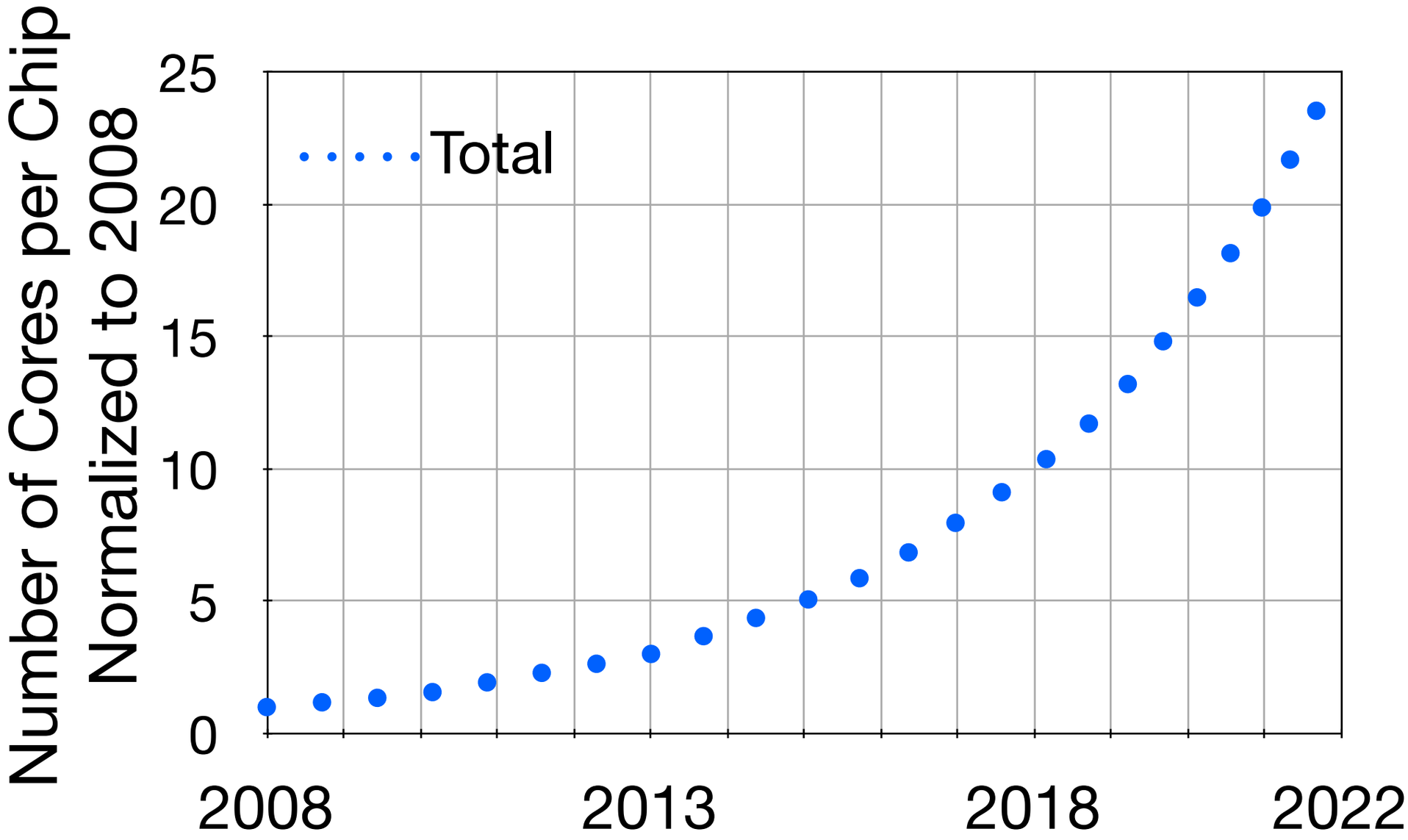

ITRS 2008

# The Power Wall



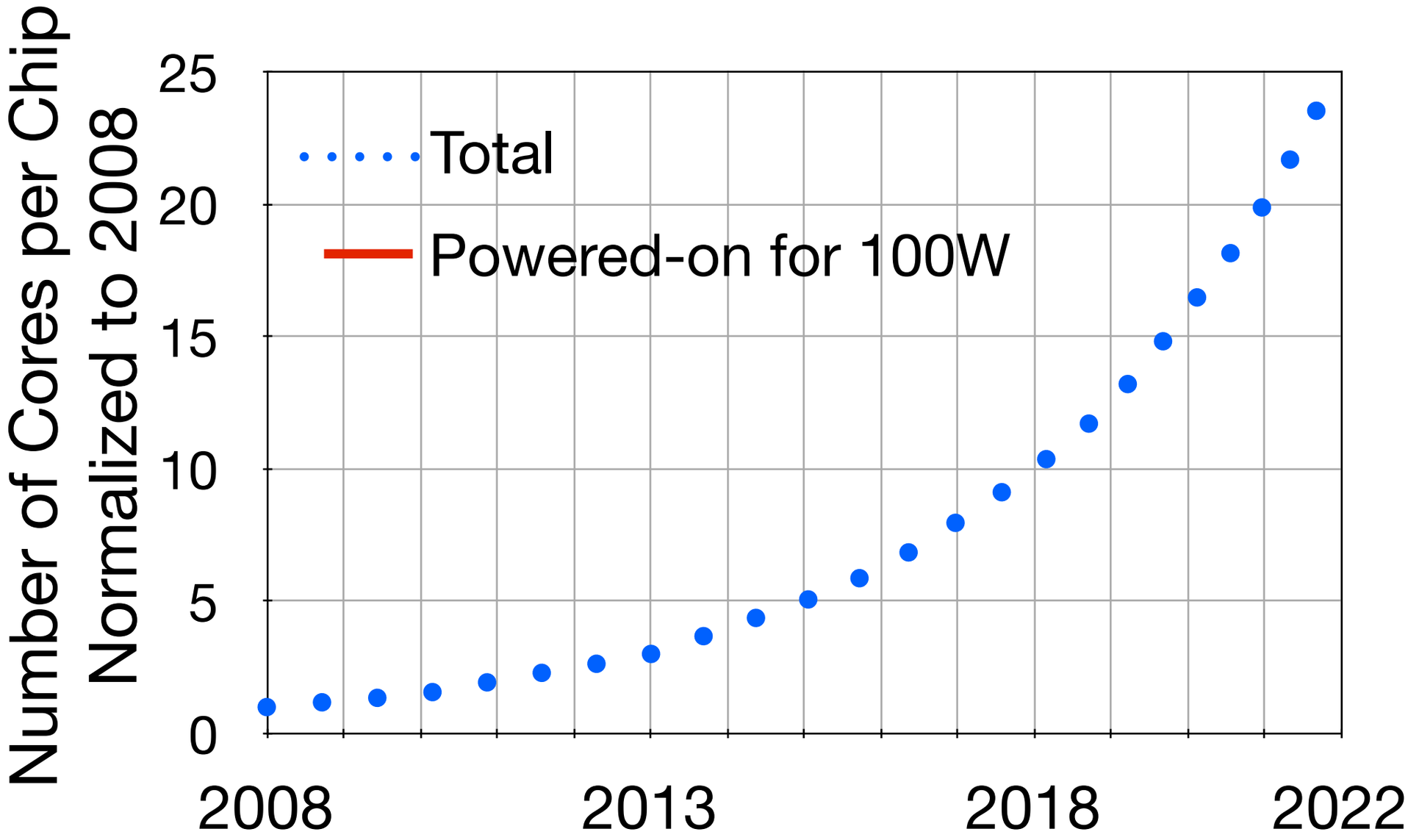

ITRS 2008

# The Power Wall



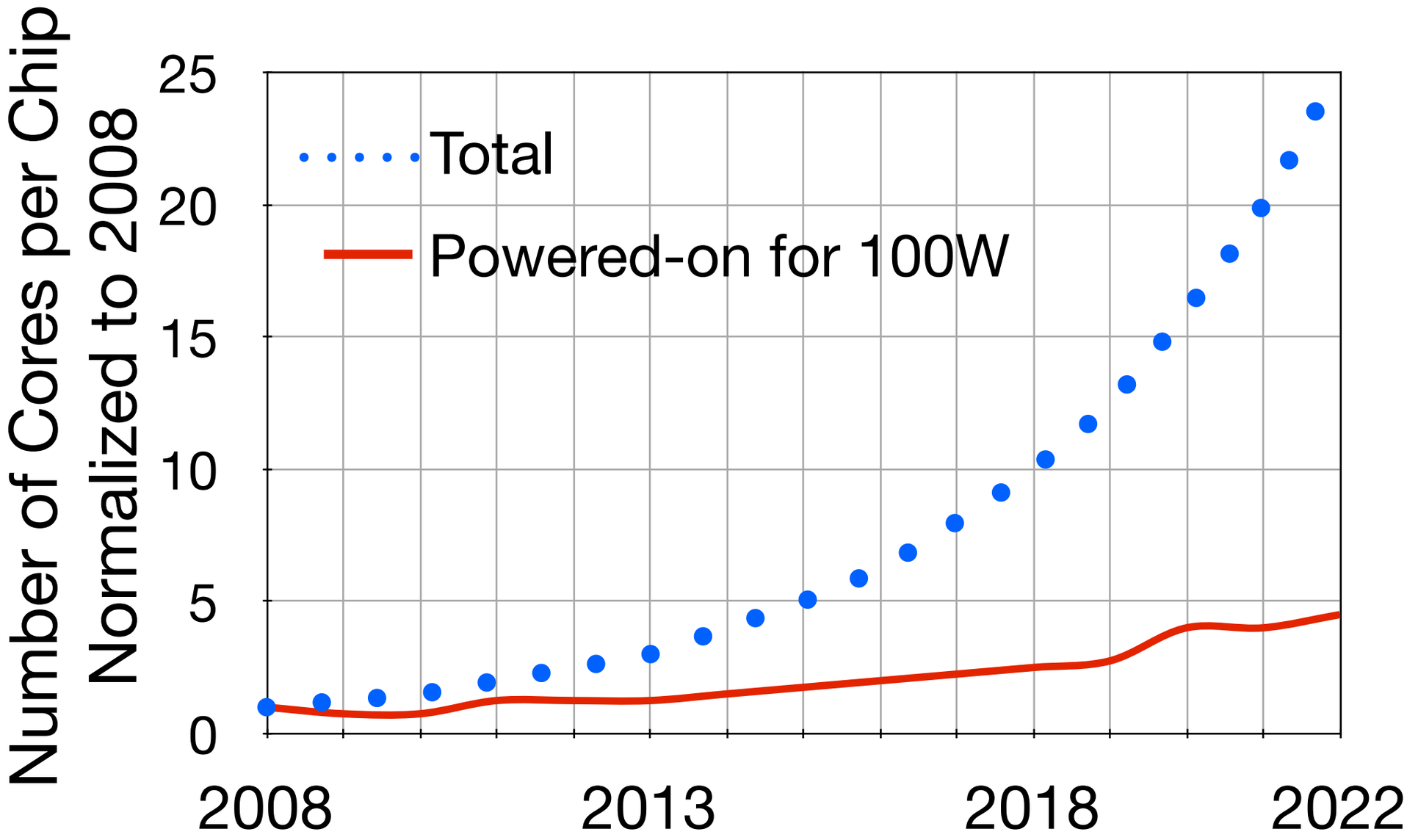

ITRS 2008

# The Power Wall



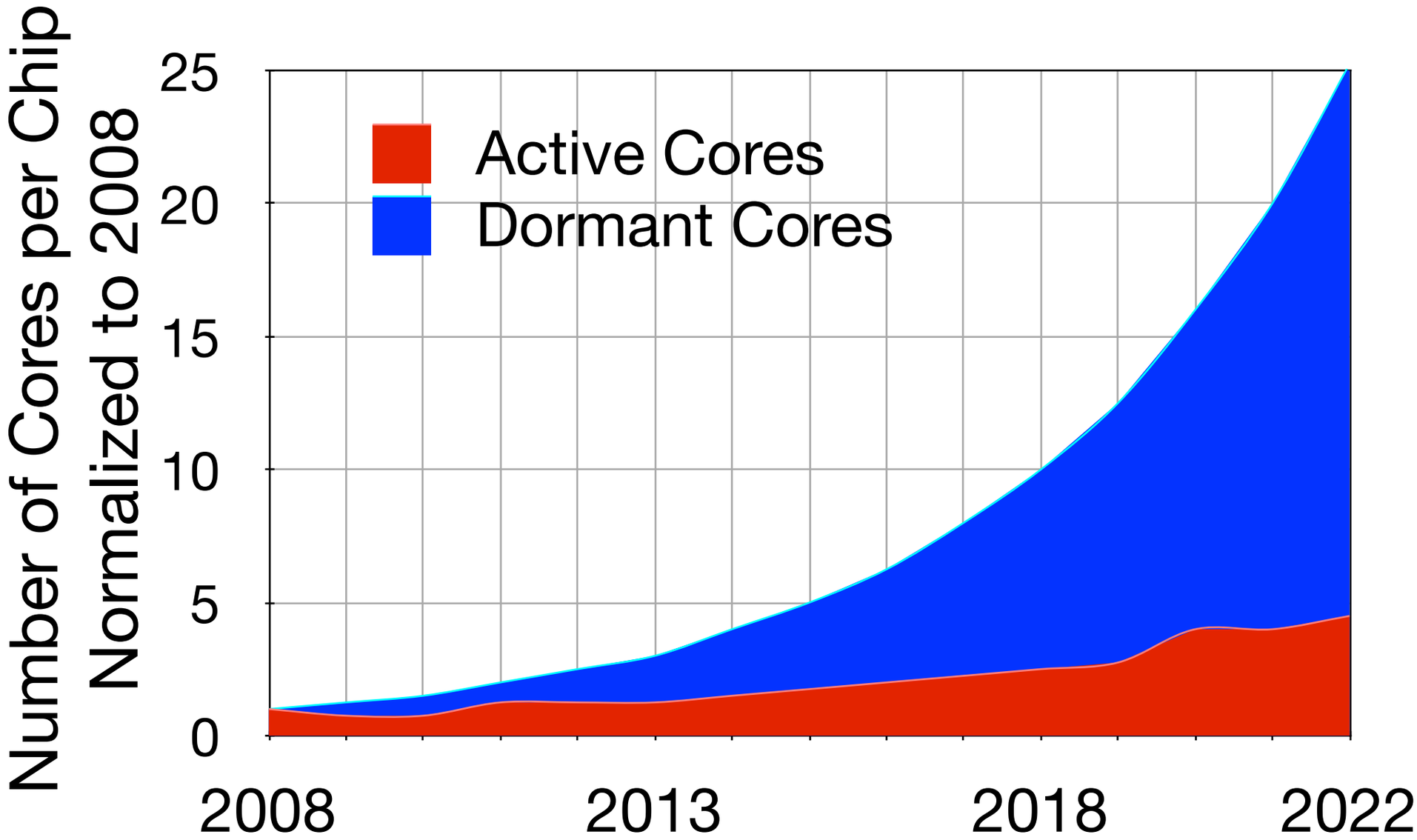

ITRS 2008

# The Power Wall



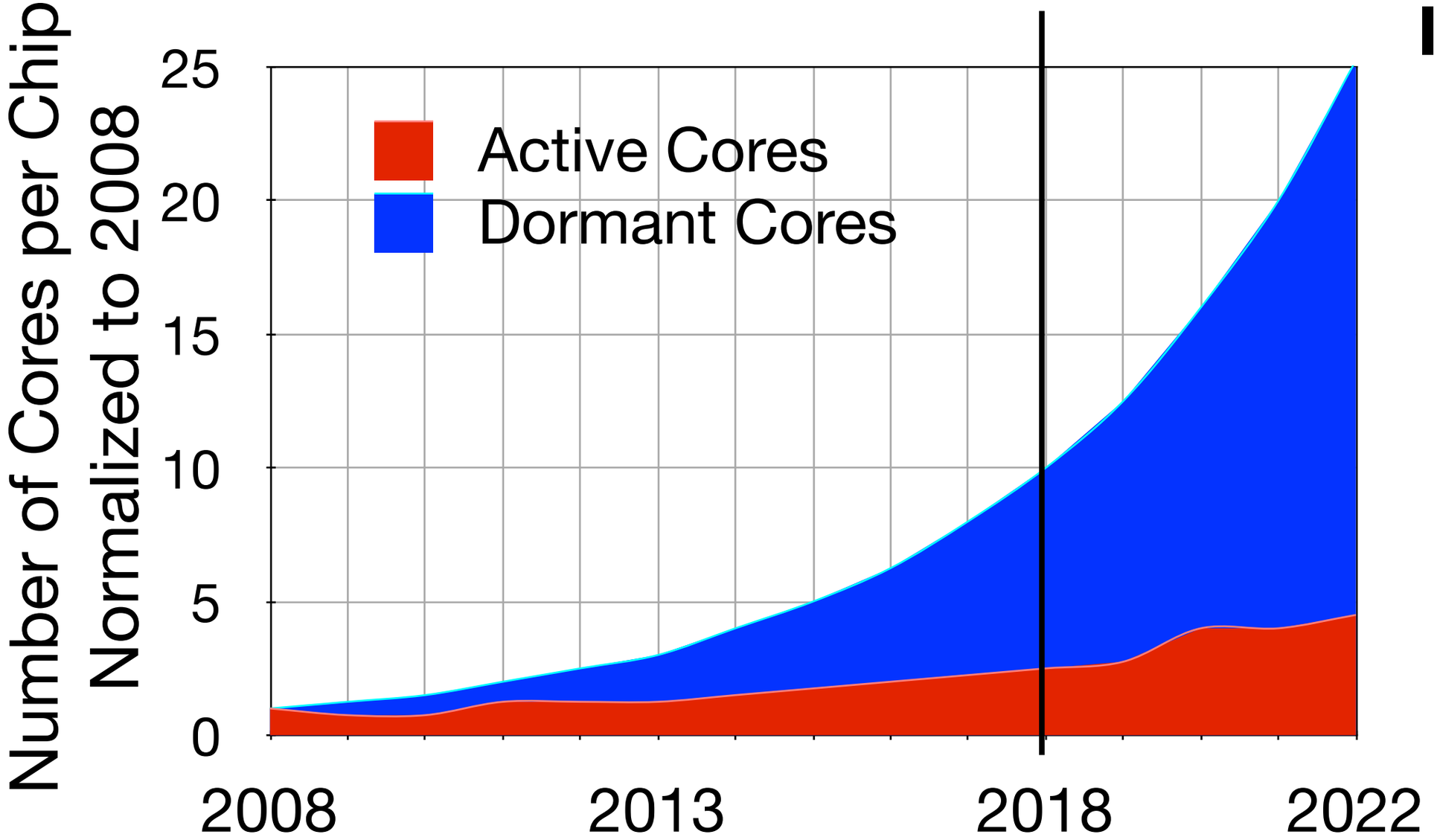

ITRS 2008

# The Power Wall



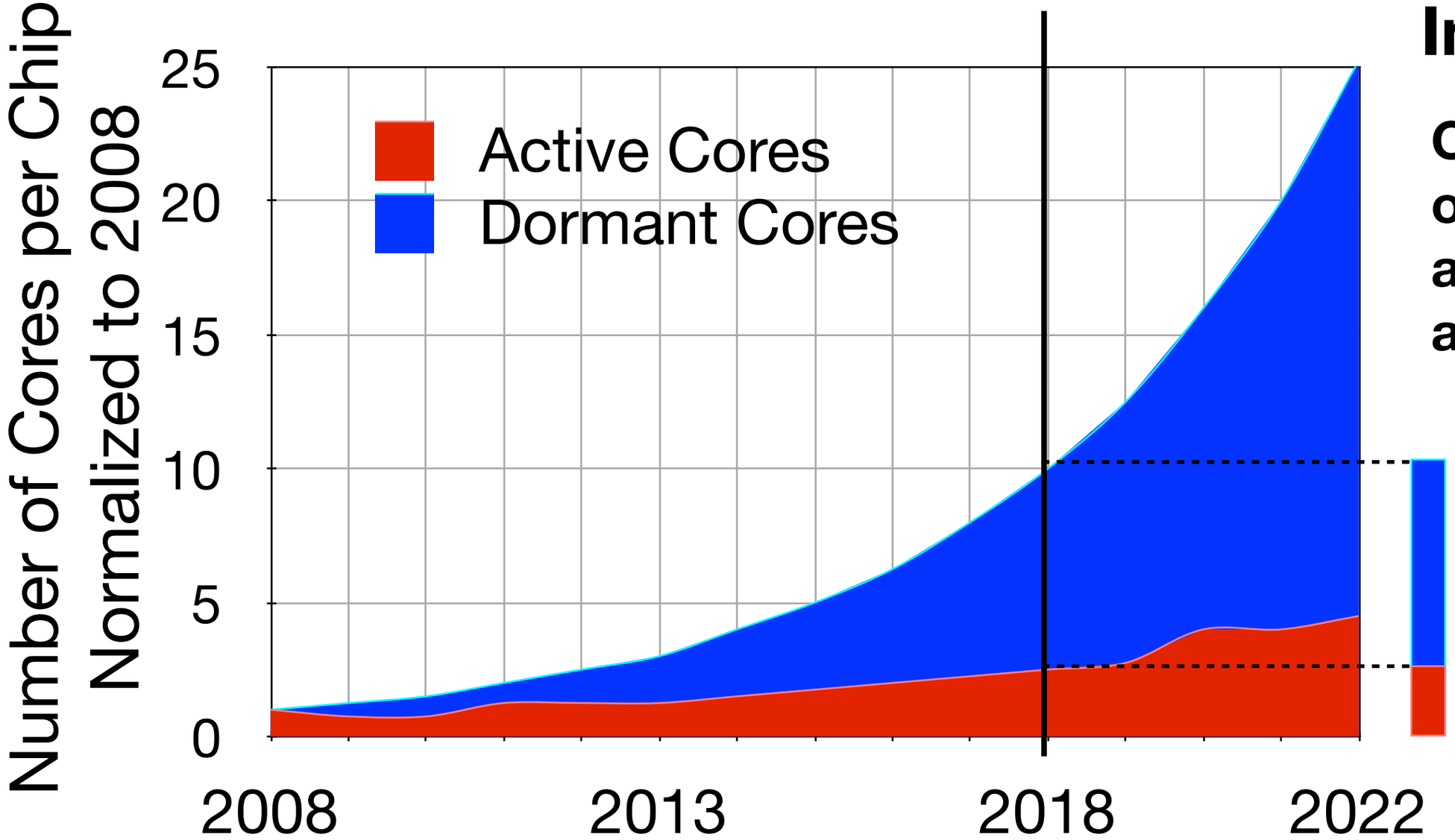

ITRS 2008

# The Power Wall




ITRS 2008

# The Power Wall




**In 9 years**

**Only ~25% of cores active at any time**

ITRS 2008

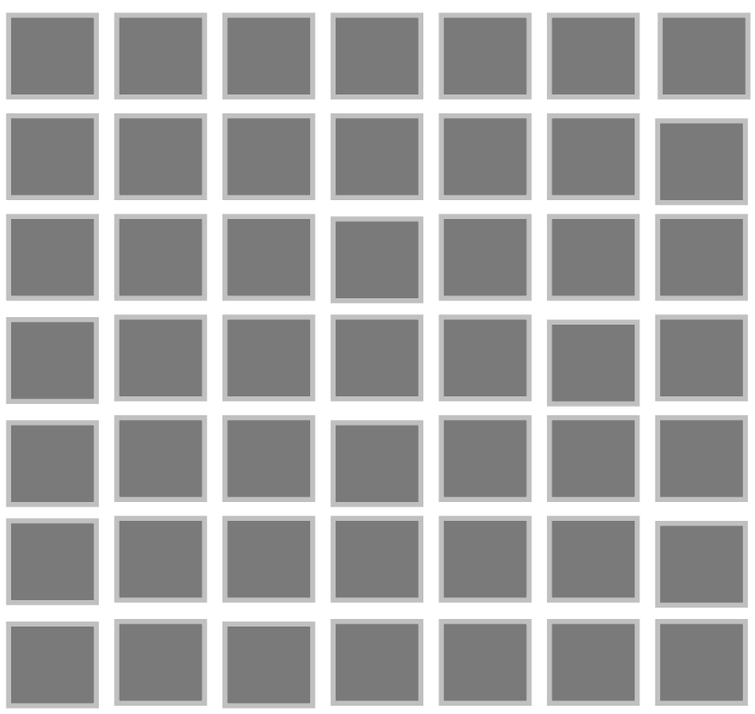
# The BubbleWrap Many-Core

# The BubbleWrap Many-Core

- Exploit dormant cores to accelerate sequential sections at the cost of a shorter per-core service life

# The BubbleWrap Many-Core

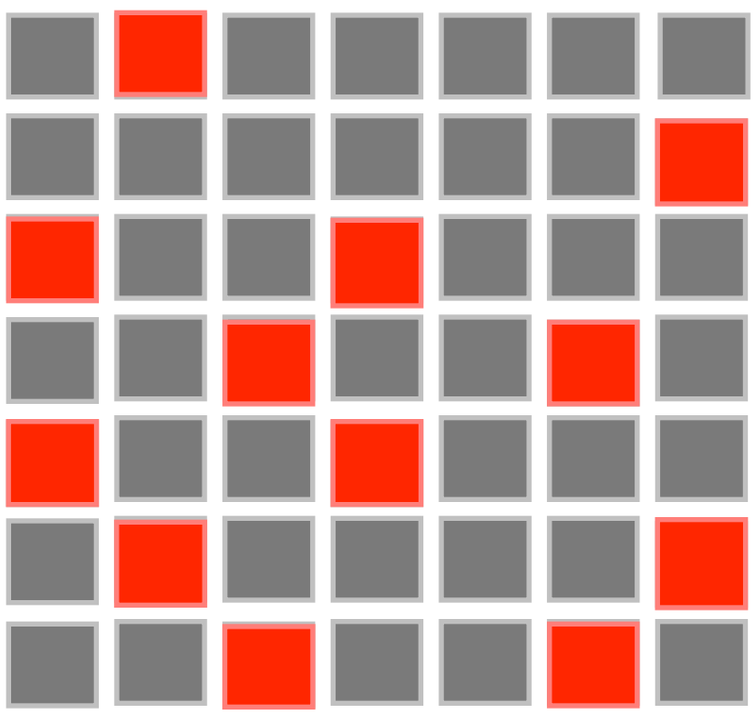
- Exploit dormant cores to accelerate sequential sections at the cost of a shorter per-core service life
  - Base: A homogeneous many-core



# The BubbleWrap Many-Core

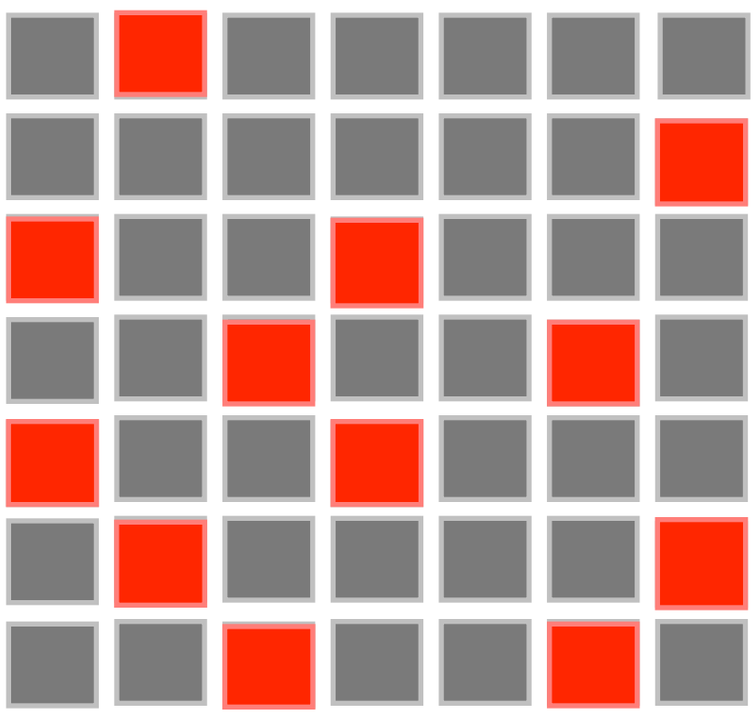
• Exploit dormant cores to accelerate sequential sections at the cost of a shorter per-core service life

- Base: A homogeneous many-core
- Throughput Cores



# The BubbleWrap Many-Core

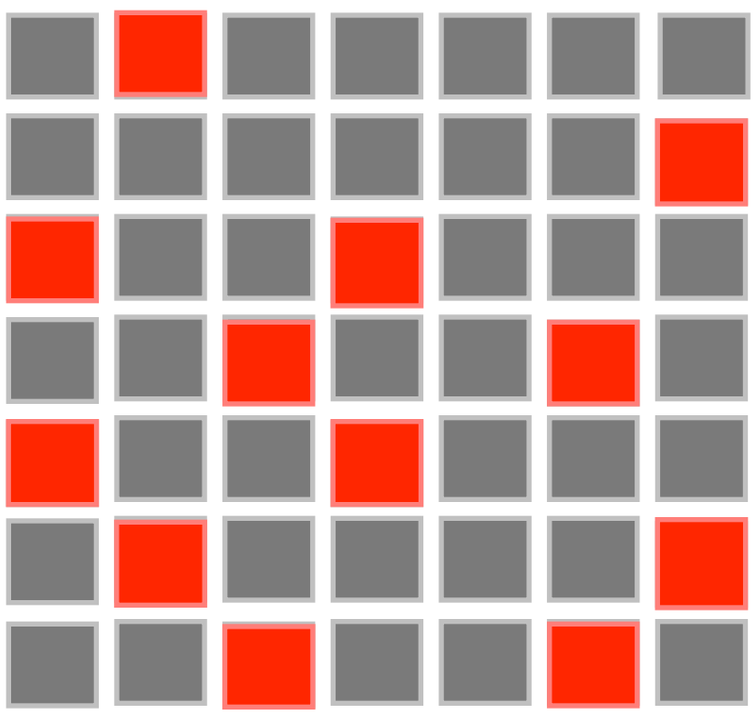
• Exploit dormant cores to accelerate sequential sections at the cost of a shorter per-core service life



- Base: A homogeneous many-core
- Throughput Cores
  - Most energy-efficient cores

# The BubbleWrap Many-Core

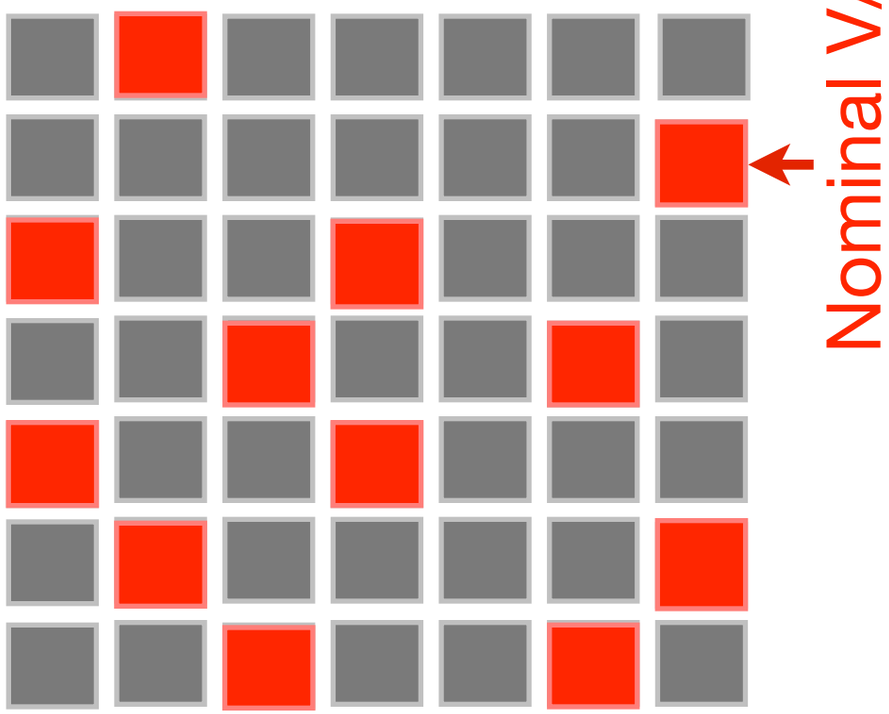
• Exploit dormant cores to accelerate sequential sections at the cost of a shorter per-core service life



- Base: A homogeneous many-core
- Throughput Cores
  - Most energy-efficient cores
  - Run parallel sections

# The BubbleWrap Many-Core

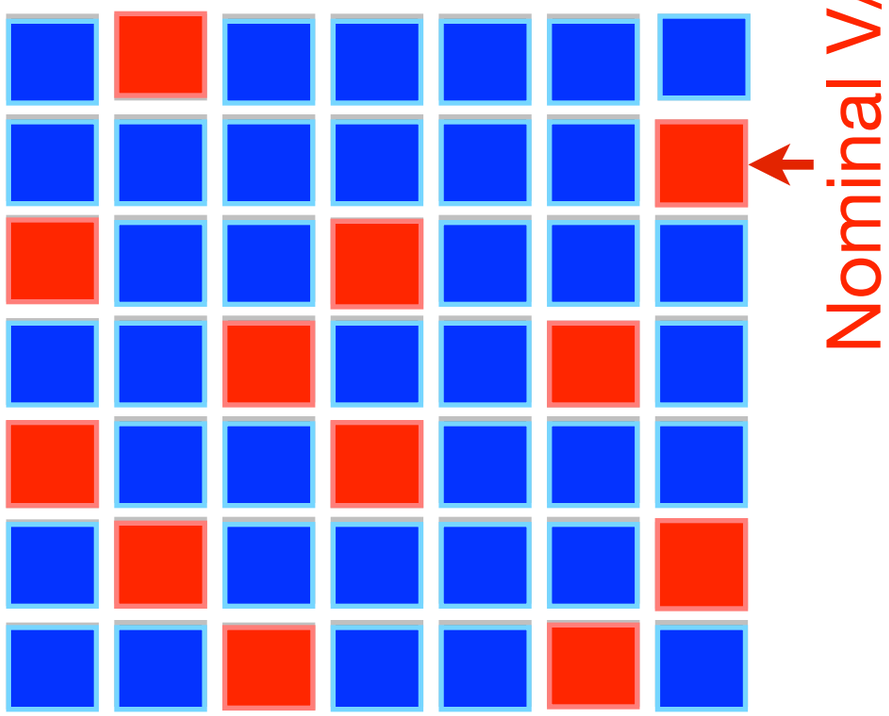
• Exploit dormant cores to accelerate sequential sections at the cost of a shorter per-core service life




- Base: A homogeneous many-core
- Throughput Cores
  - Most energy-efficient cores
  - Run parallel sections
  - Operate at nominal V/f

# The BubbleWrap Many-Core

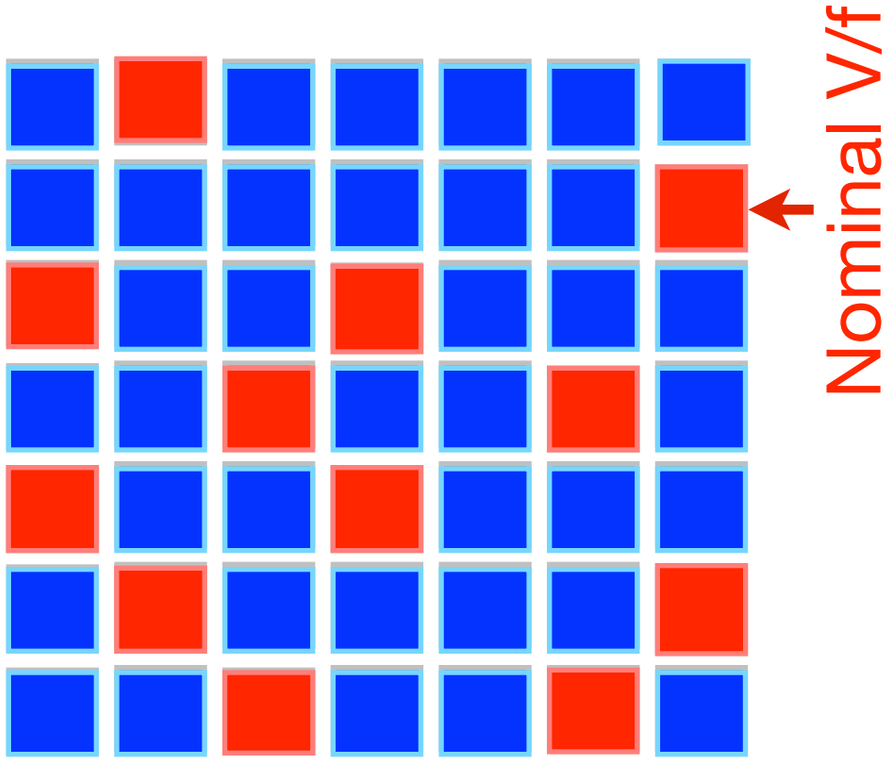
• Exploit dormant cores to accelerate sequential sections at the cost of a shorter per-core service life




- Base: A homogeneous many-core
- Throughput Cores
  - Most energy-efficient cores
  - Run parallel sections
  - Operate at nominal V/f
- Expendable Cores

# The BubbleWrap Many-Core

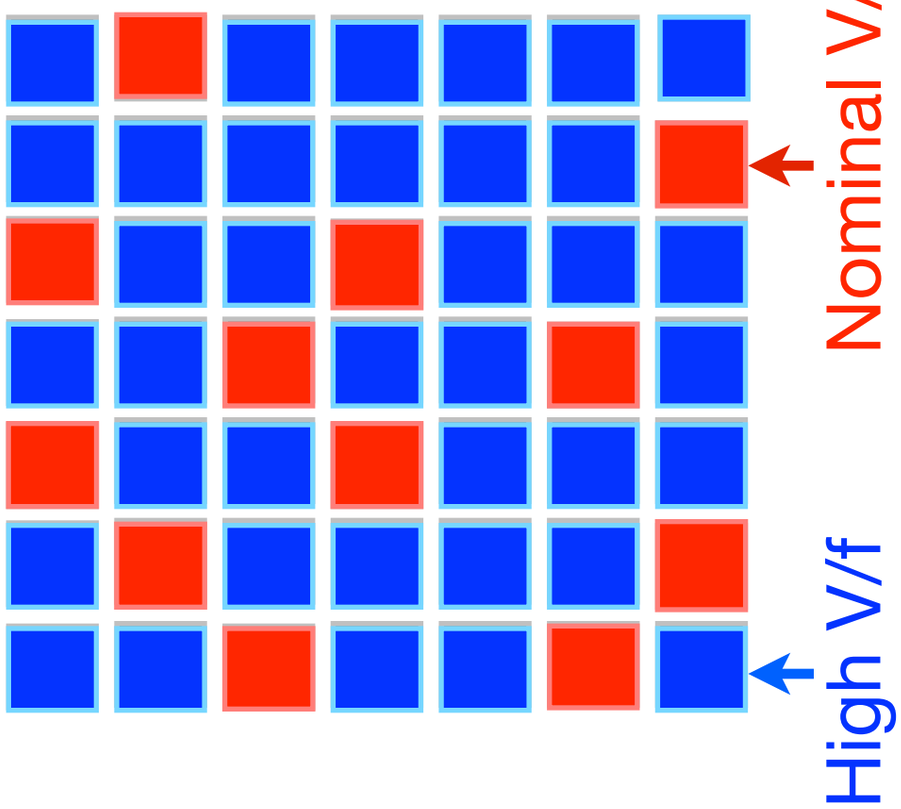
• Exploit dormant cores to accelerate sequential sections at the cost of a shorter per-core service life




- Base: A homogeneous many-core
- Throughput Cores
  - Most energy-efficient cores
  - Run parallel sections
  - Operate at nominal V/f
- Expendable Cores
  - Run sequential sections

