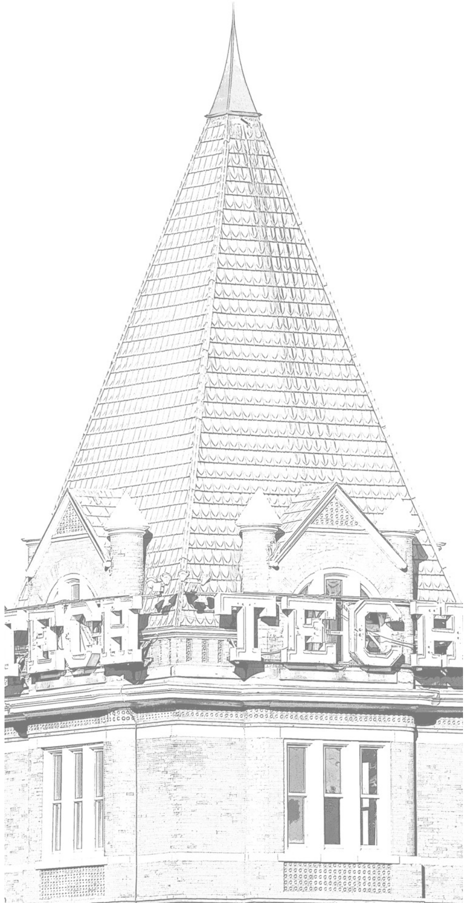
Reducing Read Latency of Phase Change Memory via Early Read and Turbo Read



Feb 9th 2015

HPCA-21 San Francisco, USA

Prashant Nair - Georgia Tech

Chiachen Chou - Georgia Tech

Bipin Rajendran – IIT Bombay

Moinuddin Qureshi - Georgia Tech

INTRODUCTION TO PCM

- Phase Change Memory (PCM) promises higher density and better scalability

Key Challenges:

- Limited Endurance (10-100M writes/cell)
- High Write Latency (4X-8X higher than PCM read)
- High Read Latency (2X of DRAM)

INTRODUCTION TO PCM

- Phase Change Memory (PCM) promises higher density and better scalability

Key Challenges:

- Limited Endurance (10-100M writes/cell)
  - Wear Leveling, Error correction, Graceful degradation
- High Write Latency (4X-8X higher than PCM read)

- High Read Latency (2X of DRAM)

INTRODUCTION TO PCM

- Phase Change Memory (PCM) promises higher density and better scalability

Key Challenges:

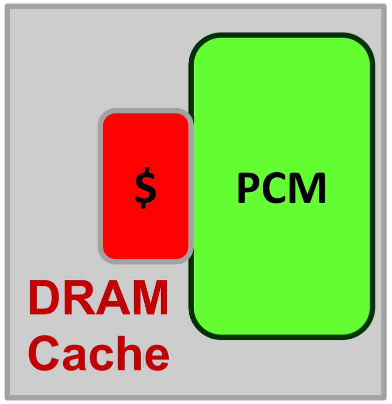
- Limited Endurance (10-100M writes/cell)
  - Wear Leveling, Error correction, Graceful degradation
- High Write Latency (4X-8X higher than PCM read)
  - PreSET, Write Cancellation, Write Pausing
- High Read Latency (2X of DRAM)

INTRODUCTION TO PCM

- Phase Change Memory (PCM) promises higher density and better scalability

Key Challenges:

- Limited Endurance (10-100M writes/cell)
  - Wear Leveling, Error correction, Graceful degradation
- High Write Latency (4X-8X higher than PCM read)
  - PreSET, Write Cancellation, Write Pausing
- High Read Latency (2X of DRAM)
  - Hybrid Memory, combining PCM and DRAM



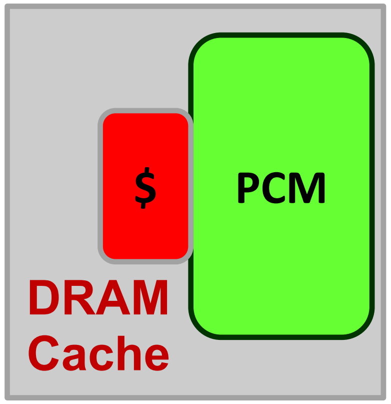
Hybrid Memory

INTRODUCTION TO PCM

- Phase Change Memory (PCM) promises higher density and better scalability

Key Challenges:

- Limited Endurance (10-100M writes/cell)
  - Wear Leveling, Error correction, Graceful degradation
- High Write Latency (4X-8X higher than PCM read)
  - PreSET, Write Cancellation, Write Pausing
- High Read Latency (2X of DRAM)
  - Hybrid Memory, combining PCM and DRAM



Hybrid Memory

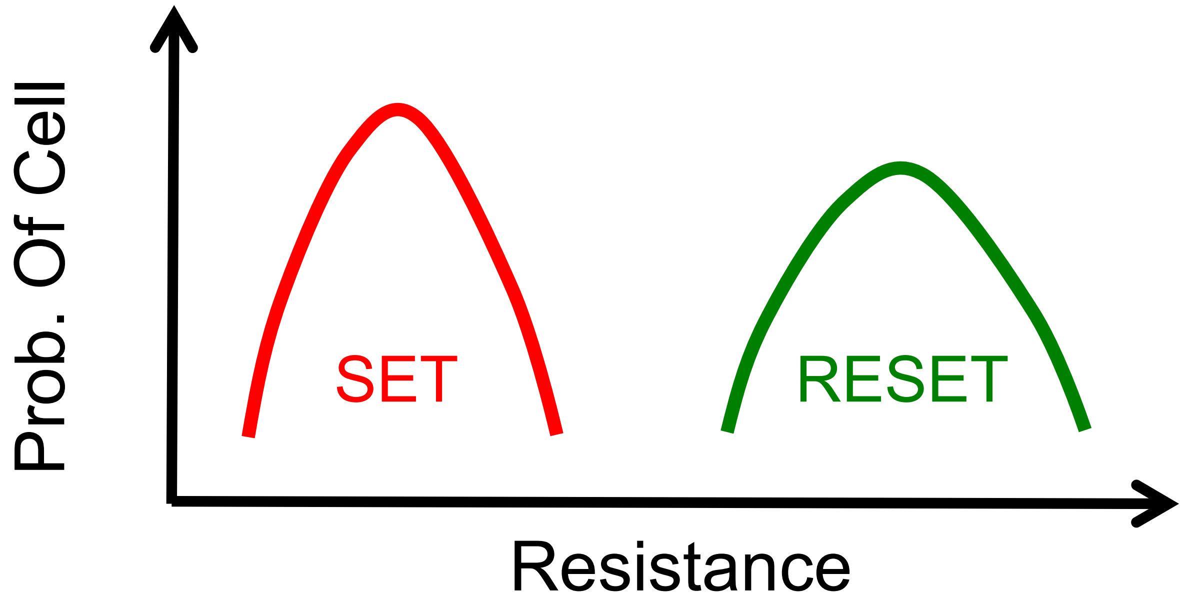
Goal ➔ Reduce the high read latency of PCM

OUTLINE

- Background ⇦
- Early Read
- Turbo Read
- Early+Turbo Read
- Results
- Summary

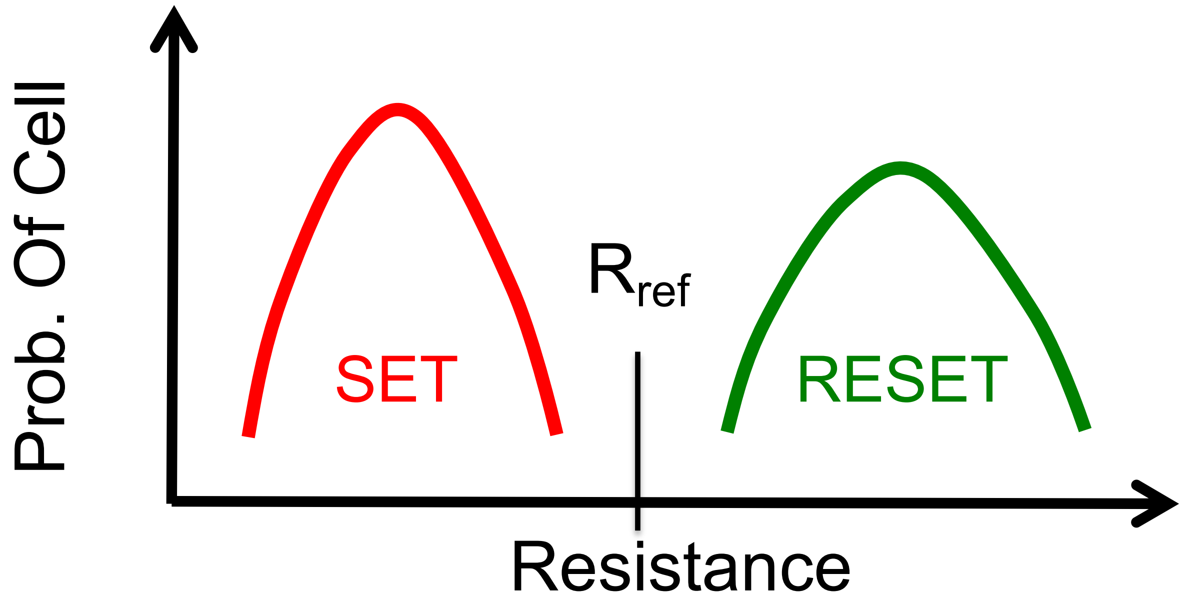
STORING DATA IN PCM CELLS

- Low (SET) and High (RESET) resistance states



STORING DATA IN PCM CELLS

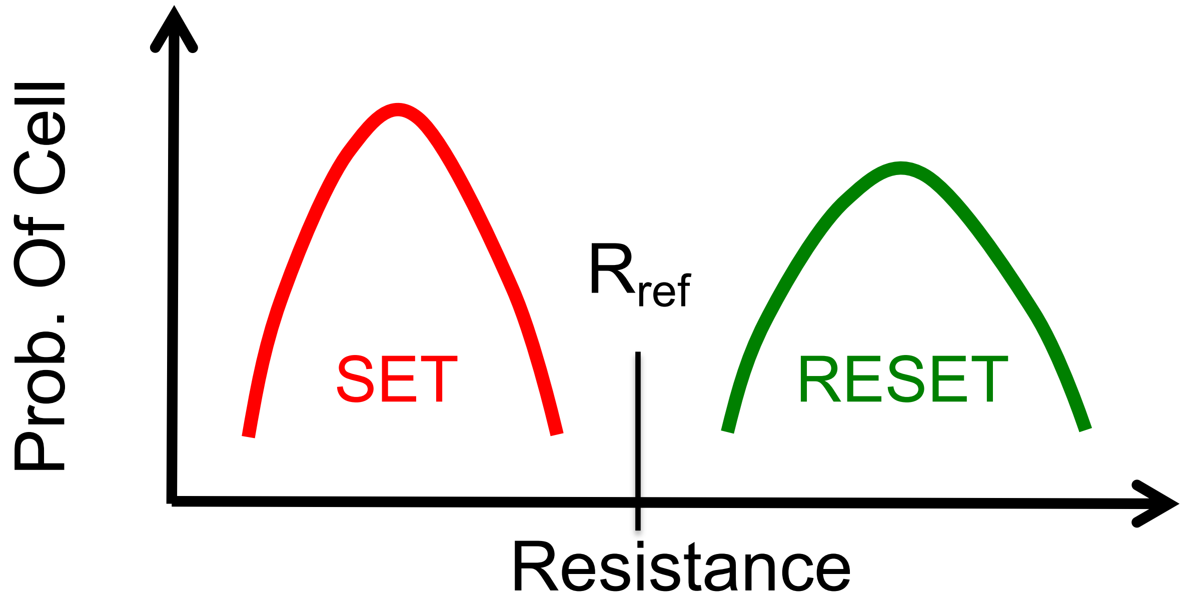
- Low (SET) and High (RESET) resistance states



- Cell states are compared to reference resistance

STORING DATA IN PCM CELLS

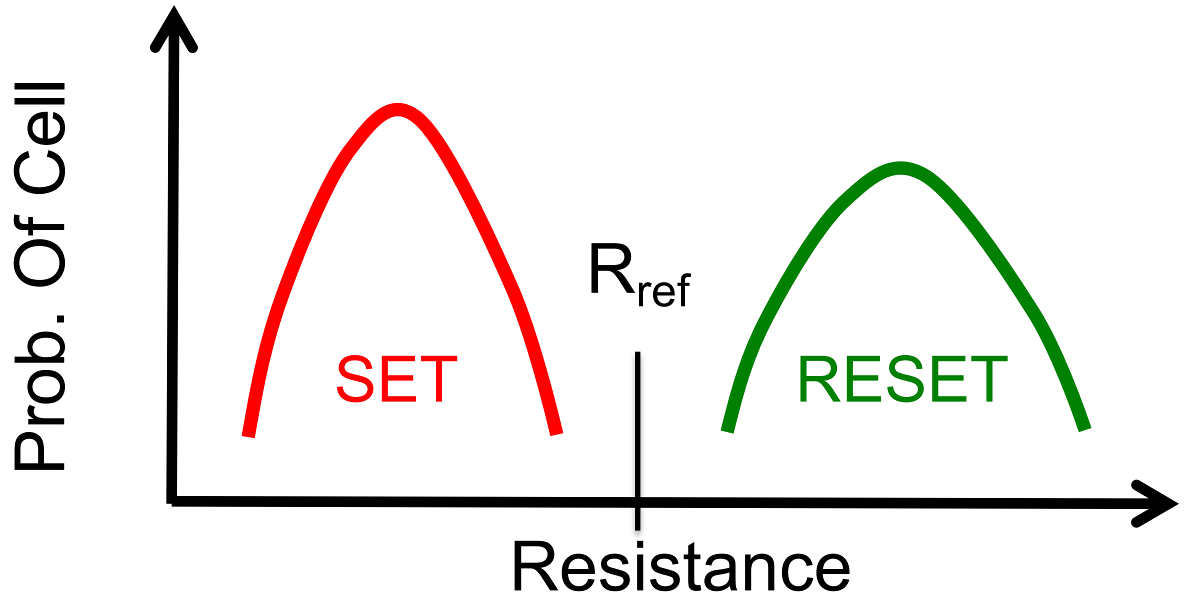
- Low (SET) and High (RESET) resistance states



- Cell states are compared to reference resistance
- The states correspond to binary values of 0 and 1

STORING DATA IN PCM CELLS

- Low (SET) and High (RESET) resistance states



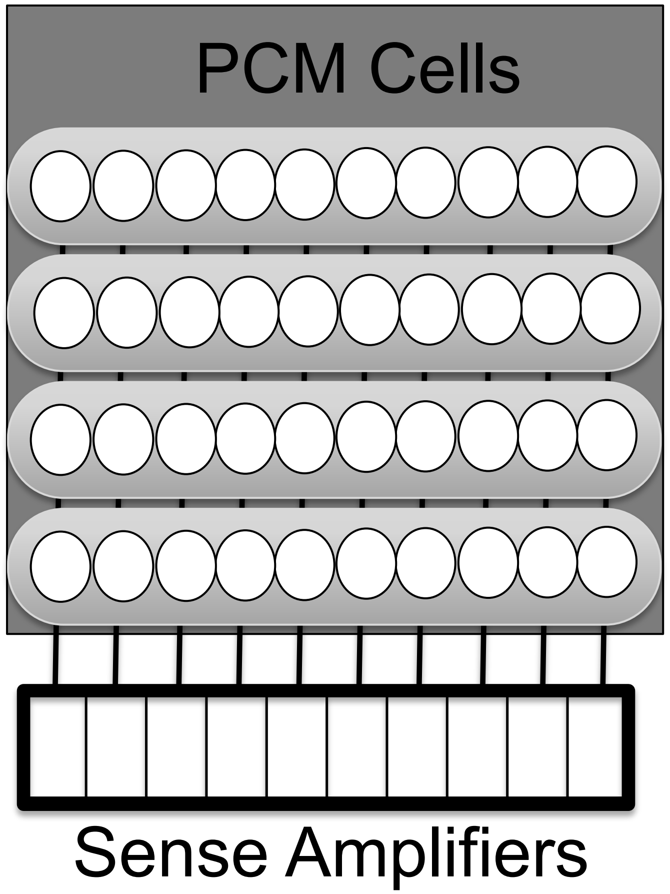
- Cell states are compared to reference resistance
- The states correspond to binary values of 0 and 1