# The BubbleWrap Many-Core

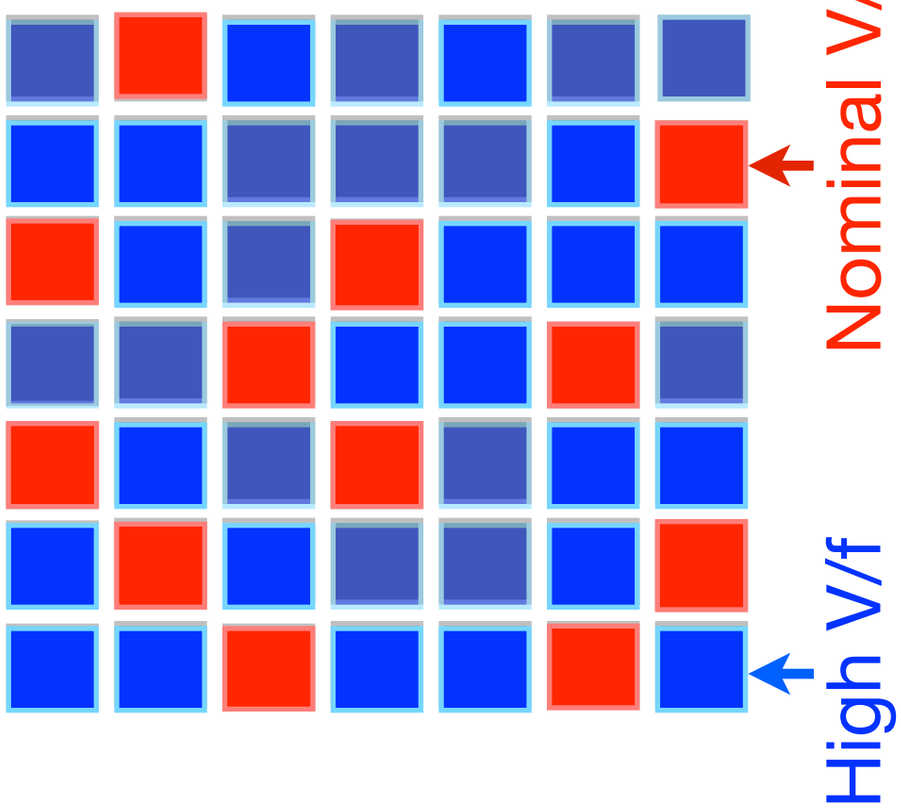
• Exploit dormant cores to accelerate sequential sections at the cost of a shorter per-core service life




- Base: A homogeneous many-core
- Throughput Cores
  - Most energy-efficient cores
  - Run parallel sections
  - Operate at nominal V/f
- Expendable Cores
  - Run sequential sections
  - Operate at elevated V/f

# The BubbleWrap Many-Core

• Exploit dormant cores to accelerate sequential sections at the cost of a shorter per-core service life




- Base: A homogeneous many-core
- Throughput Cores
  - Most energy-efficient cores
  - Run parallel sections
  - Operate at nominal V/f
- Expendable Cores
  - Run sequential sections
  - Operate at elevated V/f
  - Discarded early due to shorter service life (Popped like BubbleWrap)

# Core Aging

# Core Aging

- Manifestation: Progressive slow-down in logic as the core is being used

# Core Aging

- Manifestation: Progressive slow-down in logic as the core is being used
- Main contributor: Bias Temperature Instability (BTI)

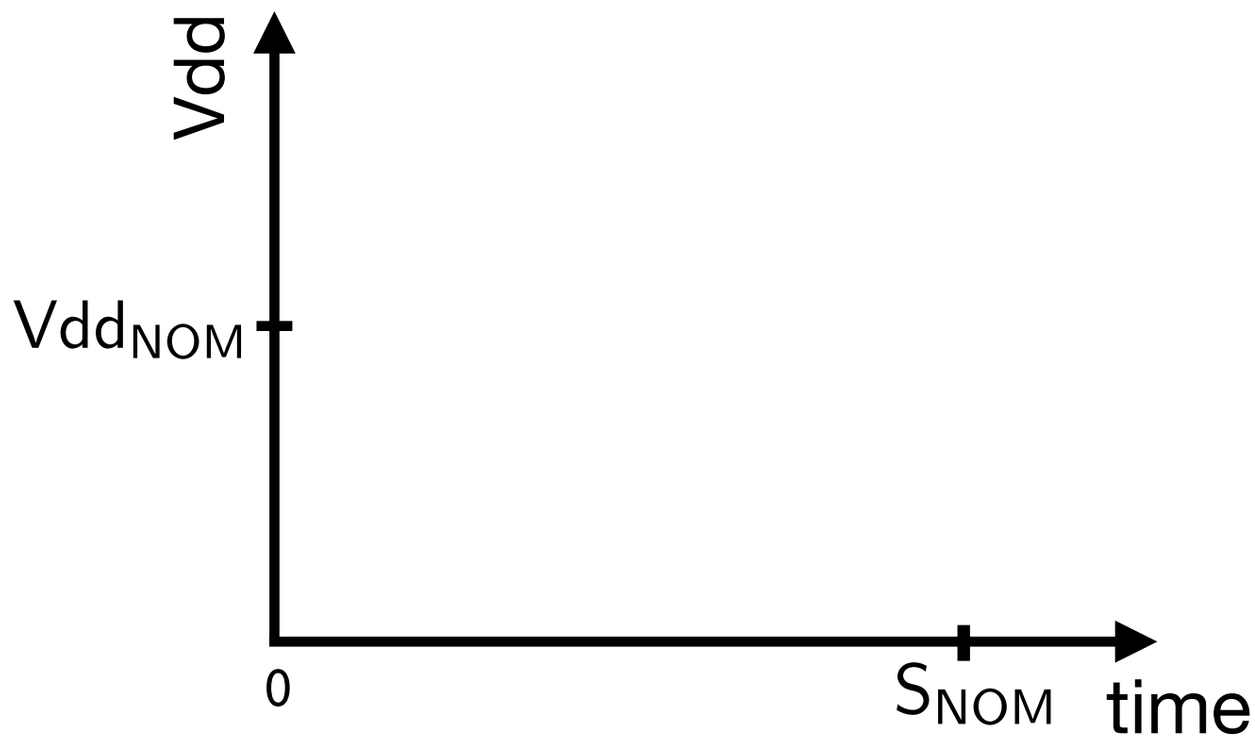
# Core Aging

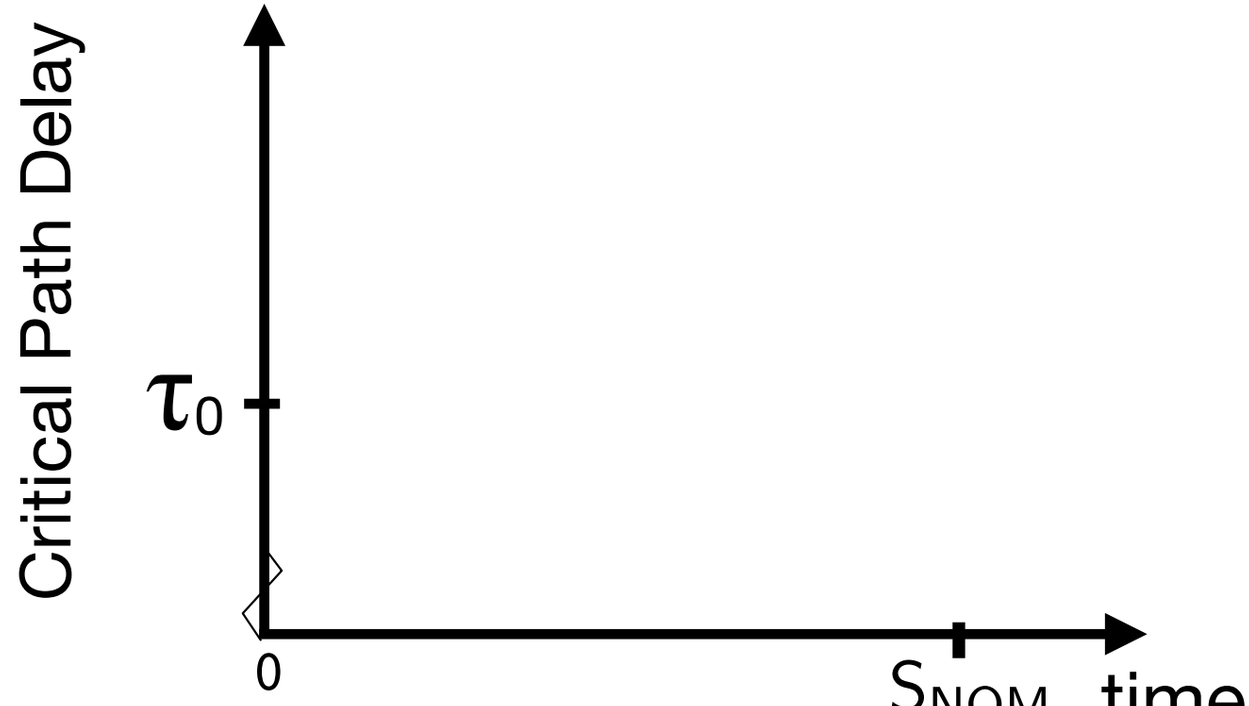
- Manifestation: Progressive slow-down in logic as the core is being used
- Main contributor: Bias Temperature Instability (BTI)
- Induces increase in critical path delays $\propto$ time$^{const.<1}$

# Core Aging

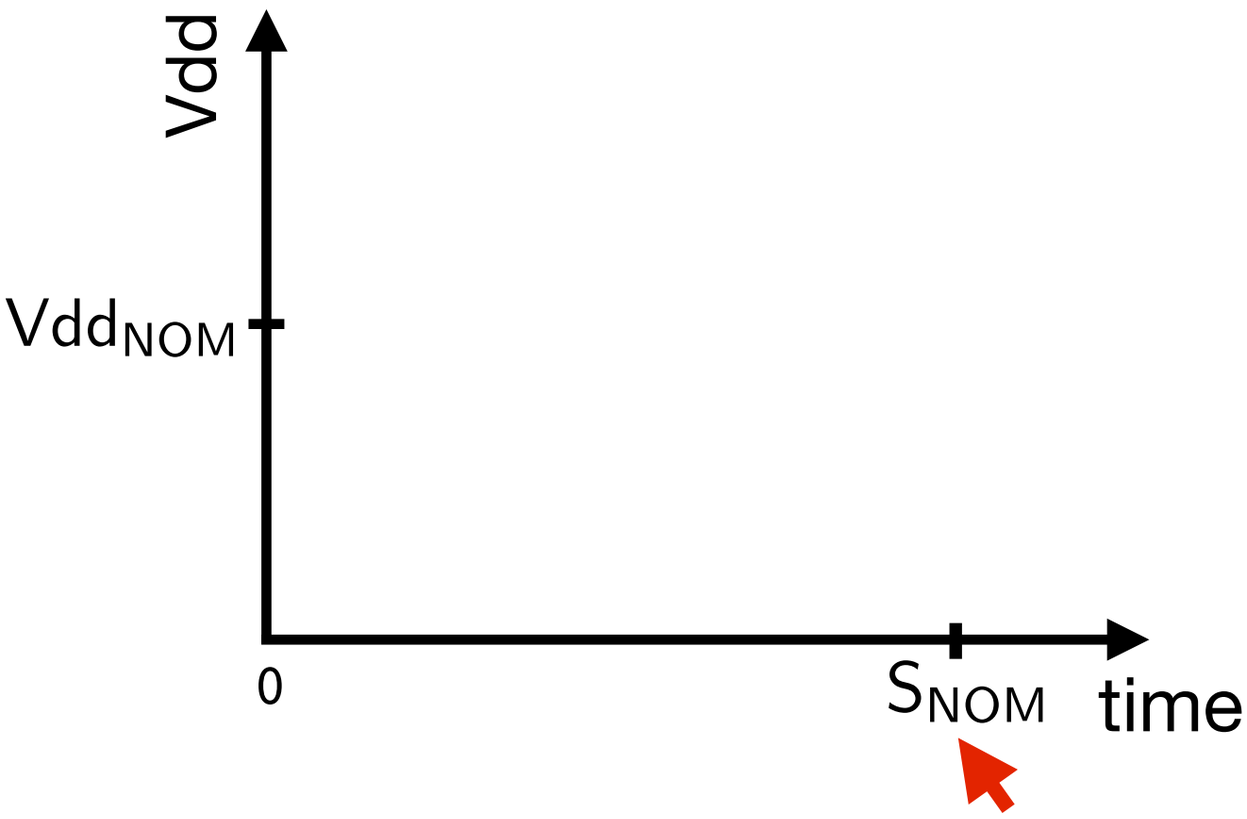
- Manifestation: Progressive slow-down in logic as the core is being used
- Main contributor: Bias Temperature Instability (BTI)
  - Induces increase in critical path delays $\propto$ time$^{const.<1}$
- Aging rate: Exponential dependence on Vdd and T

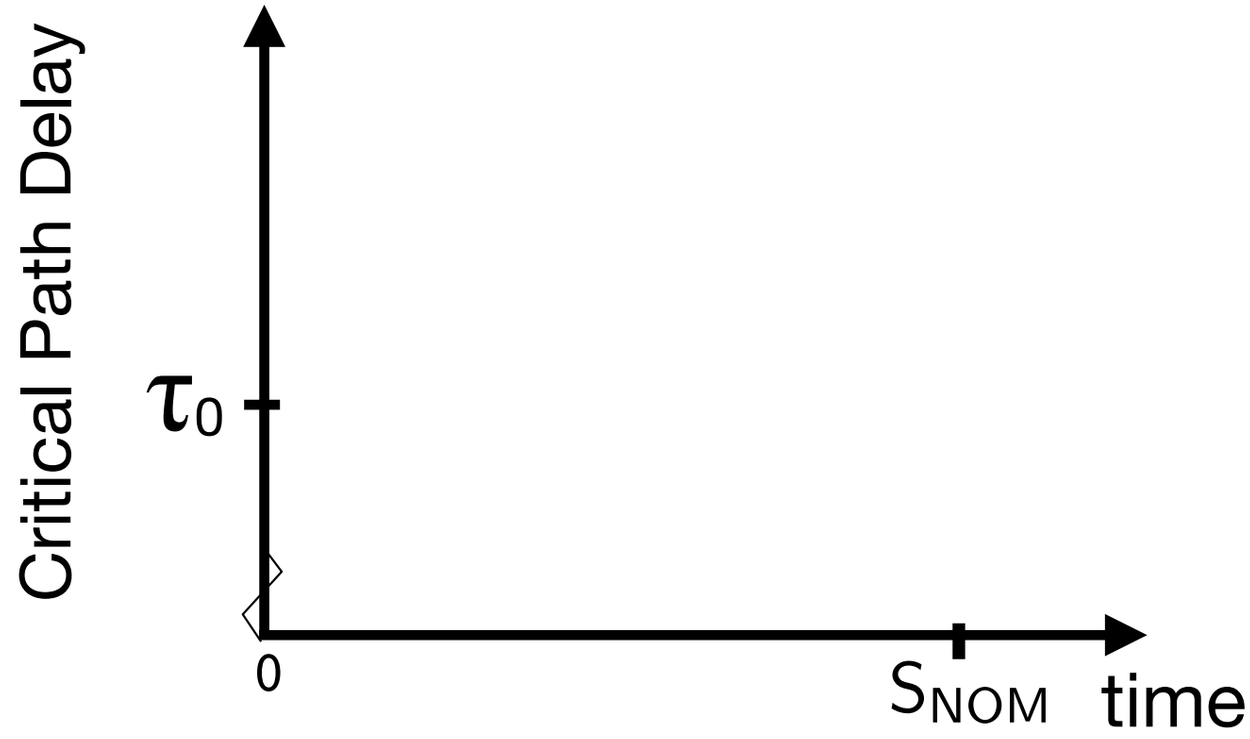
# Aging-induced Degradation



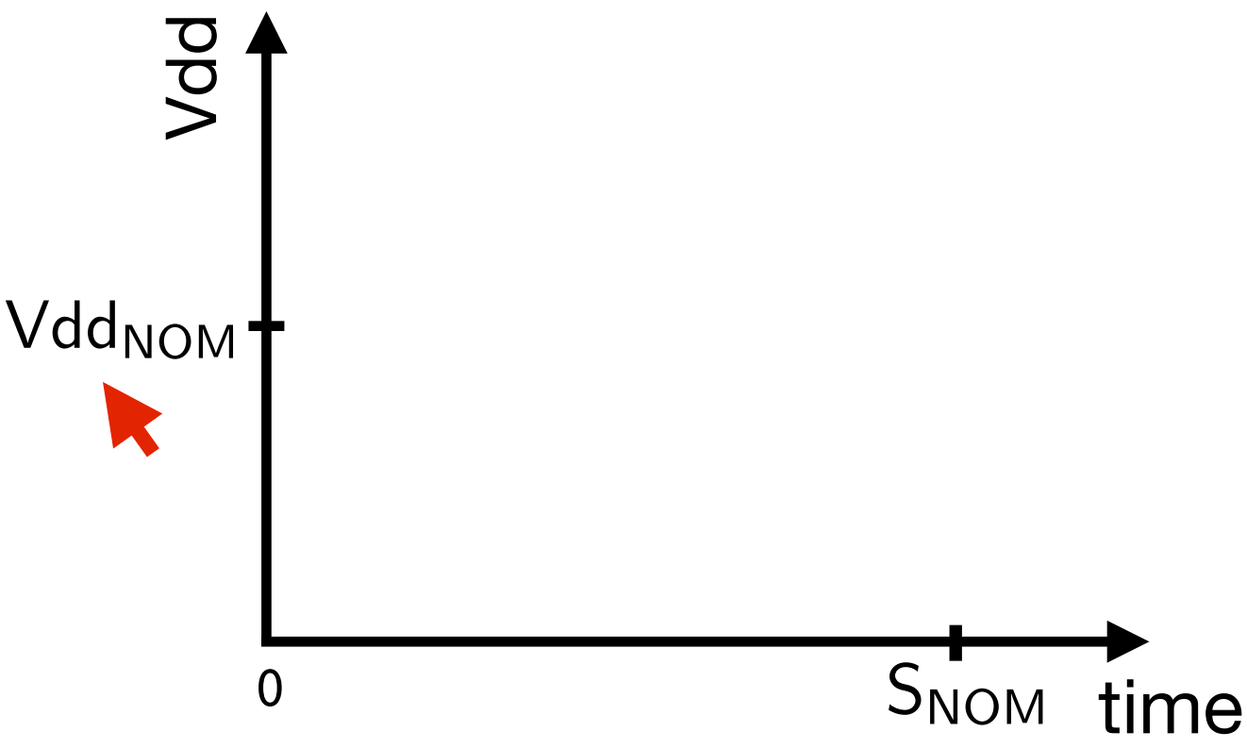



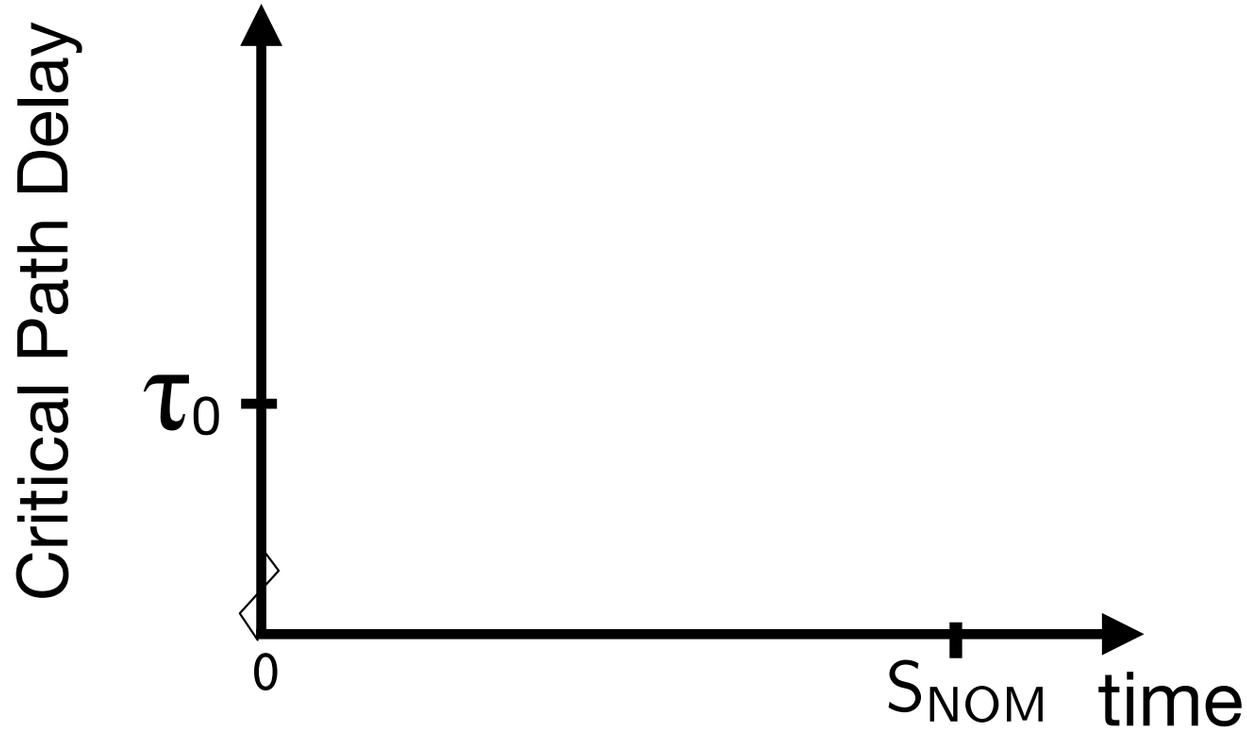
# Aging-induced Degradation



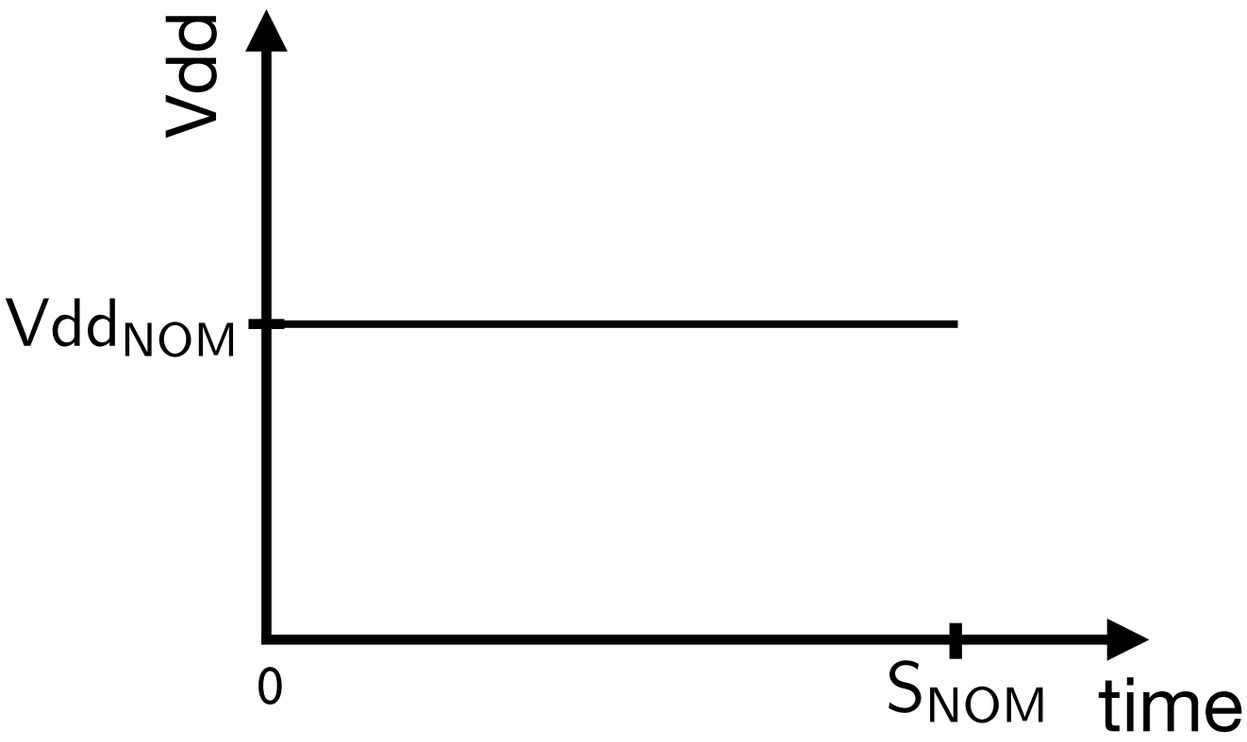



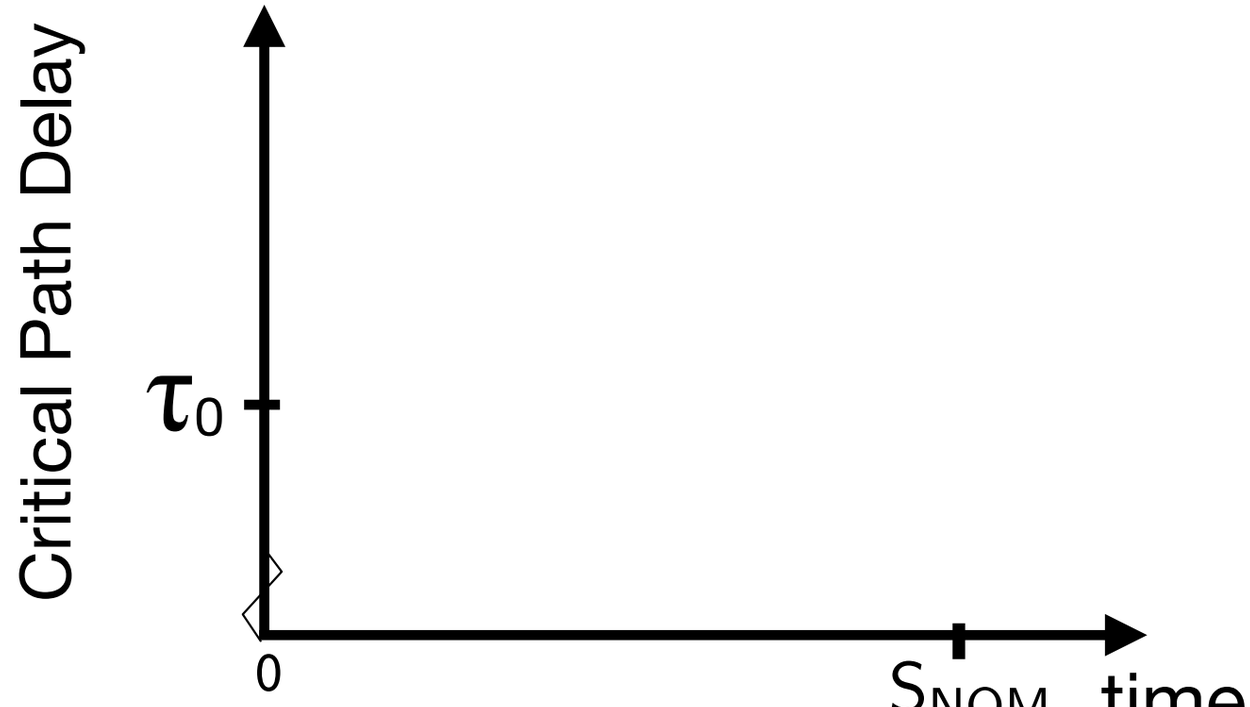
# Aging-induced Degradation



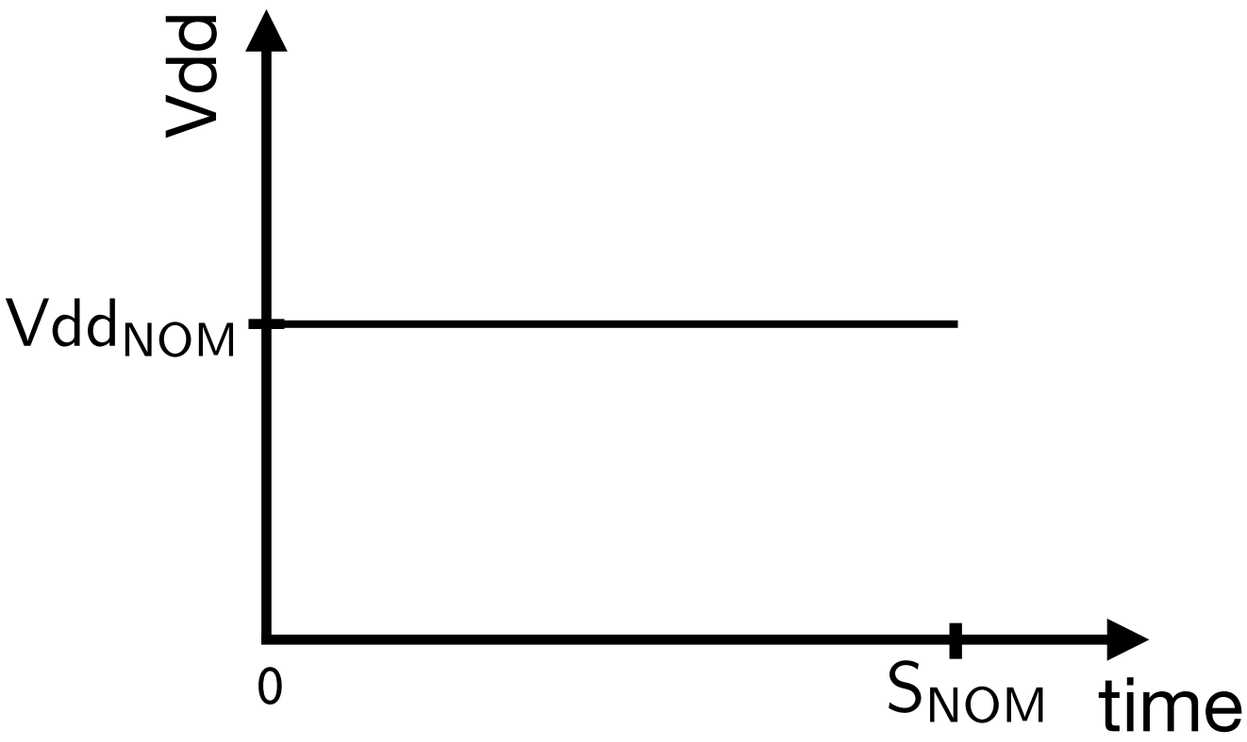



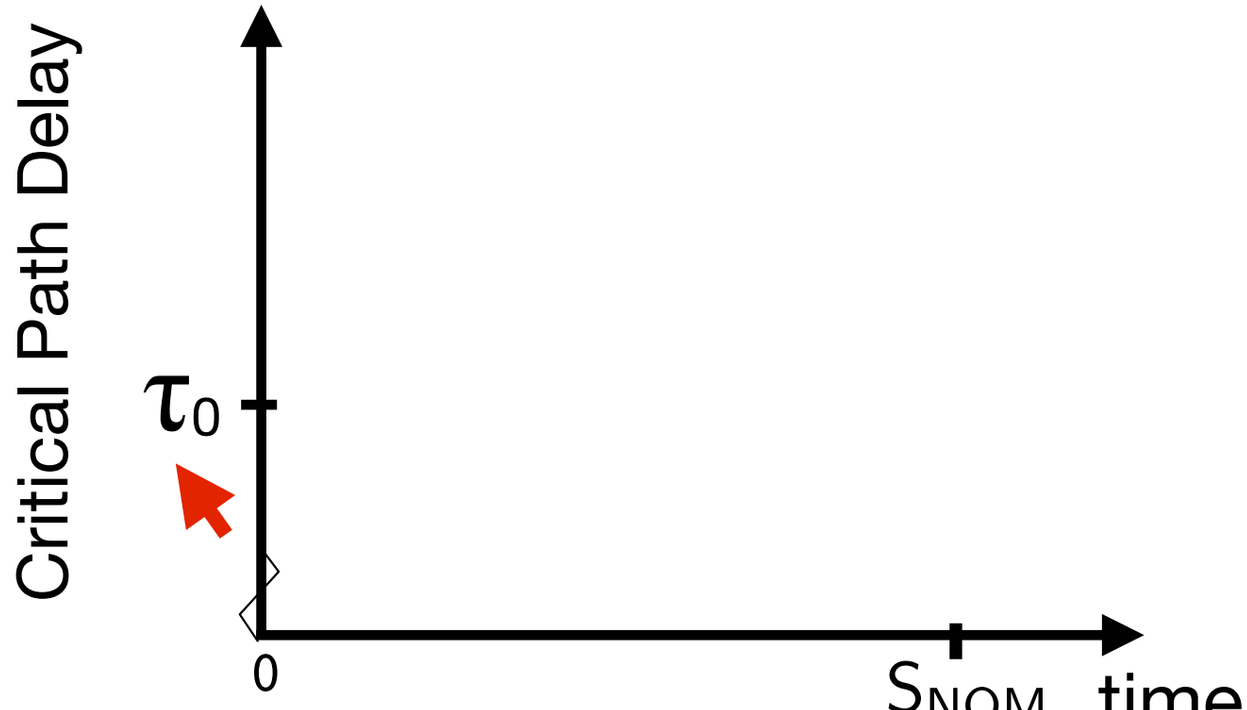
# Aging-induced Degradation



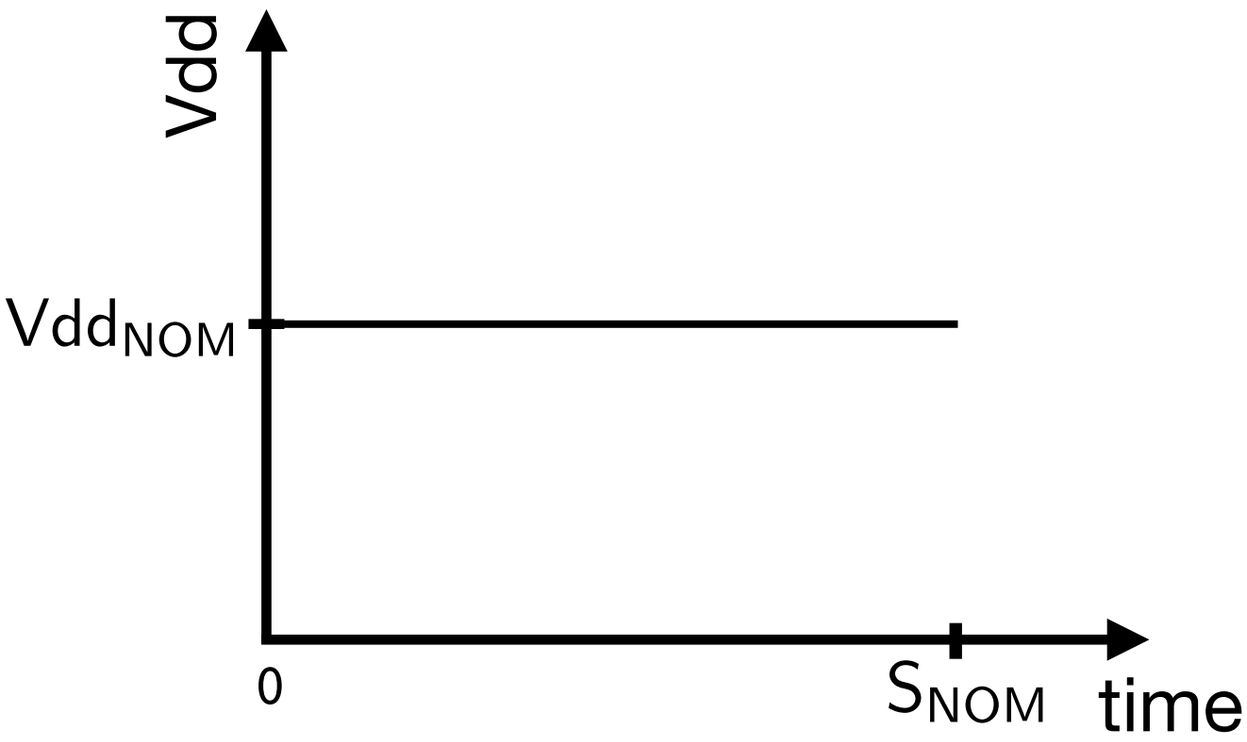



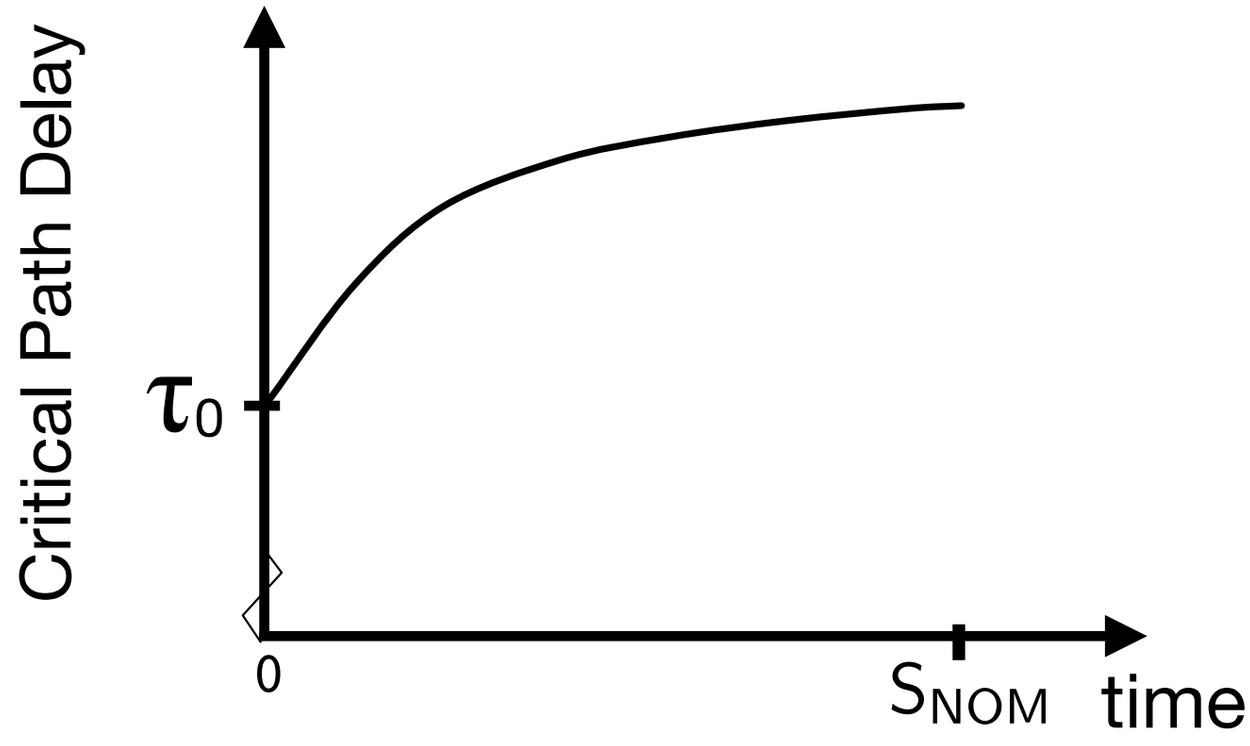
# Aging-induced Degradation



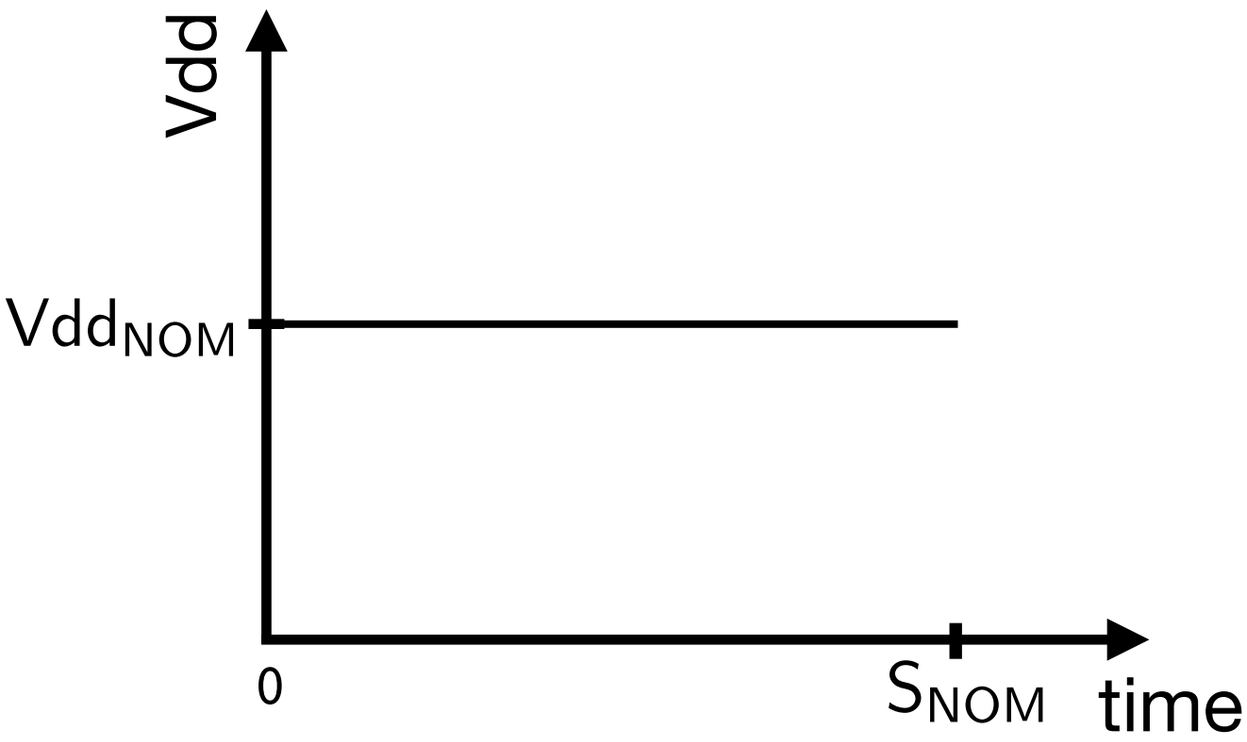
# Aging-induced Degradation

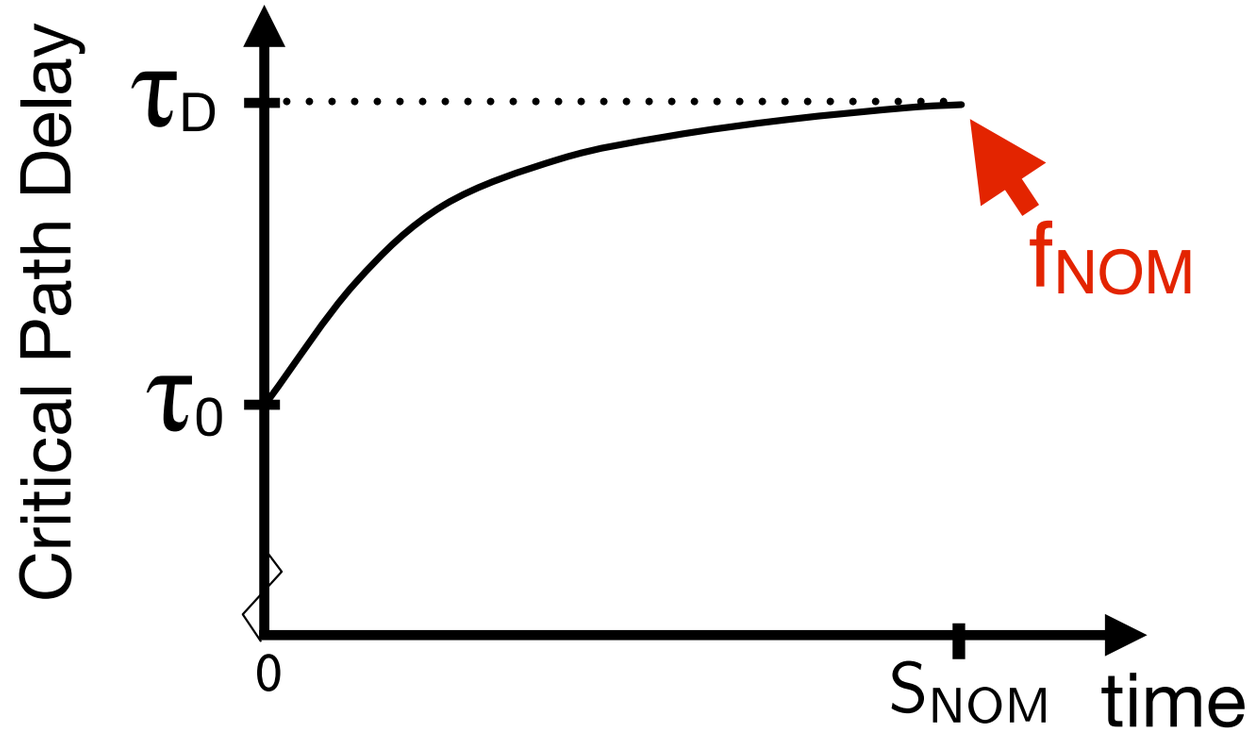



- BTI-induced increase in critical path delays $\propto$ $time^{const.<1}$

# Aging-induced Degradation

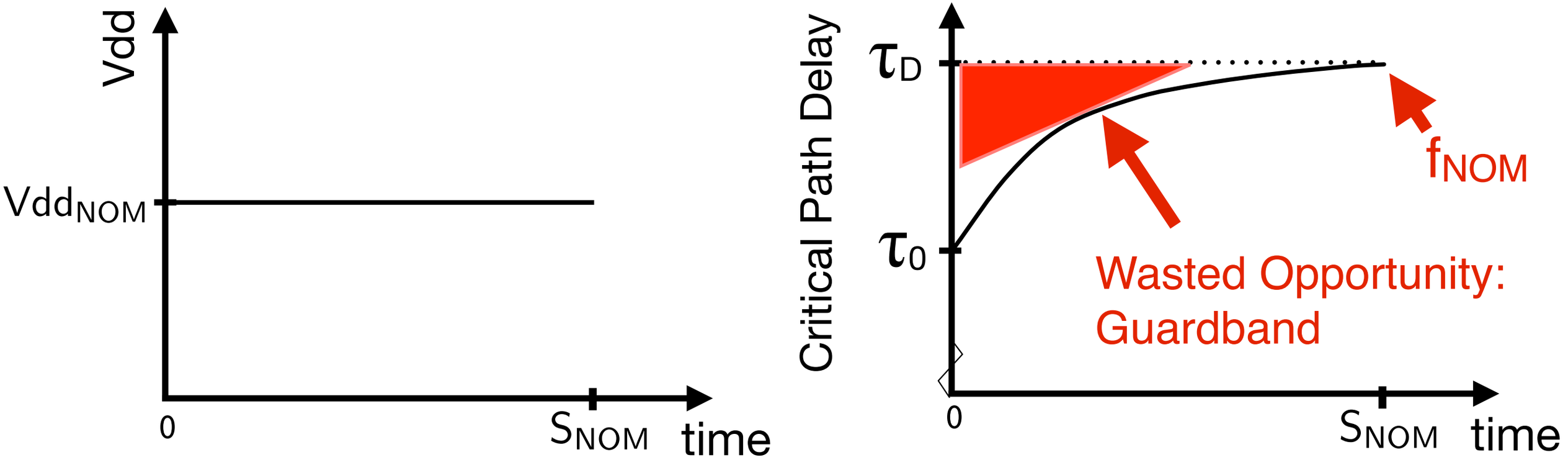



- BTI-induced increase in critical path delays $\propto$ time$^{const.<1}$
- $f_{NOM}$ set by the delay at the end of the service-life ($S_{NOM}$)

$$f_{NOM} = 1/\tau_D$$

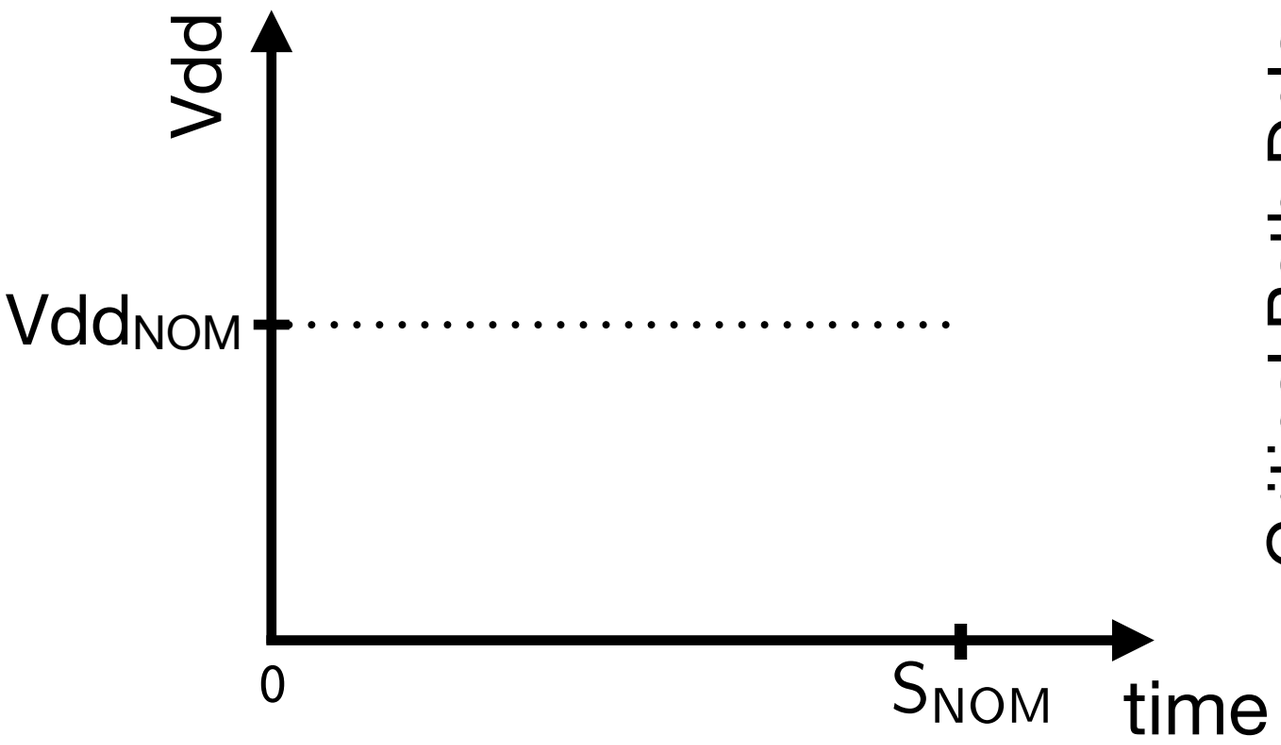
# Aging-induced Degradation

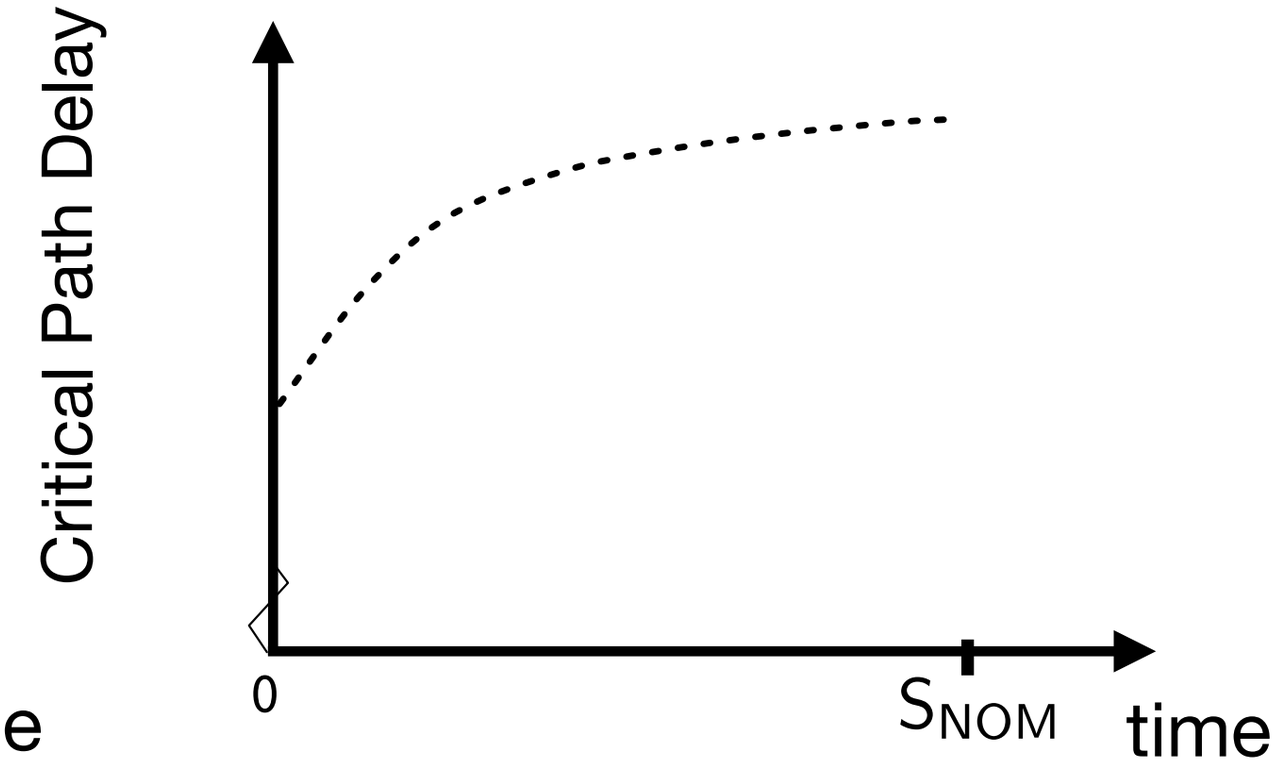



- BTI-induced increase in critical path delays $\propto$ time$^{const.<1}$
- $f_{NOM}$ set by the delay at the end of the service-life ($S_{NOM}$)

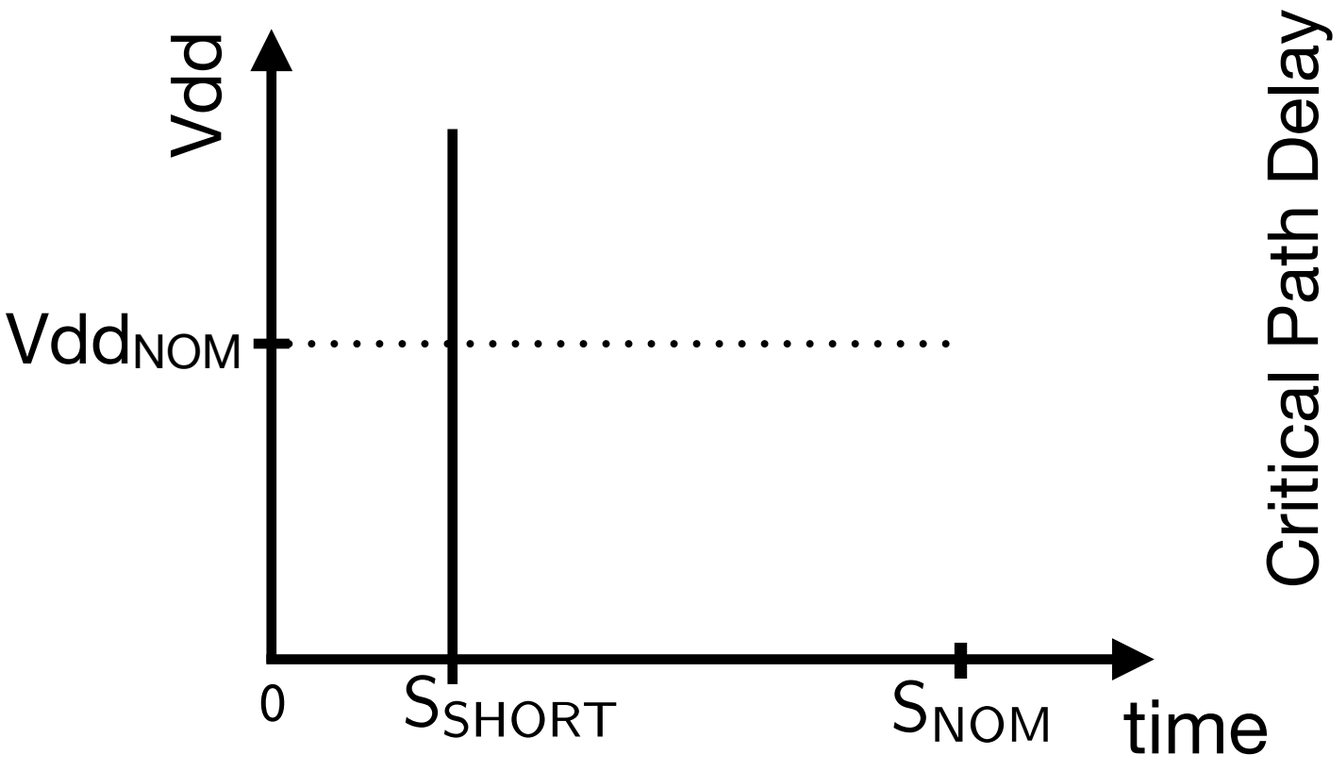
$$f_{NOM} = 1/\tau_D$$

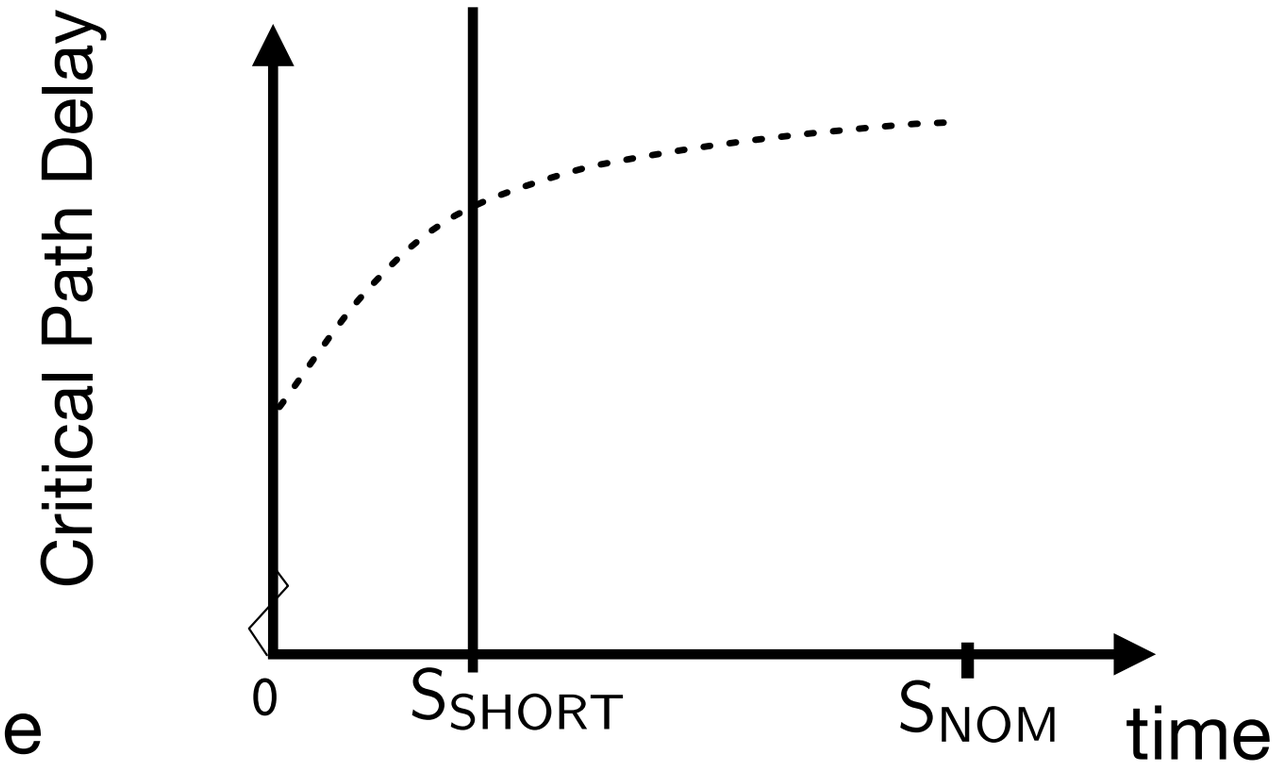
# Impact of Operation at Higher V/f on Aging



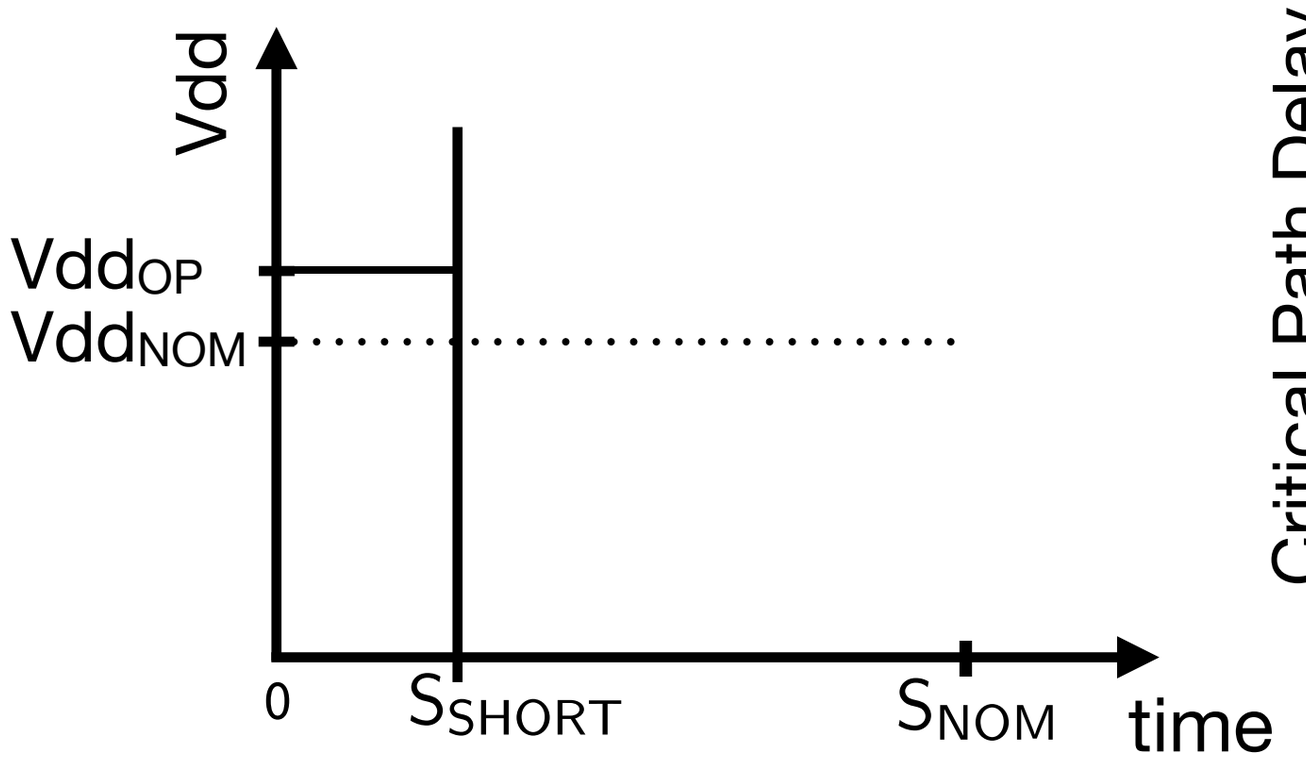



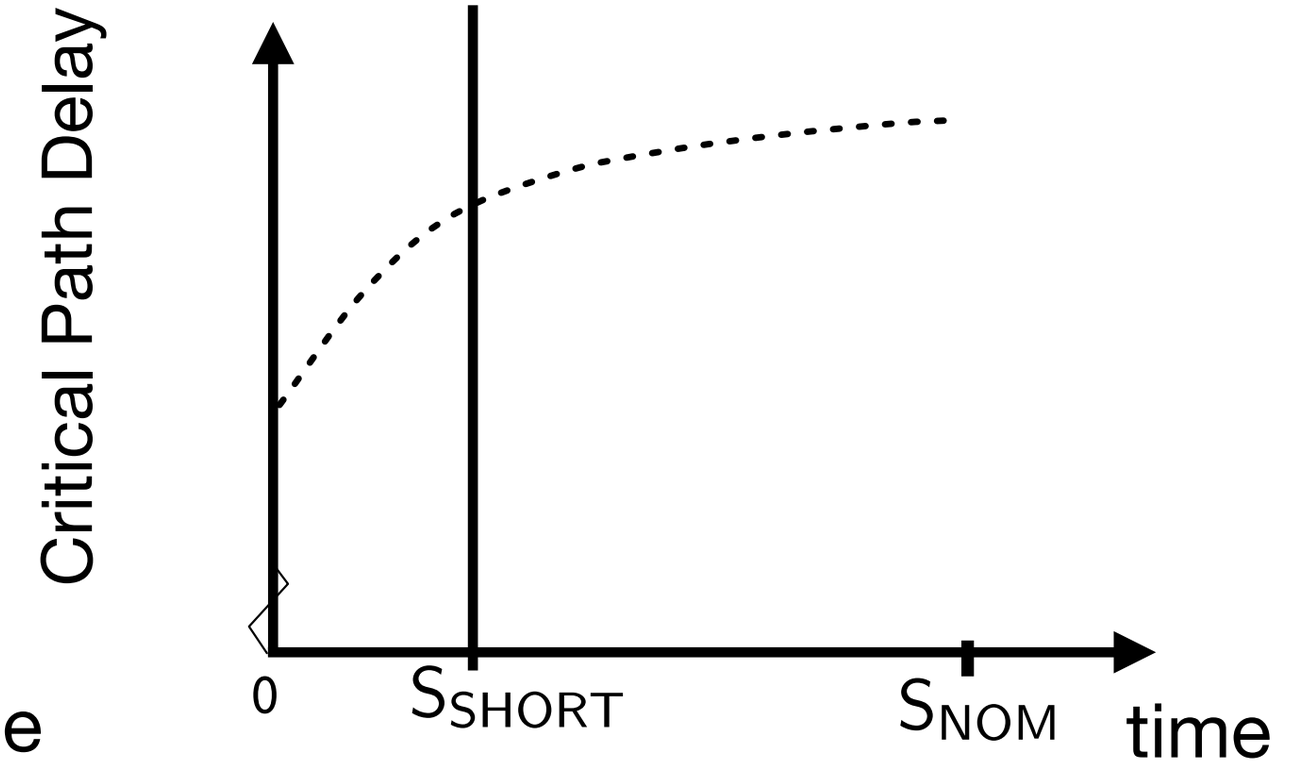
# Impact of Operation at Higher V/f on Aging



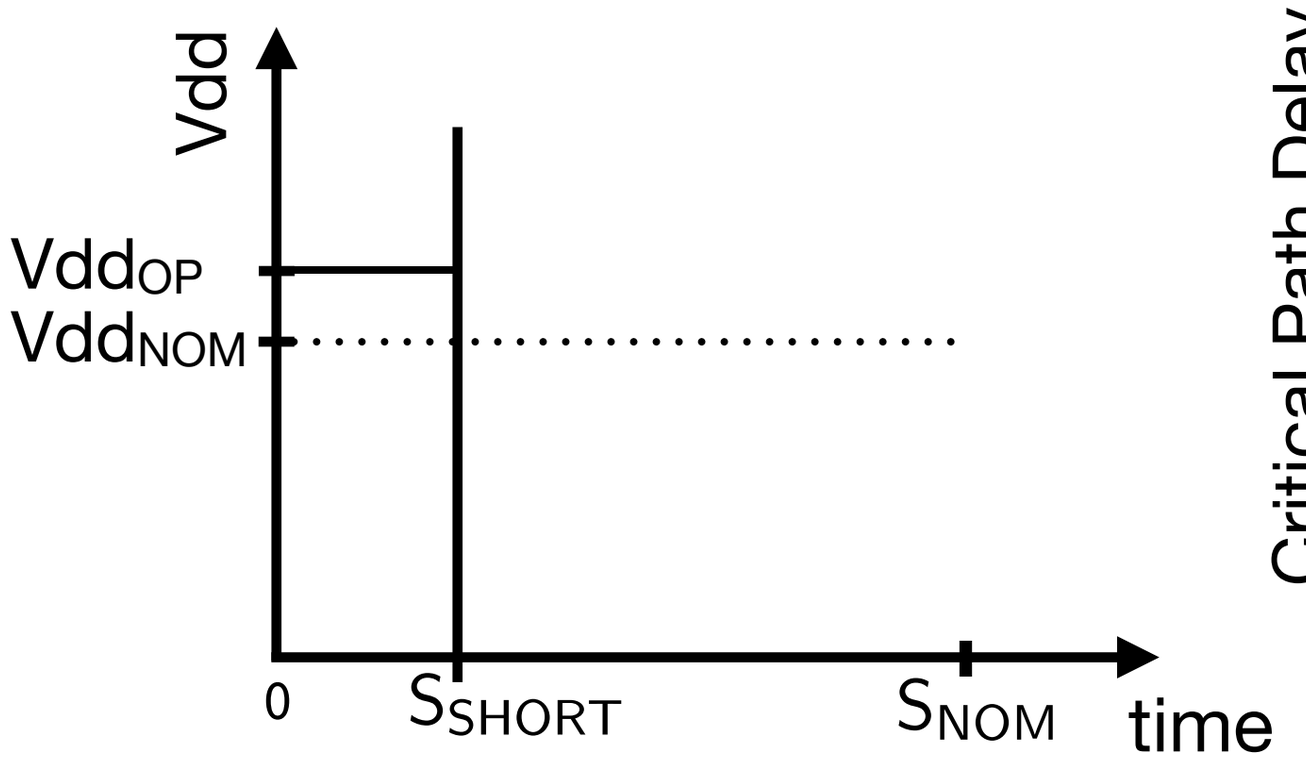
# Impact of Operation at Higher V/f on Aging

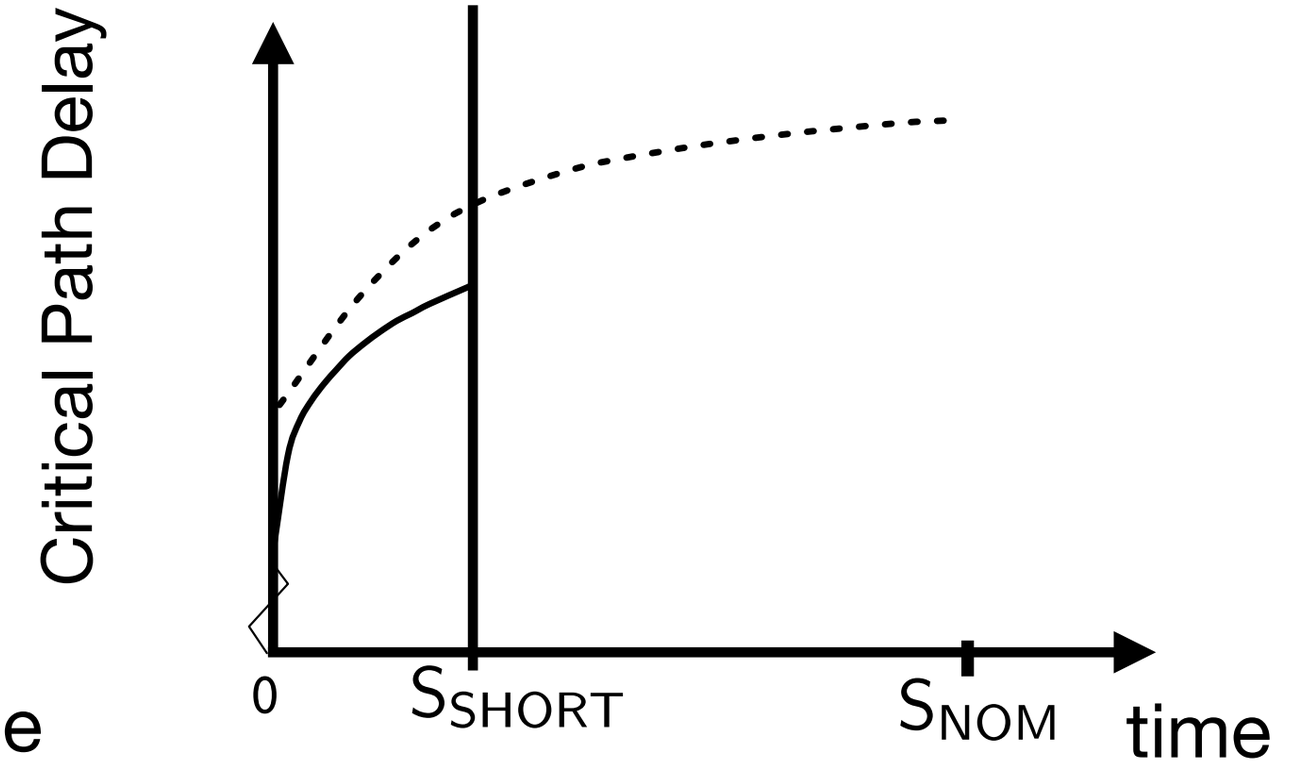



- Higher Vdd: $Vdd_{OP}$ >> $Vdd_{NOM}$

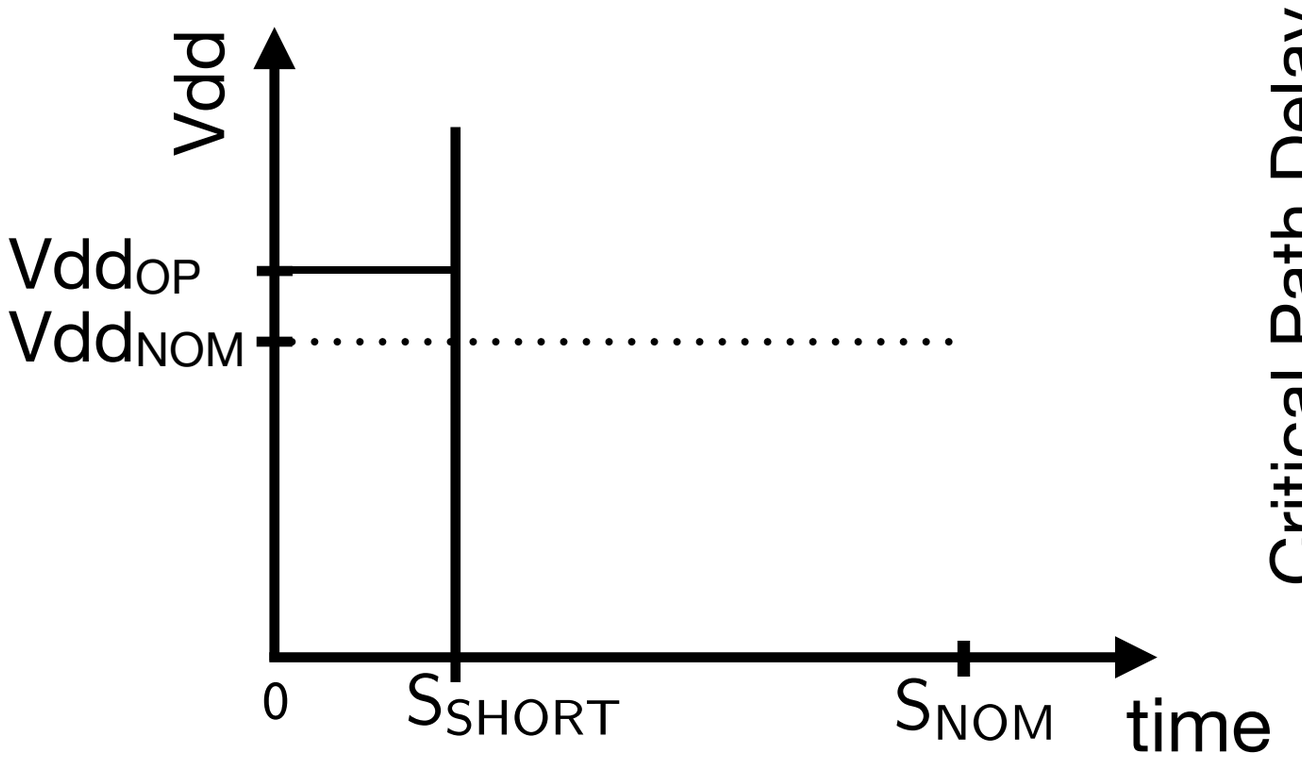
# Impact of Operation at Higher V/f on Aging

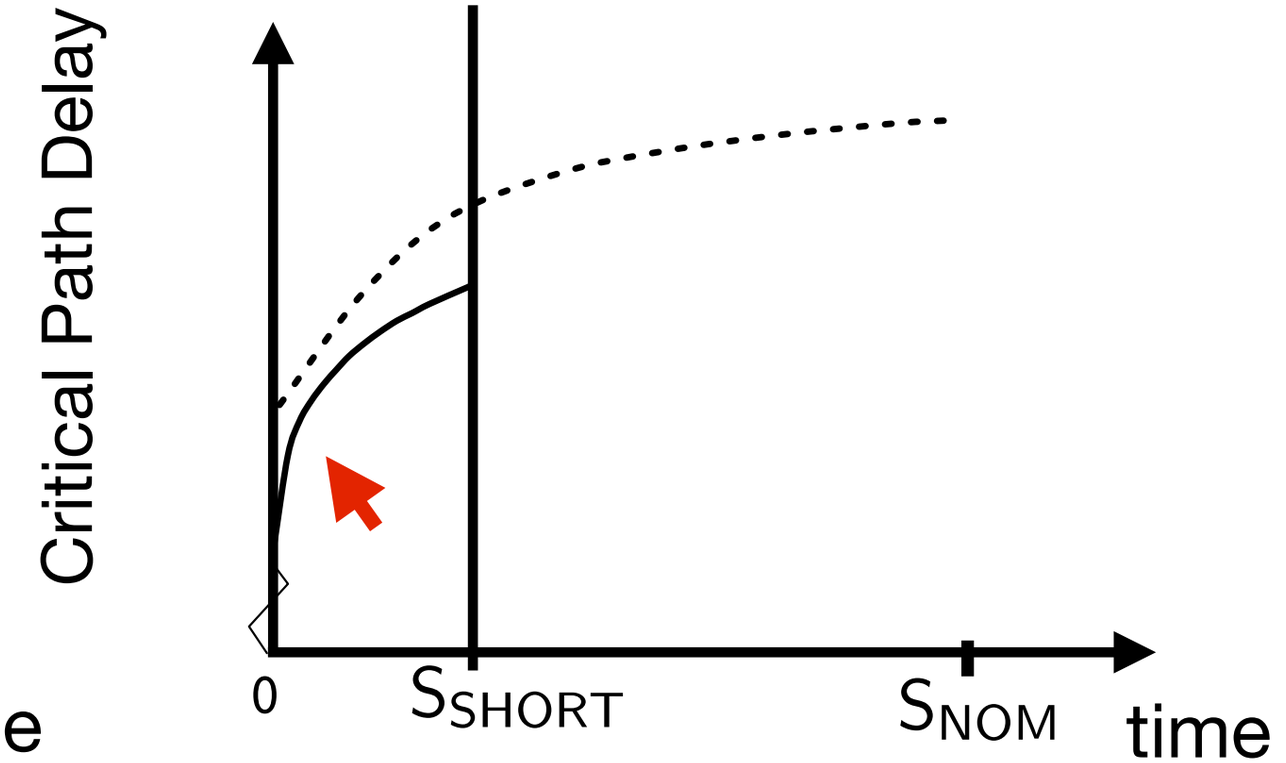



- Higher Vdd: $Vdd_{OP}$ >> $Vdd_{NOM}$

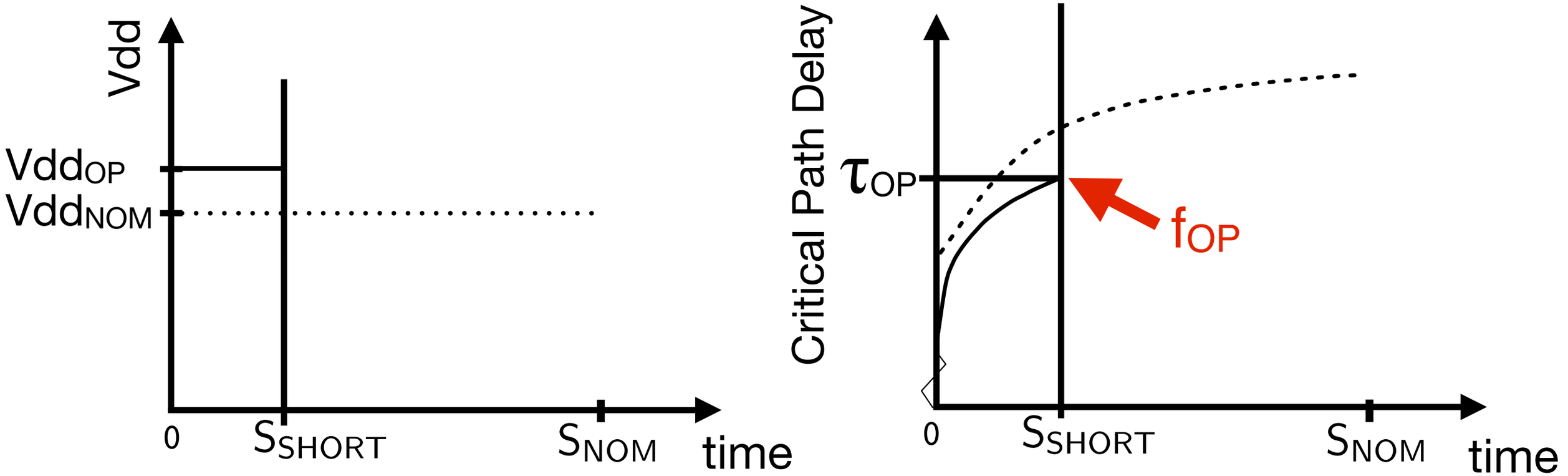
# Impact of Operation at Higher V/f on Aging