PCM stores binary values by varying resistance of cells

READ PROCESS IN PCM

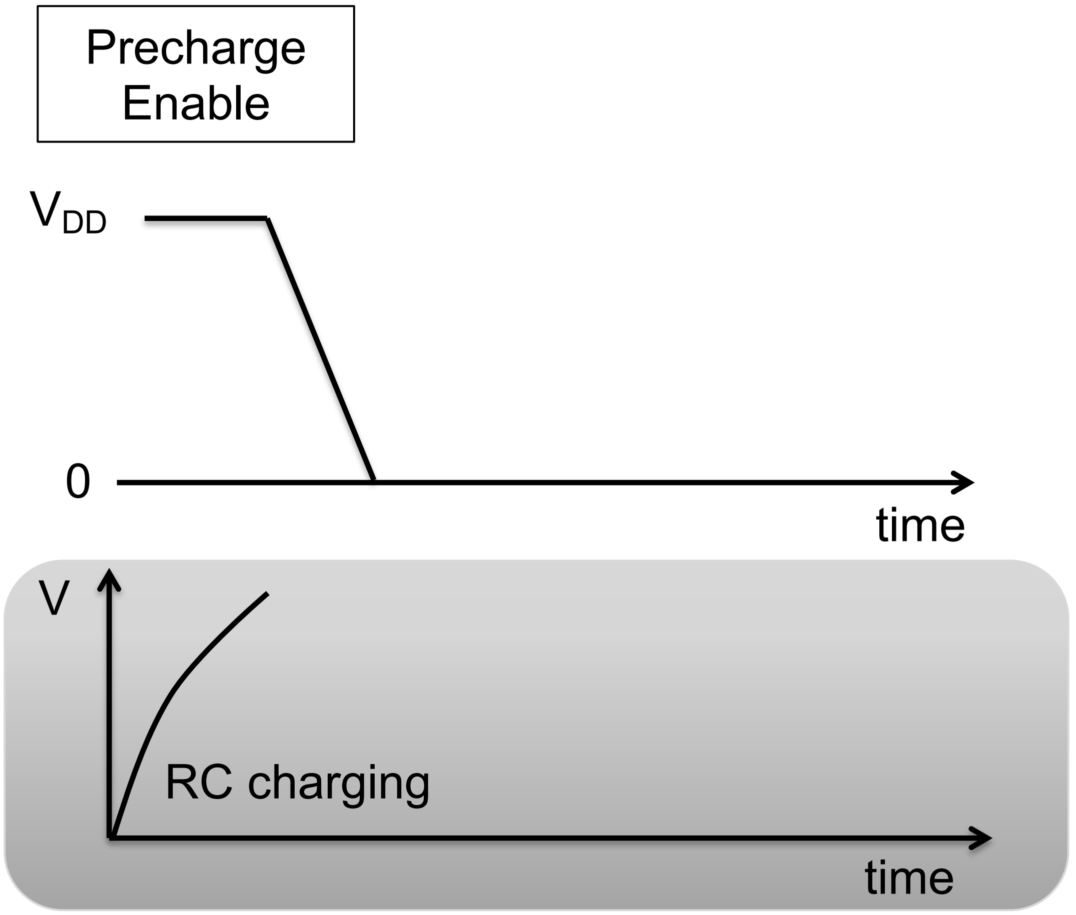
Three step process to read a PCM cell

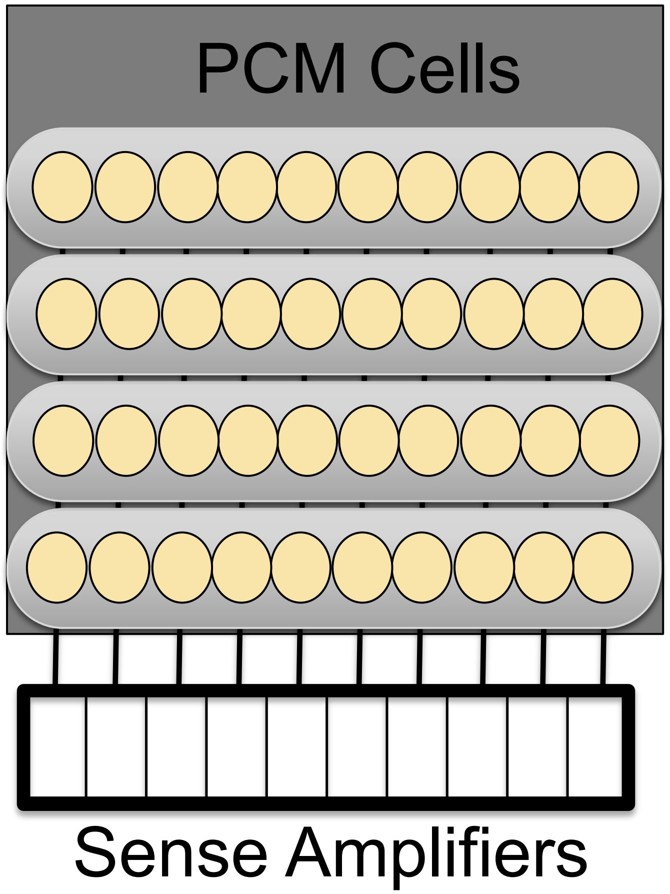
V_{DD}
0
time
V
time



READ PROCESS IN PCM

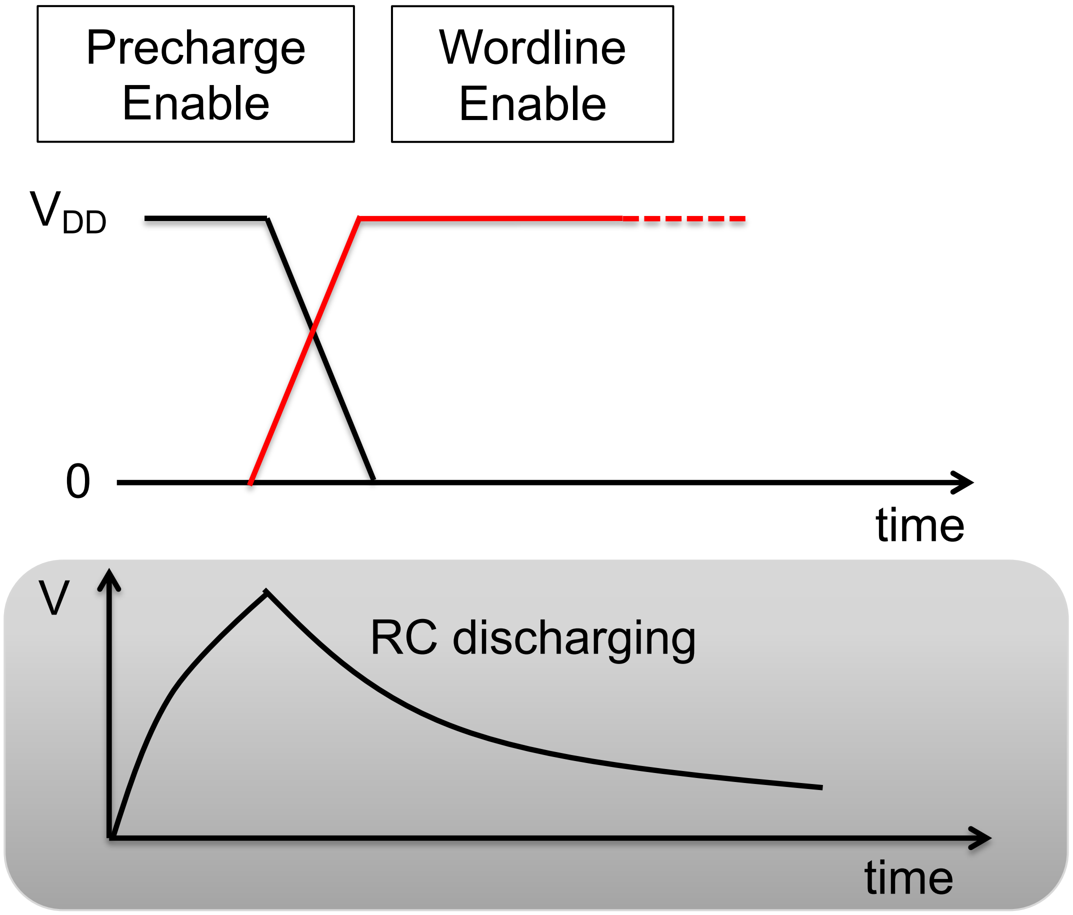
Three step process to read a PCM cell

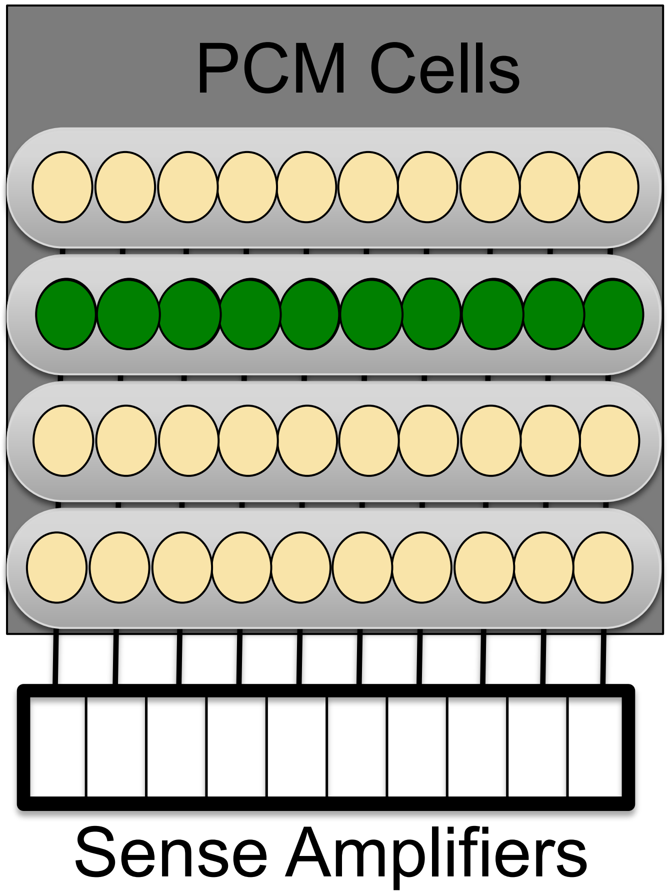




READ PROCESS IN PCM

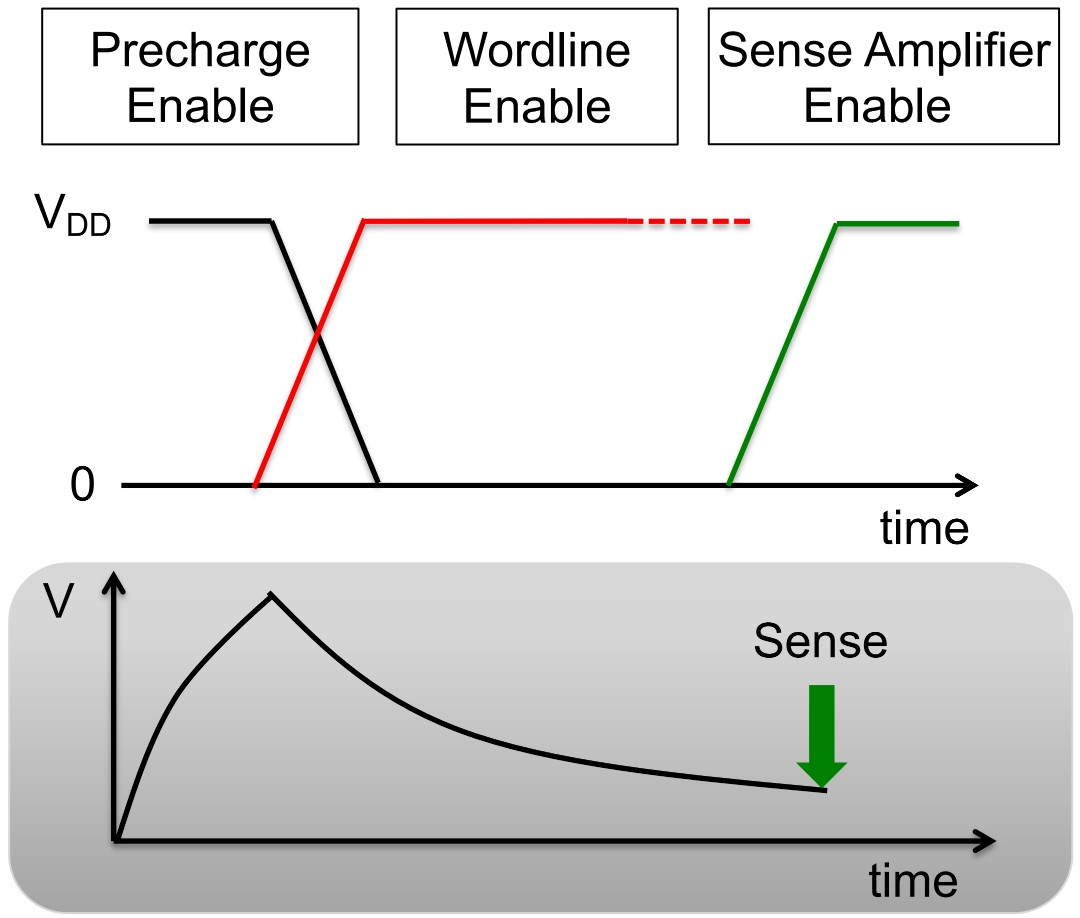