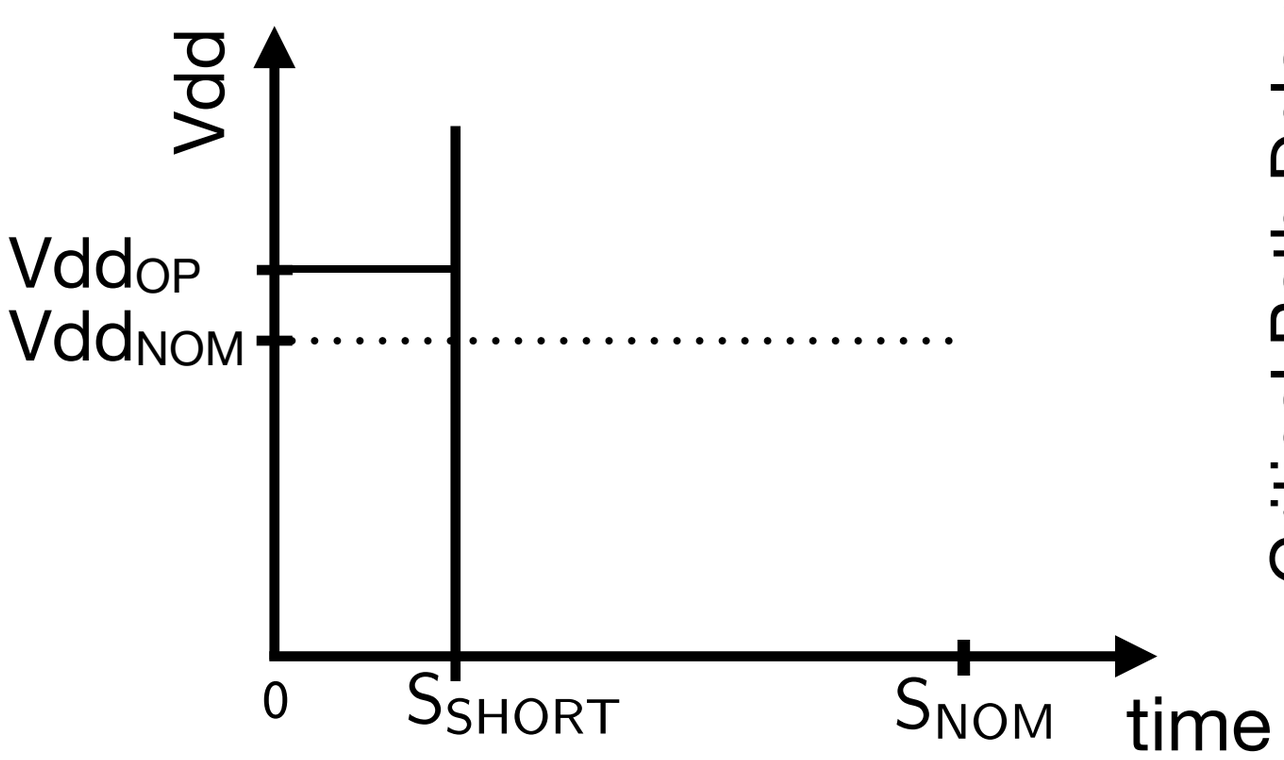

- Higher Vdd: $Vdd_{OP}$ >> $Vdd_{NOM}$
- Result: Lower critical path delay; higher aging rate

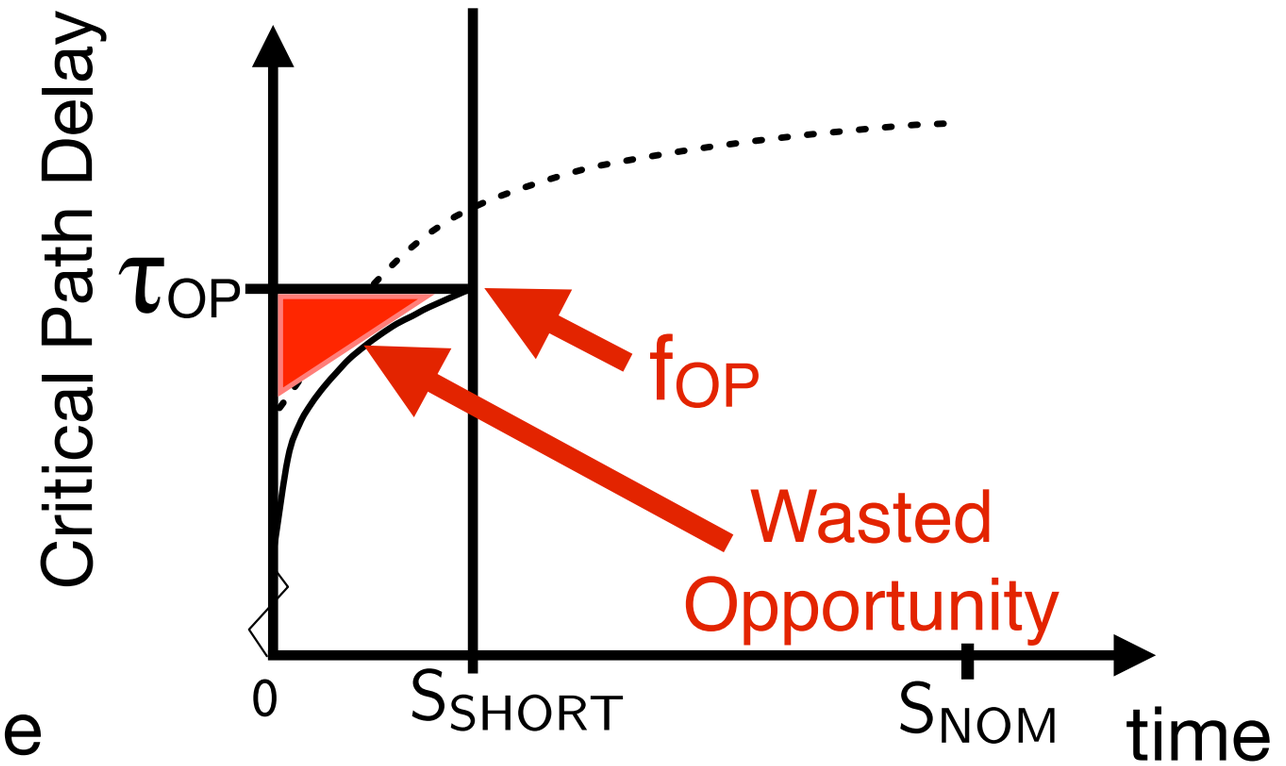
# Impact of Operation at Higher V/f on Aging




- Higher Vdd: $Vdd_{OP} >> Vdd_{NOM}$
- Result: Lower critical path delay; higher aging rate
- Run at constant $f_{OP} = 1/\tau_{OP}$ until $S_{SHORT}$; then discard

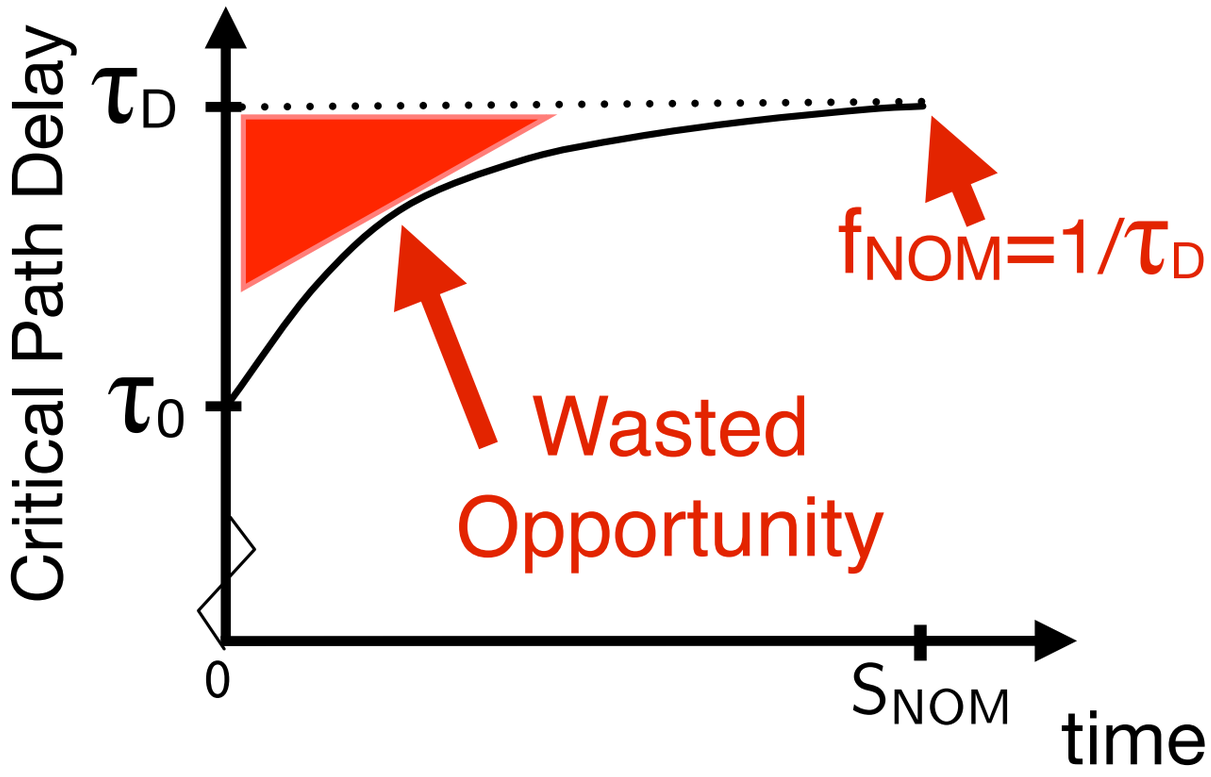
# Impact of Operation at Higher V/f on Aging



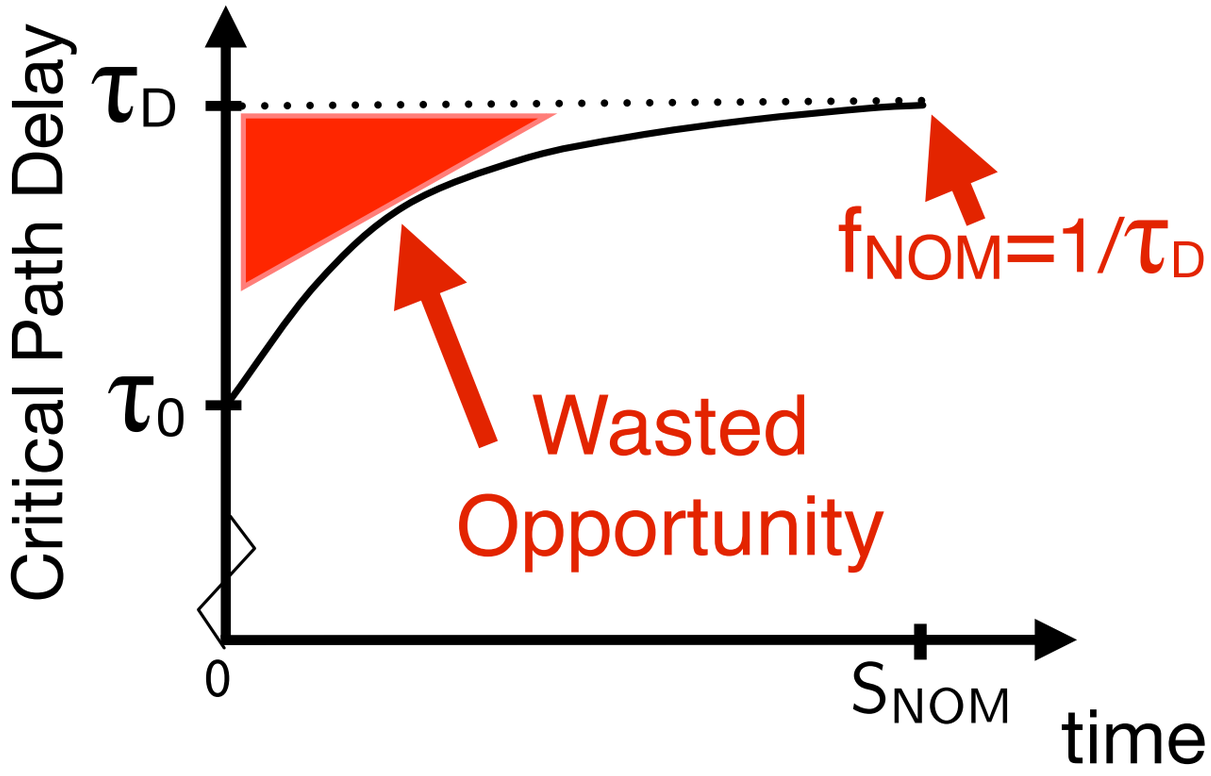

- Higher Vdd: $Vdd_{OP} >> Vdd_{NOM}$
- Result: Lower critical path delay; higher aging rate
- Run at constant $f_{OP} = 1/\tau_{OP}$ until $S_{SHORT}$; then discard

# How to Manage Aging Optimally?



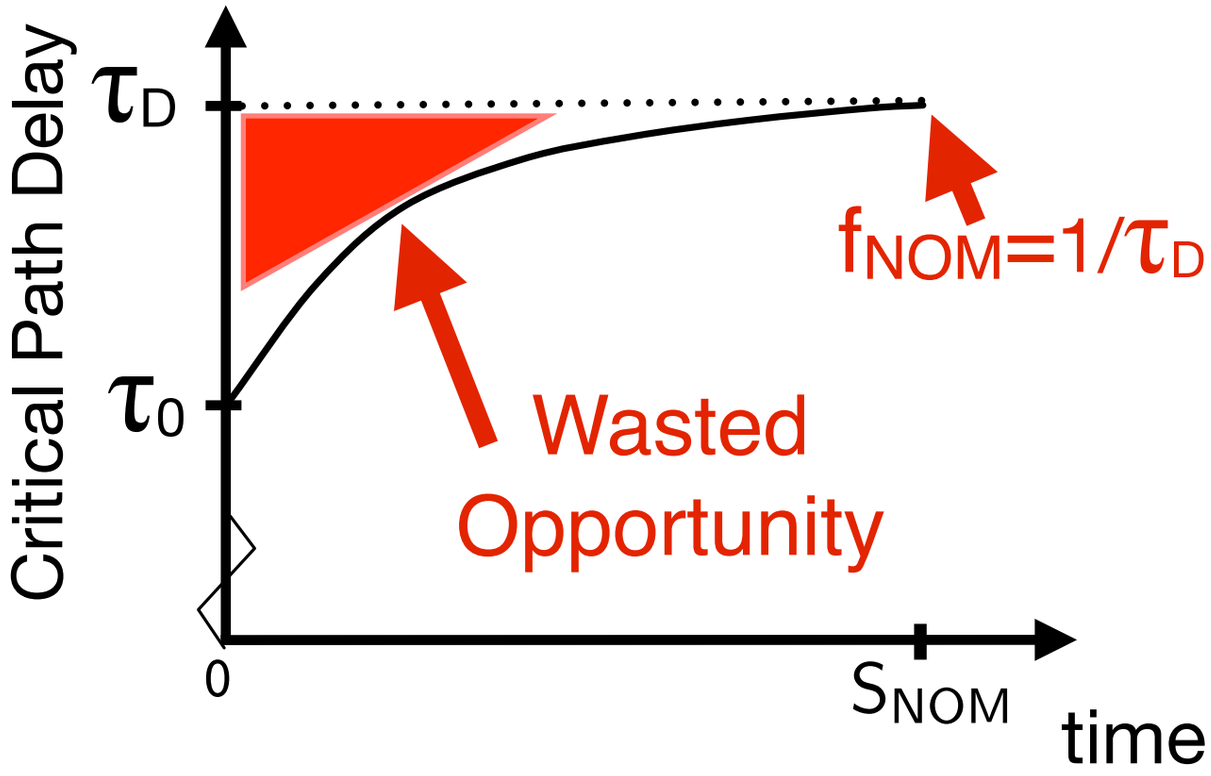
# How to Manage Aging Optimally?

Contribution: DVS for Aging Management (DVSAM)



# How to Manage Aging Optimally?

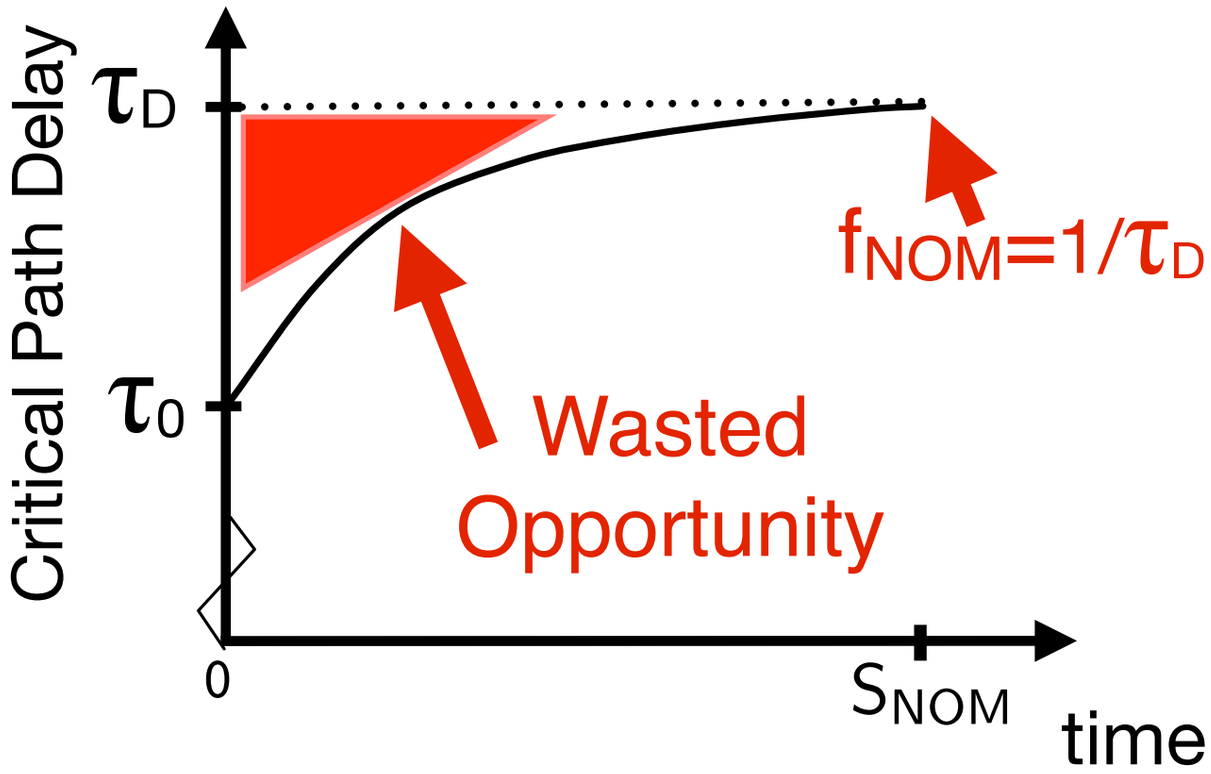
Contribution: DVS for Aging Management (DVSAM)




- Change Vdd with time to compensate for critical path degradation

# How to Manage Aging Optimally?

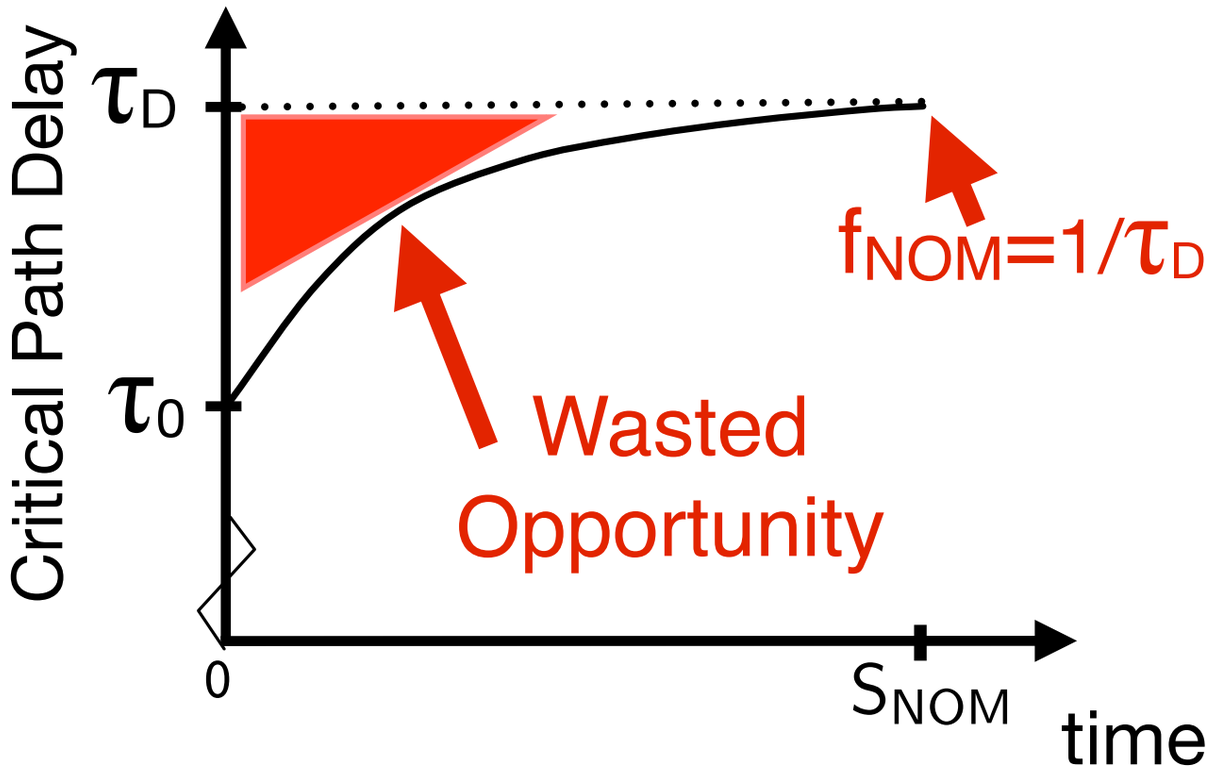
Contribution: DVS for Aging Management (DVSAM)




- Change Vdd with time to compensate for critical path degradation
- Enforce minimum Vdd needed for any f-target

# How to Manage Aging Optimally?

Contribution: DVS for Aging Management (DVSAM)

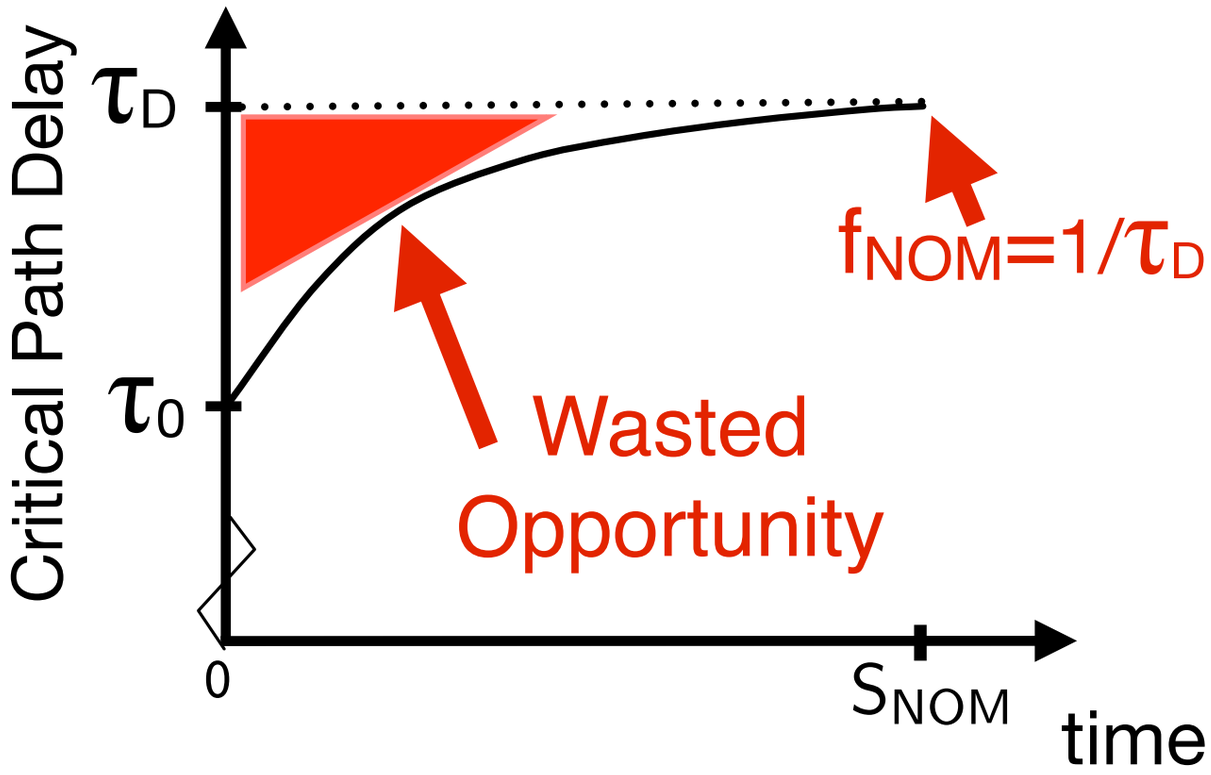



- Change Vdd with time to compensate for critical path degradation
- Enforce minimum Vdd needed for any f-target

- DVSAM-Pow: Turn wasted opportunity to power efficiency

# How to Manage Aging Optimally?

Contribution: DVS for Aging Management (DVSAM)




- Change Vdd with time to compensate for critical path degradation
- Enforce minimum Vdd needed for any f-target

- DVSAM-Pow: Turn wasted opportunity to power efficiency
- DVSAM-Perf: Turn wasted opportunity to higher frequency

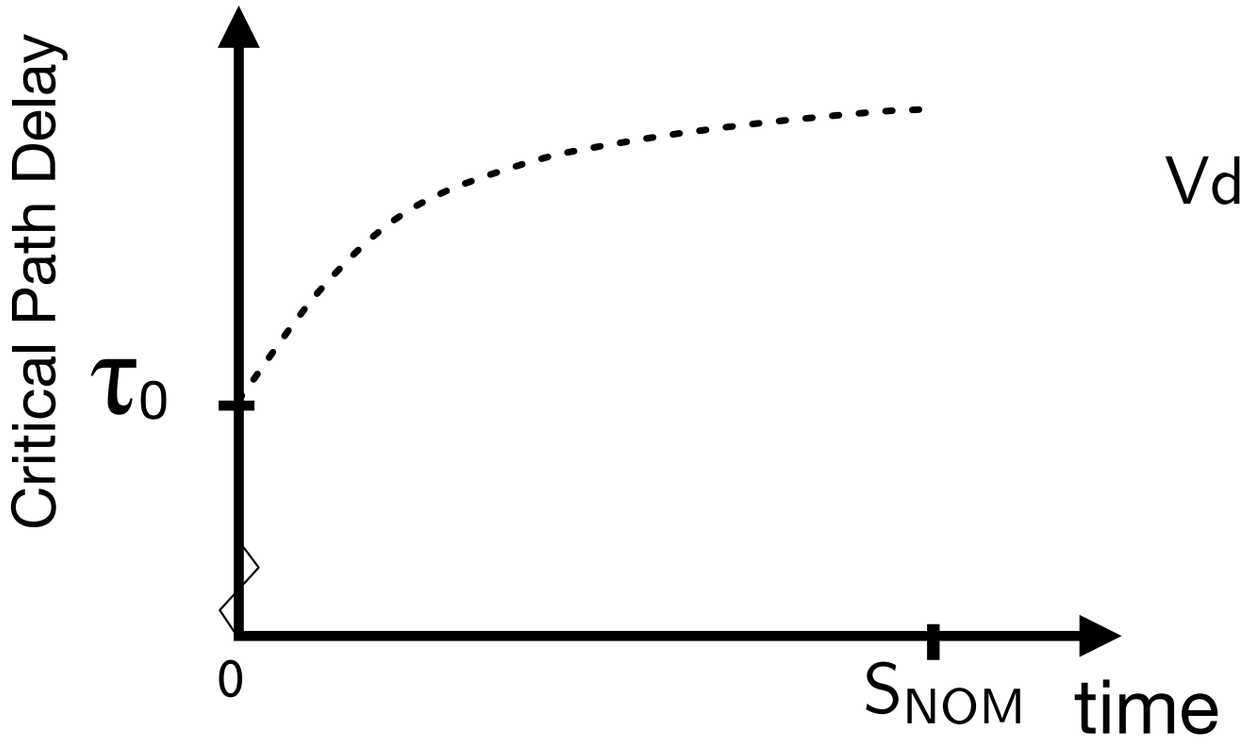
# DVSAM-Pow

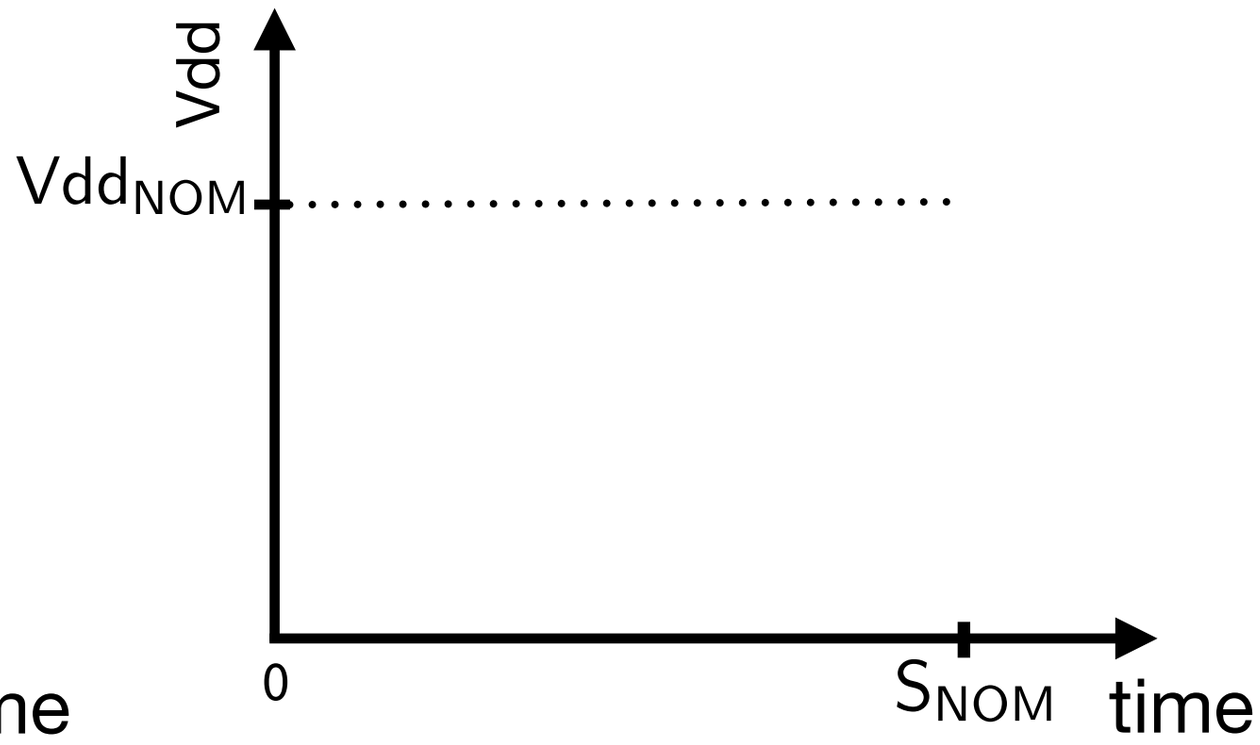
# DVSAM-Pow

Idea: Minimize power consumption at $f_{NOM} = 1/\tau_D$

# DVSAM-Pow

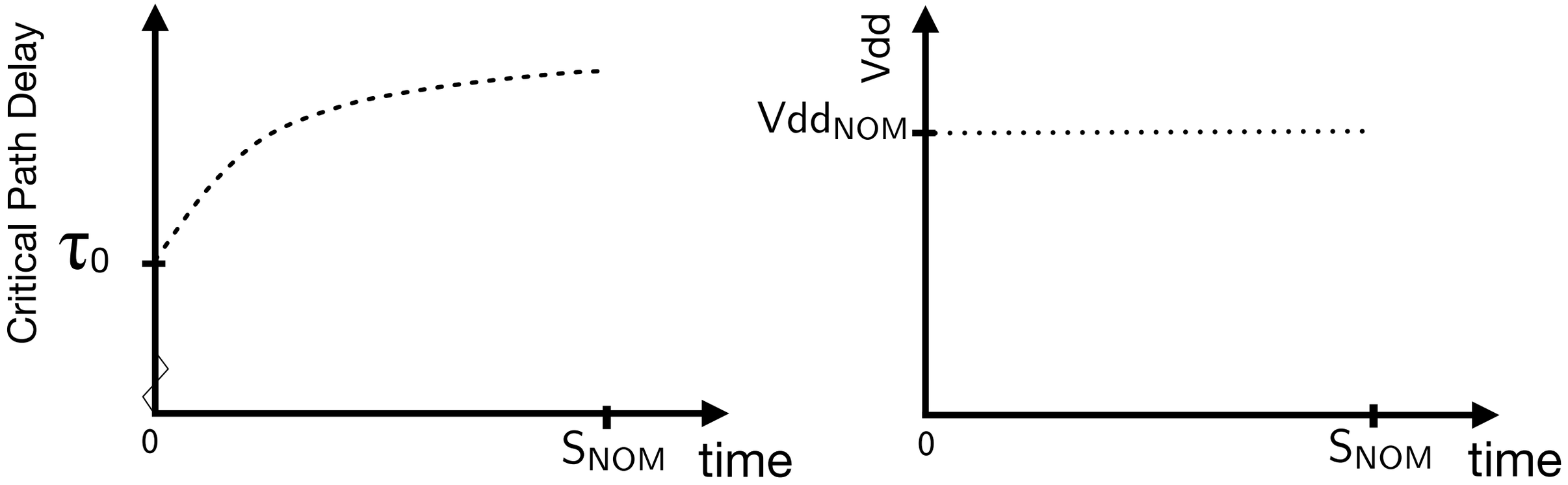
Idea: Minimize power consumption at $f_{NOM} = 1/\tau_D$



# DVSAM-Pow

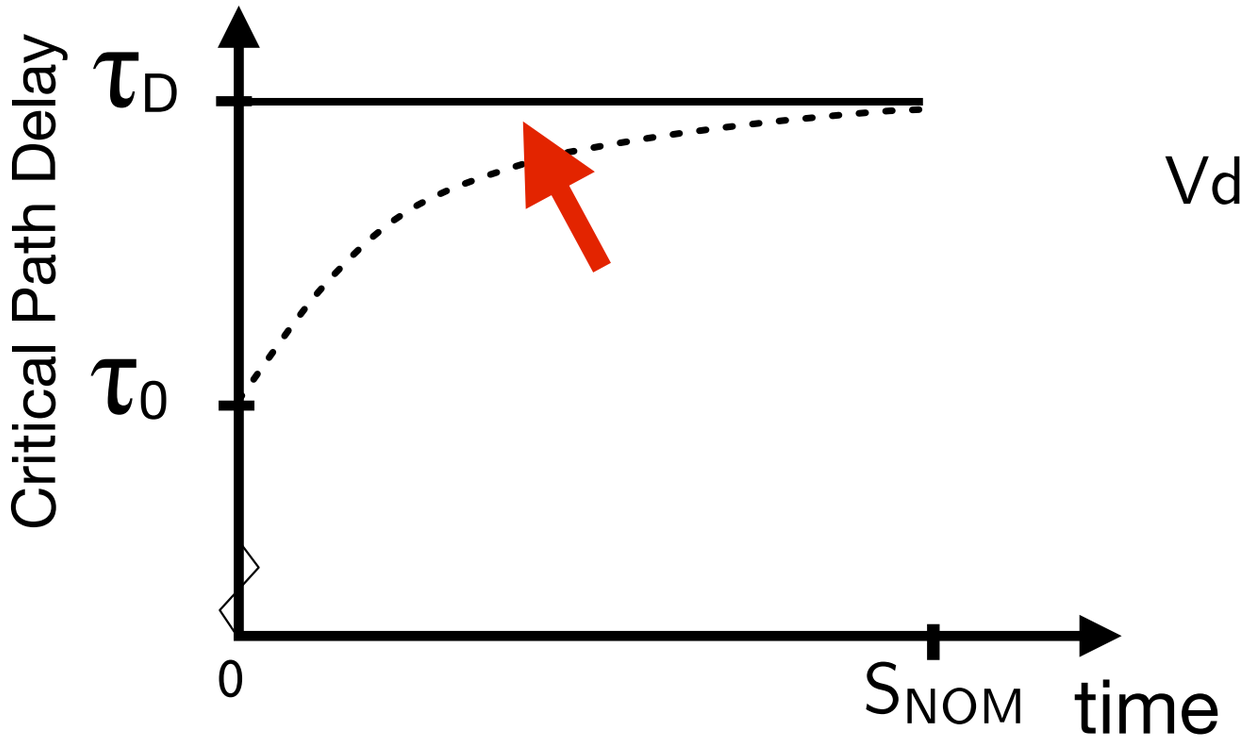
Idea: Minimize power consumption at $f_{NOM} = 1/\tau_D$

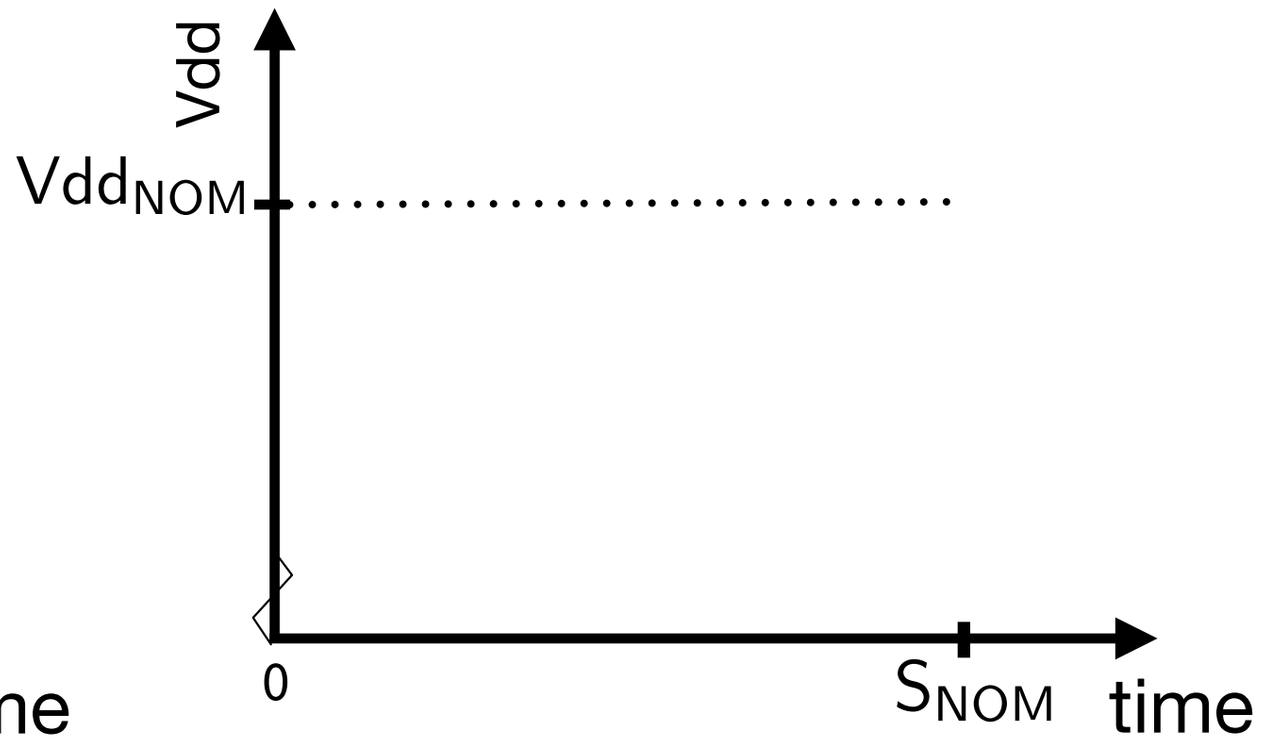



- Critical path delays are kept at $\tau_D$ until $S_{NOM}$: Run at $f_{NOM}$

# DVSAM-Pow

Idea: Minimize power consumption at $f_{NOM} = 1/\tau_D$

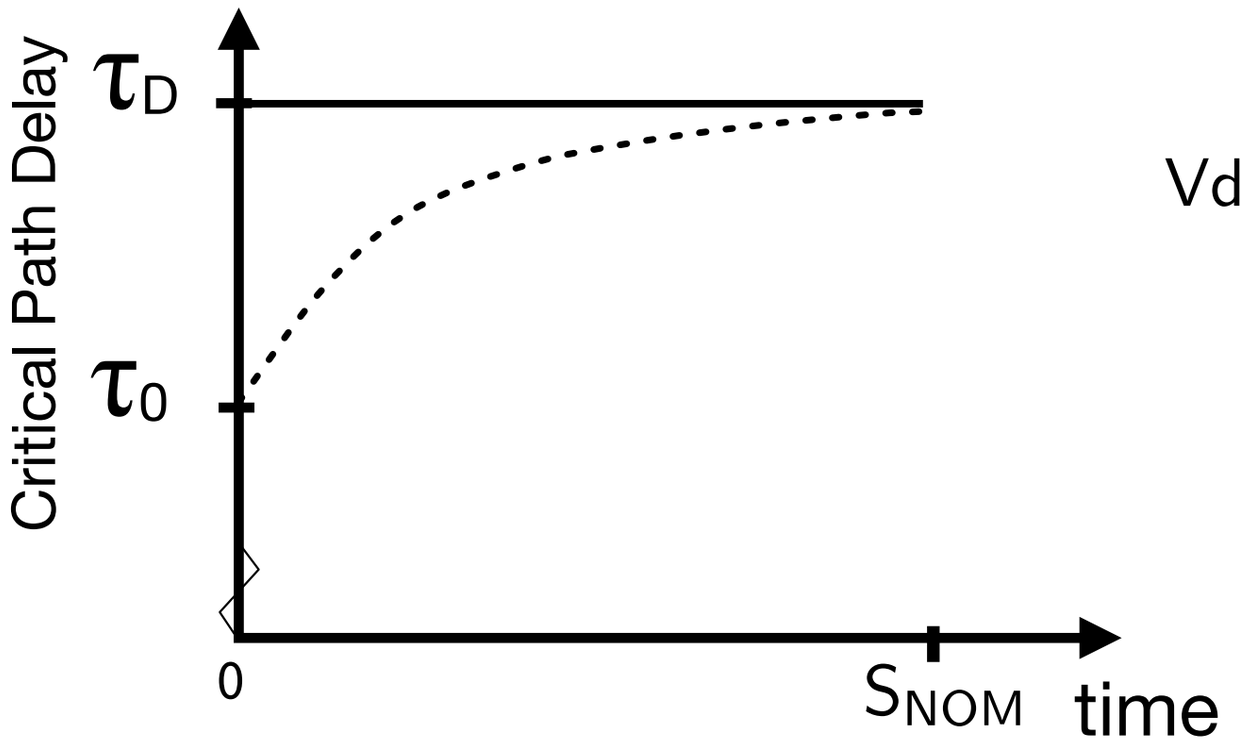


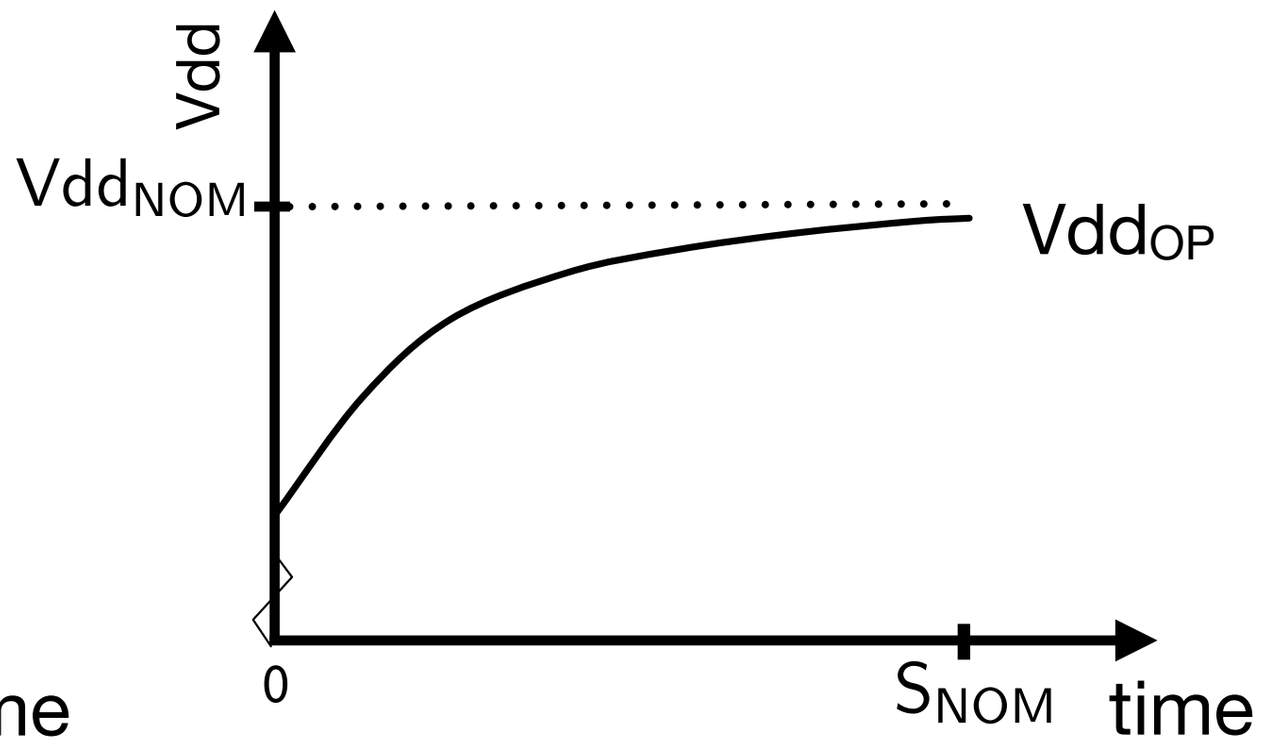




- Critical path delays are kept at $\tau_D$ until $S_{NOM}$: Run at $f_{NOM}$

# DVSAM-Pow

Idea: Minimize power consumption at $f_{NOM} = 1/\tau_D$

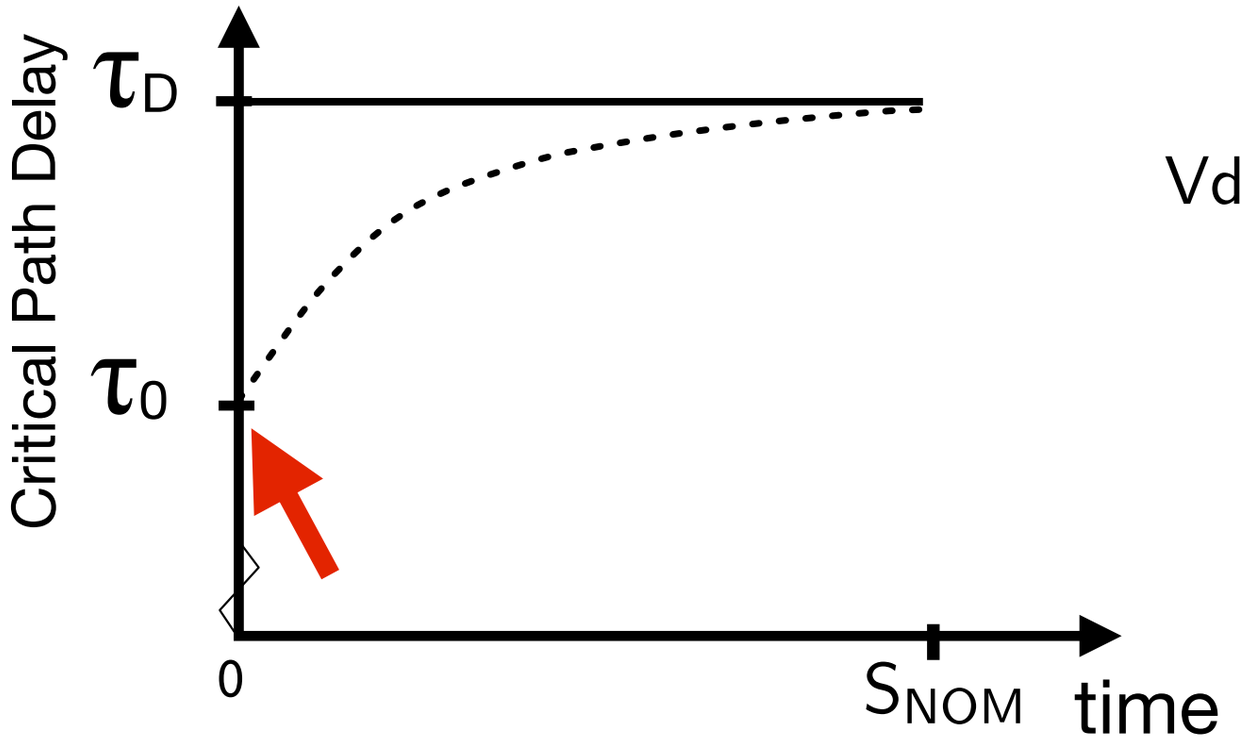


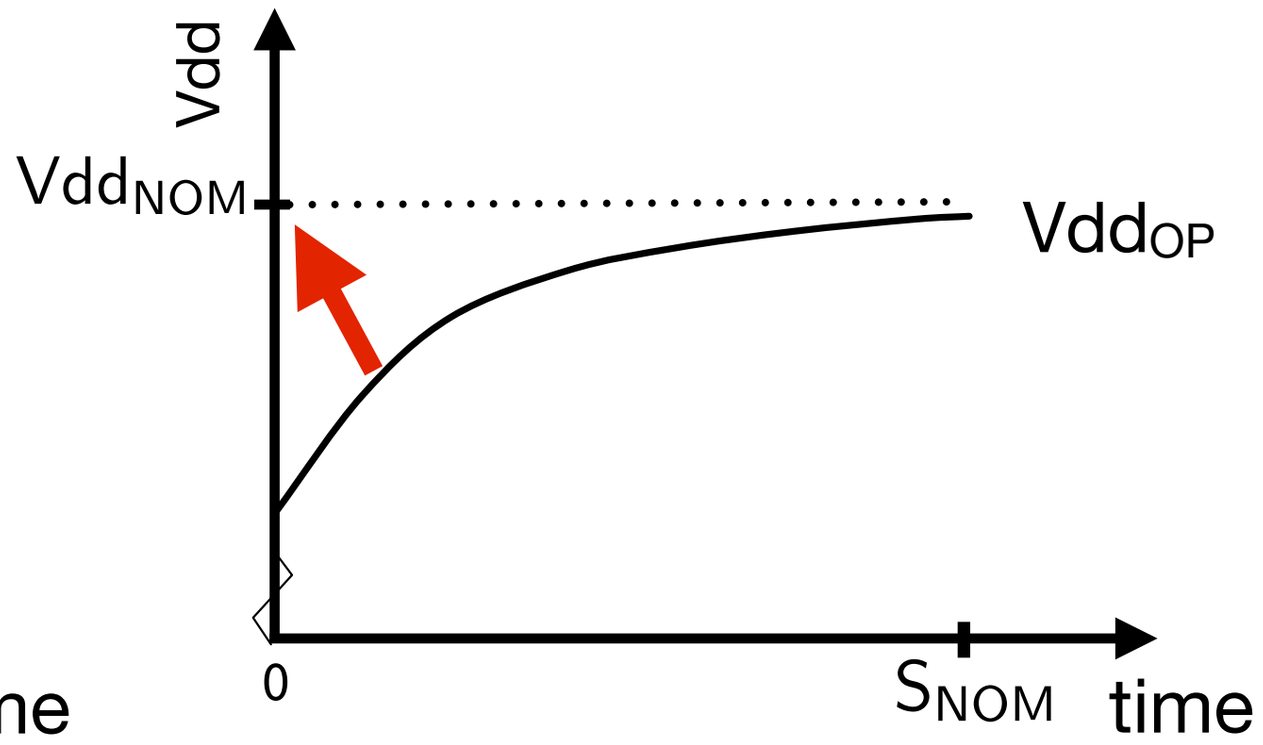




- Critical path delays are kept at $\tau_D$ until $S_{NOM}$: Run at $f_{NOM}$
- Start with low Vdd and increase slowly

# DVSAM-Pow

Idea: Minimize power consumption at $f_{NOM} = 1/\tau_D$

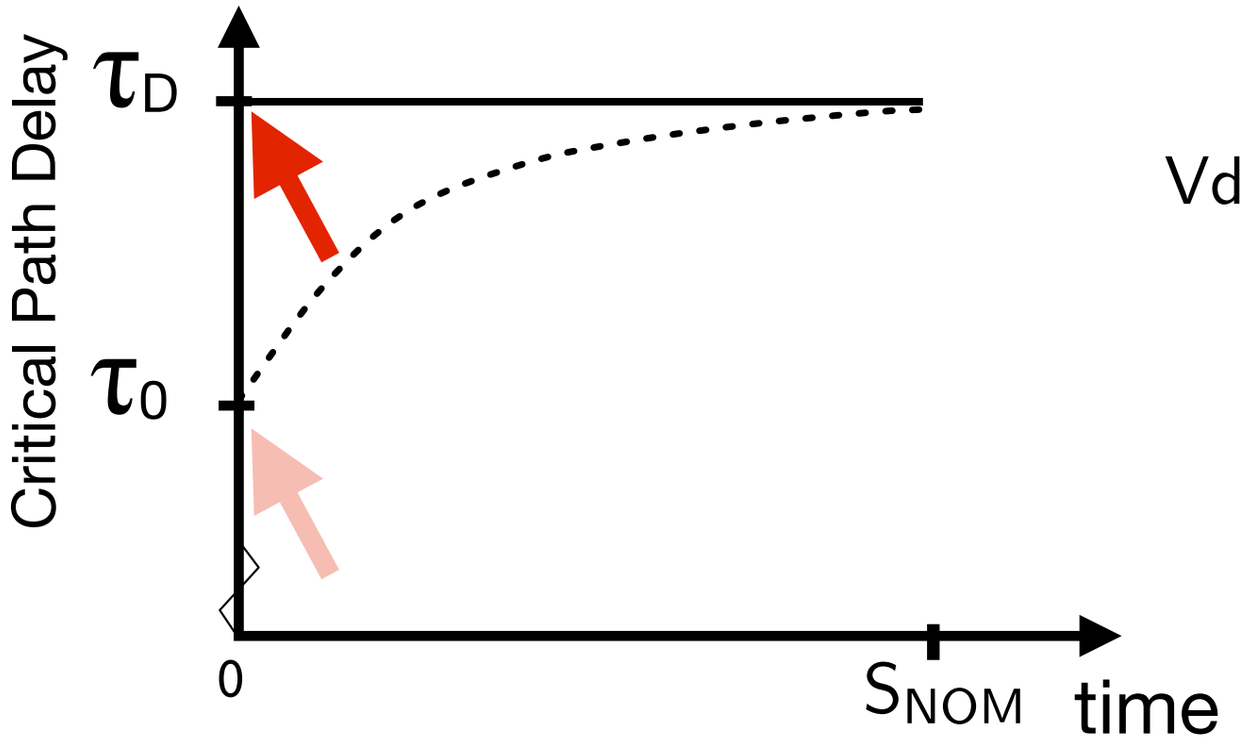


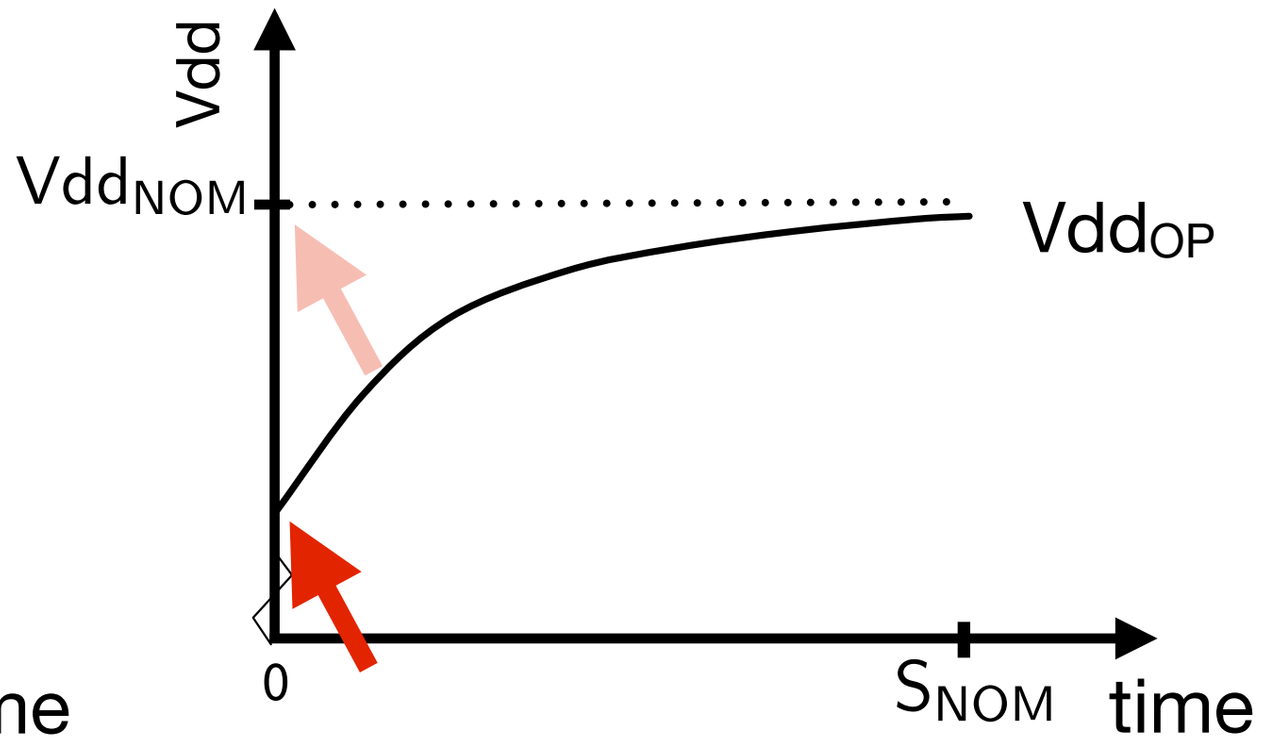




- Critical path delays are kept at $\tau_D$ until $S_{NOM}$: Run at $f_{NOM}$
- Start with low Vdd and increase slowly

# DVSAM-Pow

Idea: Minimize power consumption at $f_{NOM} = 1/\tau_D$

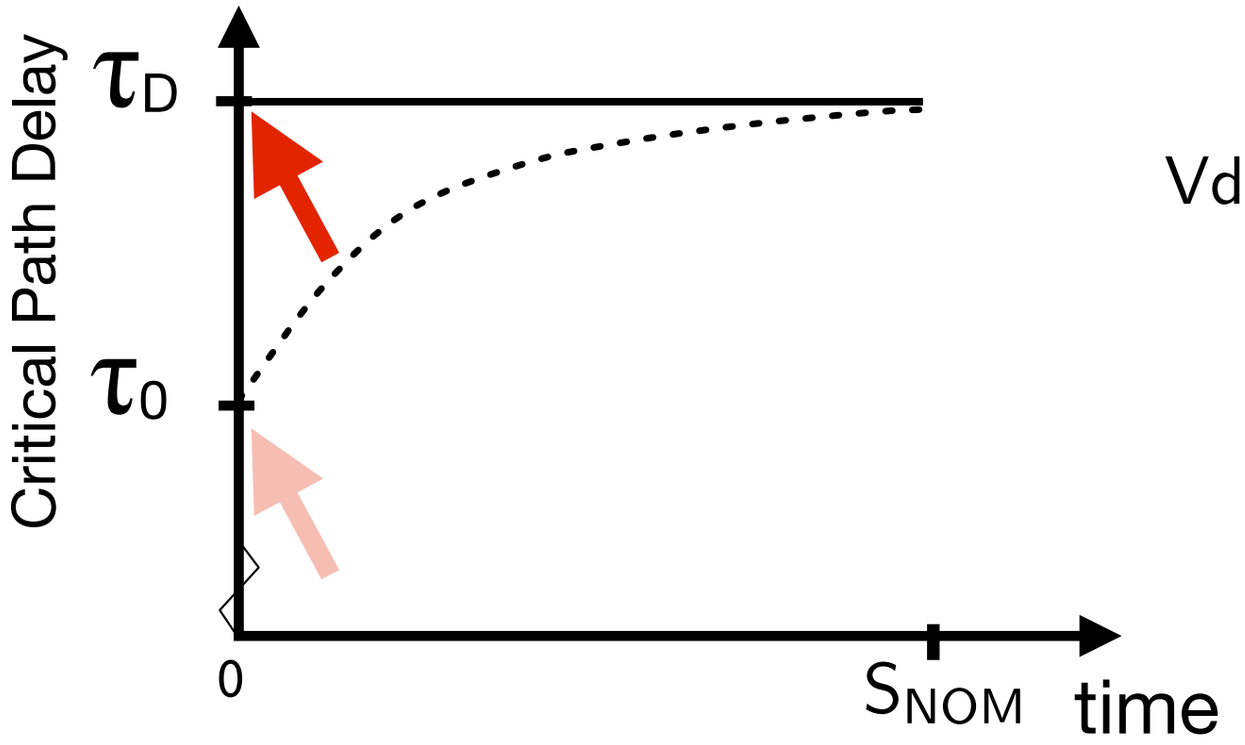


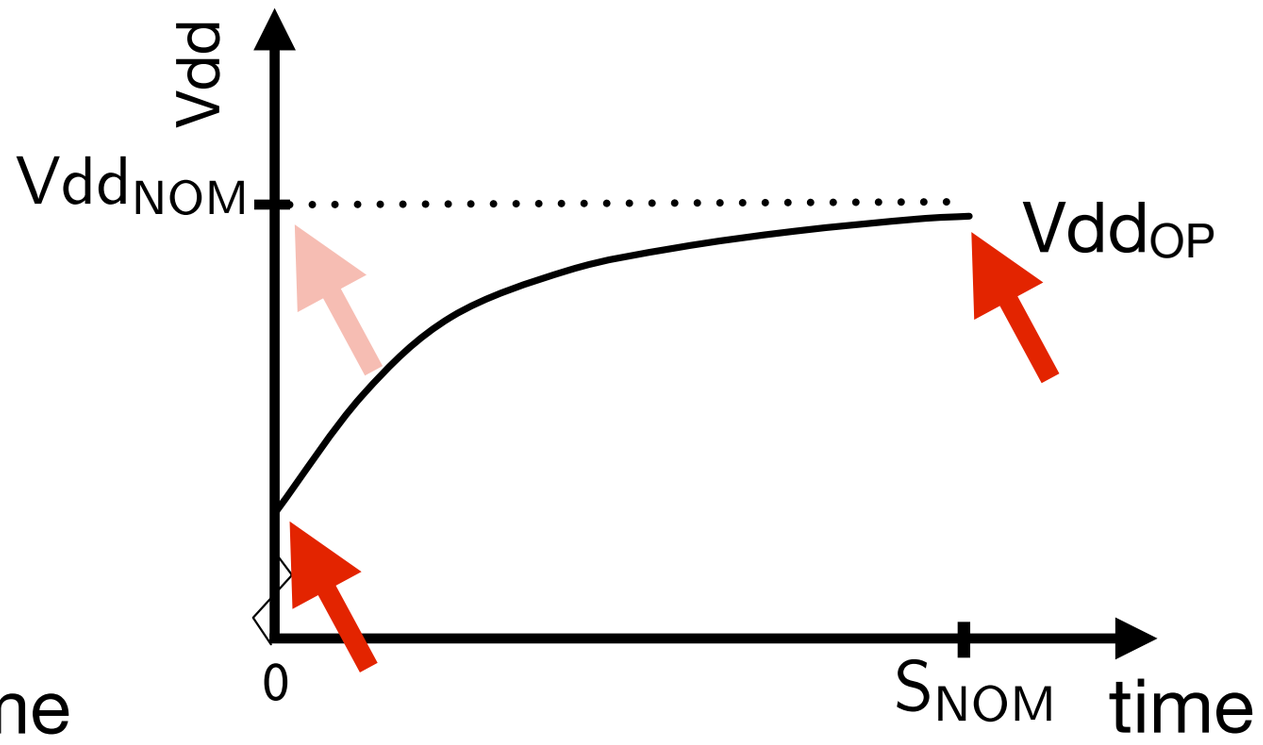




- Critical path delays are kept at $\tau_D$ until $S_{NOM}$: Run at $f_{NOM}$
- Start with low Vdd and increase slowly

# DVSAM-Pow

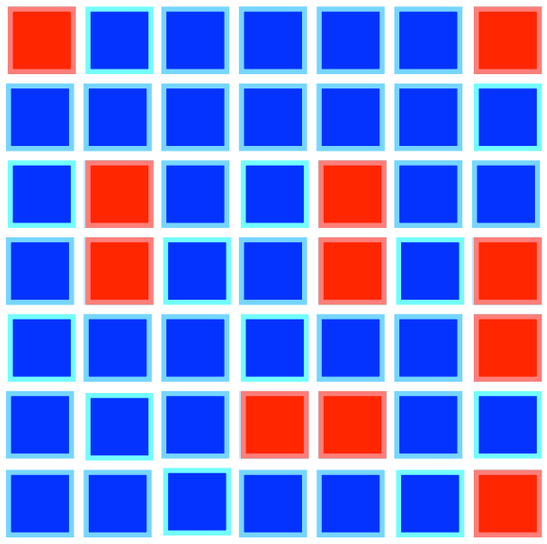
Idea: Minimize power consumption at $f_{NOM} = 1/\tau_D$



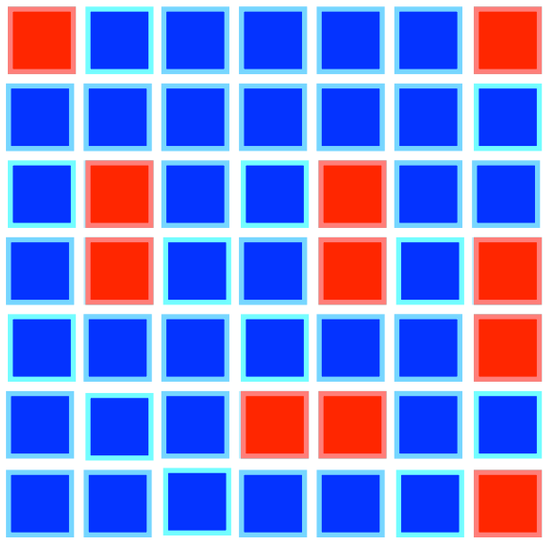




- Critical path delays are kept at $\tau_D$ until $S_{NOM}$: Run at $f_{NOM}$
- Start with low Vdd and increase slowly

# DVSAM-Pow

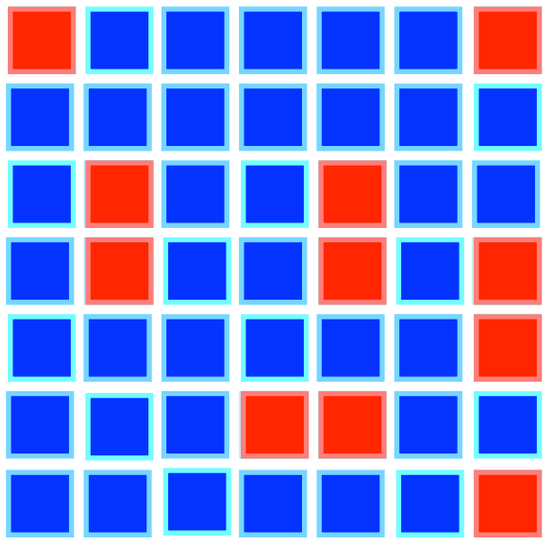


# DVSAM-Pow



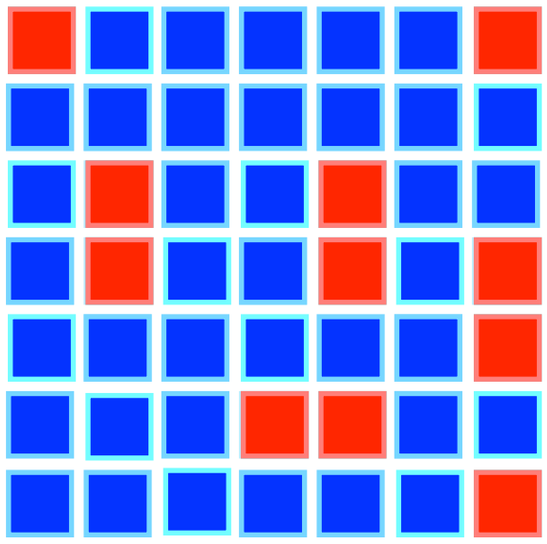
- Vdd < $Vdd_{NOM}$ and f = $f_{NOM}$ throughout $S_{NOM}$

# DVSAM-Pow



- Vdd < $Vdd_{NOM}$ and f = $f_{NOM}$ throughout $S_{NOM}$
- Power savings due to Vdd < $Vdd_{NOM}$

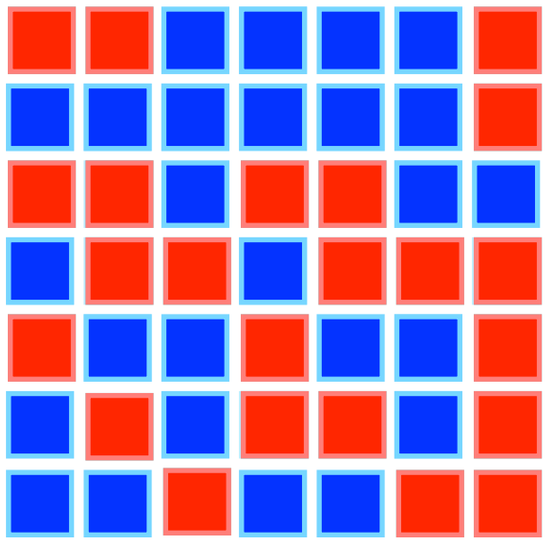
# DVSAM-Pow



- Vdd < $Vdd_{NOM}$ and f = $f_{NOM}$ throughout $S_{NOM}$
- Power savings due to Vdd < $Vdd_{NOM}$

➡ More cores active for the same P-budget

# DVSAM-Pow



- Vdd < $Vdd_{NOM}$ and f = $f_{NOM}$ throughout $S_{NOM}$
- Power savings due to Vdd < $Vdd_{NOM}$

➡ More cores active for the same P-budget

➡ Increased throughput

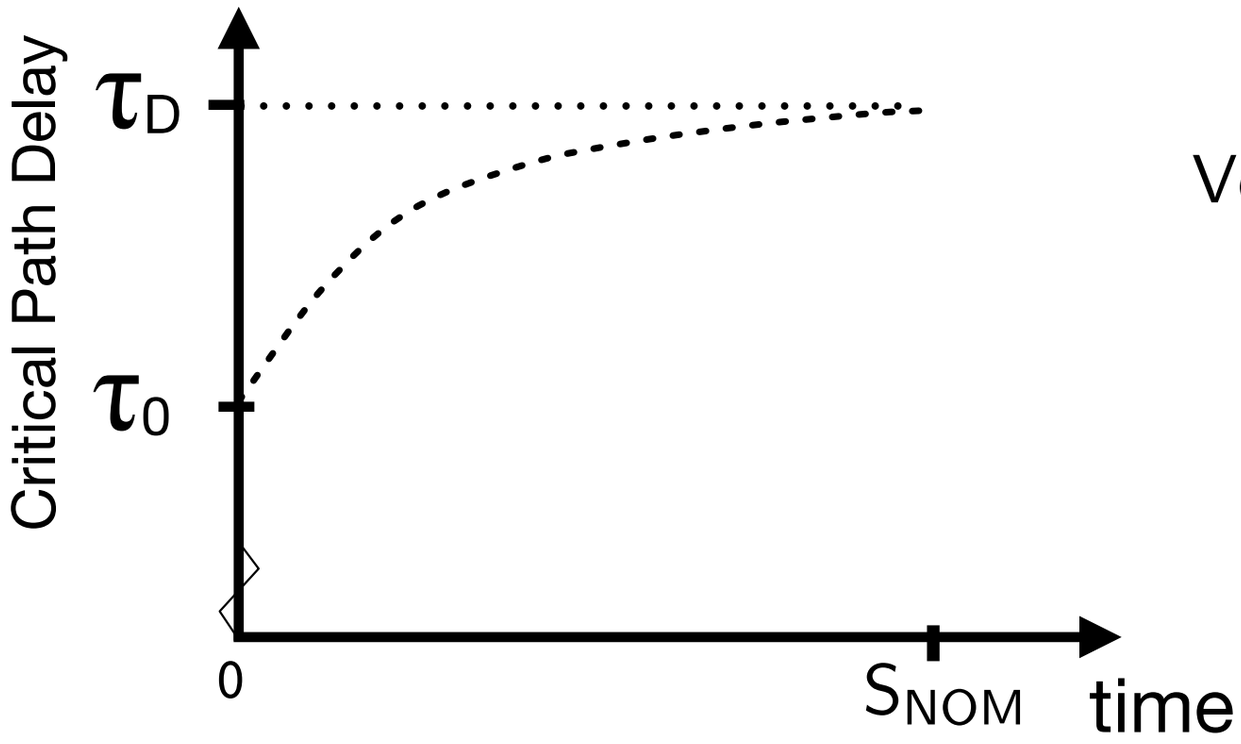
# DVSAM-Perf

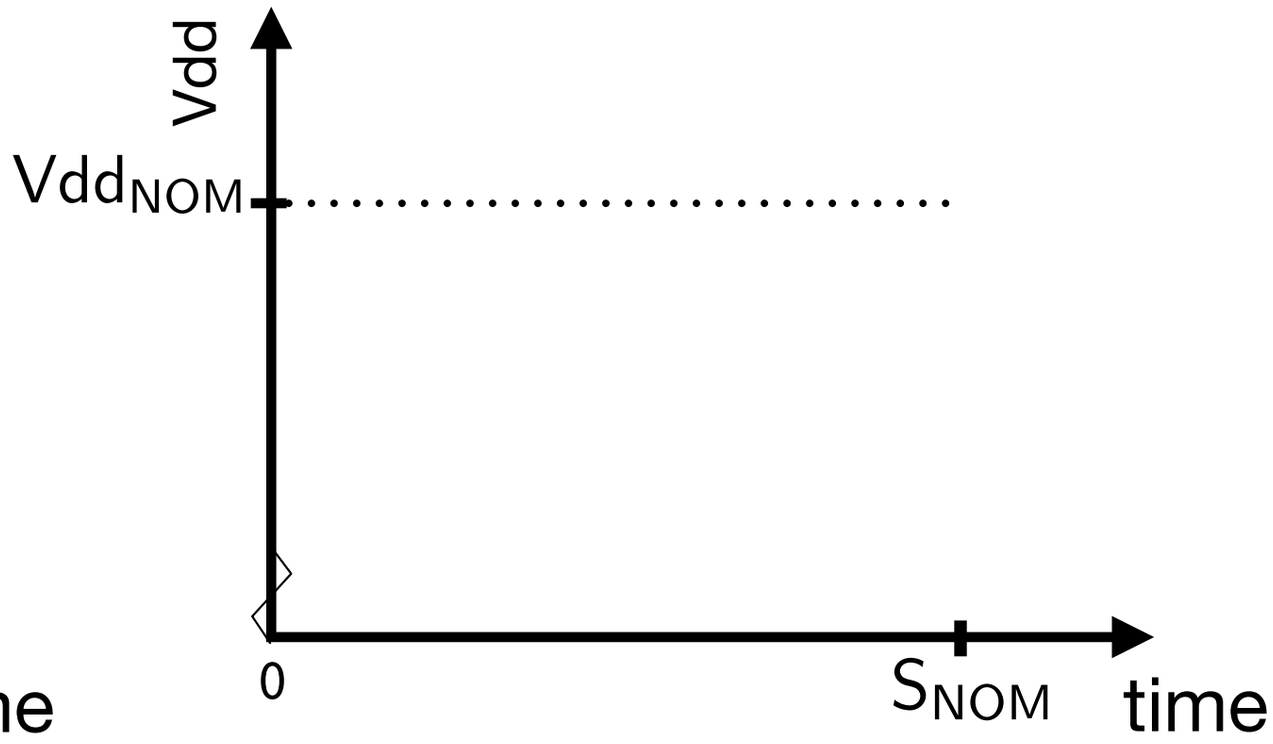
# DVSAM-Perf

Idea: Maximize frequency for the same service life

# DVSAM-Perf

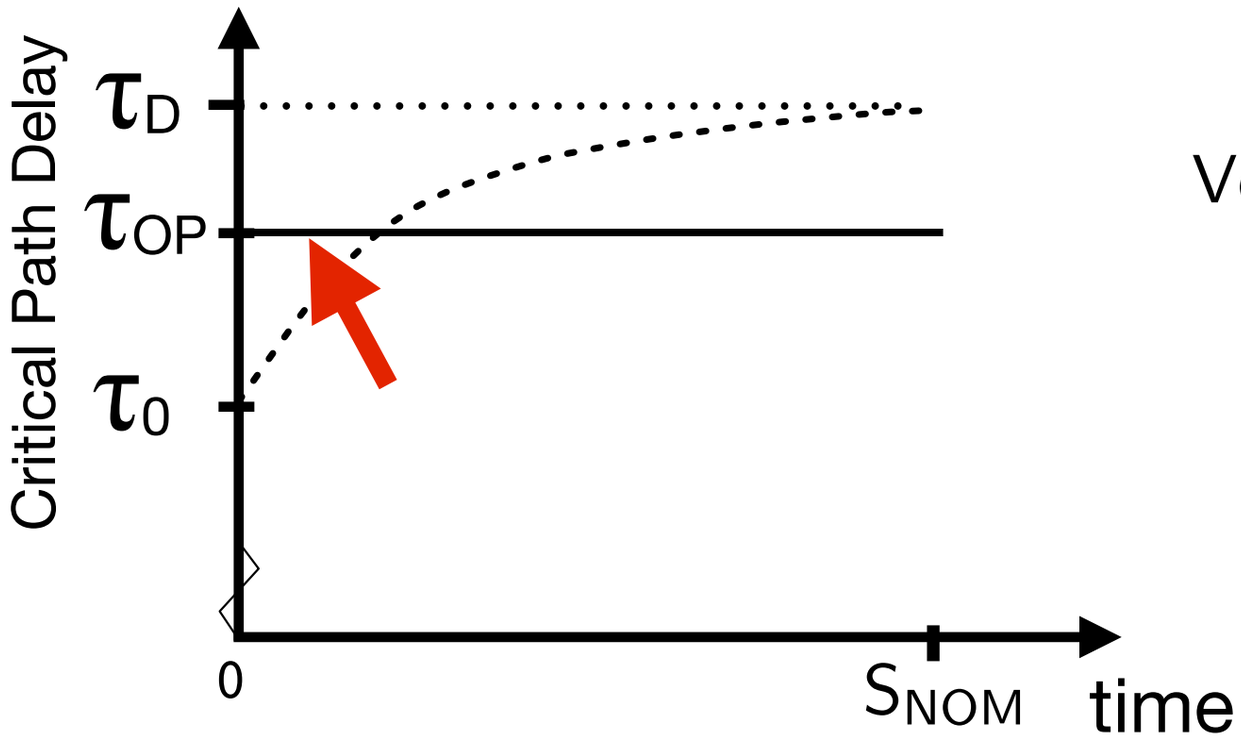
Idea: Maximize frequency for the same service life