Three step process to read a PCM cell

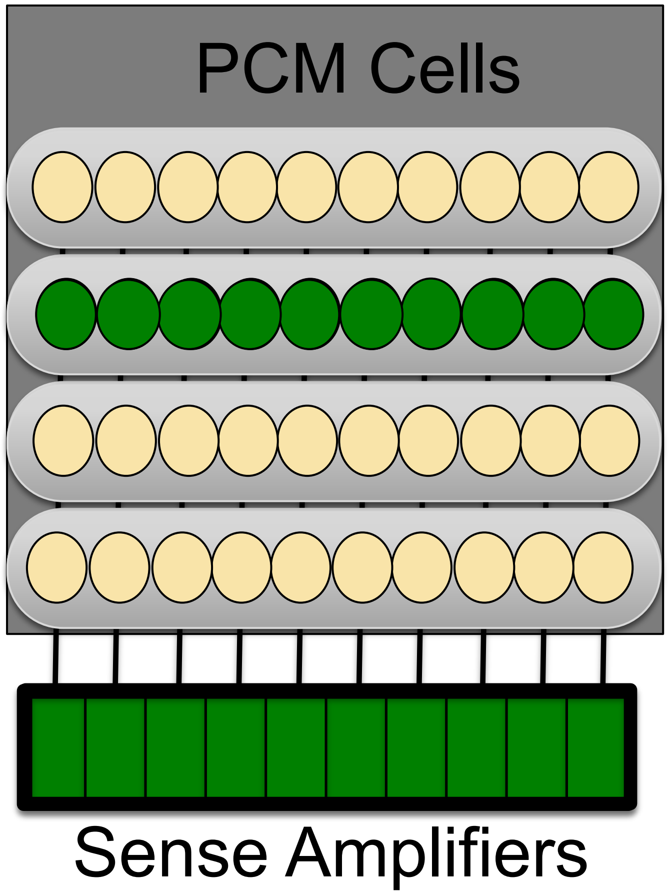




READ PROCESS IN PCM

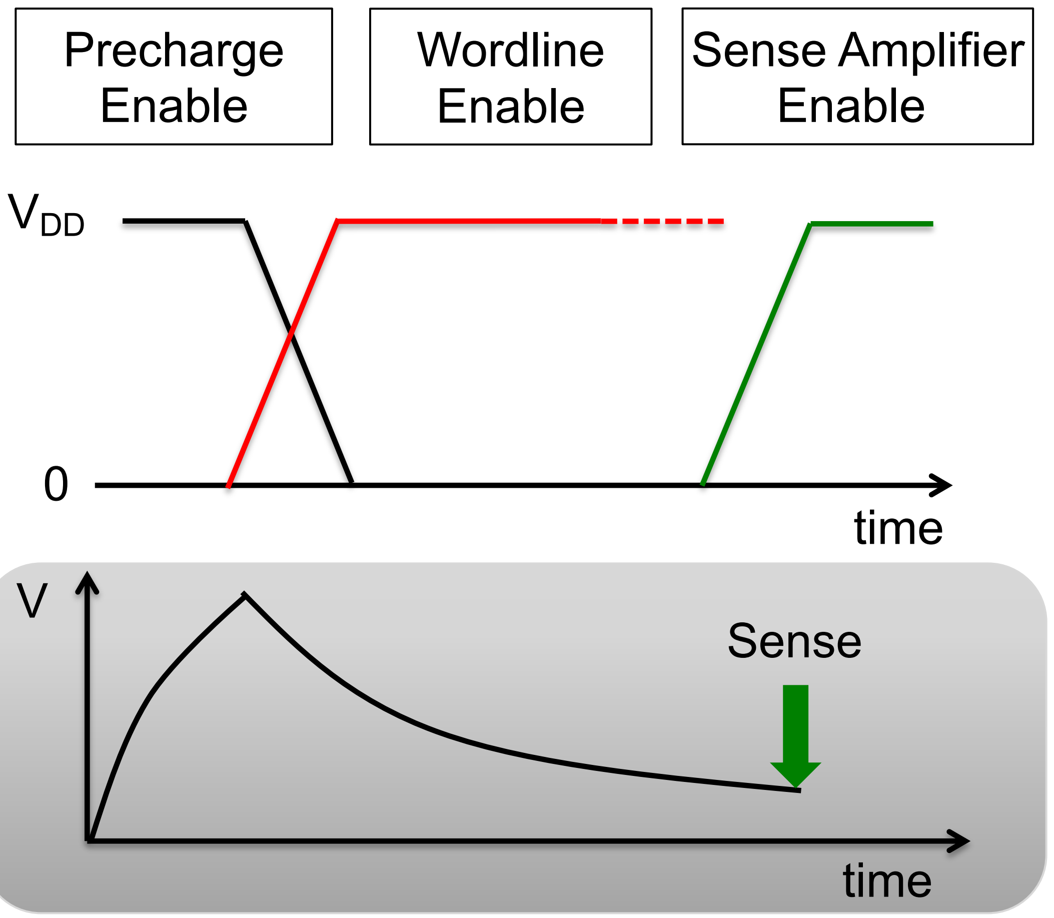
Three step process to read a PCM cell

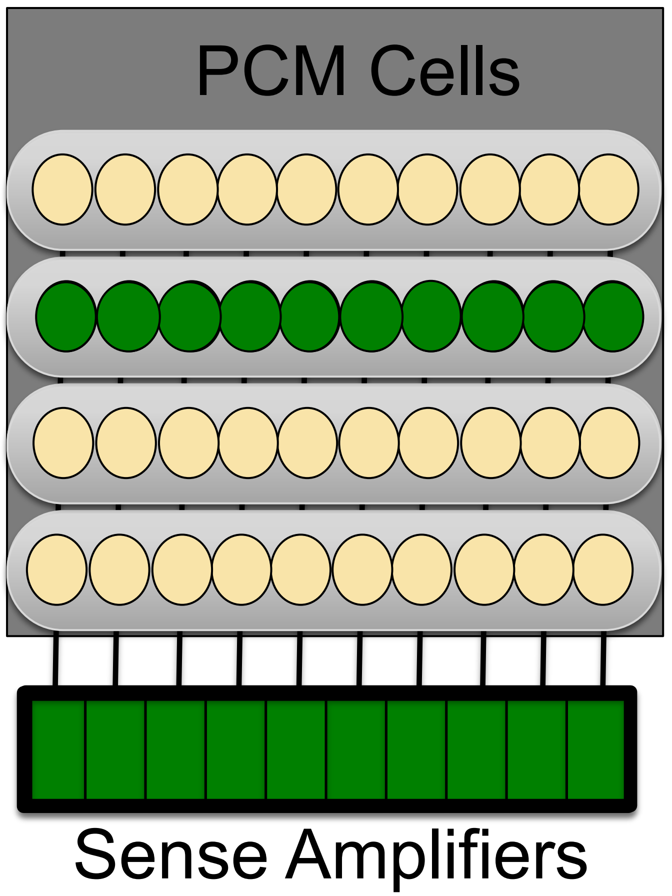




READ PROCESS IN PCM

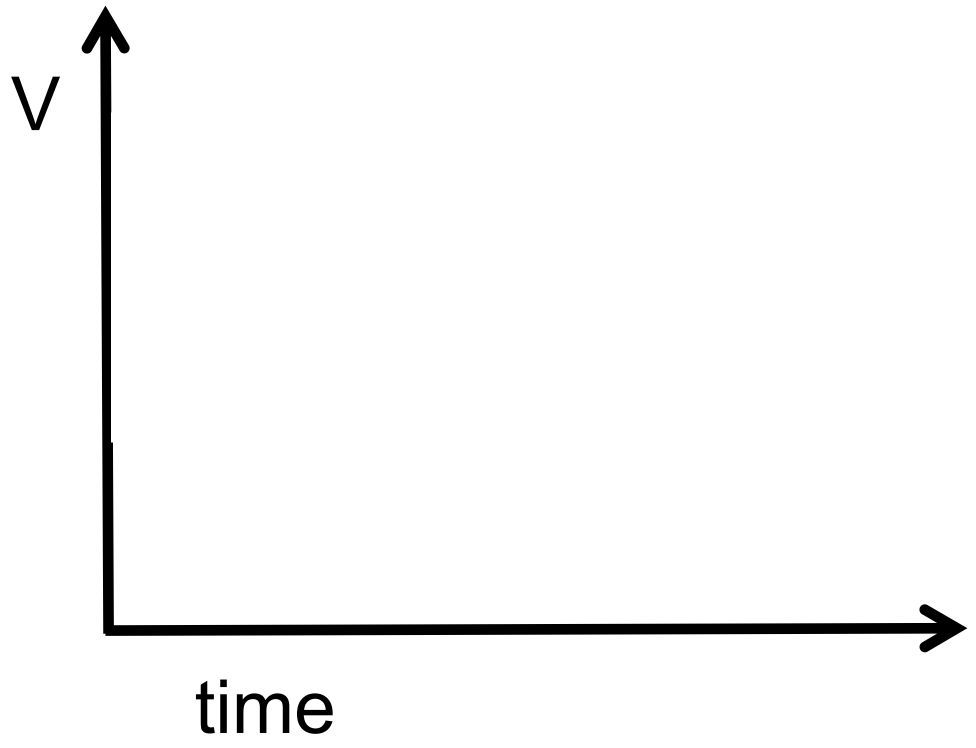
Three step process to read a PCM cell

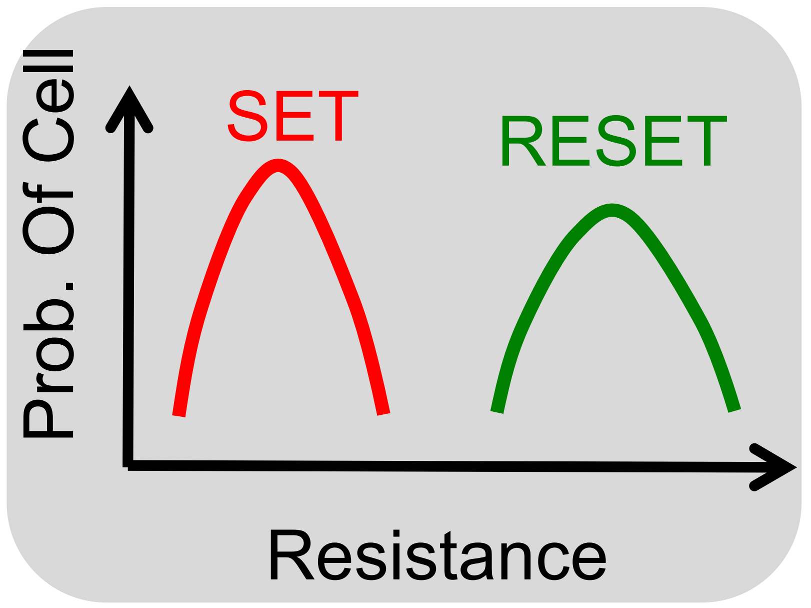




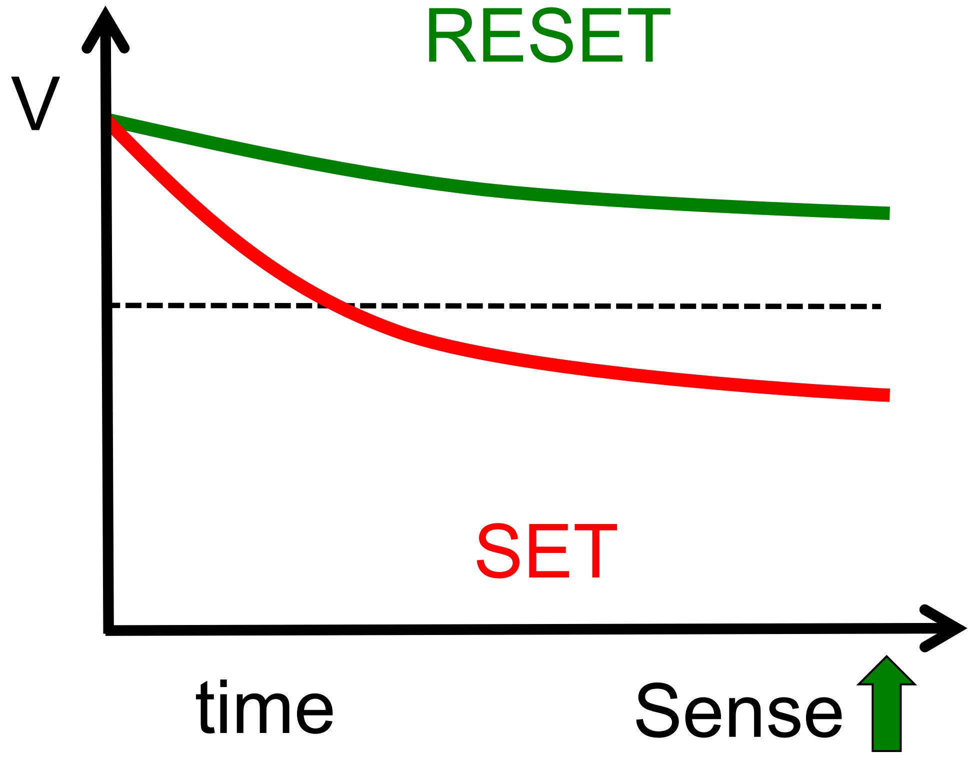
The discharging time determines the sensing time

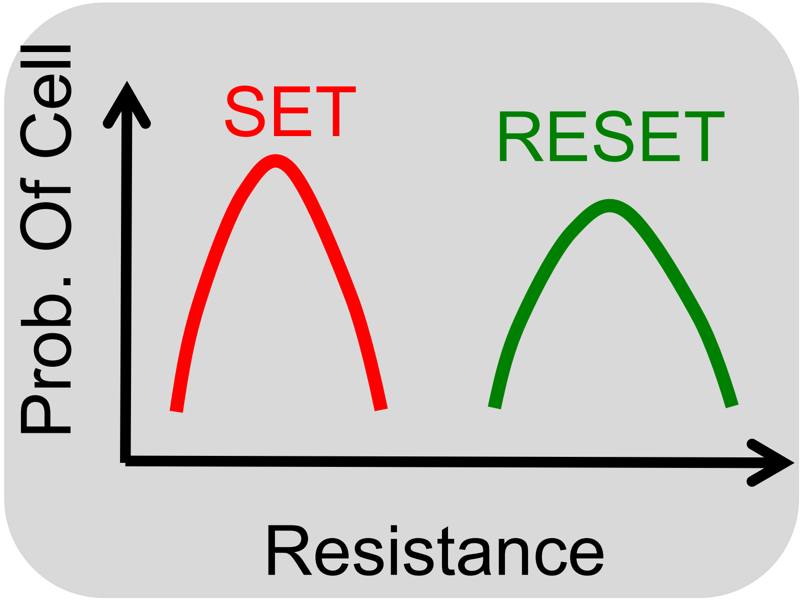
SENSING DATA FOR READ





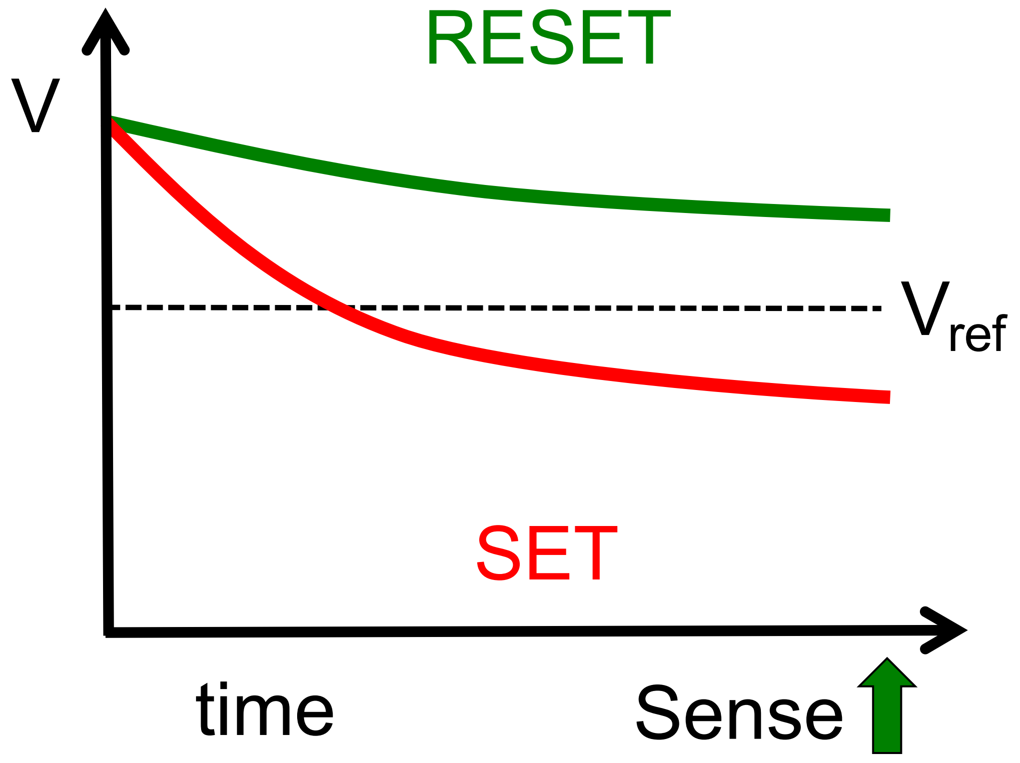
SENSING DATA FOR READ

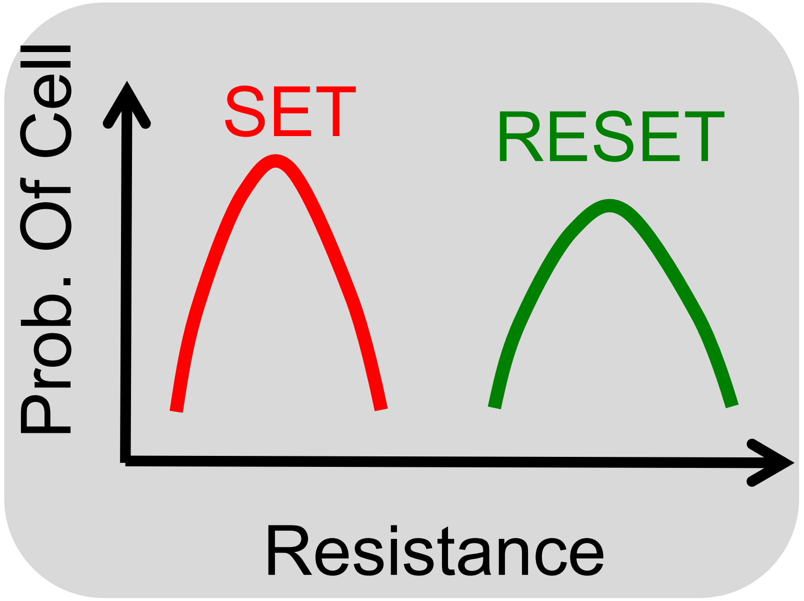




- Capacitive Discharge and compare against V_{ref}

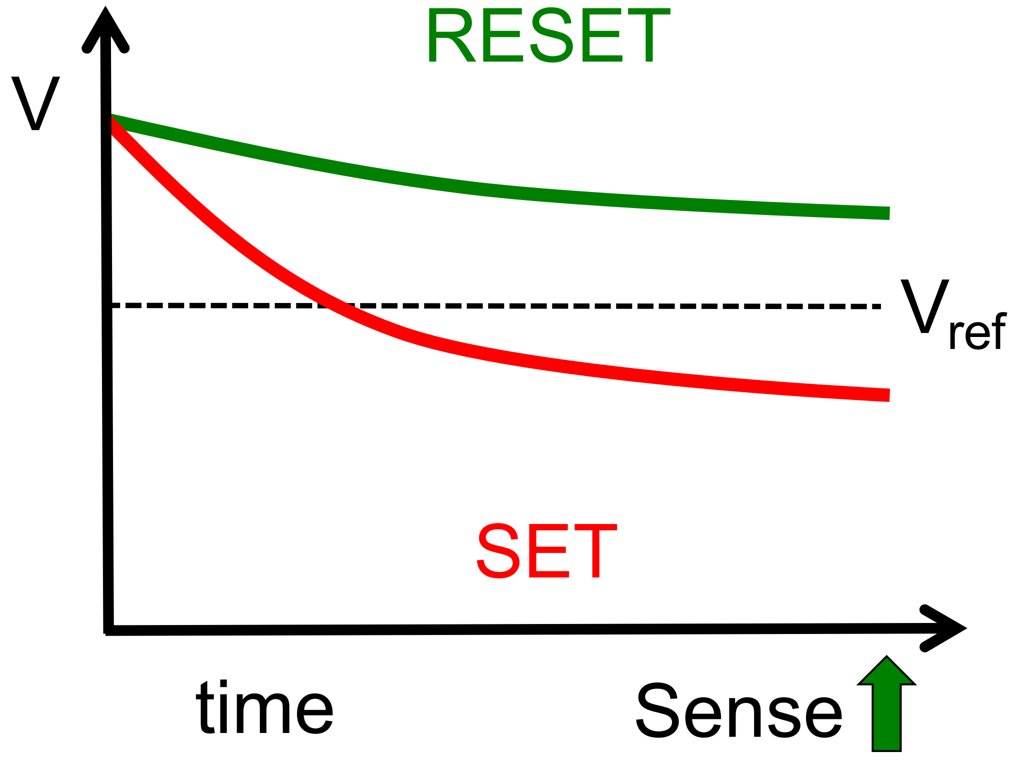
SENSING DATA FOR READ

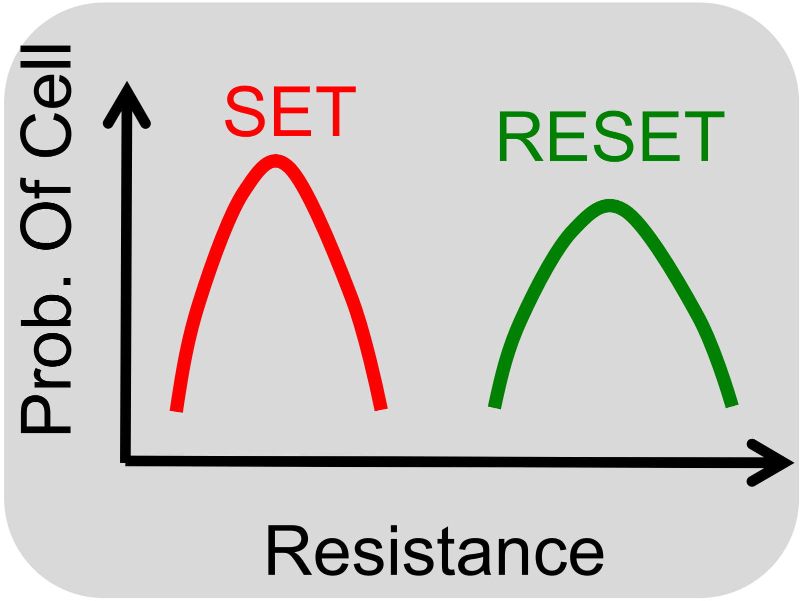




- Capacitive Discharge and compare against V_{ref}

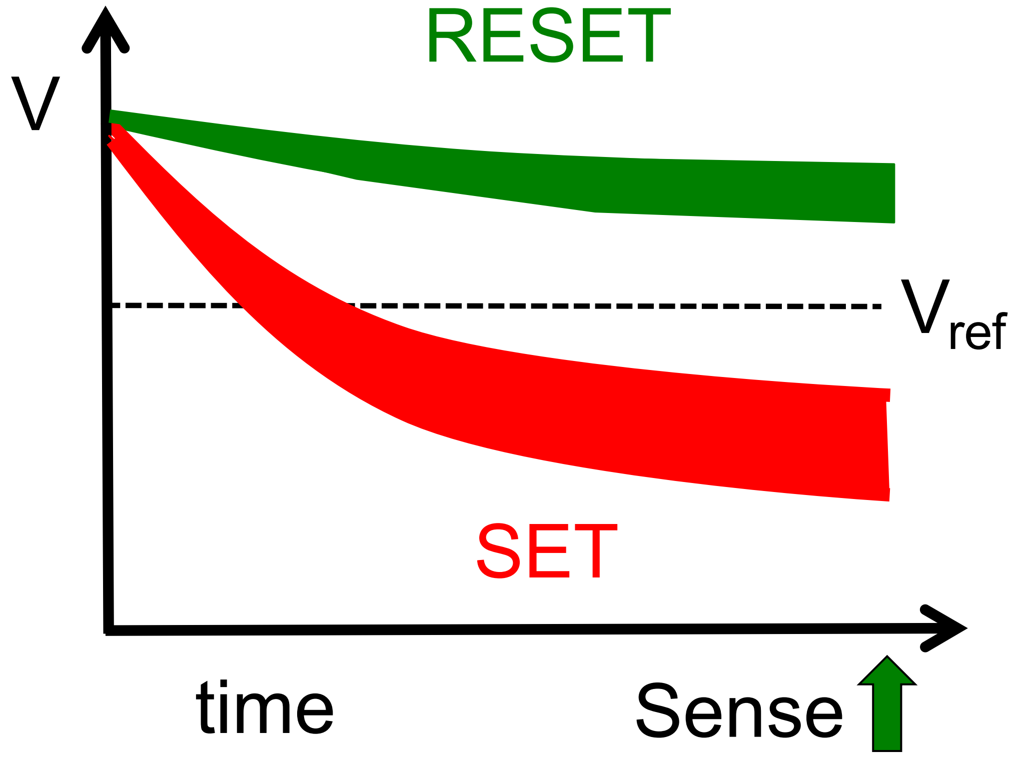
SENSING DATA FOR READ

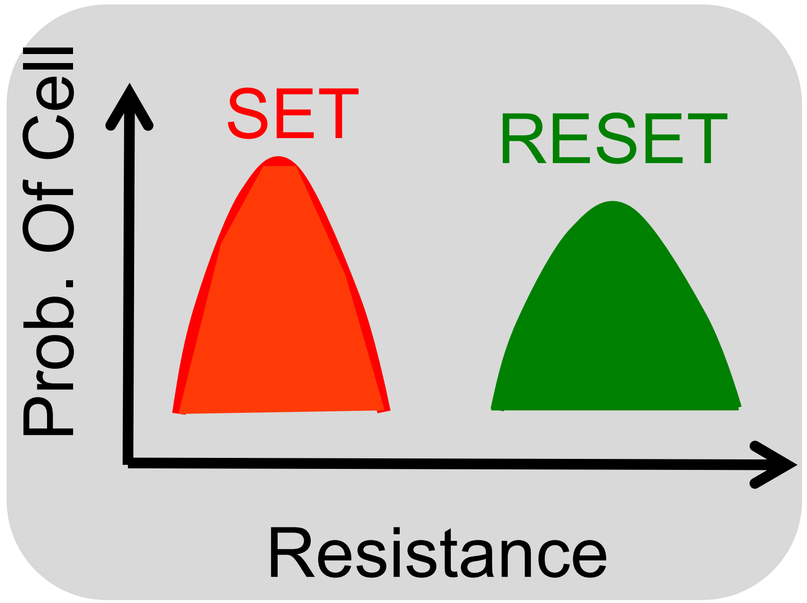




- Capacitive Discharge and compare against V_{ref}
- Variation in SET and RESET distributions

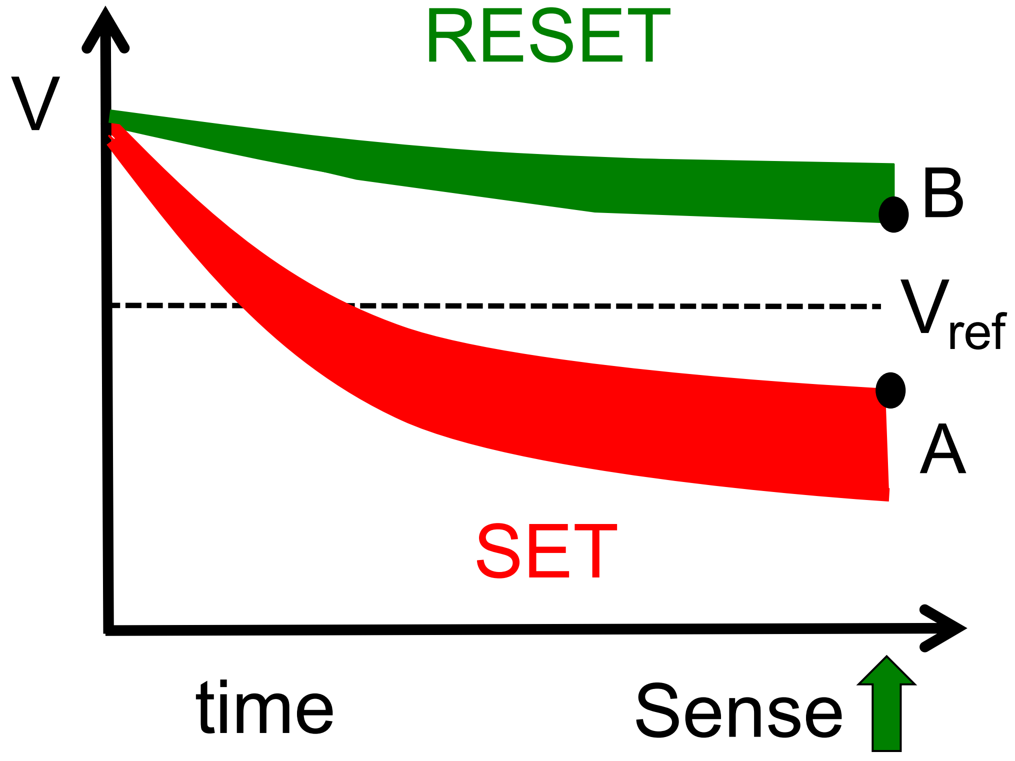
SENSING DATA FOR READ

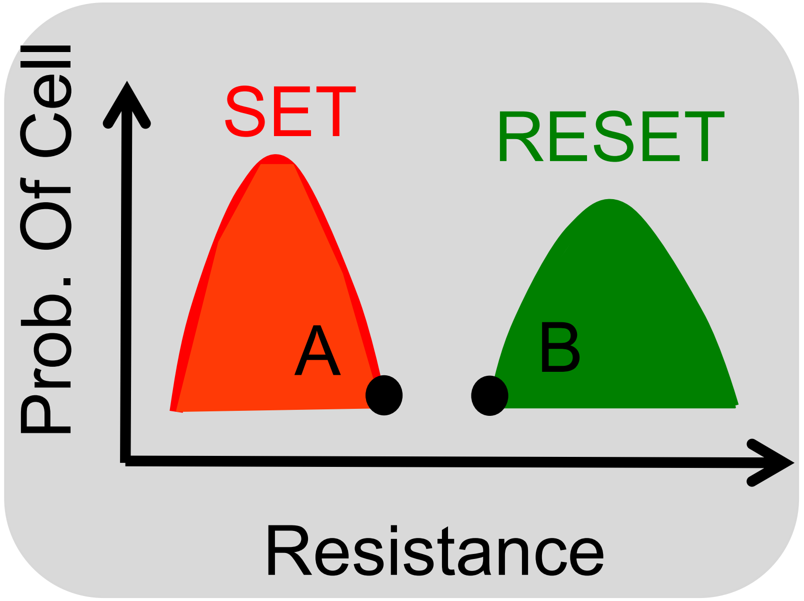




- Capacitive Discharge and compare against V_{ref}
- Variation in SET and RESET distributions

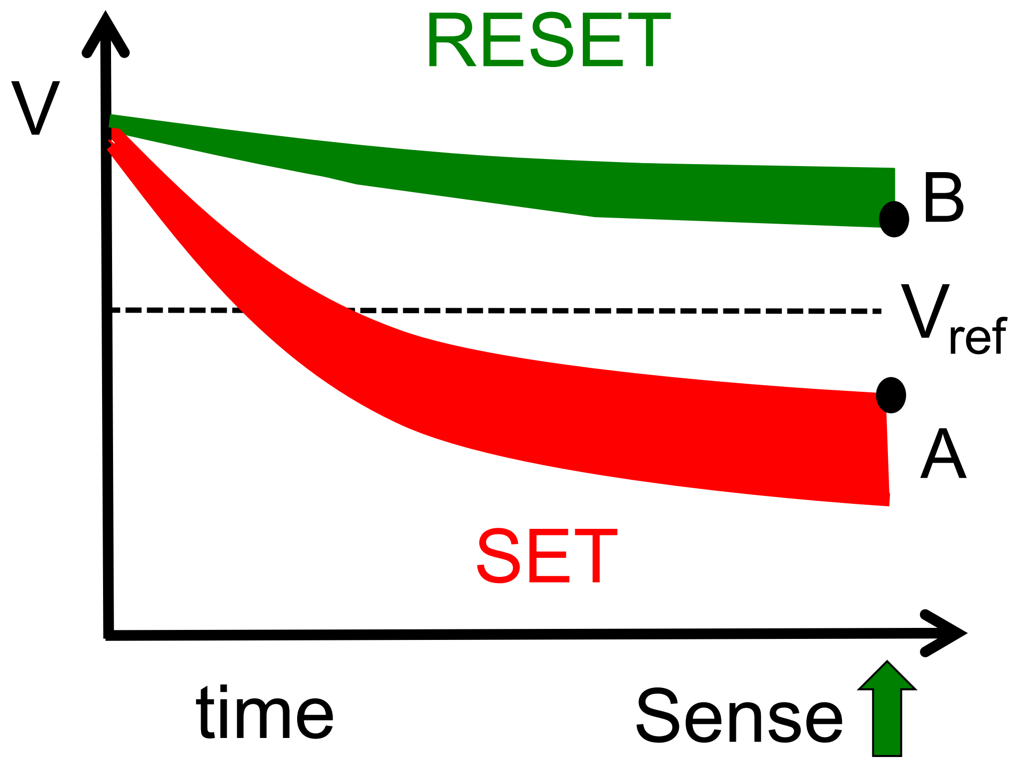
SENSING DATA FOR READ

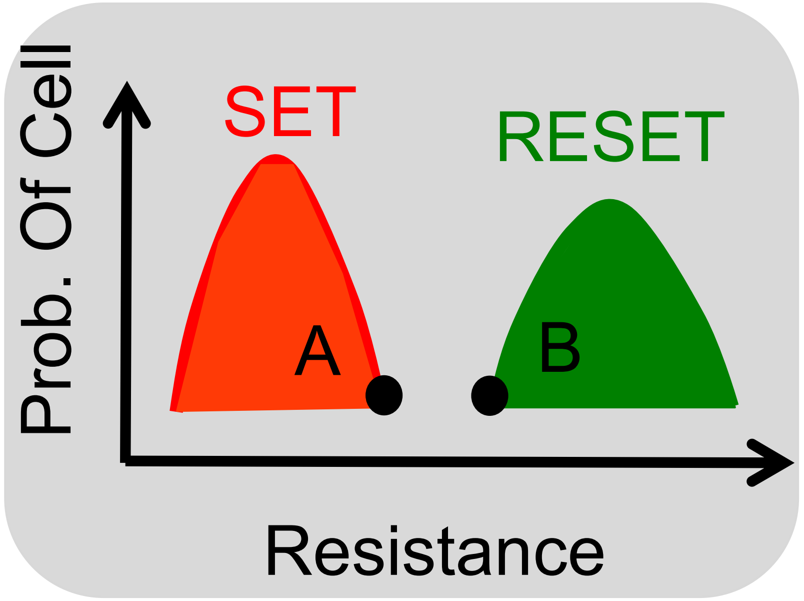




- Capacitive Discharge and compare against V_{ref}
- Variation in SET and RESET distributions

SENSING DATA FOR READ



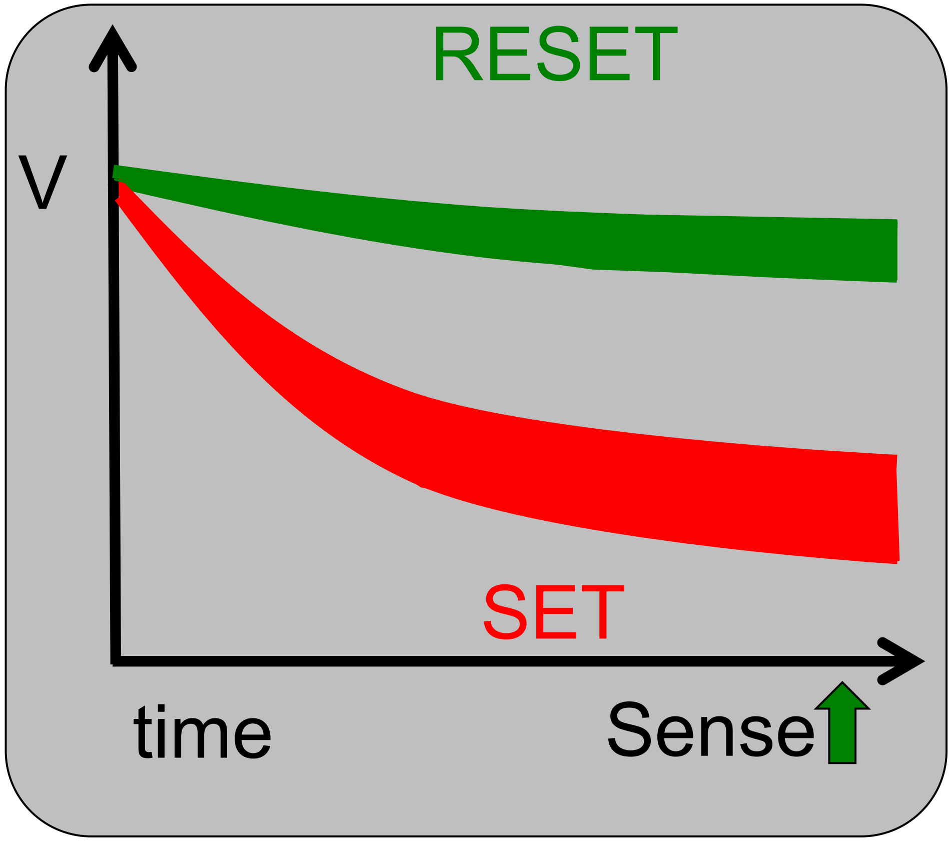


- Capacitive Discharge and compare against V_{ref}
- Variation in SET and RESET distributions