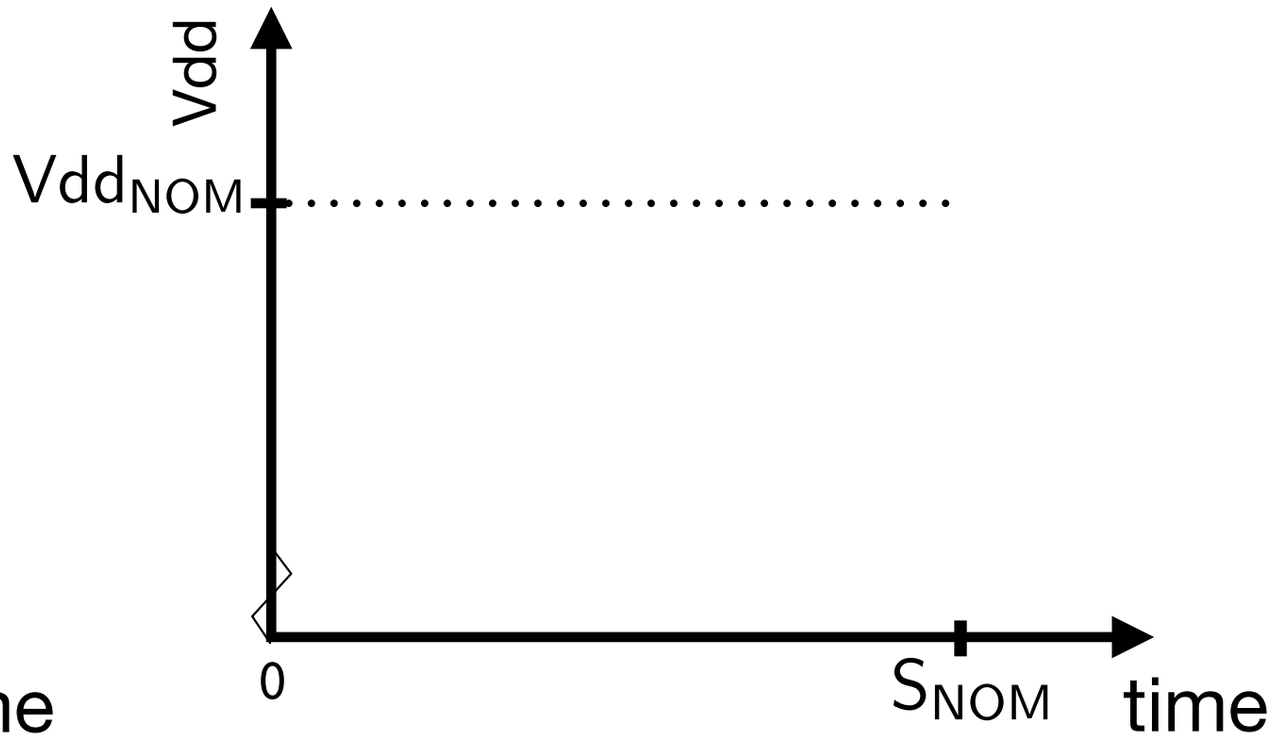


# DVSAM-Perf

Idea: Maximize frequency for the same service life

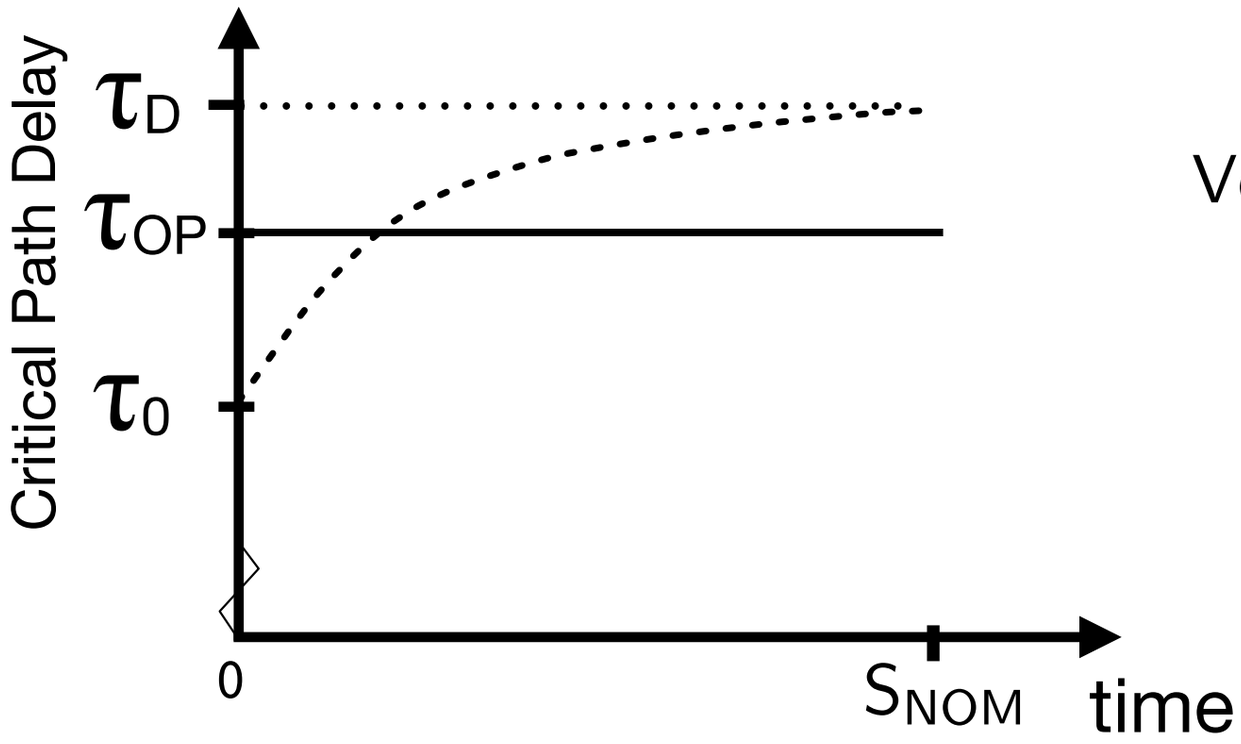


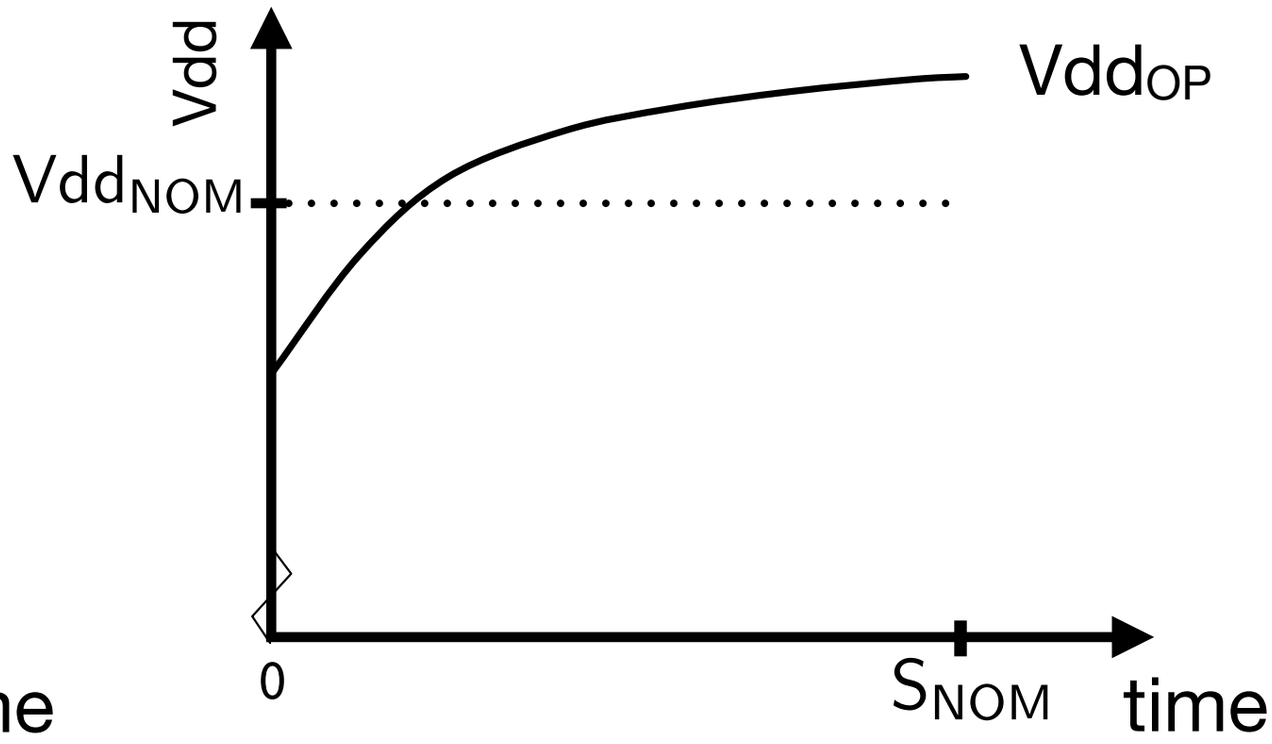




- Shorter critical path delay $\tau_{OP}$ until $S_{NOM}$: Run at higher f = 1 / $\tau_{OP}$

# DVSAM-Perf

Idea: Maximize frequency for the same service life

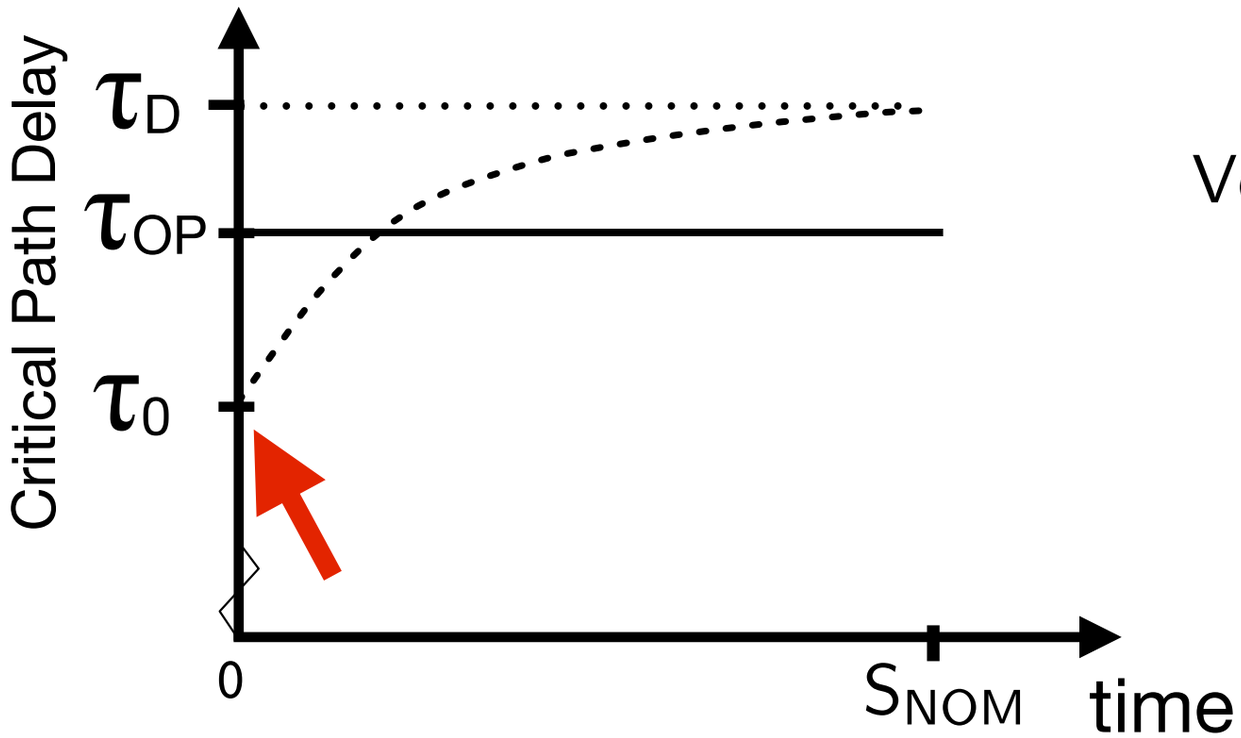


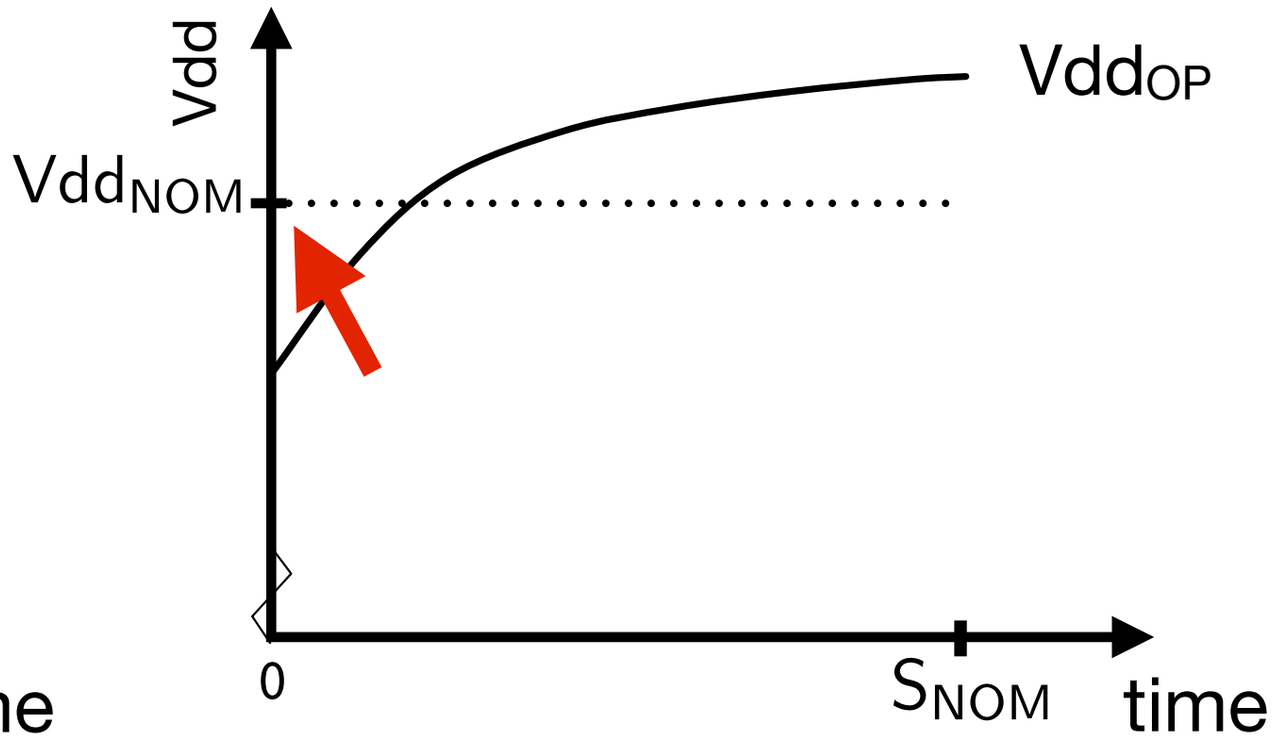




- Shorter critical path delay $\tau_{OP}$ until $S_{NOM}$: Run at higher f = 1 / $\tau_{OP}$
- Start with low Vdd and increase rapidly

# DVSAM-Perf

Idea: Maximize frequency for the same service life

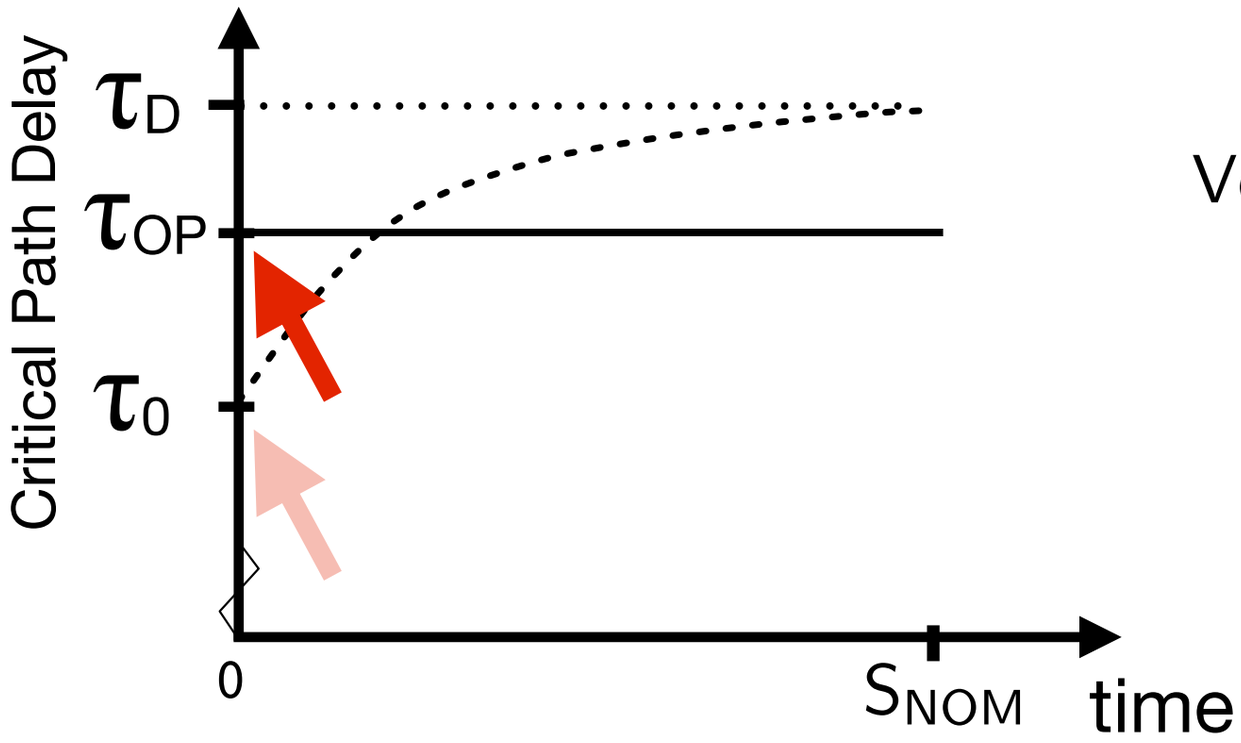


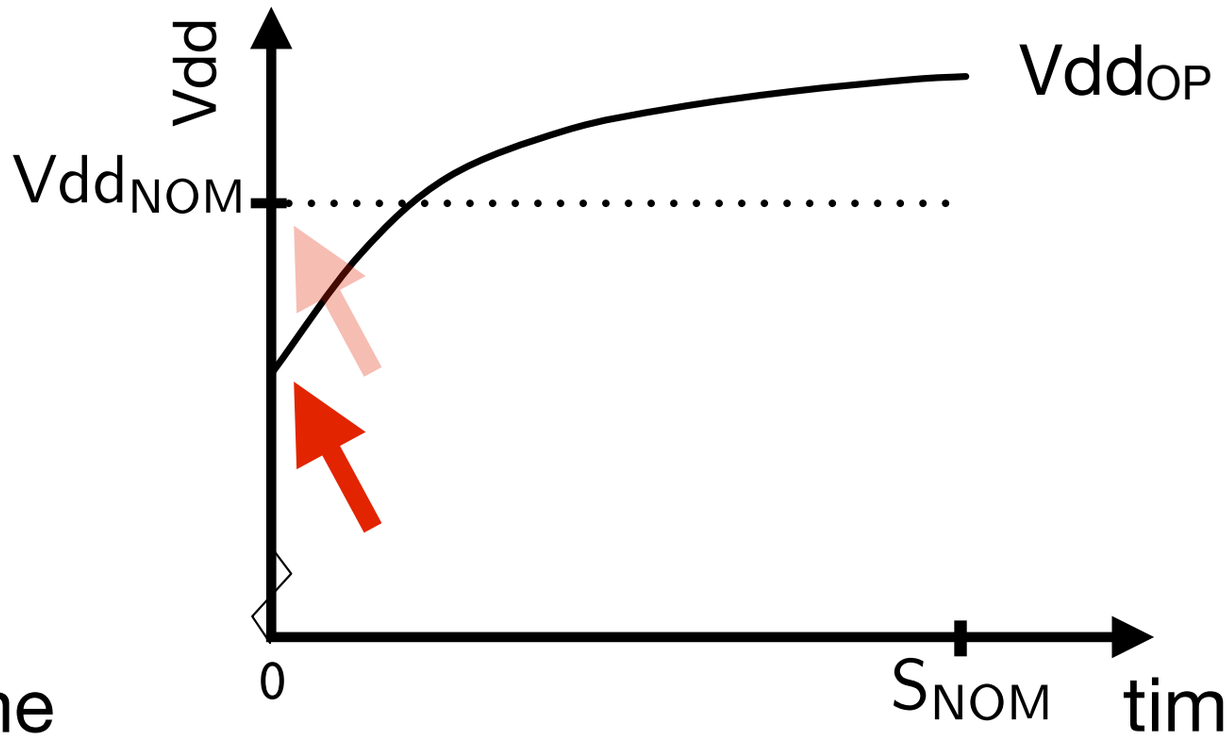




- Shorter critical path delay $\tau_{OP}$ until $S_{NOM}$: Run at higher f = 1 / $\tau_{OP}$
- Start with low Vdd and increase rapidly

# DVSAM-Perf

Idea: Maximize frequency for the same service life

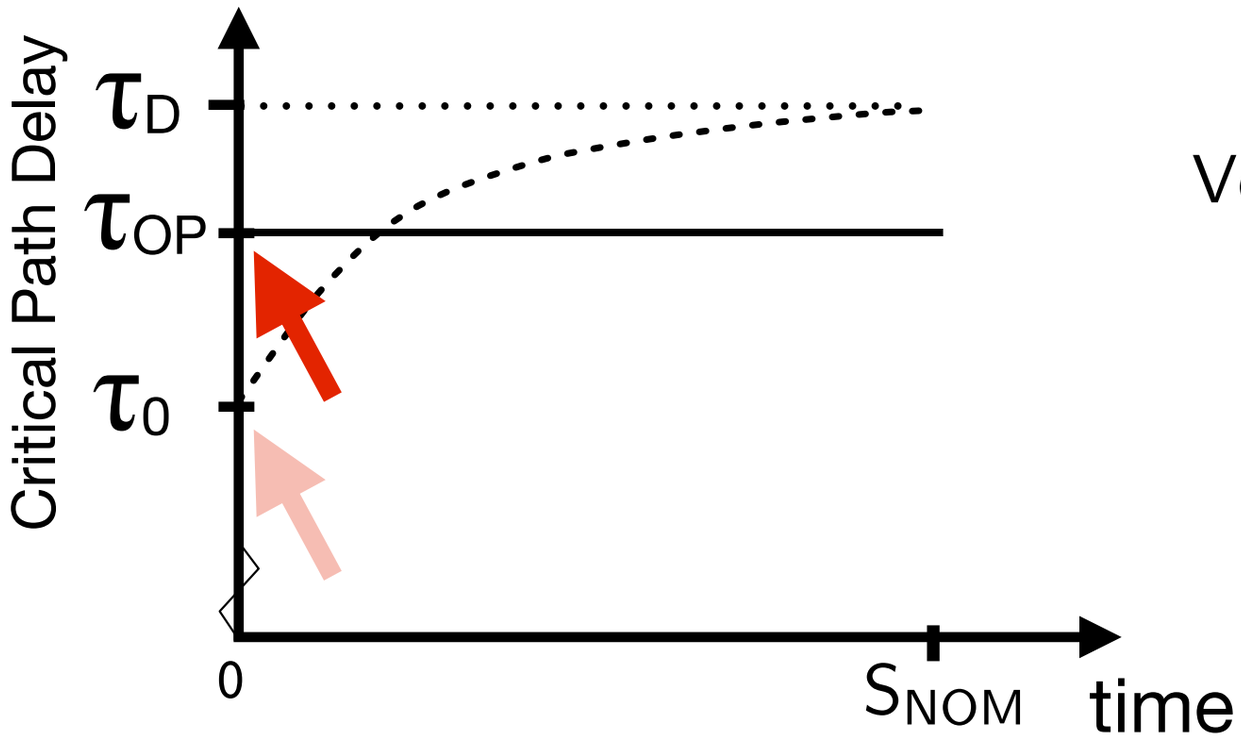


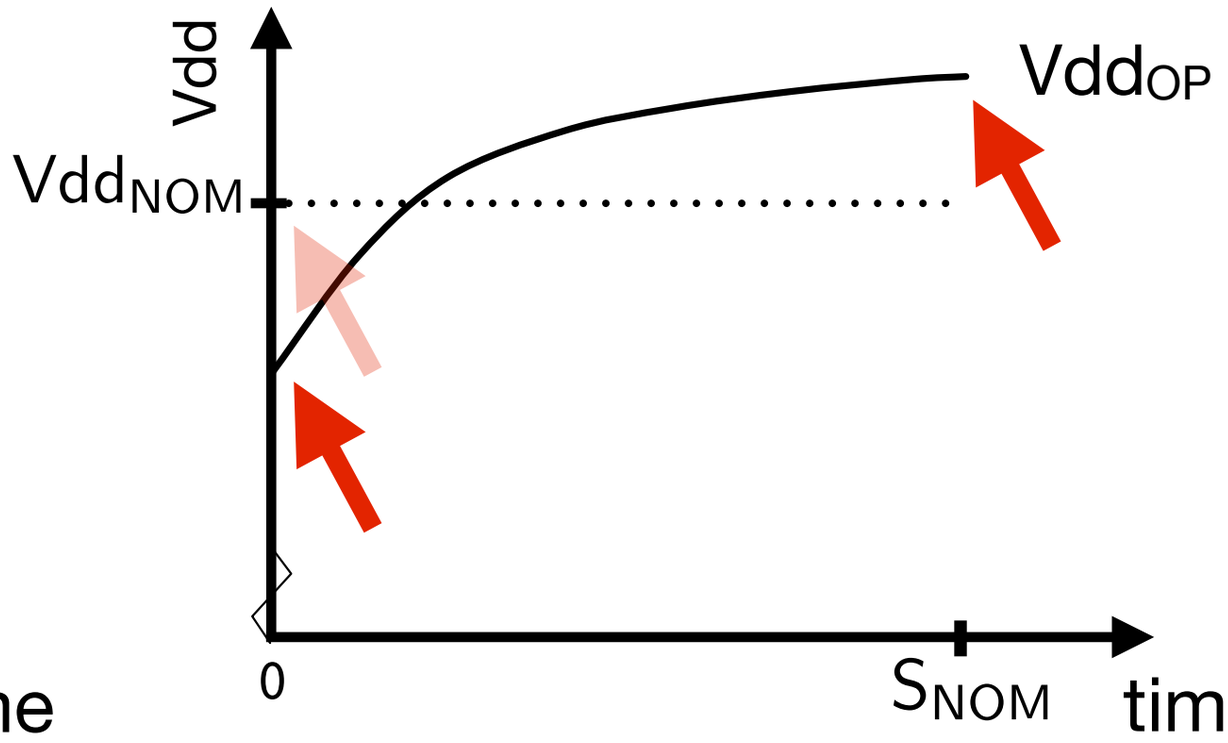




- Shorter critical path delay $\tau_{OP}$ until $S_{NOM}$: Run at higher f = 1 / $\tau_{OP}$
- Start with low Vdd and increase rapidly

# DVSAM-Perf

Idea: Maximize frequency for the same service life




- Shorter critical path delay $\tau_{OP}$ until $S_{NOM}$: Run at higher f = 1 / $\tau_{OP}$
- Start with low Vdd and increase rapidly

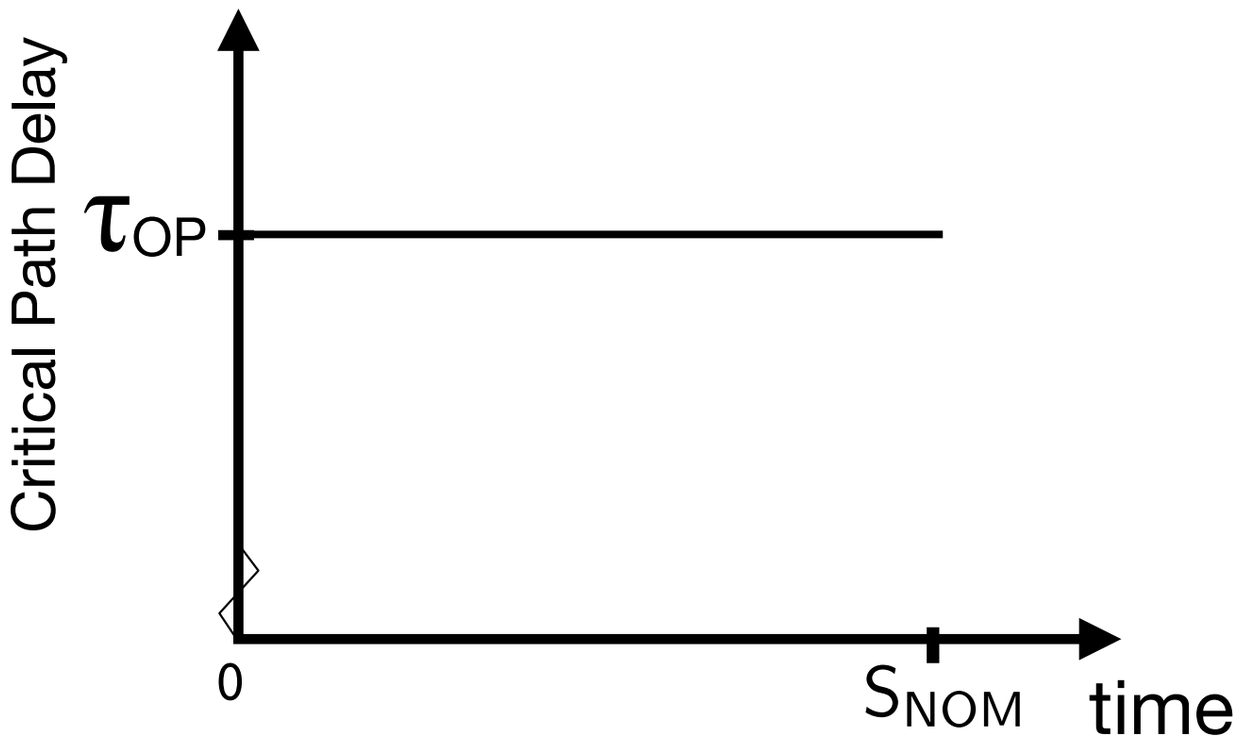
# DVSAM-Perf for SSHORT

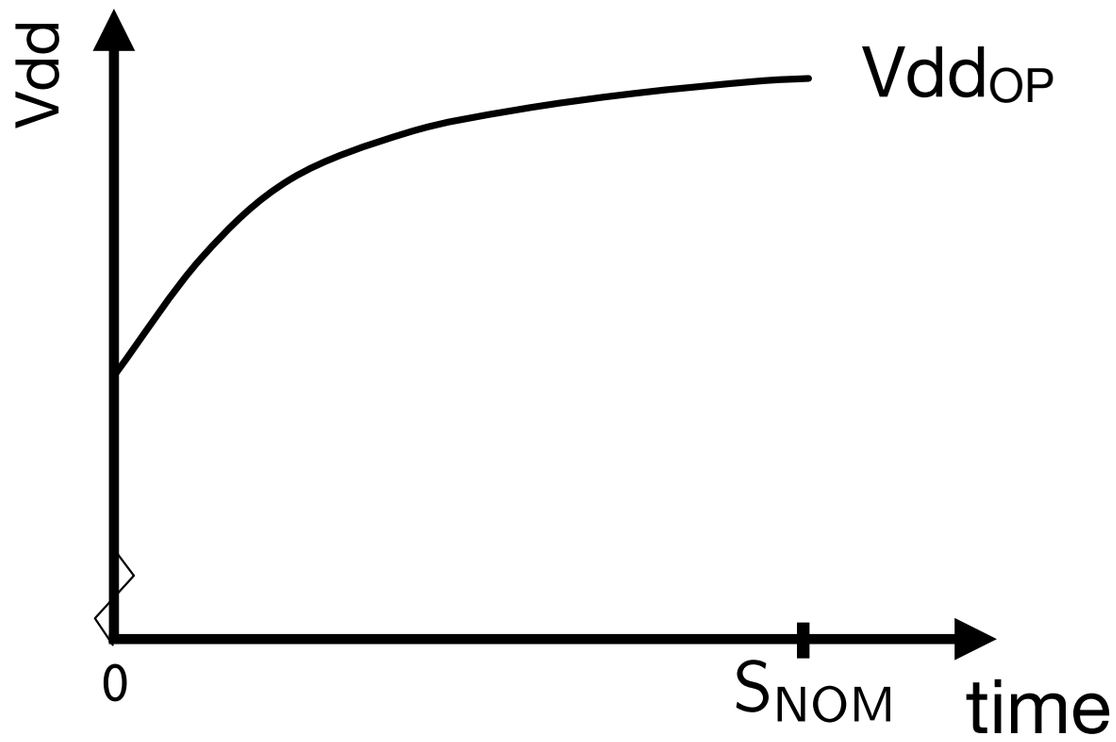
# DVSAM-Perf for SSHORT

Idea: Aggressive DVSAM-Perf for a short service life to get even higher performance

# DVSAM-Perf for $S_{SHORT}$

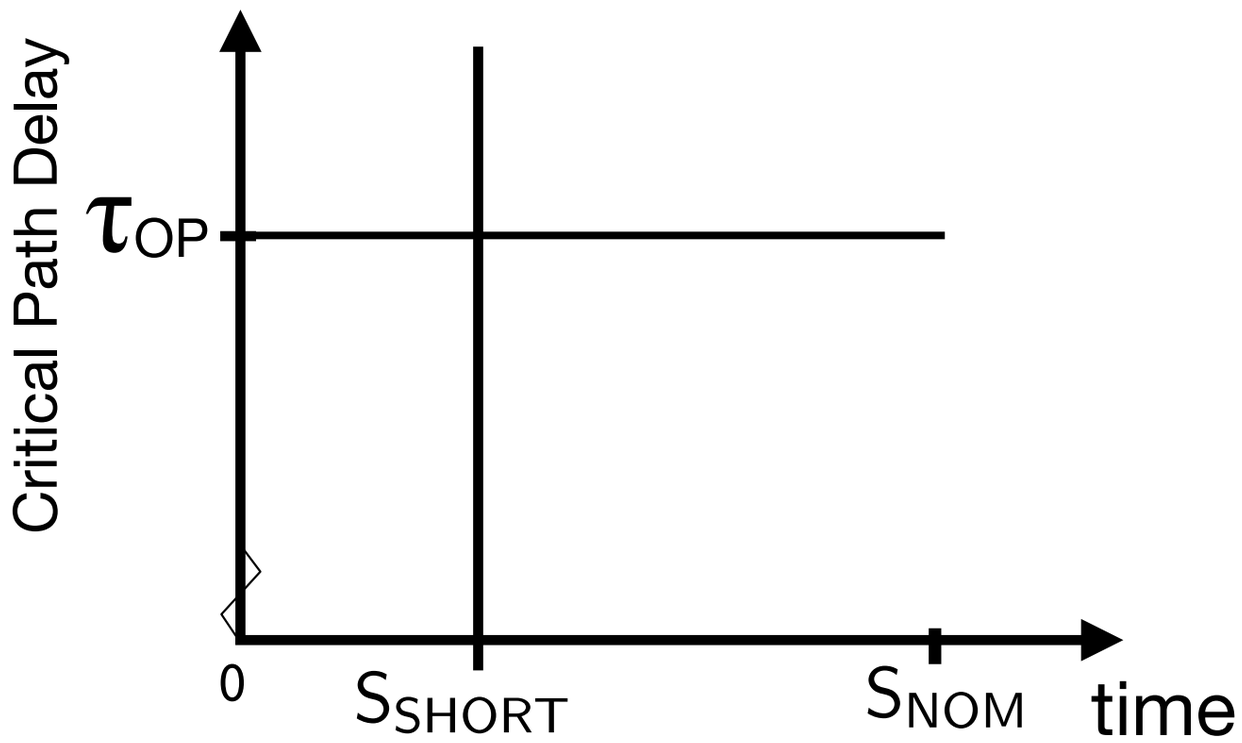
Idea: Aggressive DVSAM-Perf for a short service life to get even higher performance

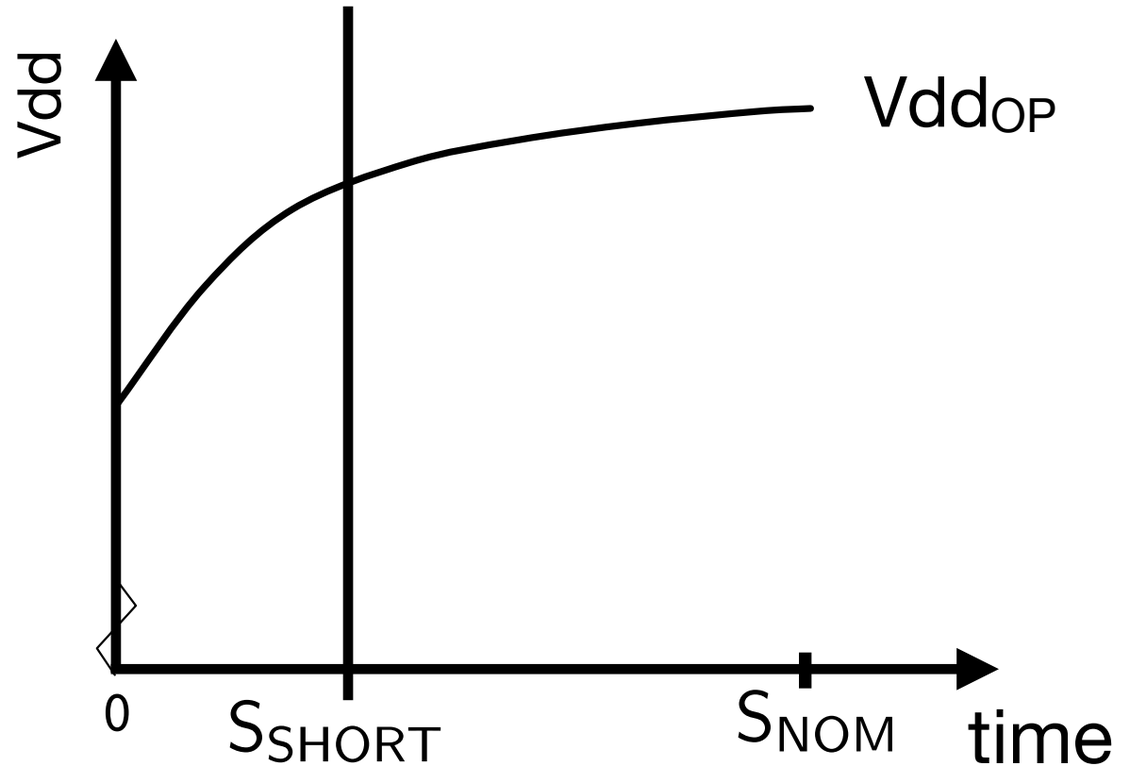


# DVSAM-Perf for $S_{SHORT}$

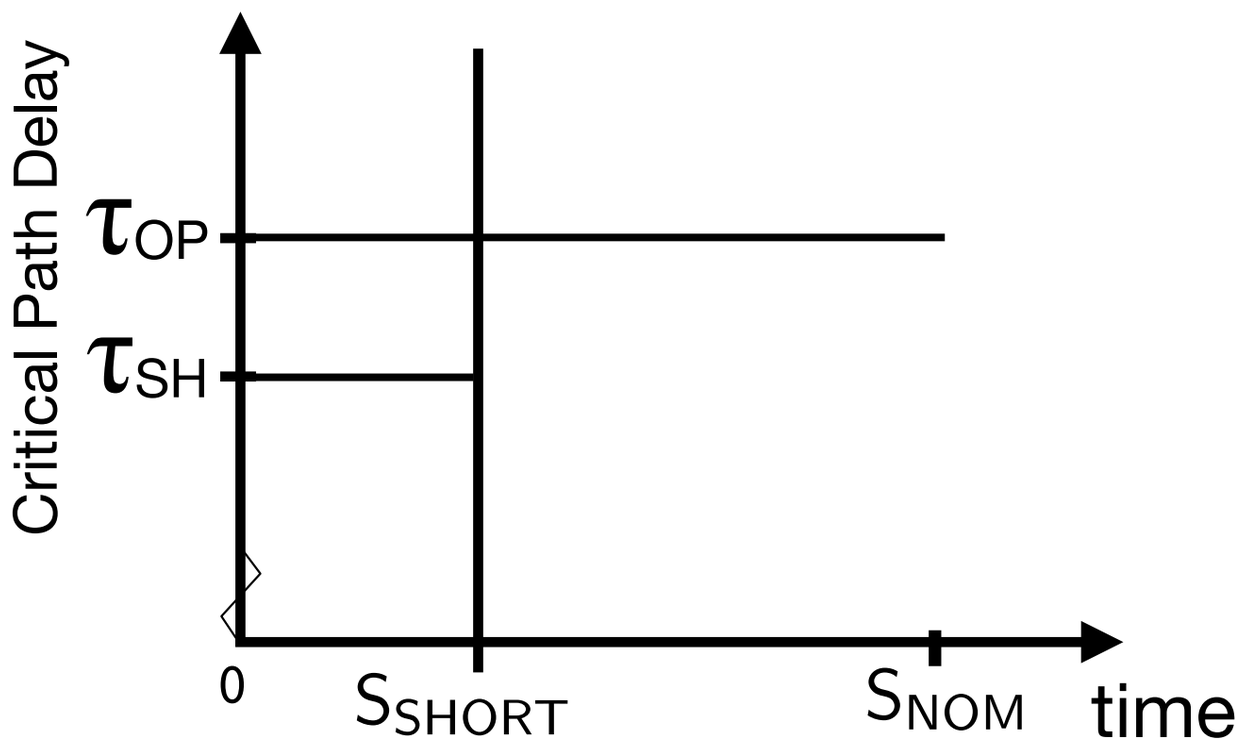
Idea: Aggressive DVSAM-Perf for a short service life to get even higher performance