Sensing time is determined by worst case cells

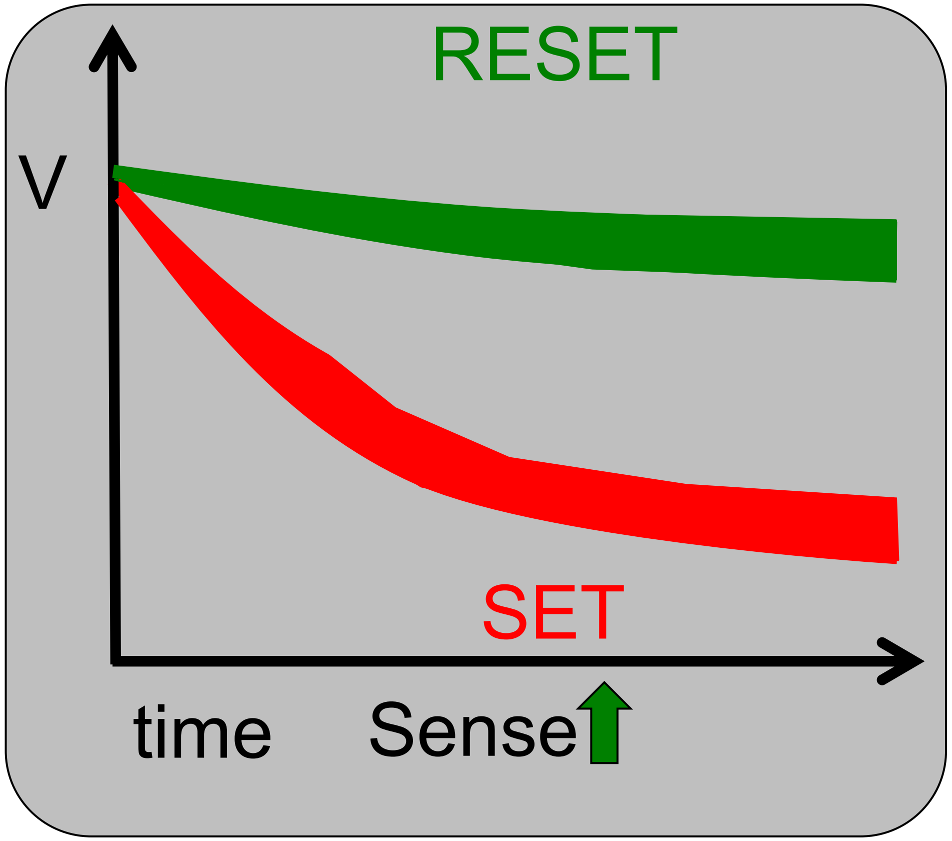
REDUCE READ LATENCY : SENSE EARLIER

- Sense data earlier than the provisioned time



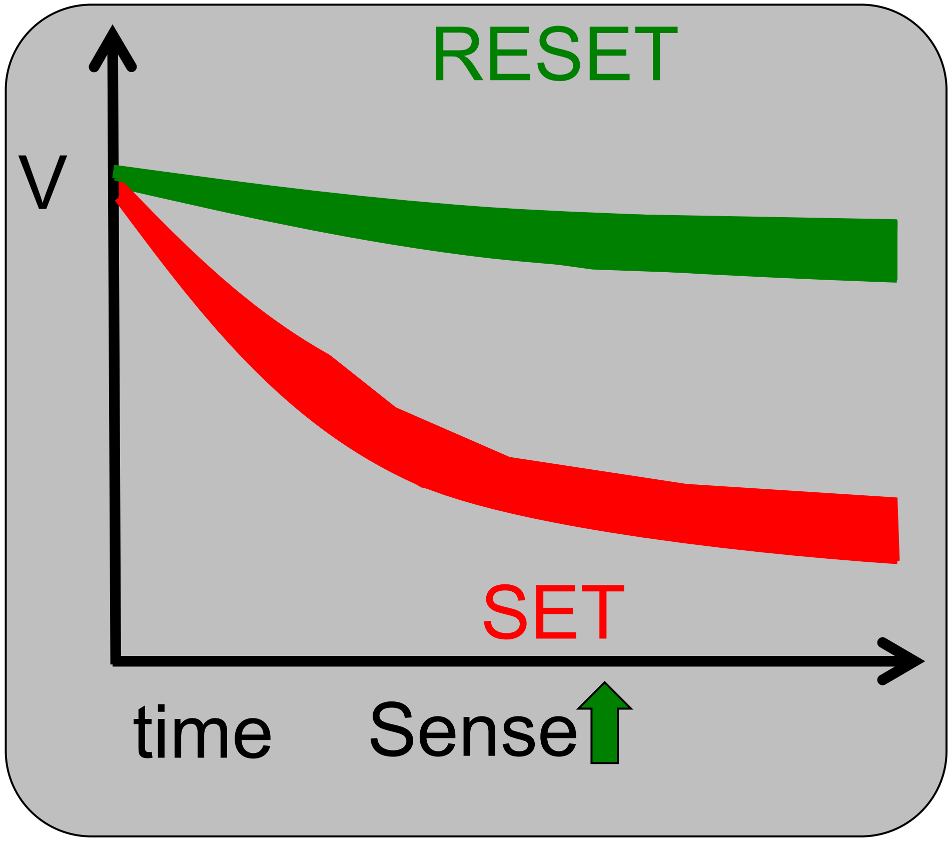
REDUCE READ LATENCY : SENSE EARLIER

- Sense data earlier than the provisioned time
- Lower Resistance ➔ Lower RC time to discharge



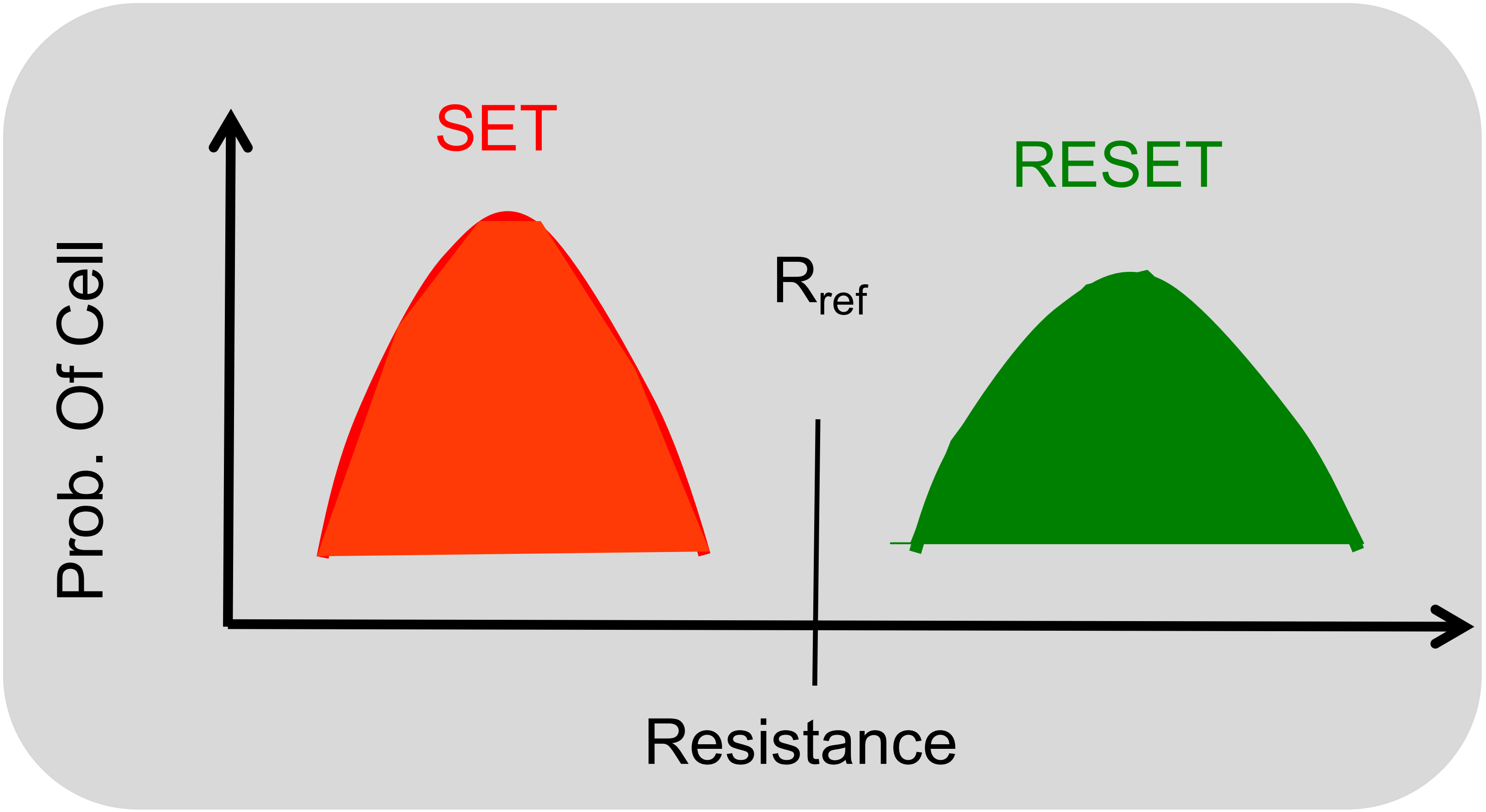
REDUCE READ LATENCY : SENSE EARLIER

- Sense data earlier than the provisioned time
- Lower Resistance ➔ Lower RC time to discharge

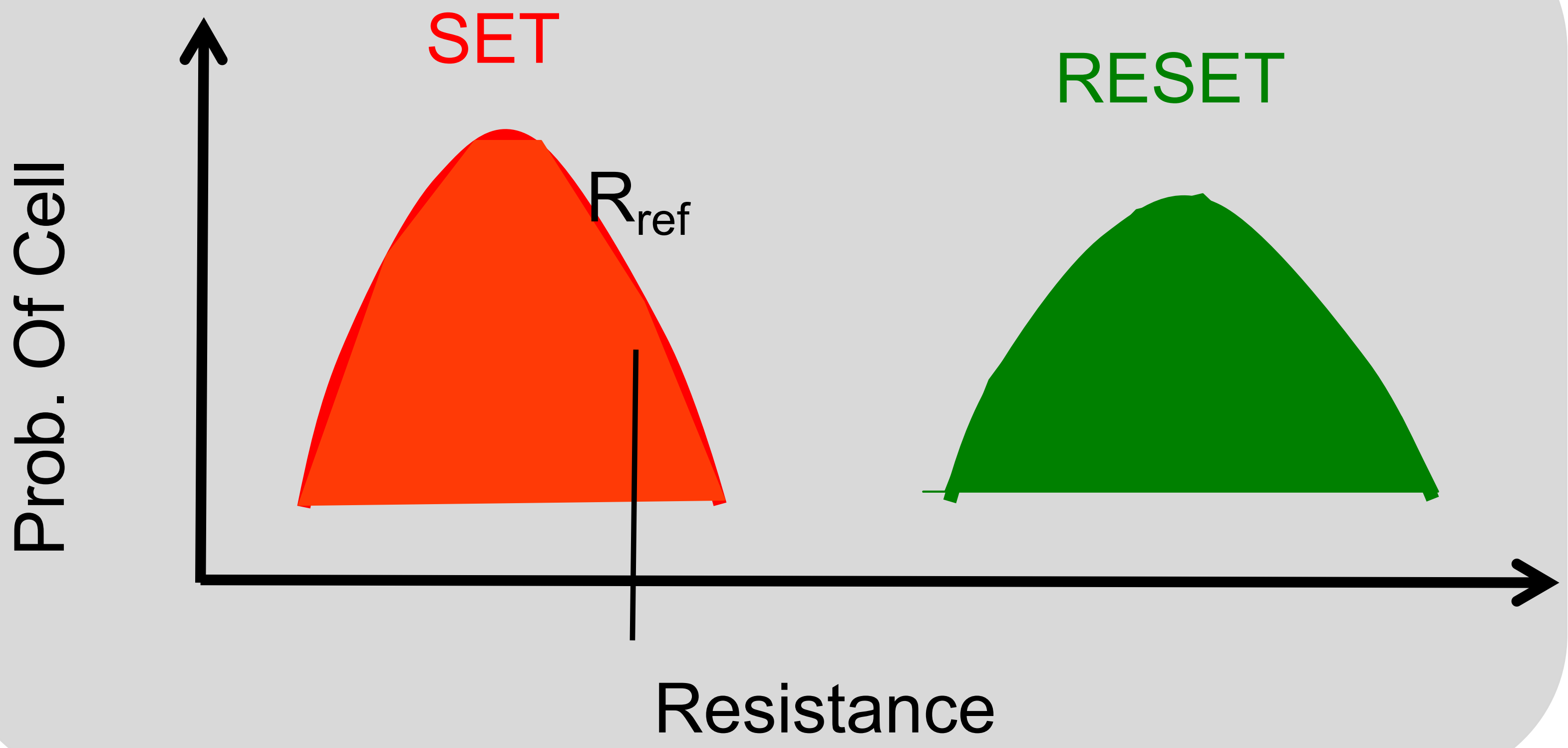


Reduce time to sense by lowering the RC time

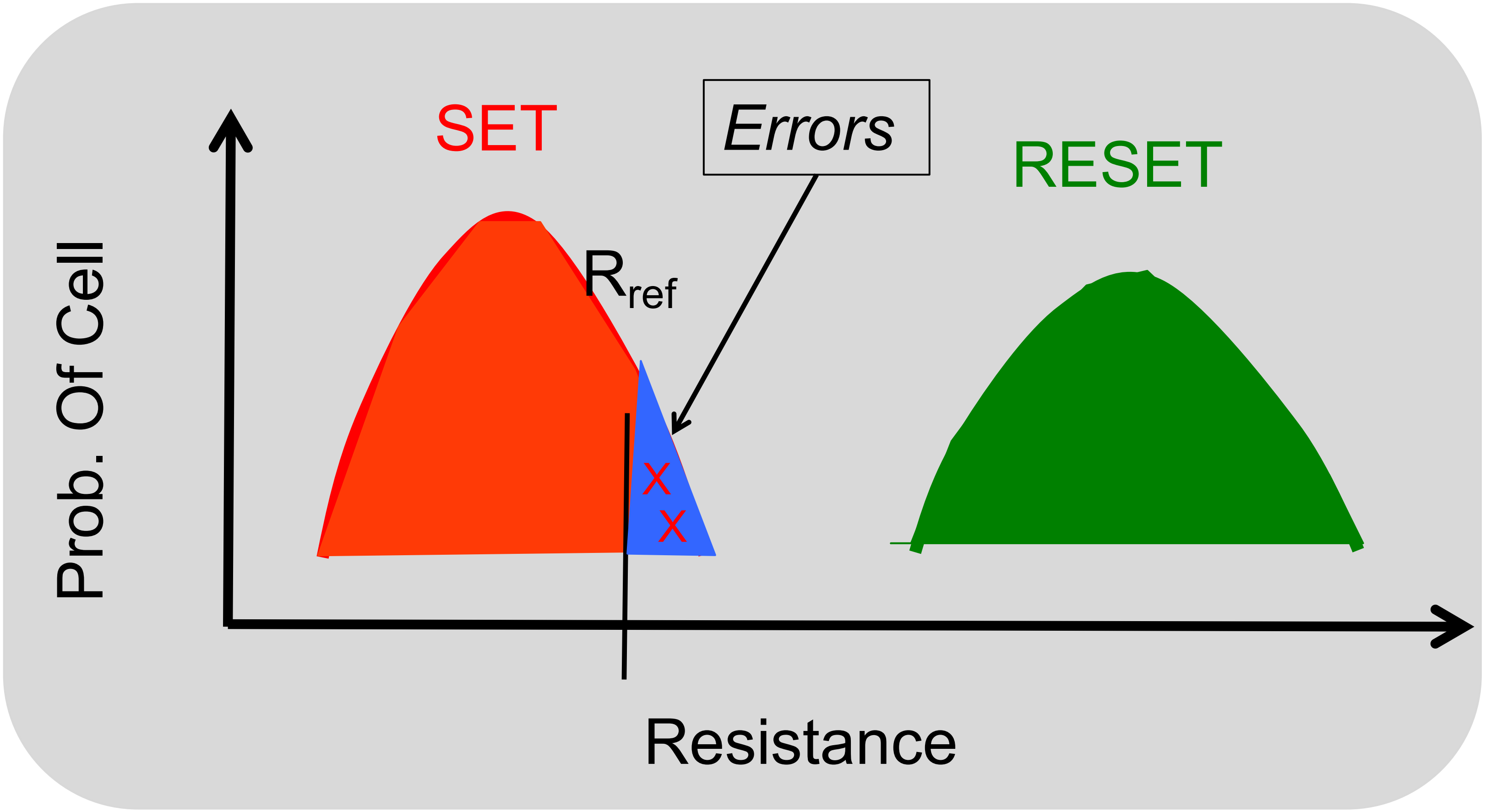
EFFECT OF SENSING EARLIER



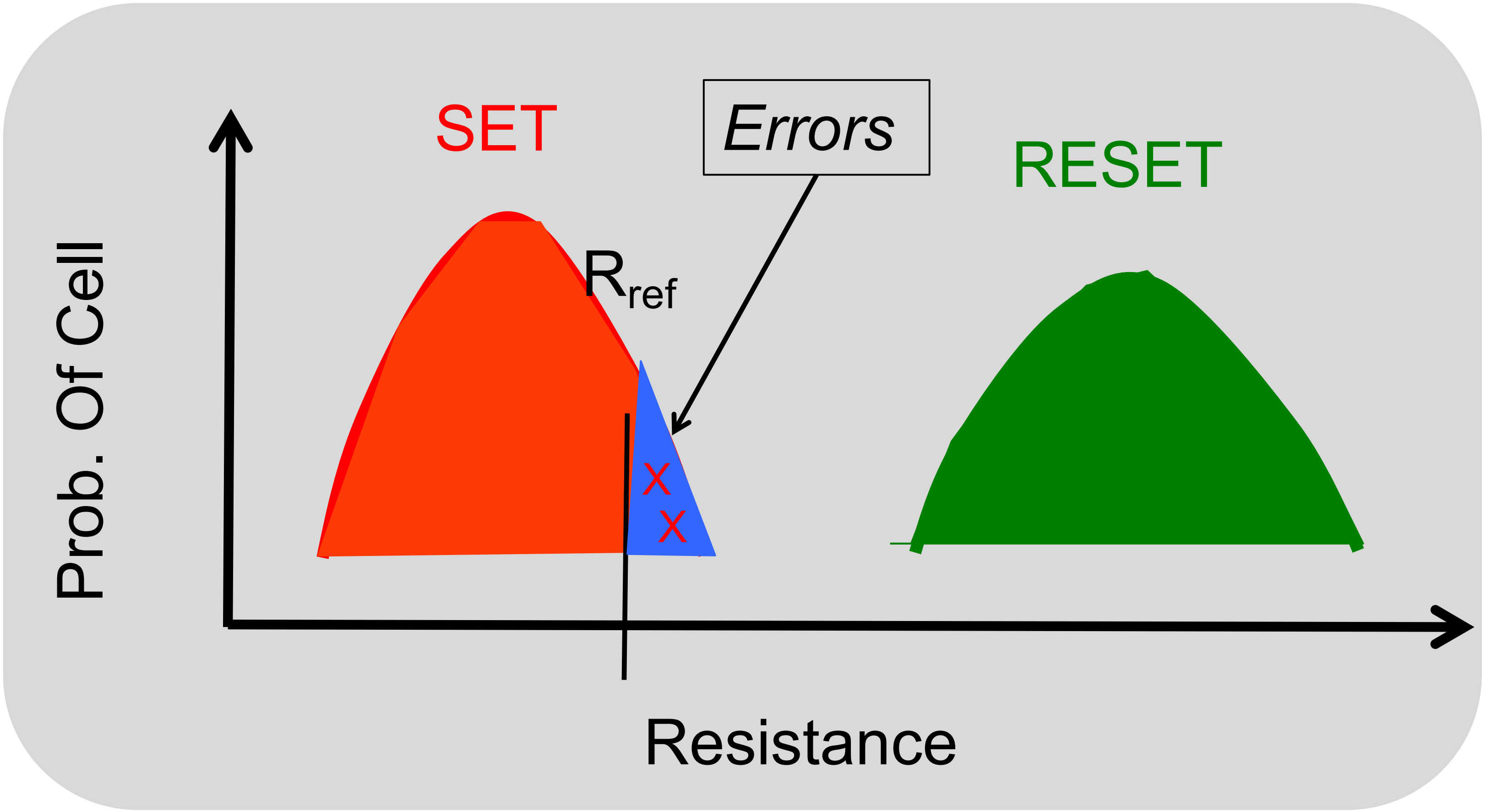
EFFECT OF SENSING EARLIER



EFFECT OF SENSING EARLIER



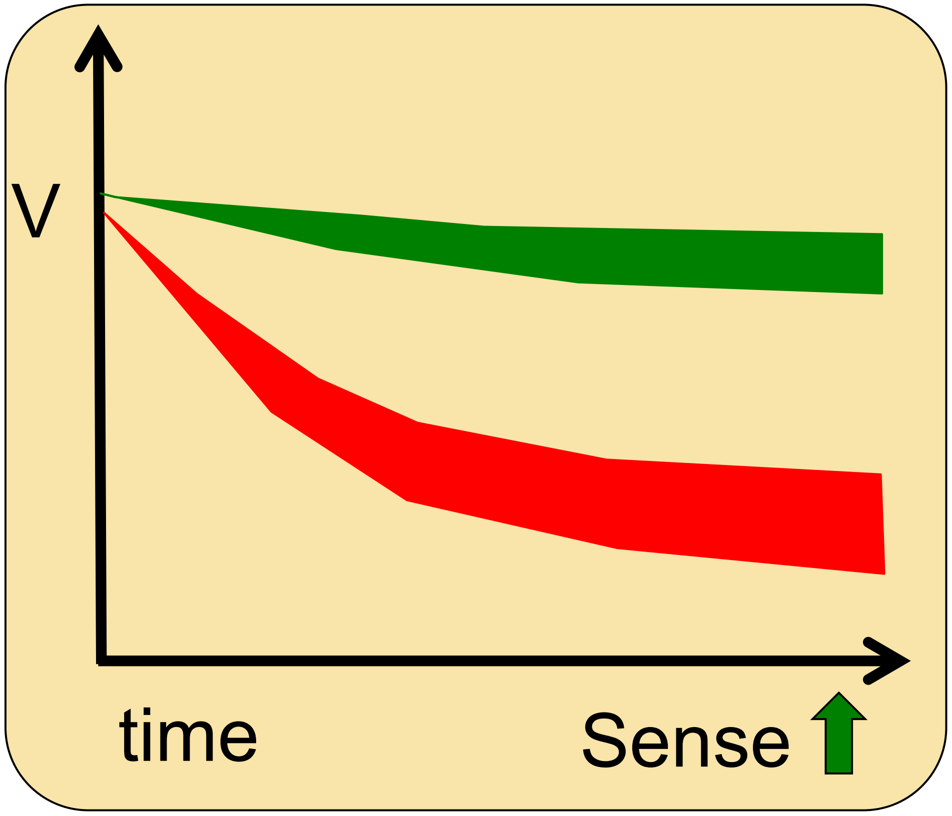
EFFECT OF SENSING EARLIER



Sensing earlier causes errors while reading higher resistances

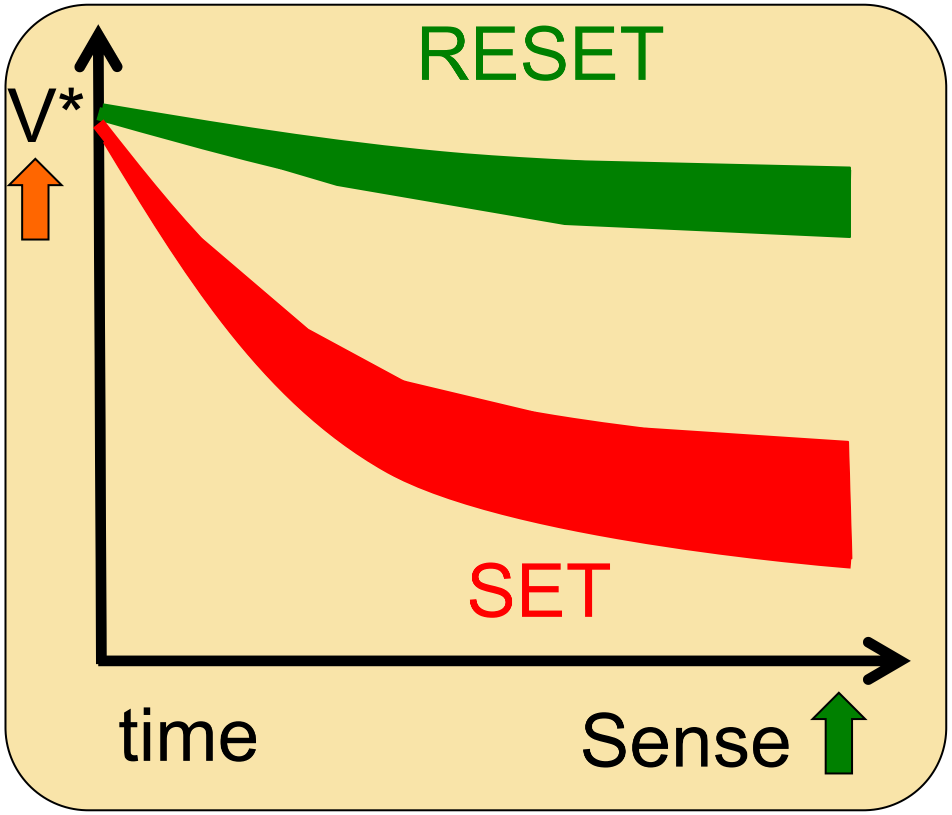
REDUCE READ LATENCY: HIGHER VOLTAGE

- Increase bitline voltage more than the provisioned value



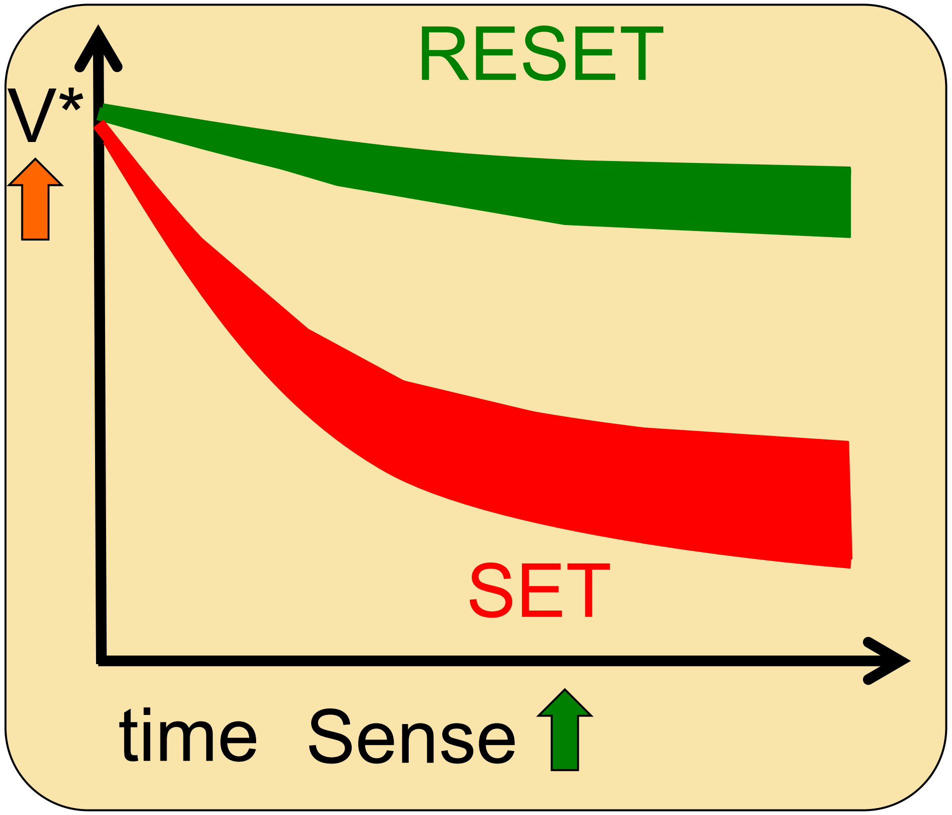
REDUCE READ LATENCY: HIGHER VOLTAGE

- Increase bitline voltage more than the provisioned value



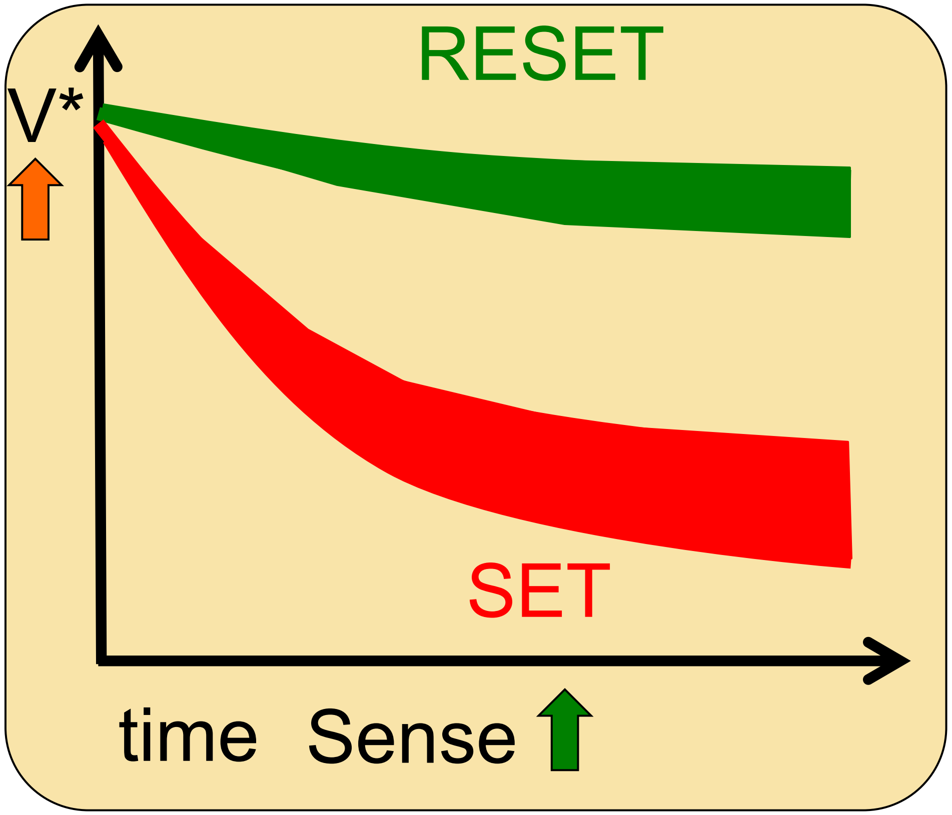
REDUCE READ LATENCY: HIGHER VOLTAGE

- Increase bitline voltage more than the provisioned value
- Higher Voltage ➔ Higher Current ➔ Low Read Latency



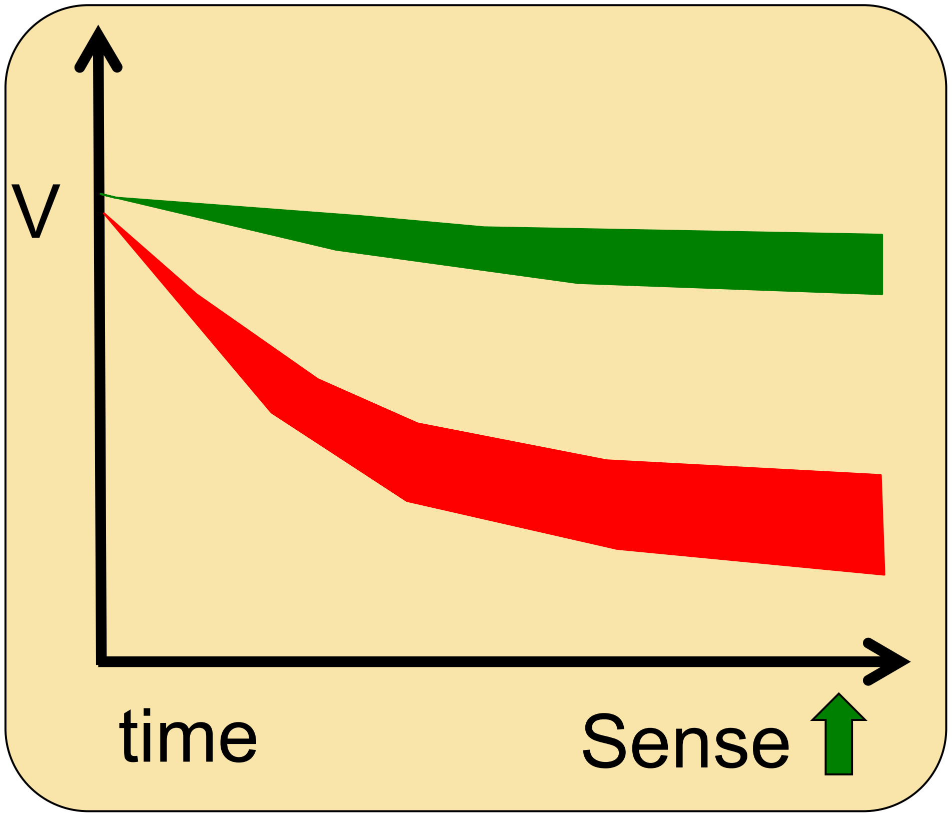
REDUCE READ LATENCY: HIGHER VOLTAGE

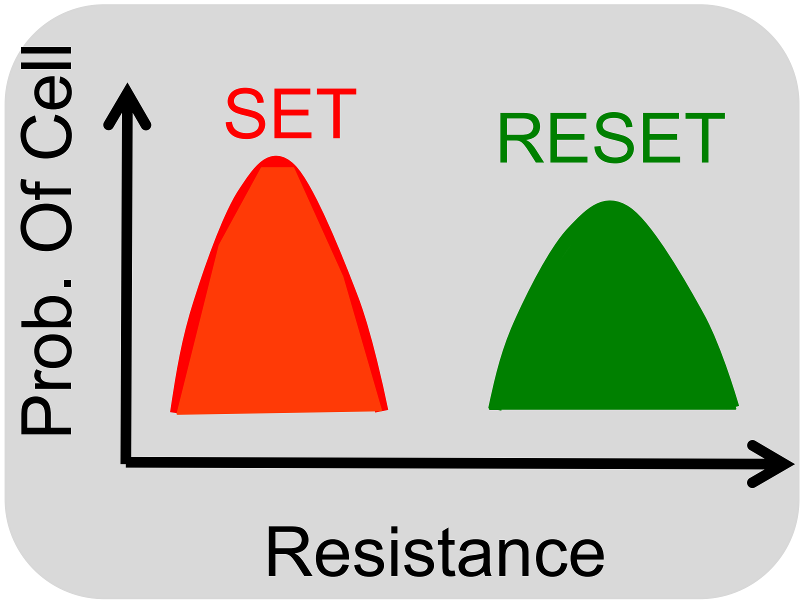
- Increase bitline voltage more than the provisioned value
- Higher Voltage ➔ Higher Current ➔ Low Read Latency