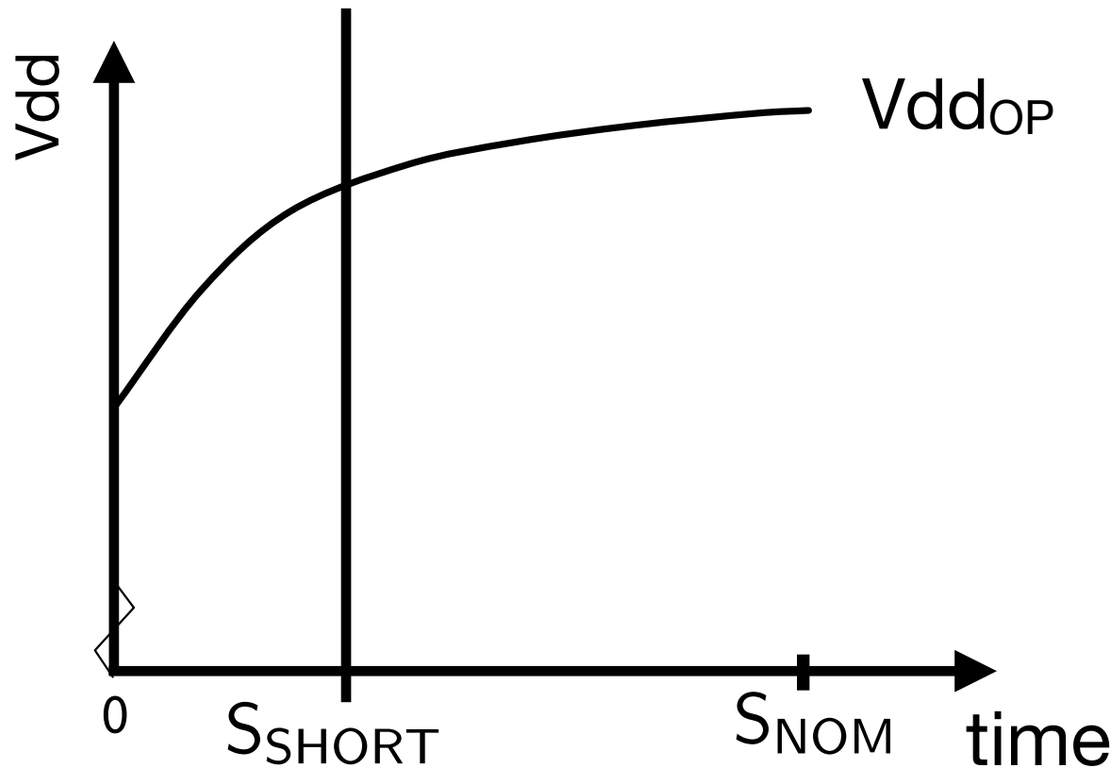


# DVSAM-Perf for $S_{SHORT}$

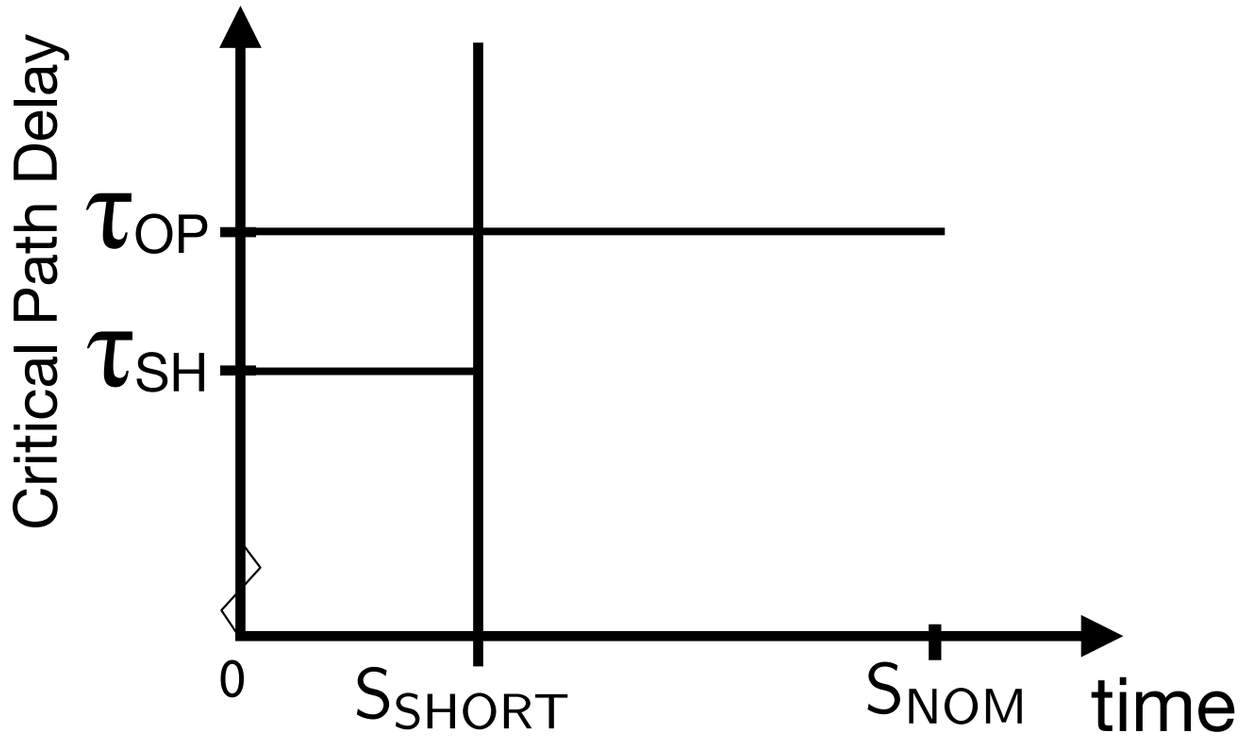
Idea: Aggressive DVSAM-Perf for a short service life to get even higher performance

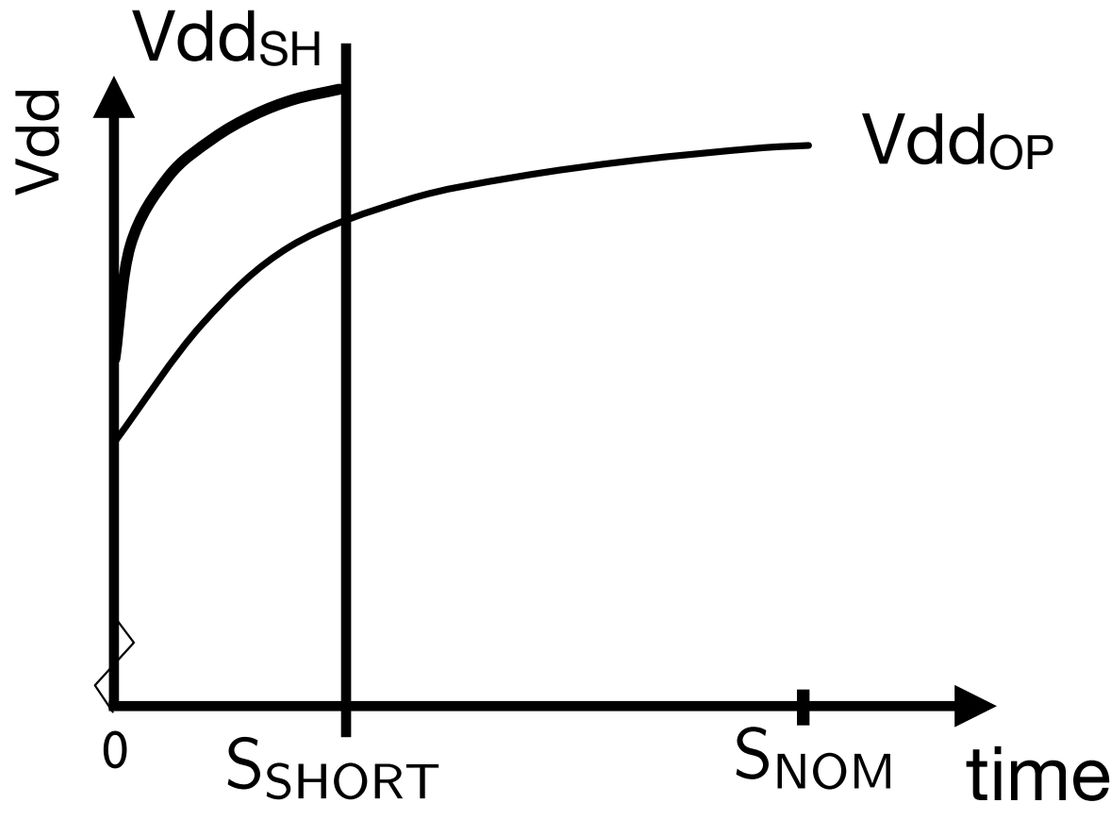


# DVSAM-Perf for $S_{SHORT}$

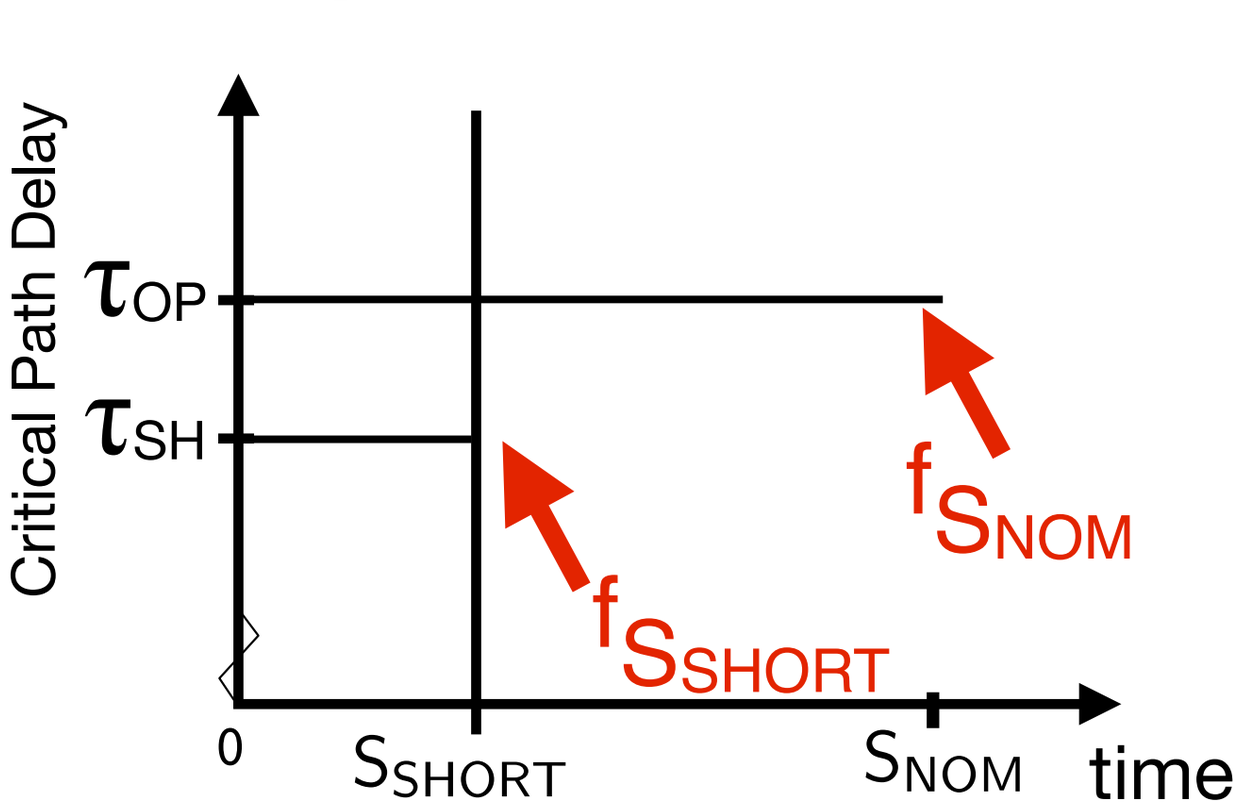
Idea: Aggressive DVSAM-Perf for a short service life to get even higher performance

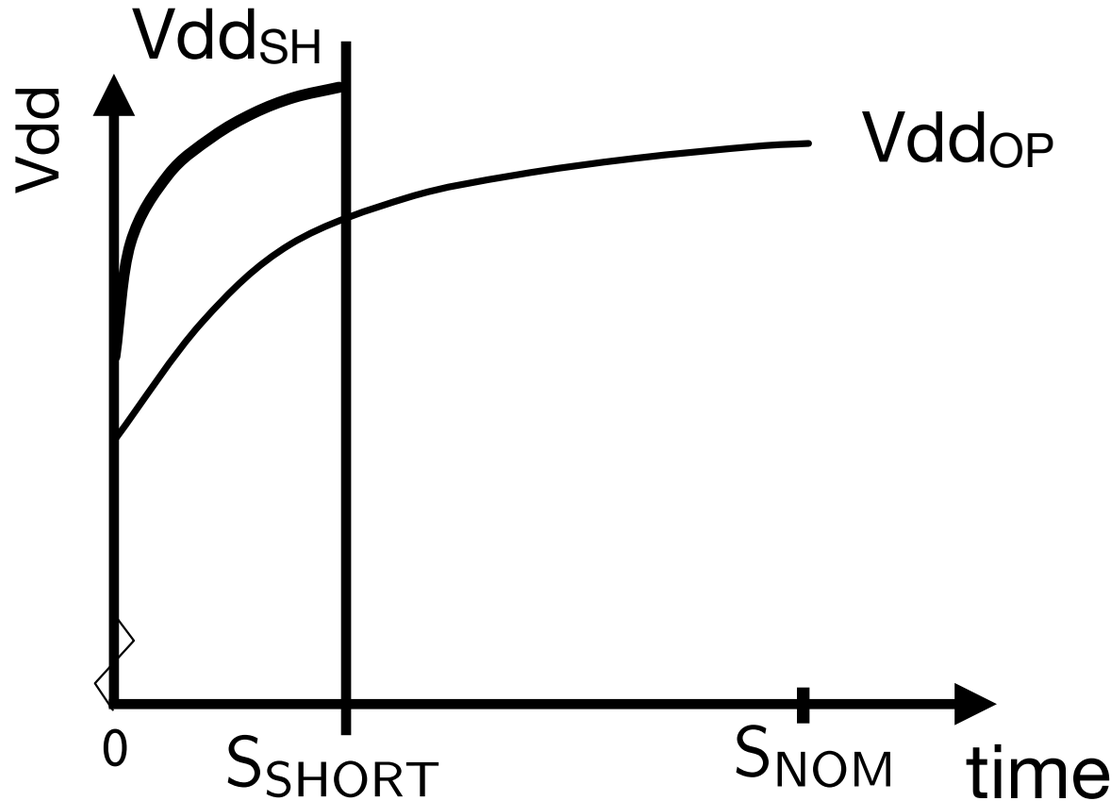


# DVSAM-Perf for $S_{SHORT}$

Idea: Aggressive DVSAM-Perf for a short service life to get even higher performance

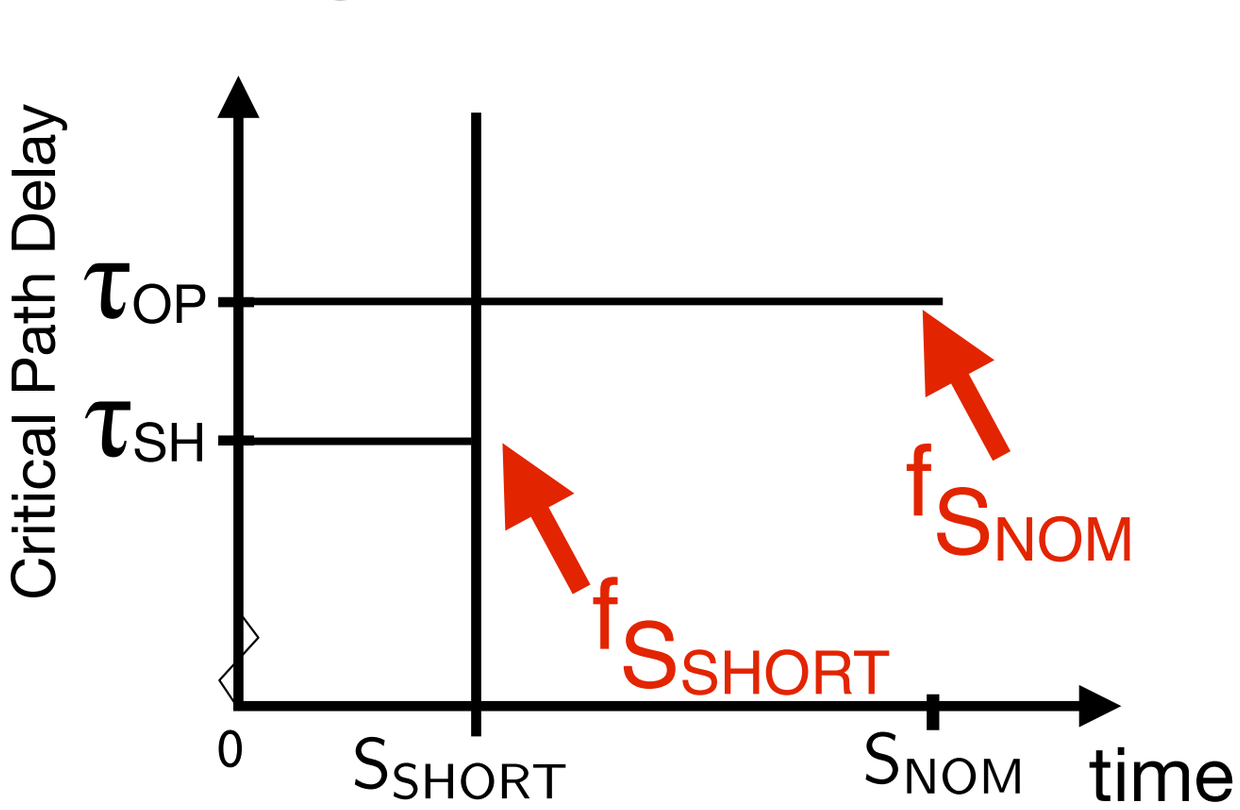


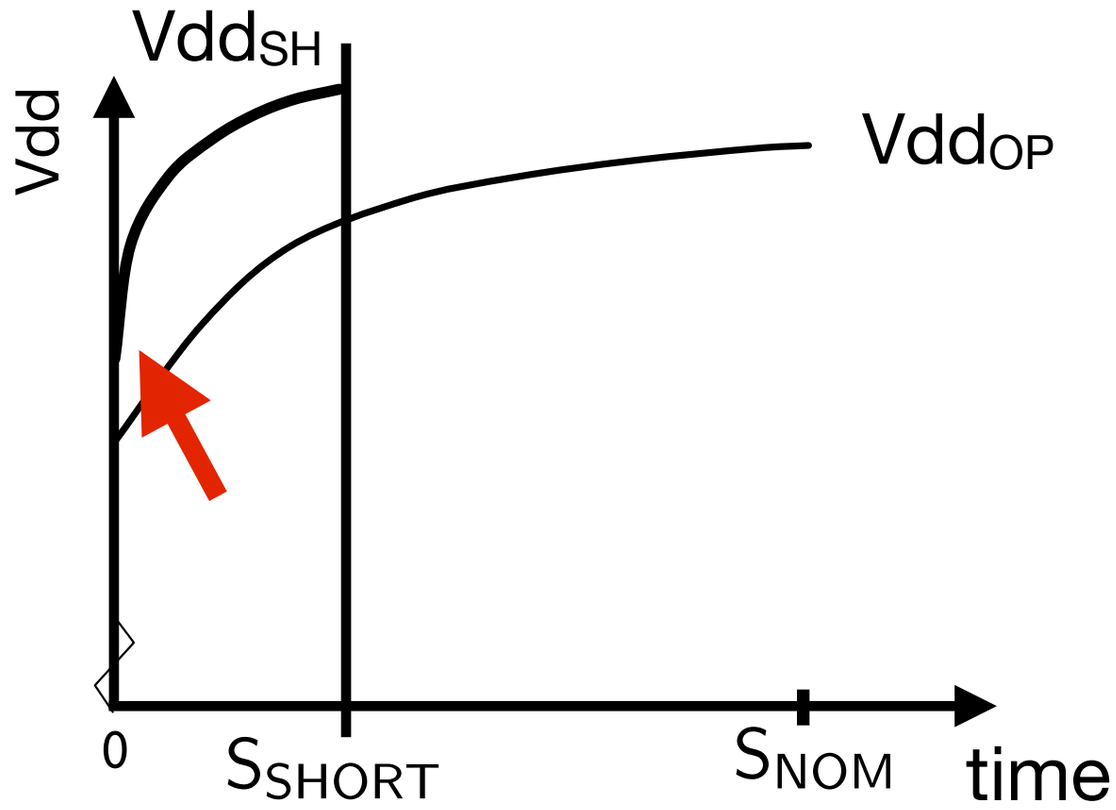




- Even higher frequency than DVSAM-Perf for short service life

# DVSAM-Perf for $S_{SHORT}$

Idea: Aggressive DVSAM-Perf for a short service life to get even higher performance

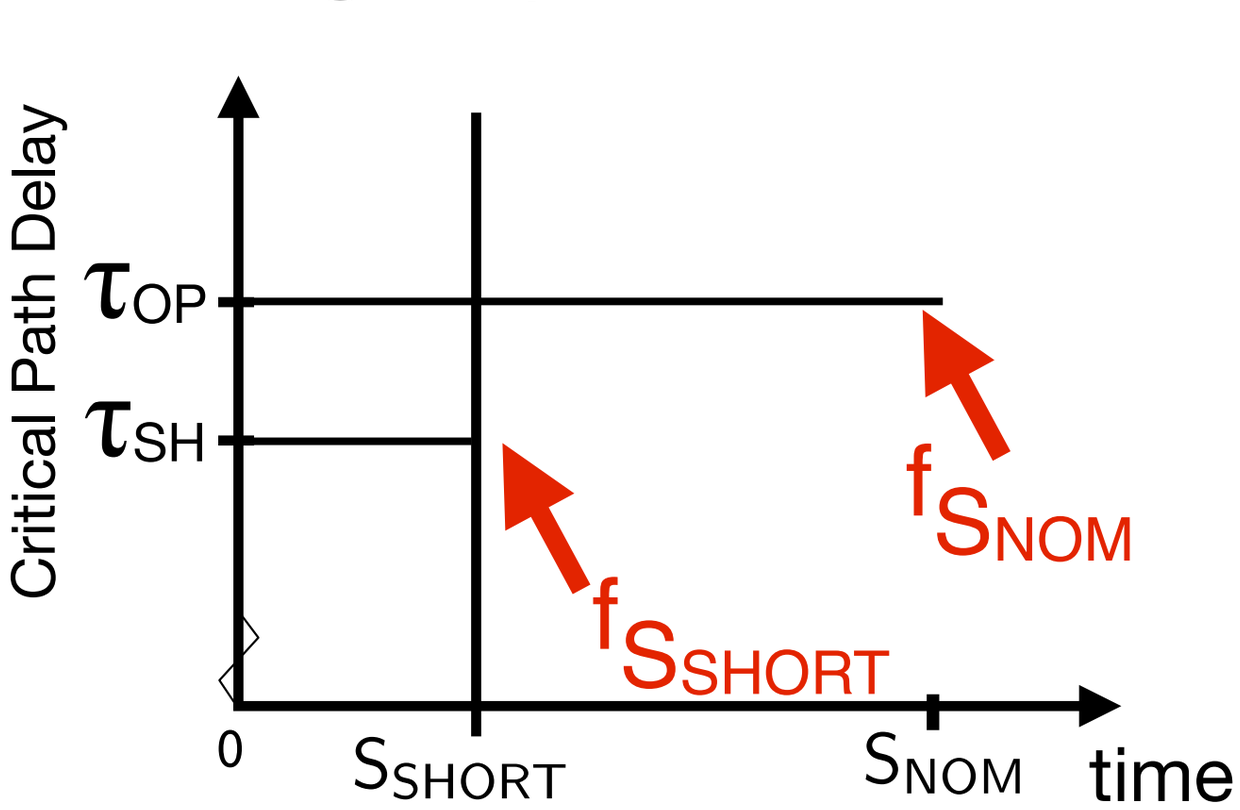


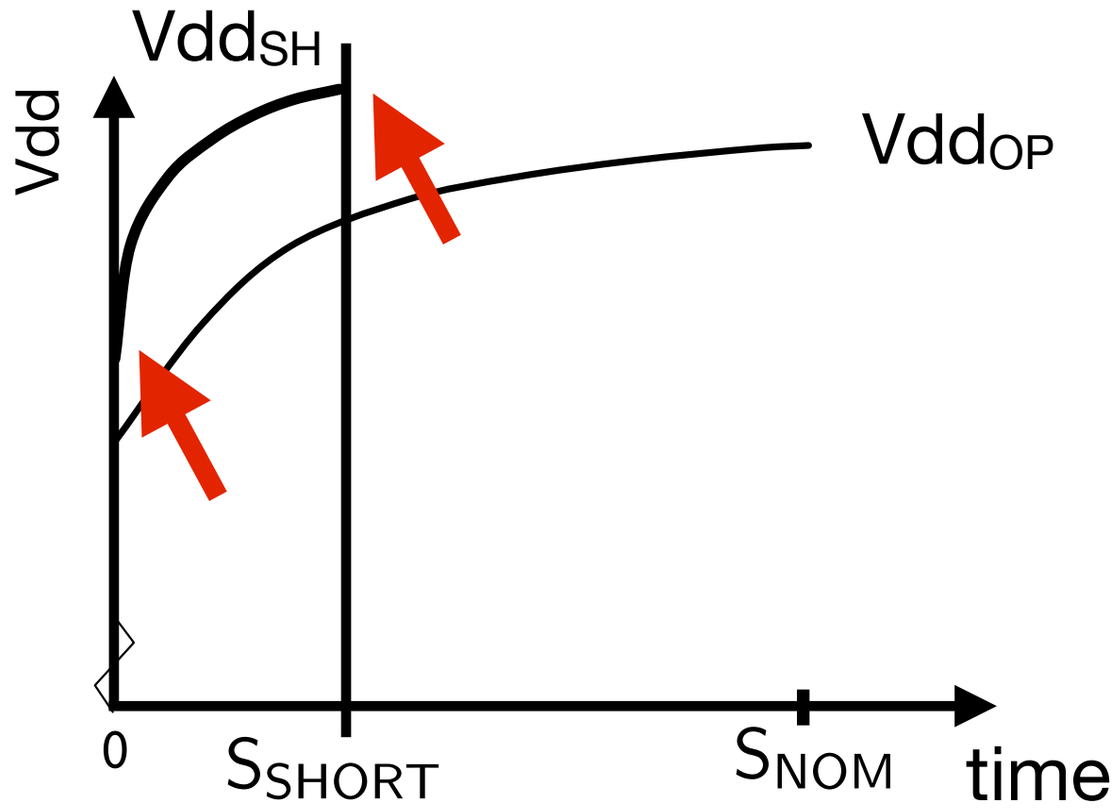




- Even higher frequency than DVSAM-Perf for short service life

# DVSAM-Perf for $S_{SHORT}$

Idea: Aggressive DVSAM-Perf for a short service life to get even higher performance

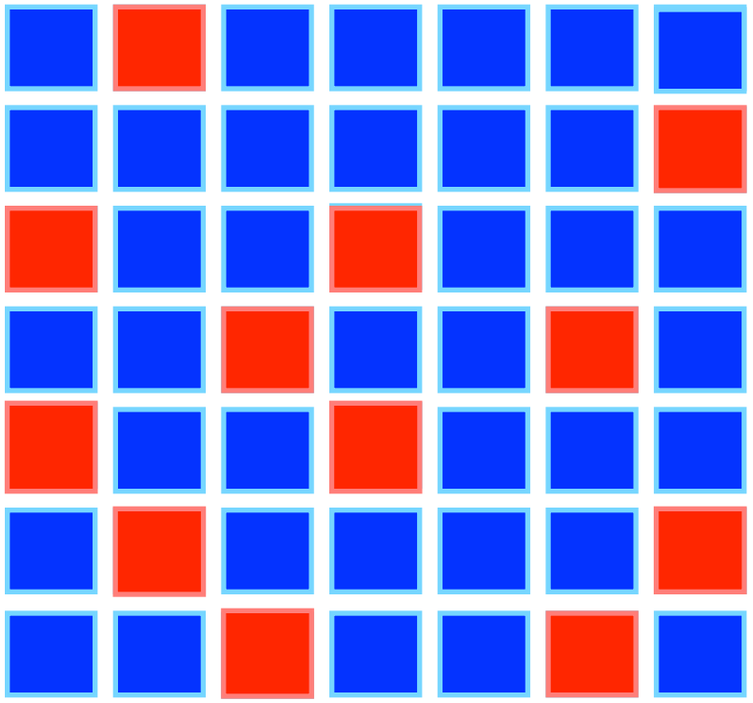



- Even higher frequency than DVSAM-Perf for short service life

# BubbleWrap Environments

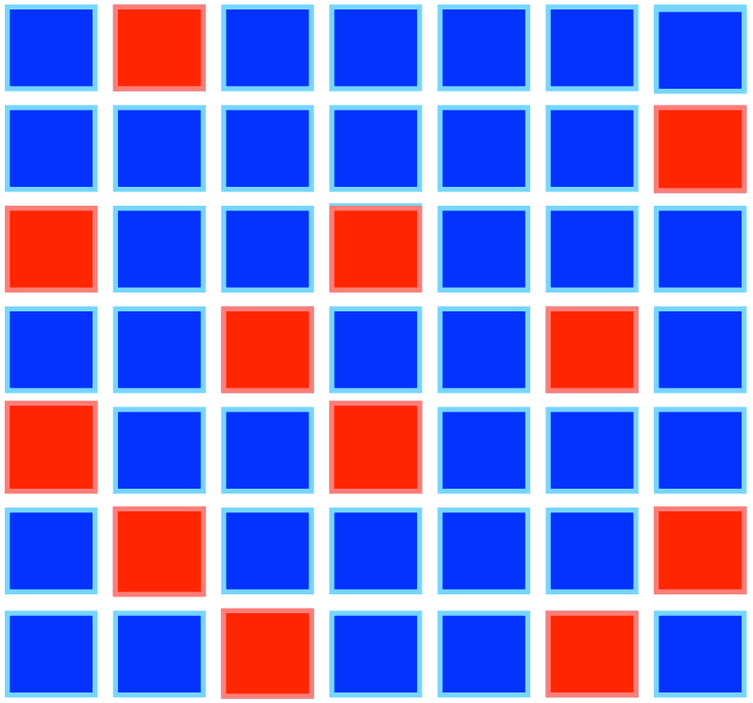
Throughput Cores



Expendable Cores

# BubbleWrap Environments

Throughput Cores

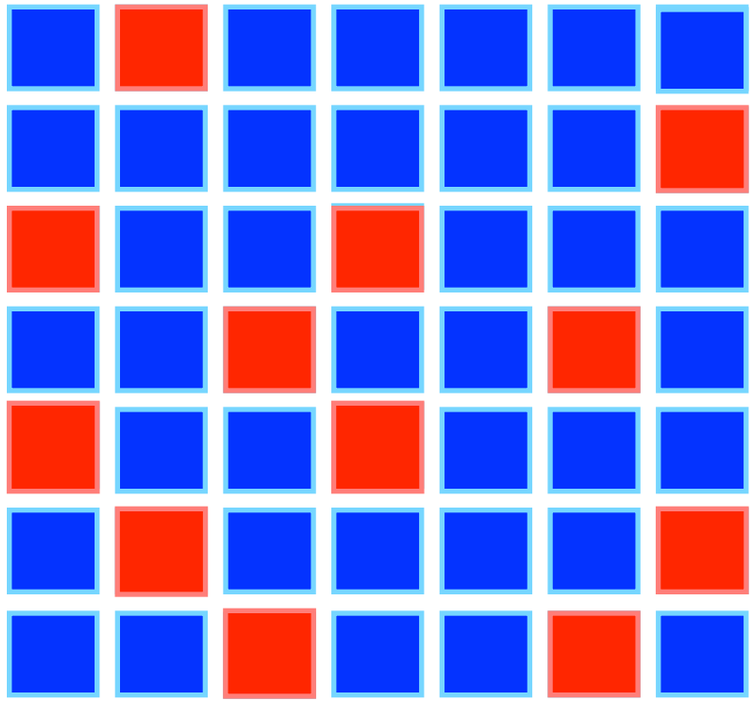


Expendable Cores

- Two choices for Throughput Cores

# BubbleWrap Environments

Throughput Cores

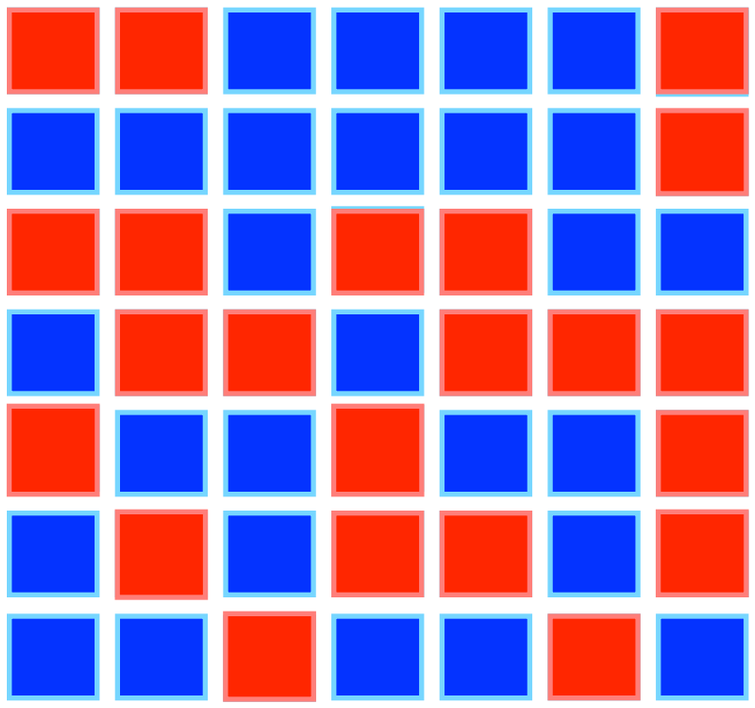


Expendable Cores

- Two choices for Throughput Cores
  - Nominal operation

# BubbleWrap Environments

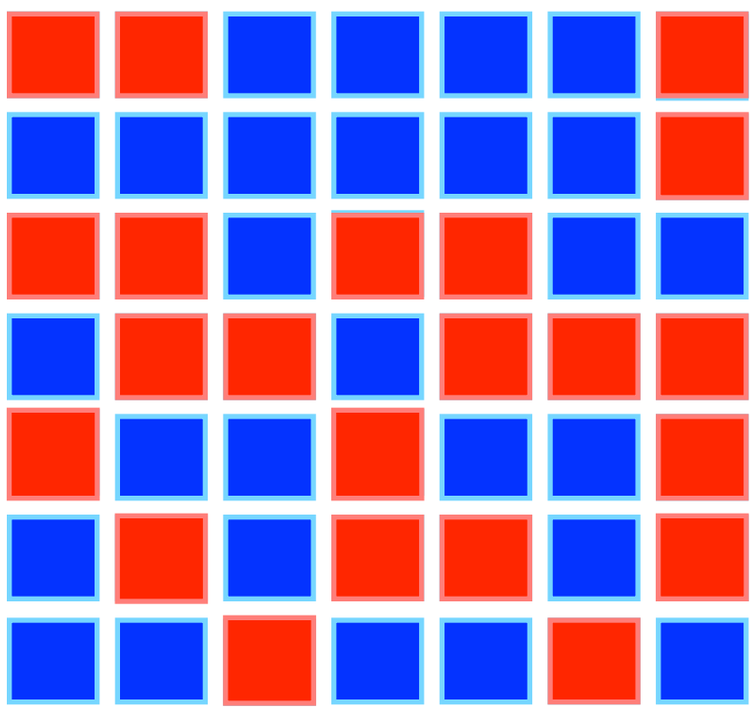
Throughput Cores



Expendable Cores

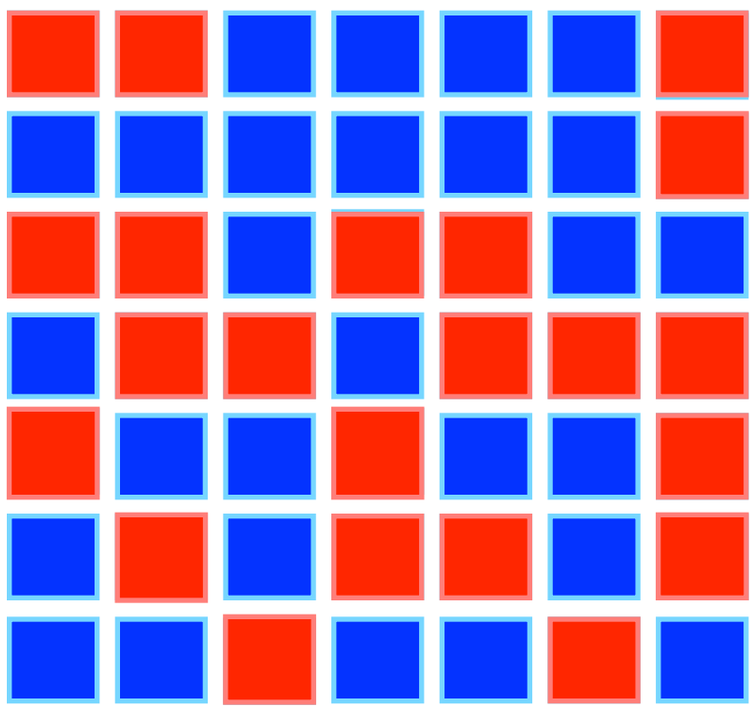
- Two choices for Throughput Cores
  - Nominal operation
  - Use DVSAM-Pow and expand the set of throughout cores for the same power budget

# BubbleWrap Environments



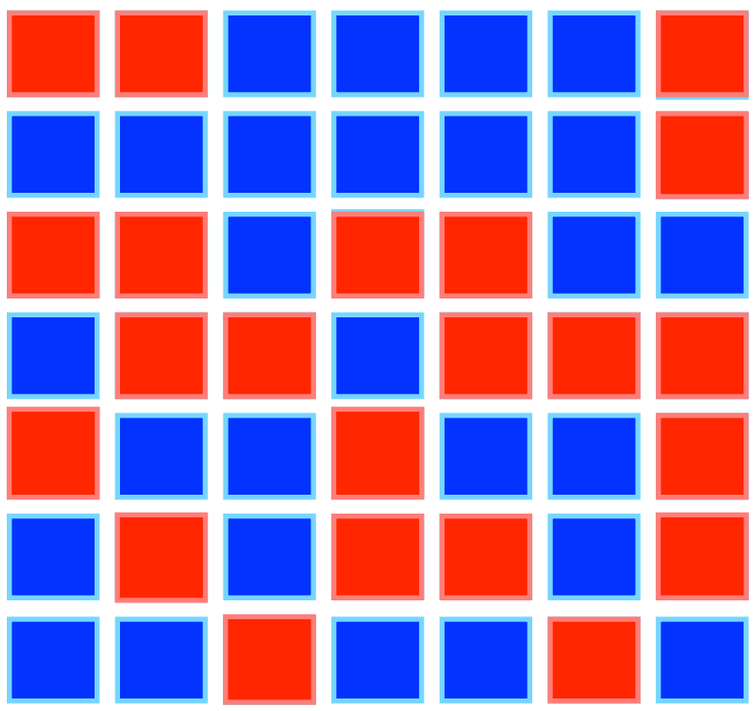

- Two choices for Throughput Cores
  - Nominal operation
  - Use DVSAM-Pow and expand the set of throughout cores for the same power budget
- Two choices for Expendable Cores

# BubbleWrap Environments




- Two choices for Throughput Cores
  - Nominal operation
  - Use DVSAM-Pow and expand the set of throughout cores for the same power budget
- Two choices for Expendable Cores
  - Higher, constant Vdd until $S_{SHORT}$; then discard

# BubbleWrap Environments




- Two choices for Throughput Cores
  - Nominal operation
  - Use DVSAM-Pow and expand the set of throughout cores for the same power budget
- Two choices for Expendable Cores
  - Higher, constant Vdd until $S_{SHORT}$; then discard
  - DVSAM-Perf until $S_{SHORT}$; then discard