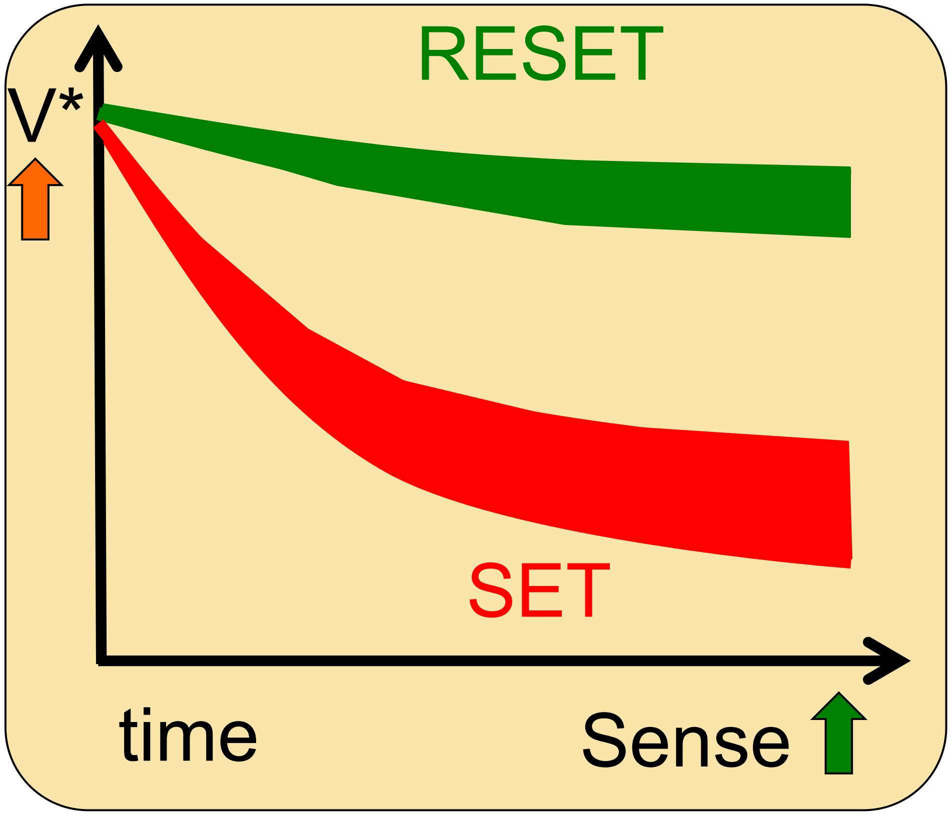
Increase bitline voltage and reduce sensing time

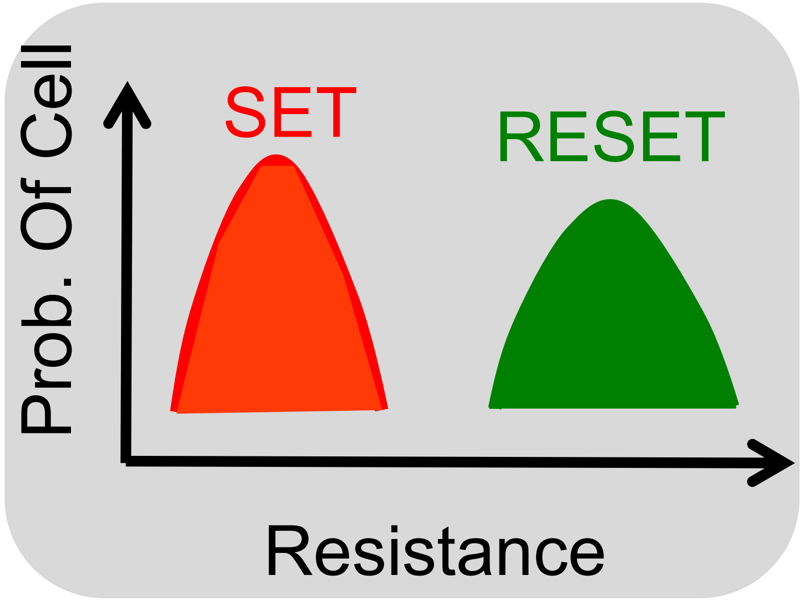
EFFECT OF HIGH BITLINE VOLTAGE



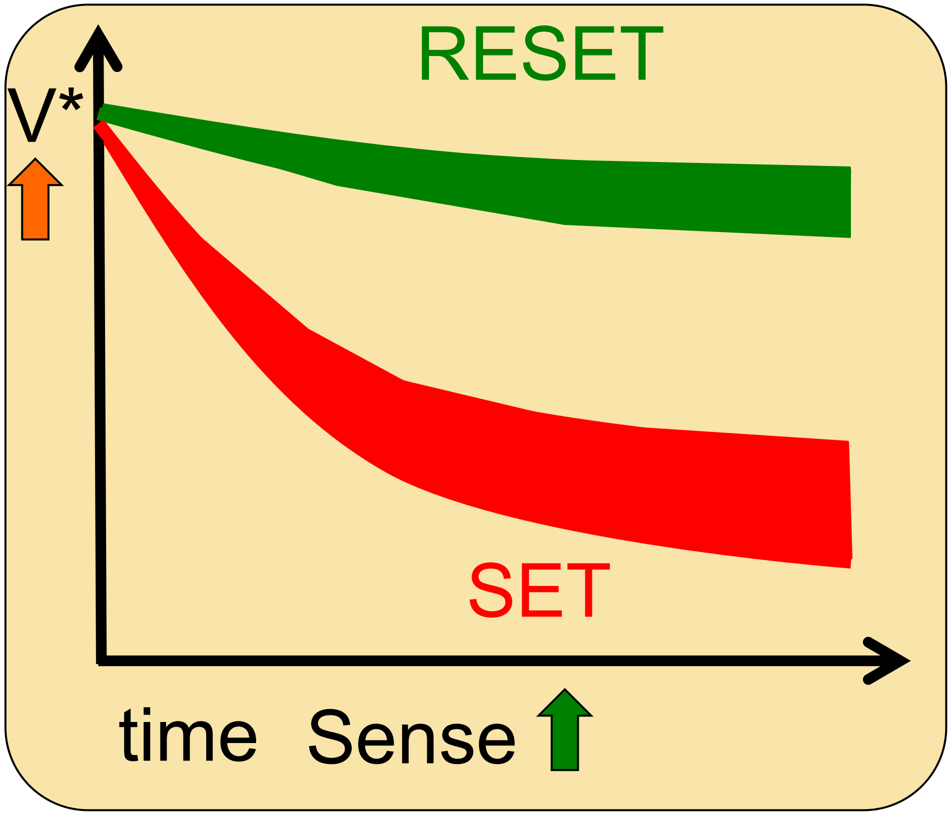


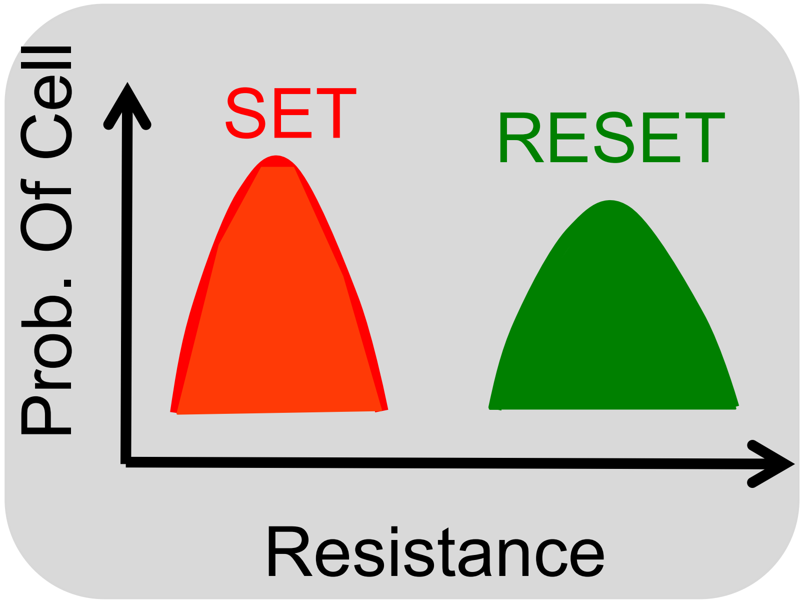
EFFECT OF HIGH BITLINE VOLTAGE



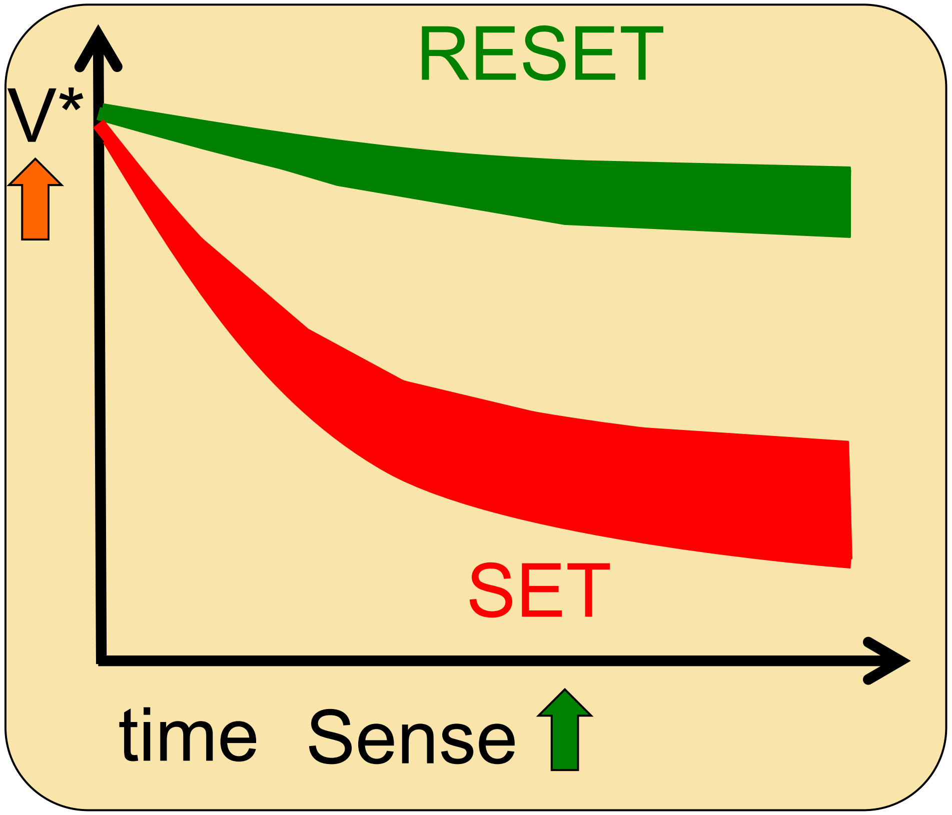


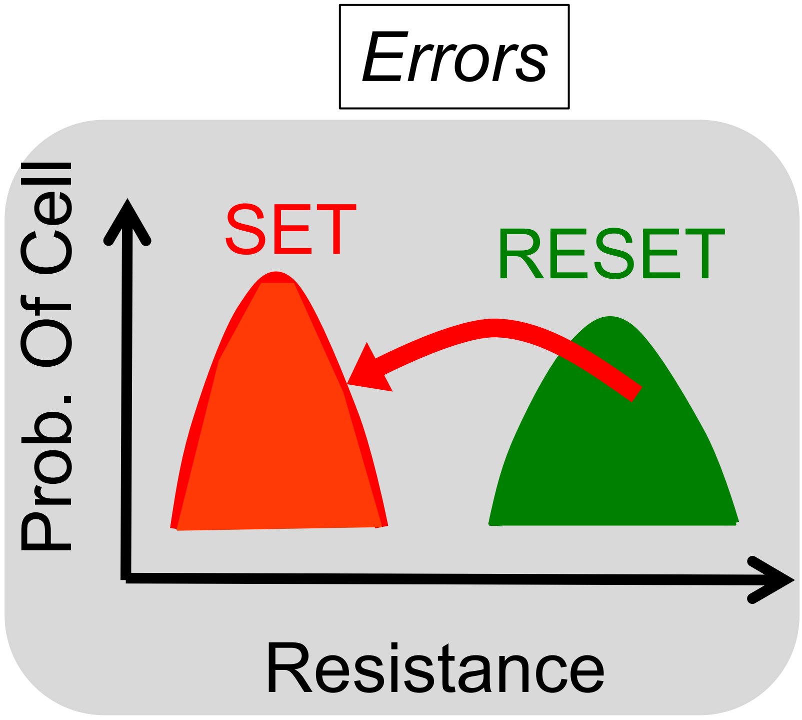
EFFECT OF HIGH BITLINE VOLTAGE





EFFECT OF HIGH BITLINE VOLTAGE





Increasing bitline voltage causes errors

GOAL

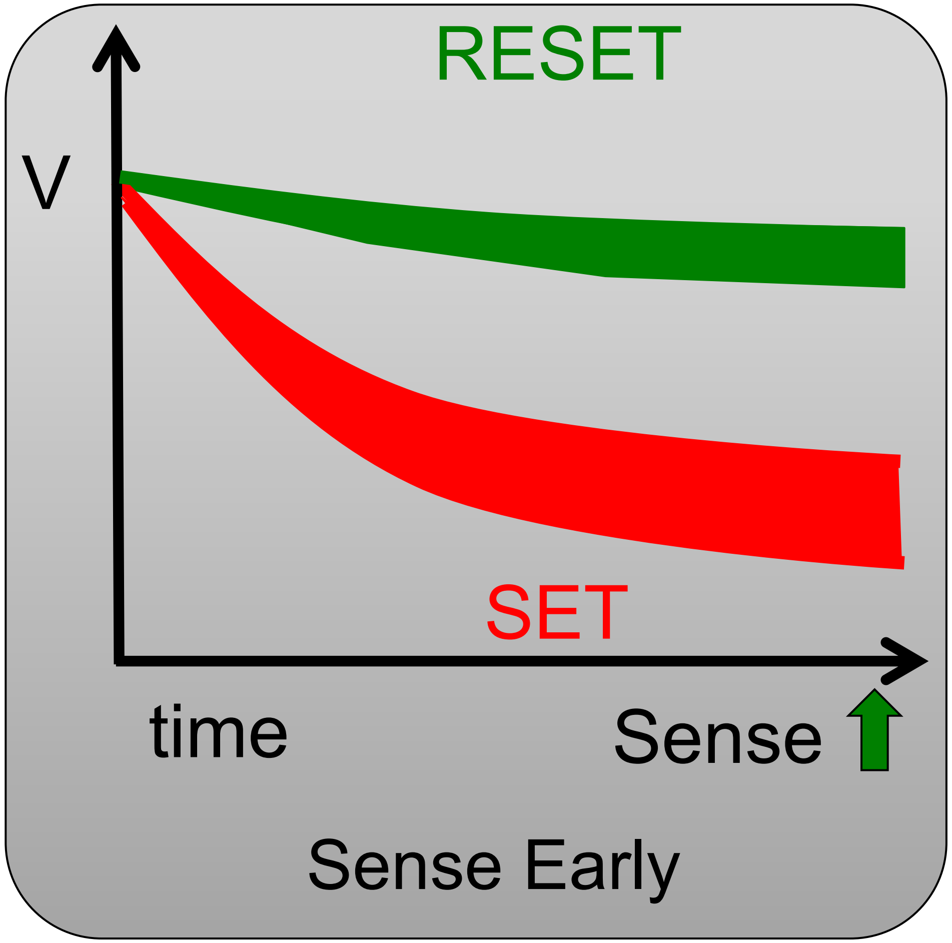
Reduce read latency by

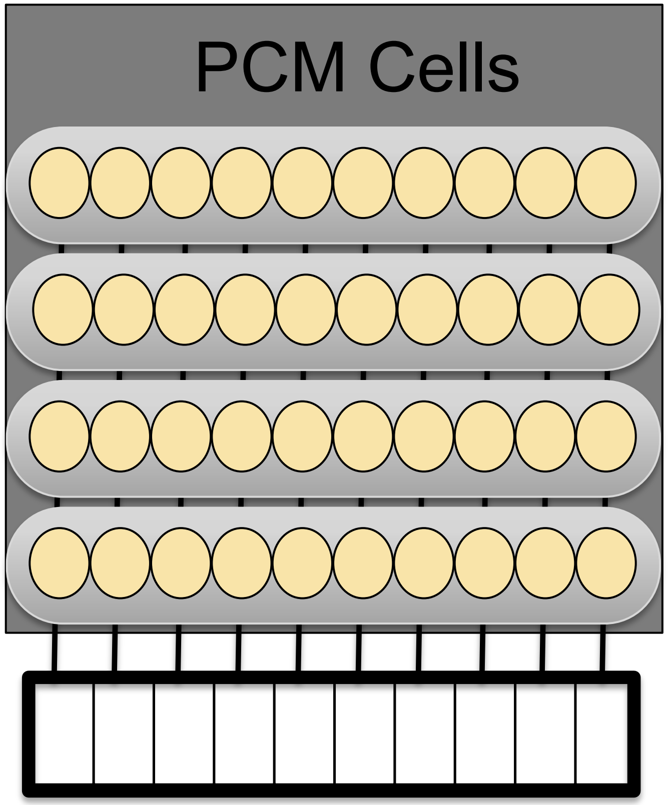
1. Exploiting variability in PCM cells → Early Read
2. Higher voltage to read PCM cells → Turbo Read