# Hardware Support?

# Hardware Support?

- No change in the core architecture

# Hardware Support?

- No change in the core architecture
- Need circuits to measure aging

# Hardware Support?

- No change in the core architecture
- Need circuits to measure aging
- Need high-precision DVS

# Hardware Support?

- No change in the core architecture
- Need circuits to measure aging
- Need high-precision DVS
- Clock and power distribution

# Hardware Support?

- No change in the core architecture
- Need circuits to measure aging
- Need high-precision DVS
- Clock and power distribution
- Two separate V/f domains: One for Expendable and one for Throughput Cores

# BubbleWrap Evaluation

# BubbleWrap Evaluation

- 32 core chip: $N_T$ = 16 Throughput and $N_E$ = 16 Expendable cores

# BubbleWrap Evaluation

- 32 core chip: $N_T$ = 16 Throughput and $N_E$ = 16 Expendable cores
- 22nm high-k metal-gate process

# BubbleWrap Evaluation

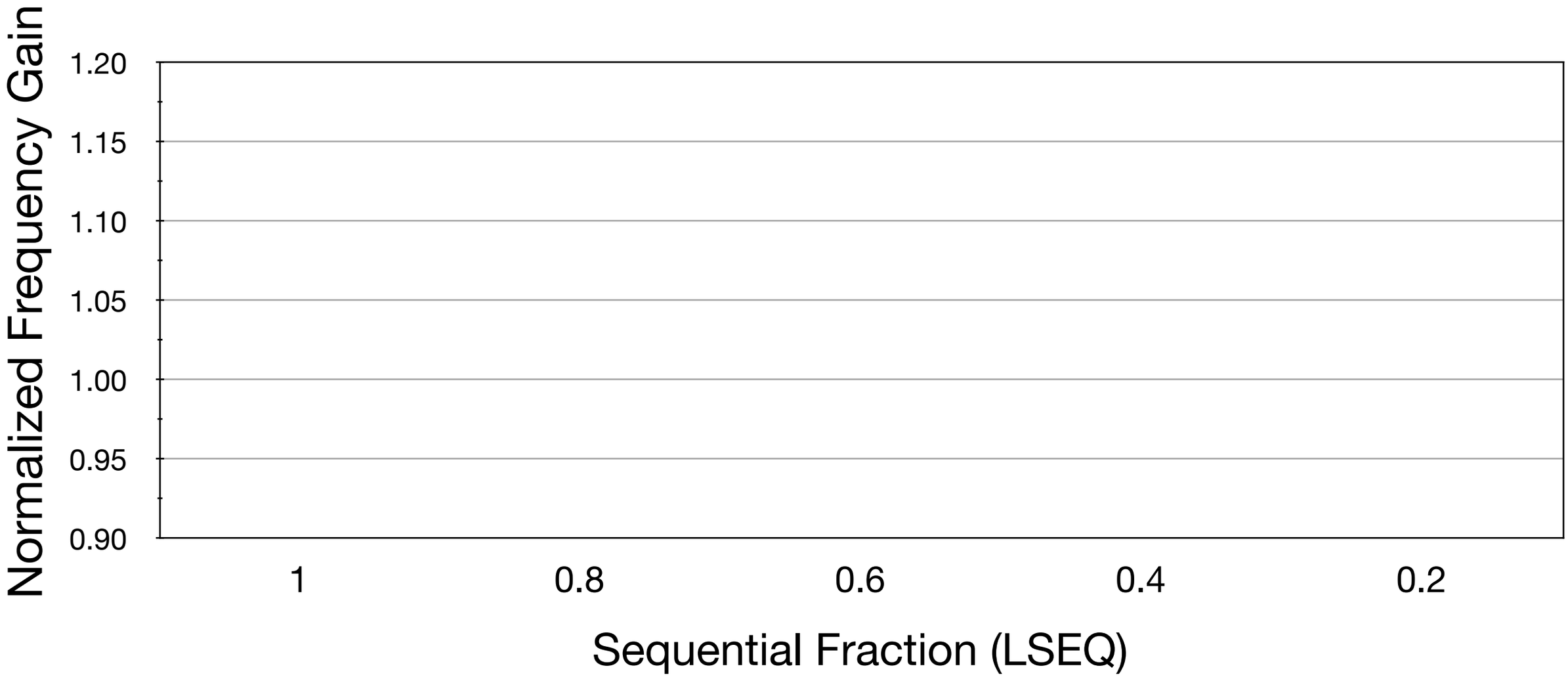
- 32 core chip: $N_T$ = 16 Throughput and $N_E$ = 16 Expendable cores
- 22nm high-k metal-gate process
- Multiprogrammed workload synthesized from SPEC2000

# BubbleWrap Evaluation

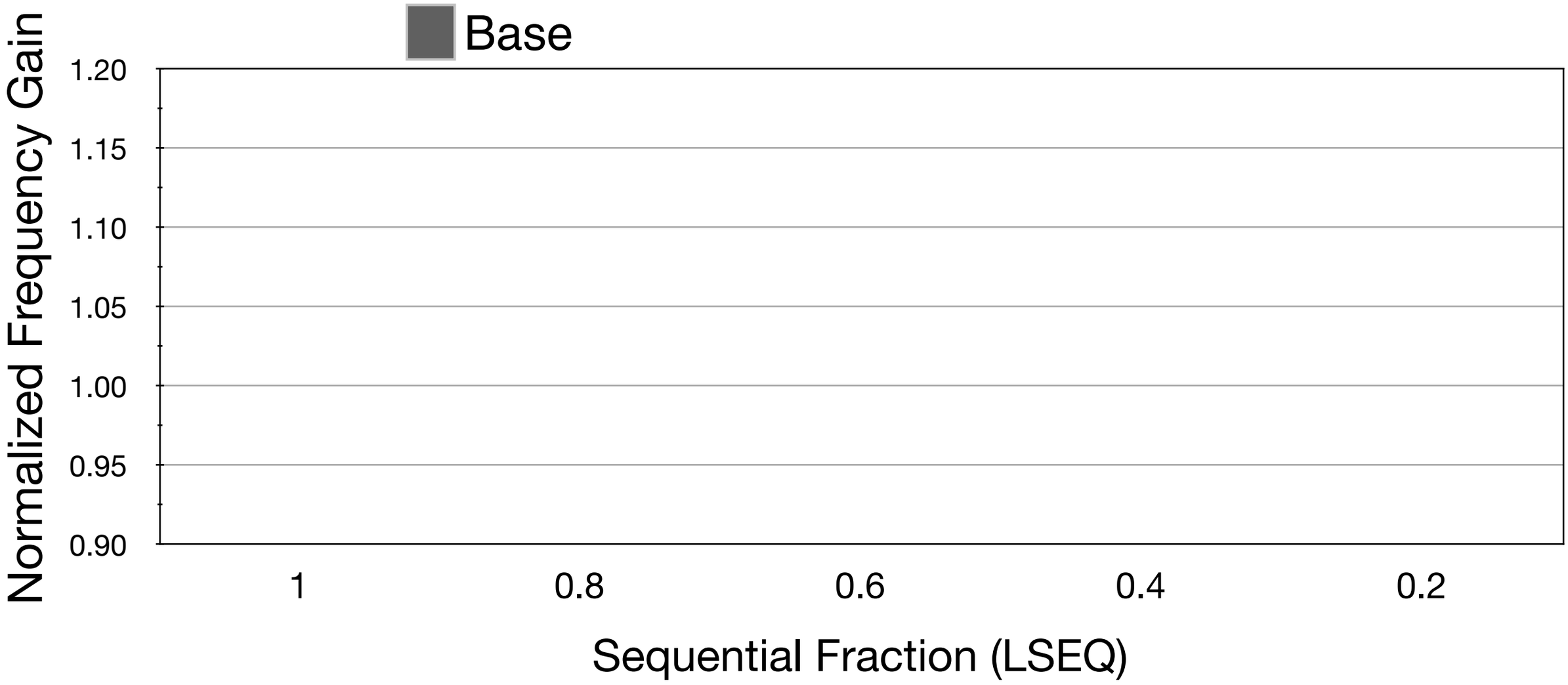
- 32 core chip: $N_T$ = 16 Throughput and $N_E$ = 16 Expendable cores
- 22nm high-k metal-gate process
- Multiprogrammed workload synthesized from SPEC2000
- SESC enhanced by a power & thermal model

# Frequency Gains of Sequential Section

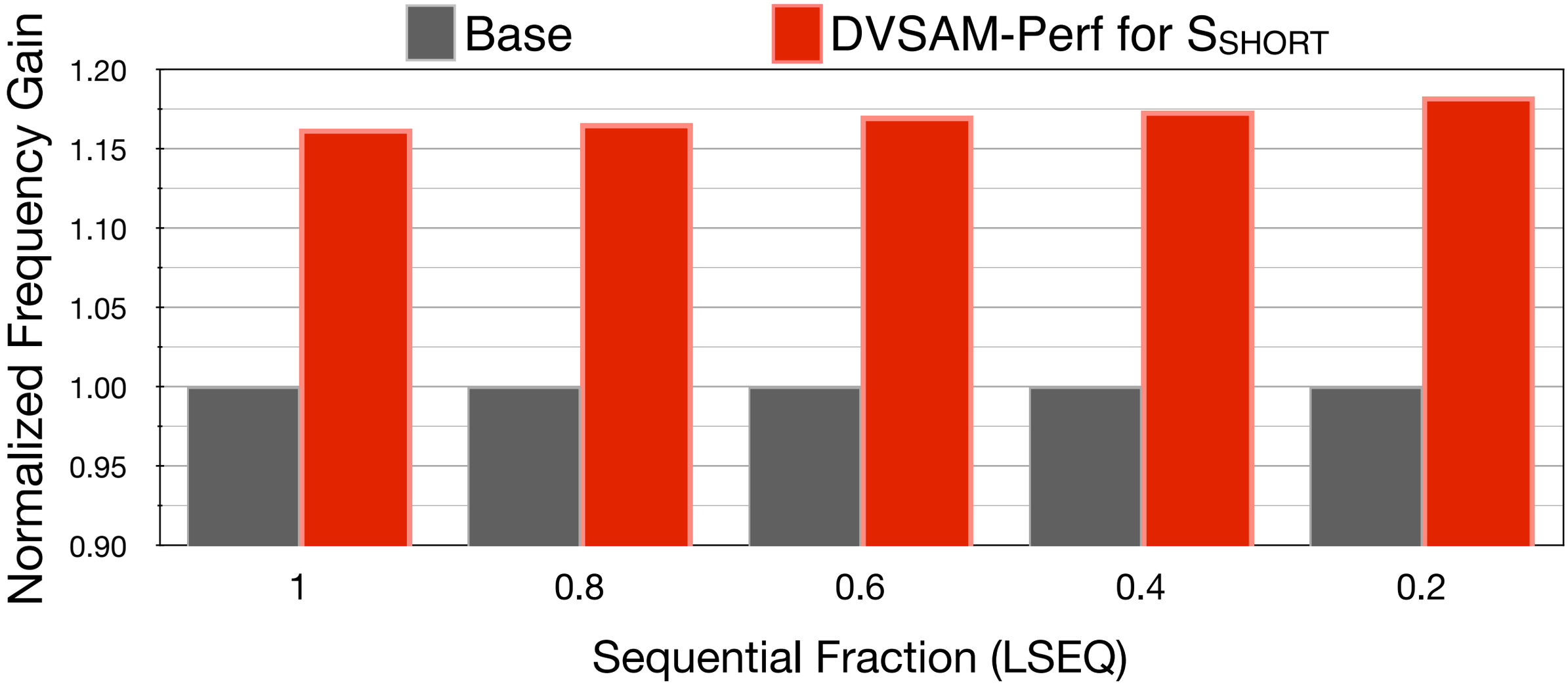
# Frequency Gains of Sequential Section



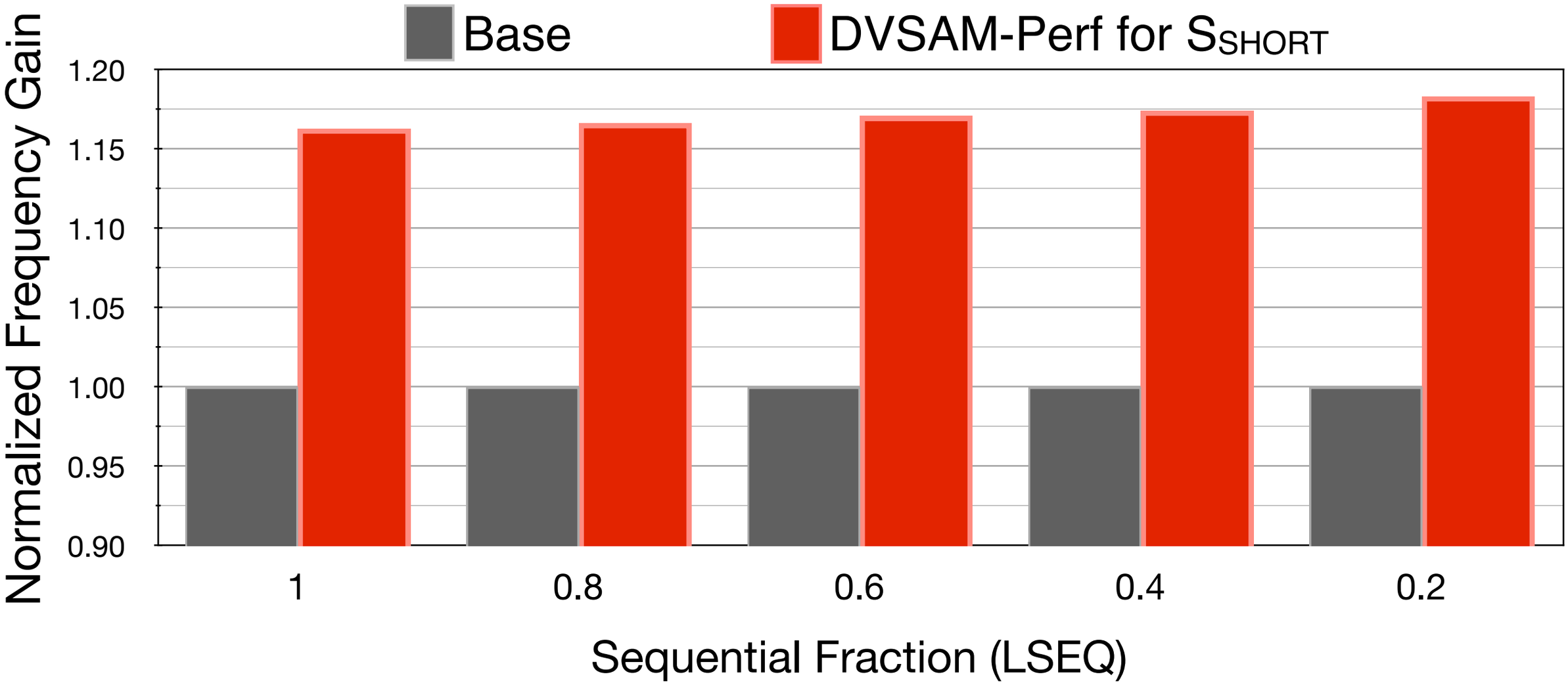
# Frequency Gains of Sequential Section



# Frequency Gains of Sequential Section

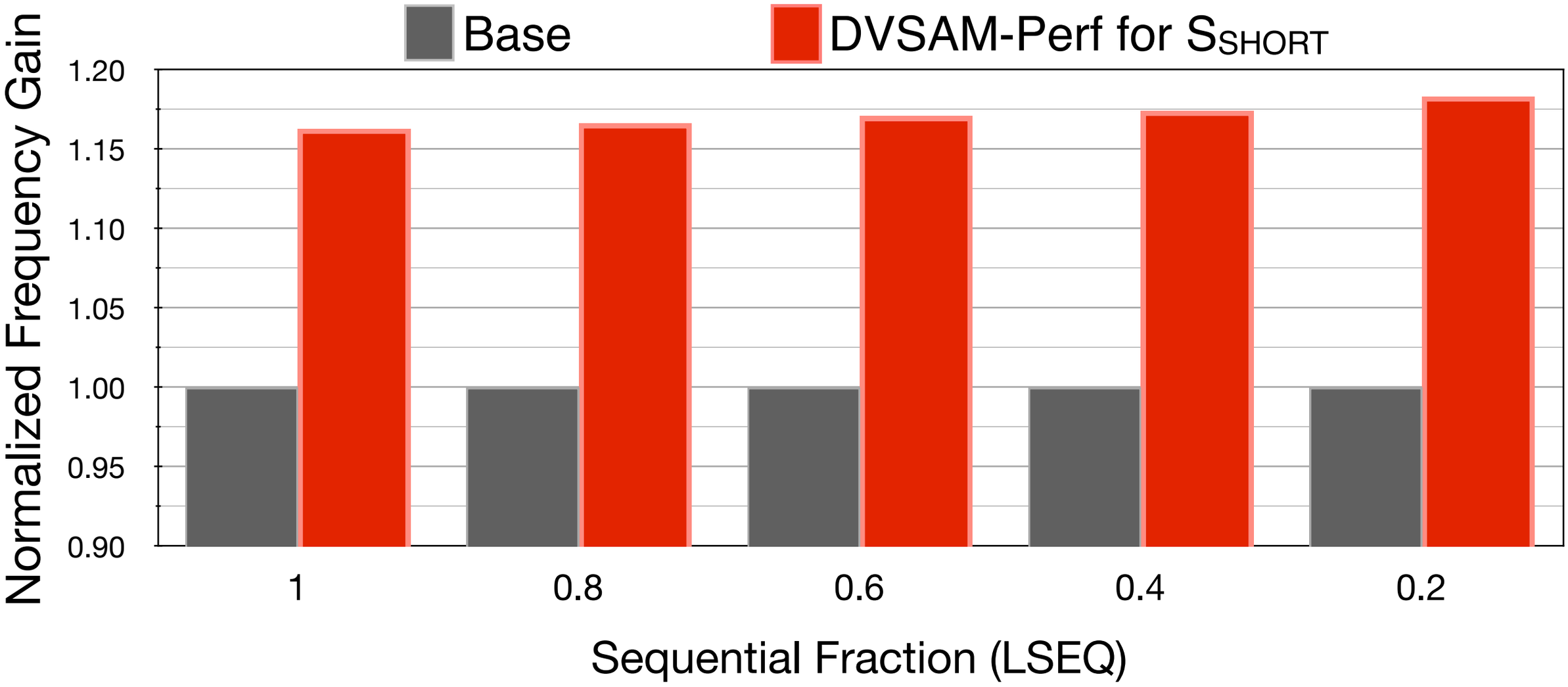


# Frequency Gains of Sequential Section



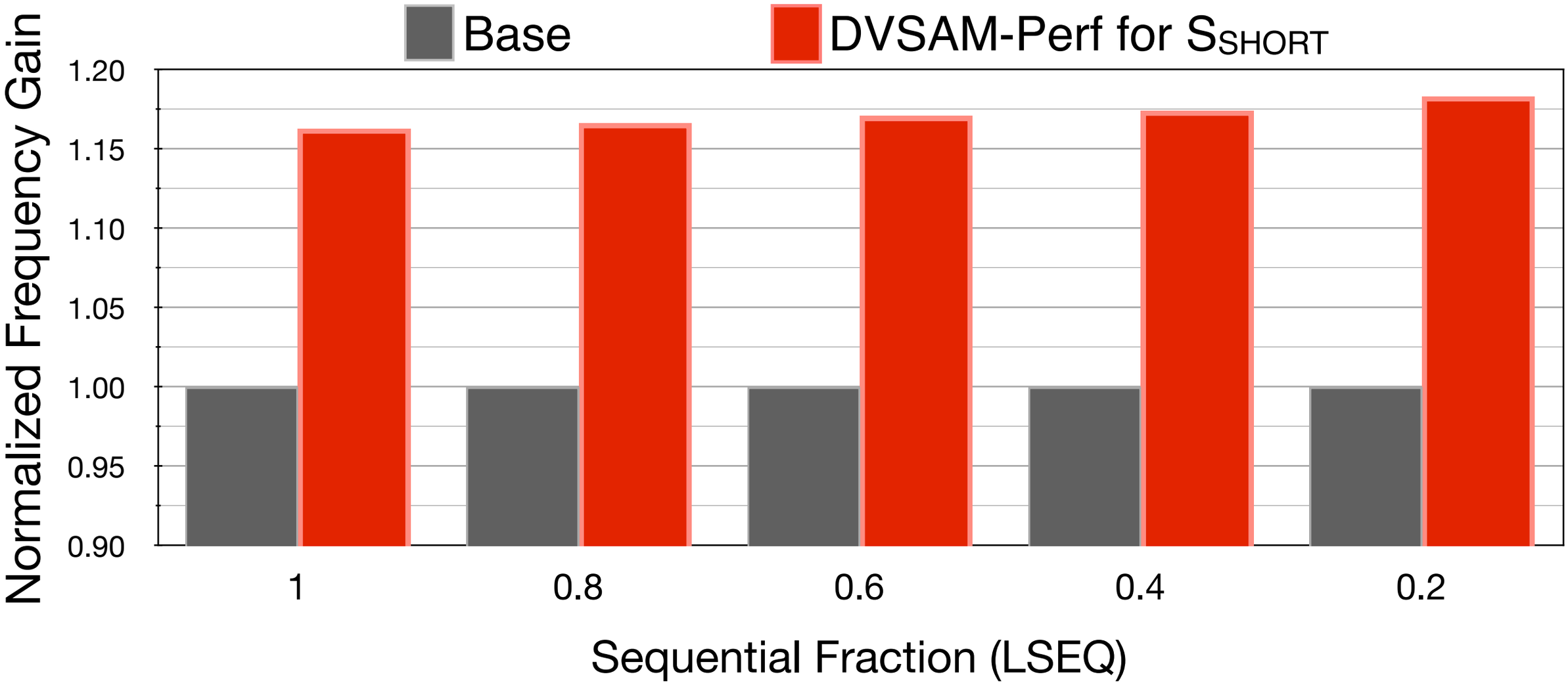

- Large f gains are feasible

# Frequency Gains of Sequential Section




- Large f gains are feasible
- f increases with smaller sequential section

# Frequency Gains of Sequential Section




- Large f gains are feasible
- f increases with smaller sequential section
- For DVSAM-Perf, each expendable core runs for $L_{SEQ}/N_E$ x $S_{NOM}$

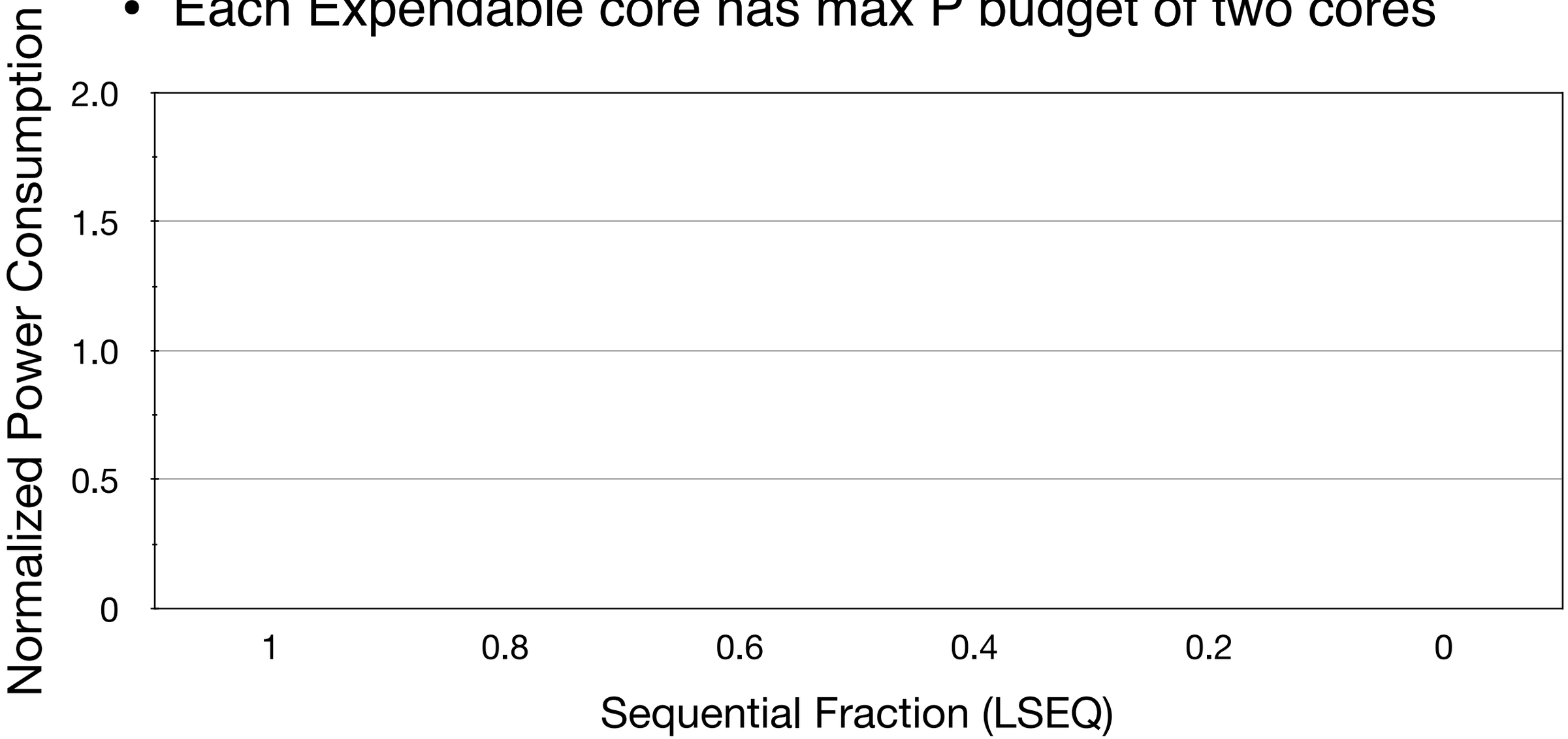
# Power Consumption of Sequential Section

# Power Consumption of Sequential Section

- Each Expendable core has max P budget of two cores

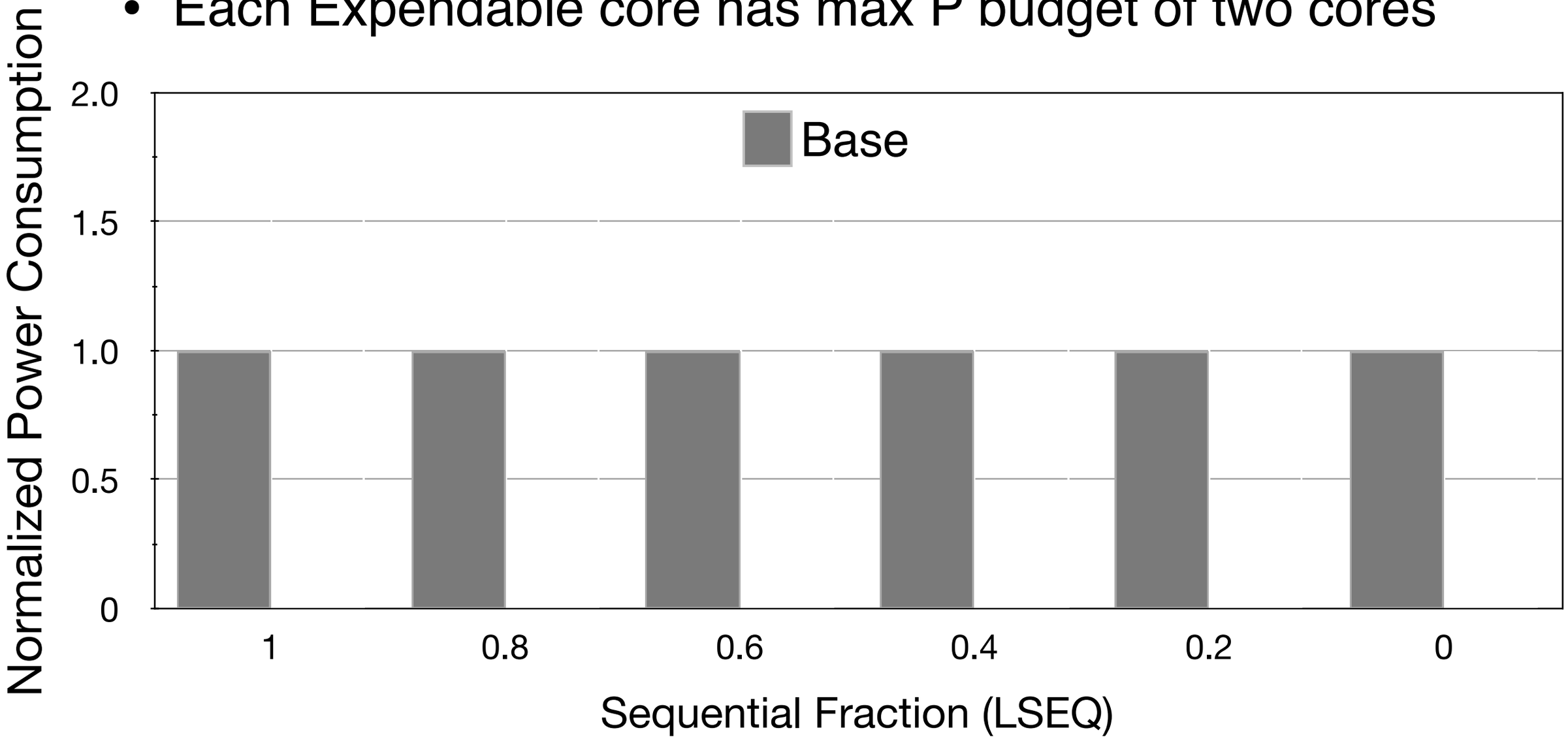
# Power Consumption of Sequential Section

- Each Expendable core has max P budget of two cores



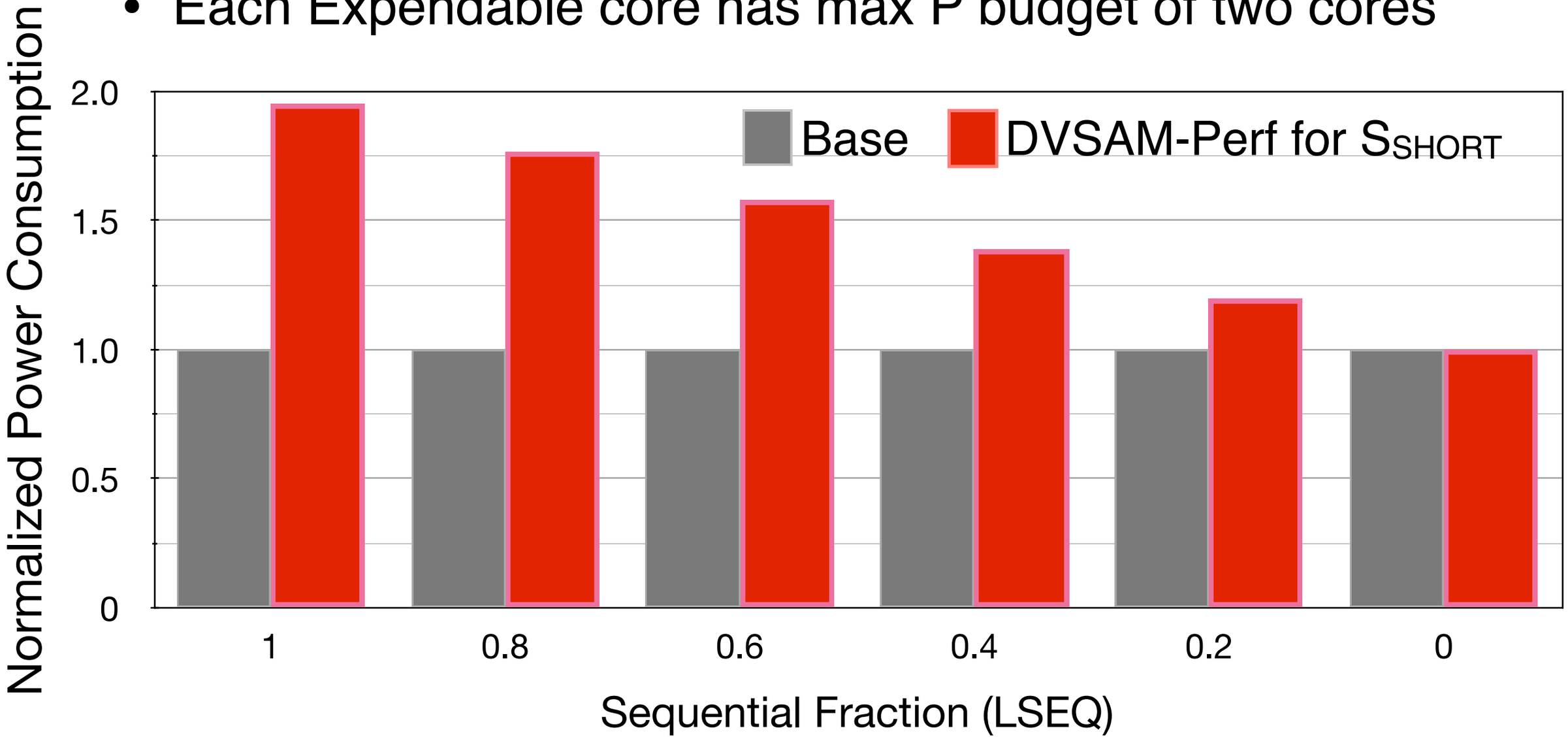
# Power Consumption of Sequential Section

- Each Expendable core has max P budget of two cores



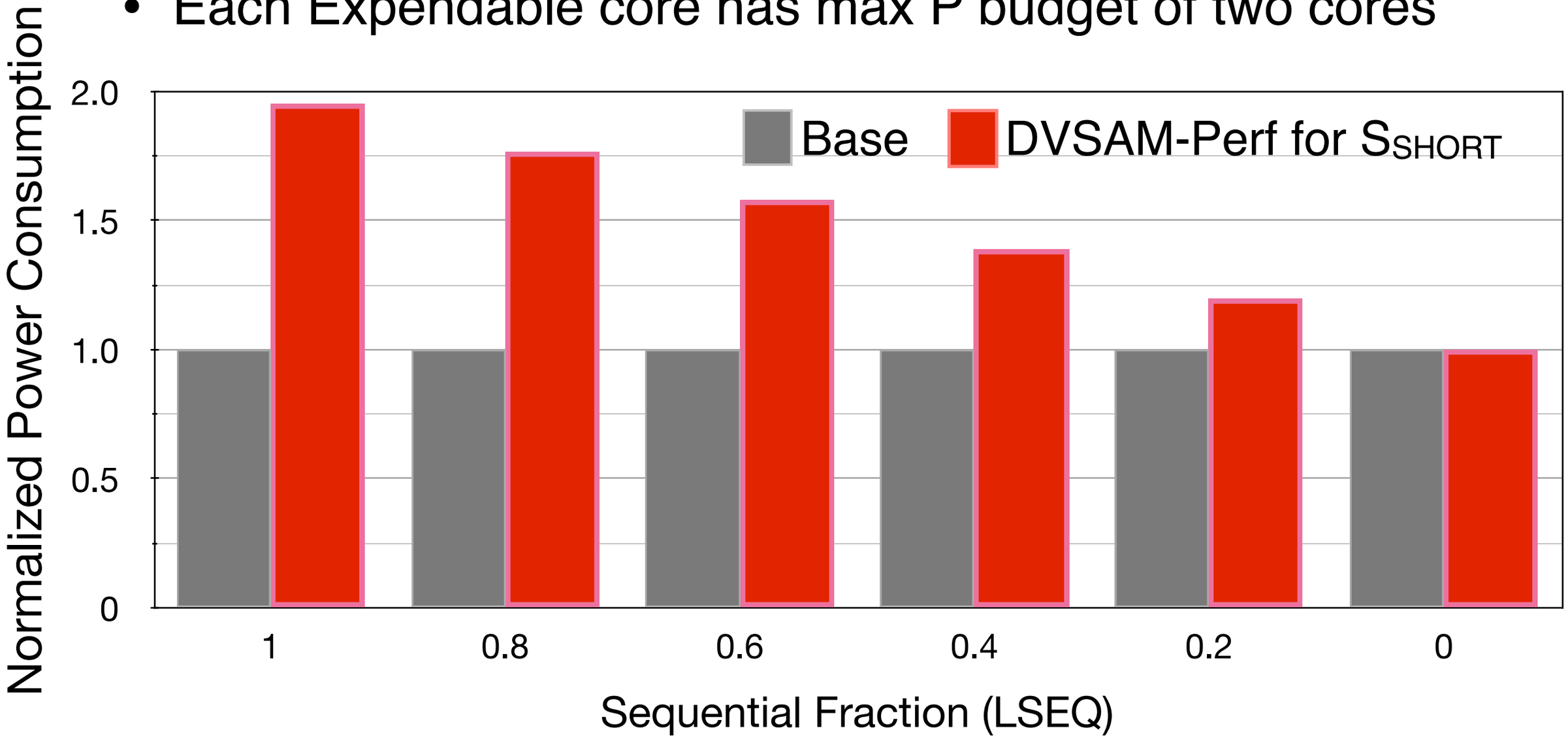
# Power Consumption of Sequential Section

- Each Expendable core has max P budget of two cores



# Power Consumption of Sequential Section

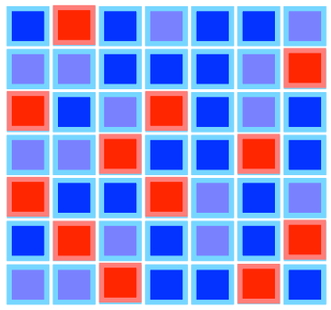
- Each Expendable core has max P budget of two cores




- Tolerable power cost for the frequency gains

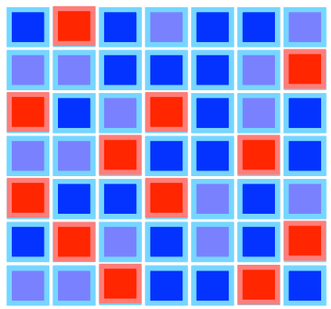
# Conclusion

The BubbleWrap Many-Core:
Exploiting dormant cores for sequential acceleration



# Conclusion

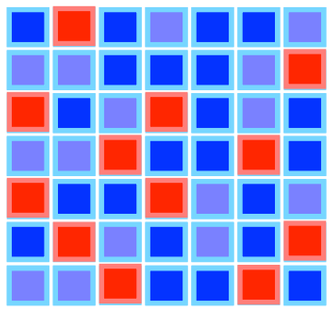
The BubbleWrap Many-Core:
Exploiting dormant cores for sequential acceleration



- Simple homogeneous design

# Conclusion

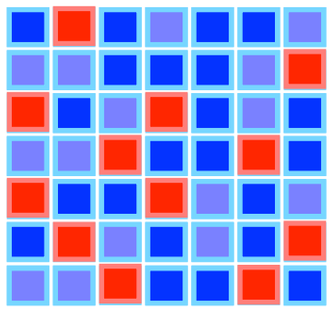
The BubbleWrap Many-Core:
Exploiting dormant cores for sequential acceleration



- Simple homogeneous design
- No architectural or software changes

# Conclusion

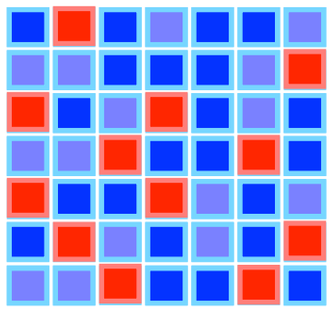
The BubbleWrap Many-Core:
Exploiting dormant cores for sequential acceleration



- Simple homogeneous design
- No architectural or software changes
- Improves sequential and parallel performance

# Conclusion

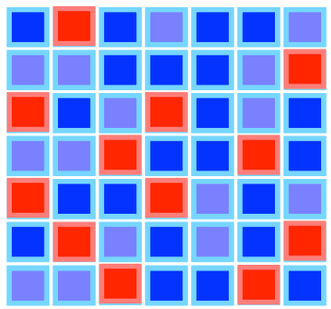
The BubbleWrap Many-Core:
Exploiting dormant cores for sequential acceleration



- Simple homogeneous design
- No architectural or software changes
- Improves sequential and parallel performance
  - Fully sequential applications at 16% higher f

# Conclusion

The BubbleWrap Many-Core:
Exploiting dormant cores for sequential acceleration



- Simple homogeneous design
- No architectural or software changes
- Improves sequential and parallel performance
  - Fully sequential applications at 16% higher f
  - Fully parallel applications at 30% higher throughput

# The BubbleWrap Many-Core:
## Popping Cores for Sequential Acceleration

**Ulya Karpuzcu**, Brian Greskamp, Josep Torrellas
University of Illinois

http://iacoma.cs.uiuc.edu/

University of Illinois

i-acoma group