OUTLINE

- Background
- Early Read ⇦
- Turbo Read
- Early+Turbo Read
- Results
- Summary

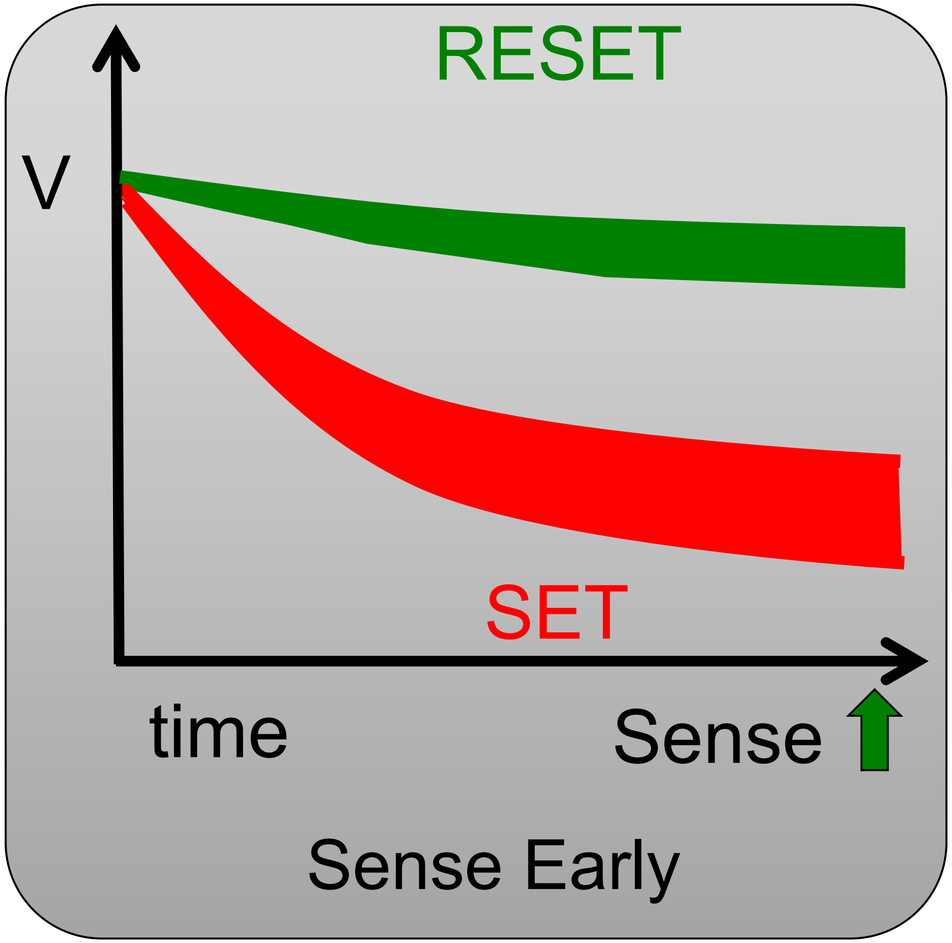
SENSING EARLY: OBSERVATION

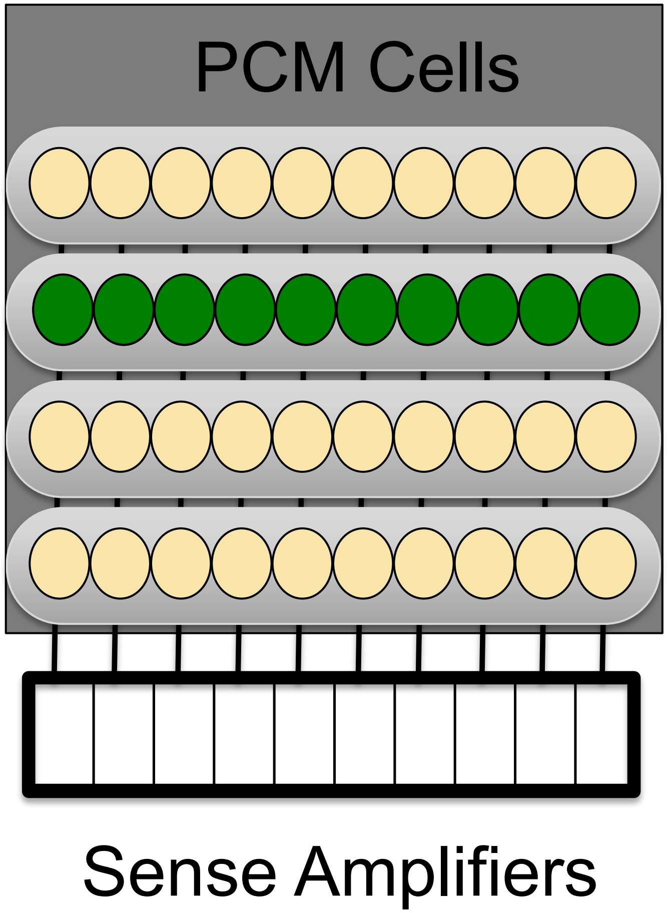




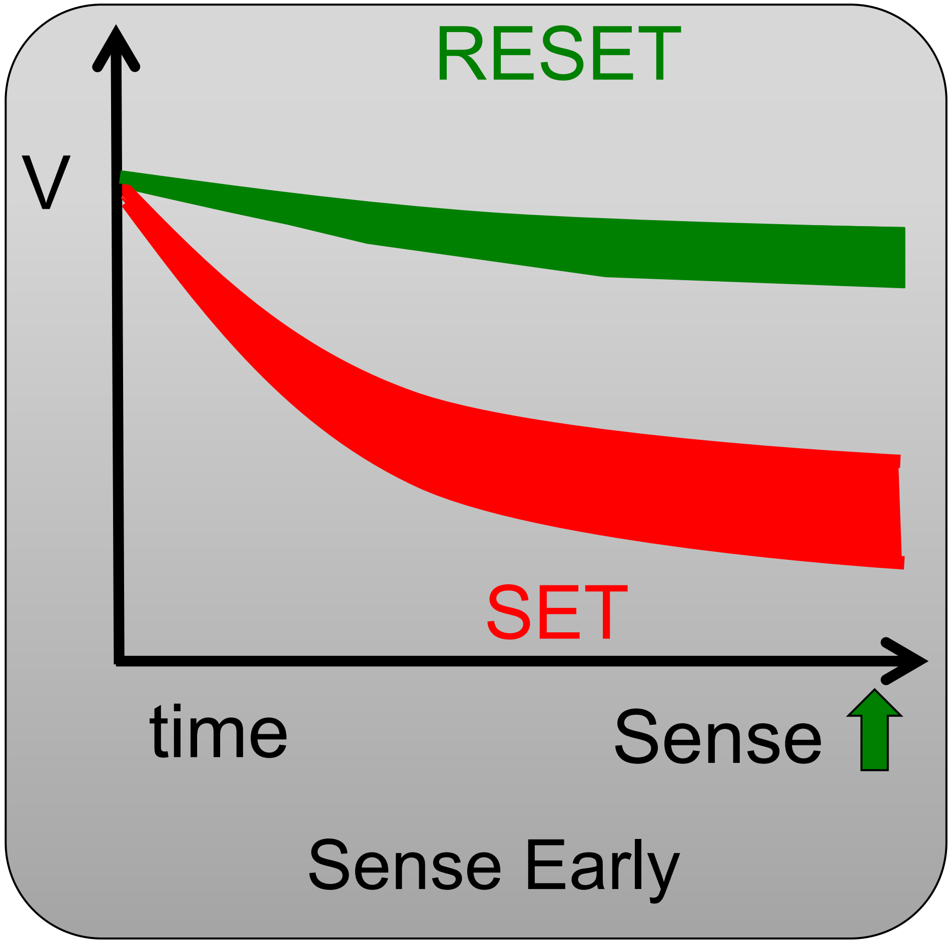
Sense Amplifiers

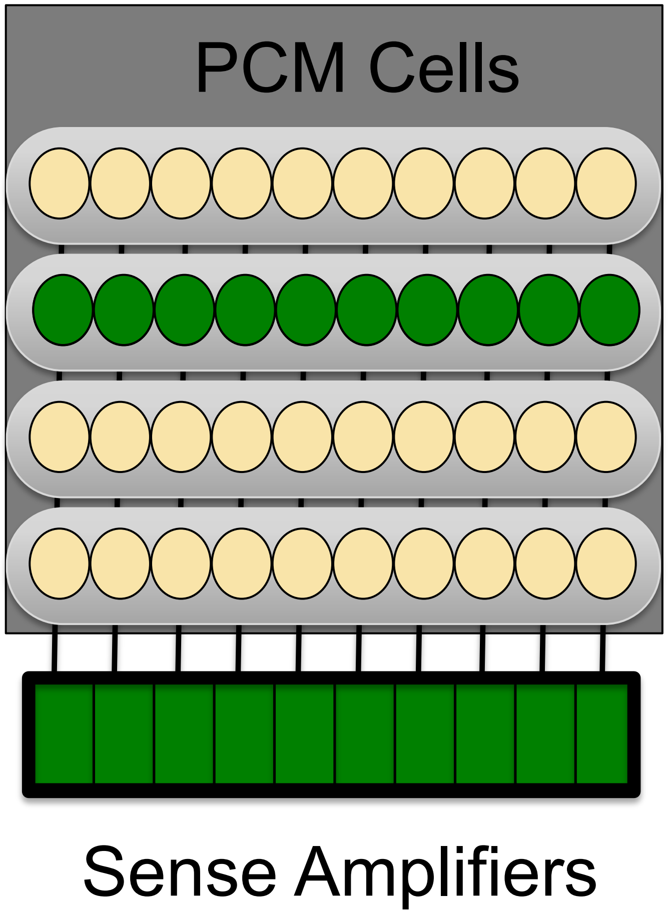
SENSING EARLY: OBSERVATION



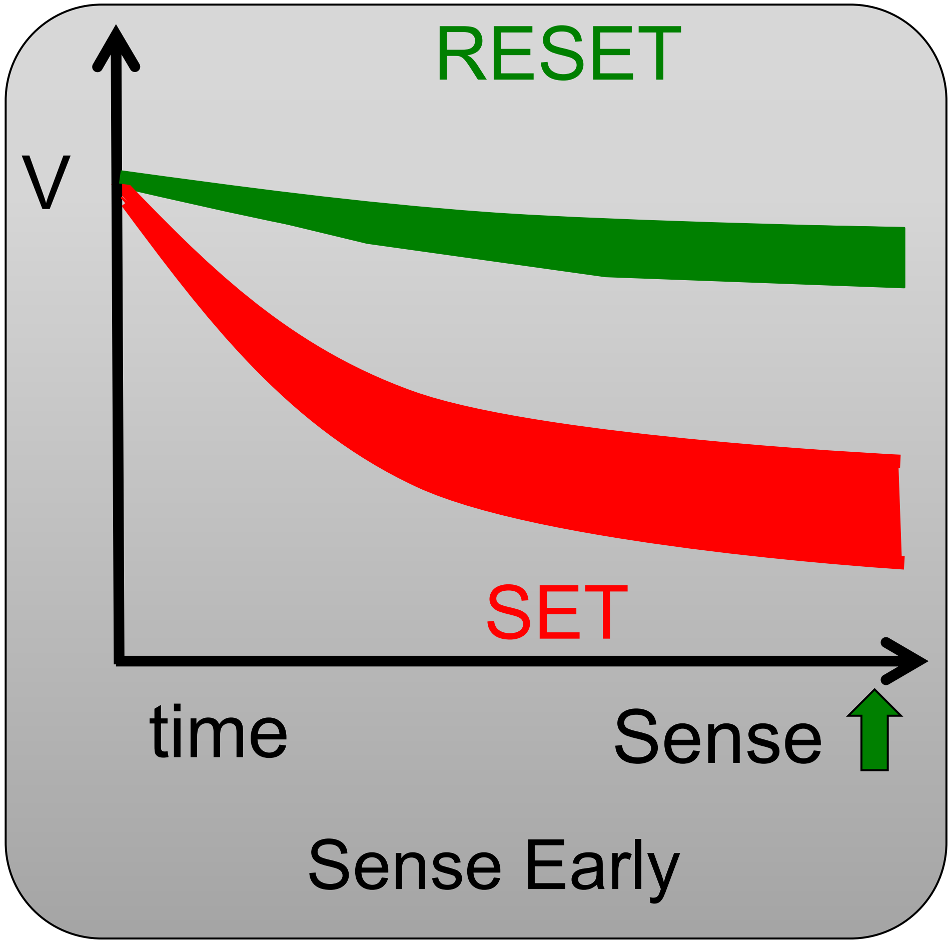


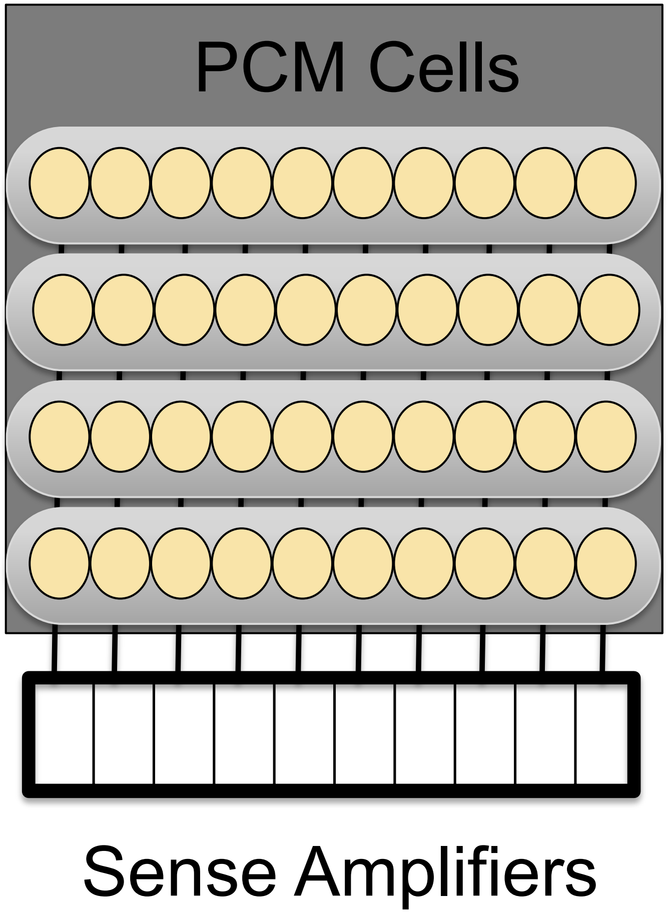
SENSING EARLY: OBSERVATION



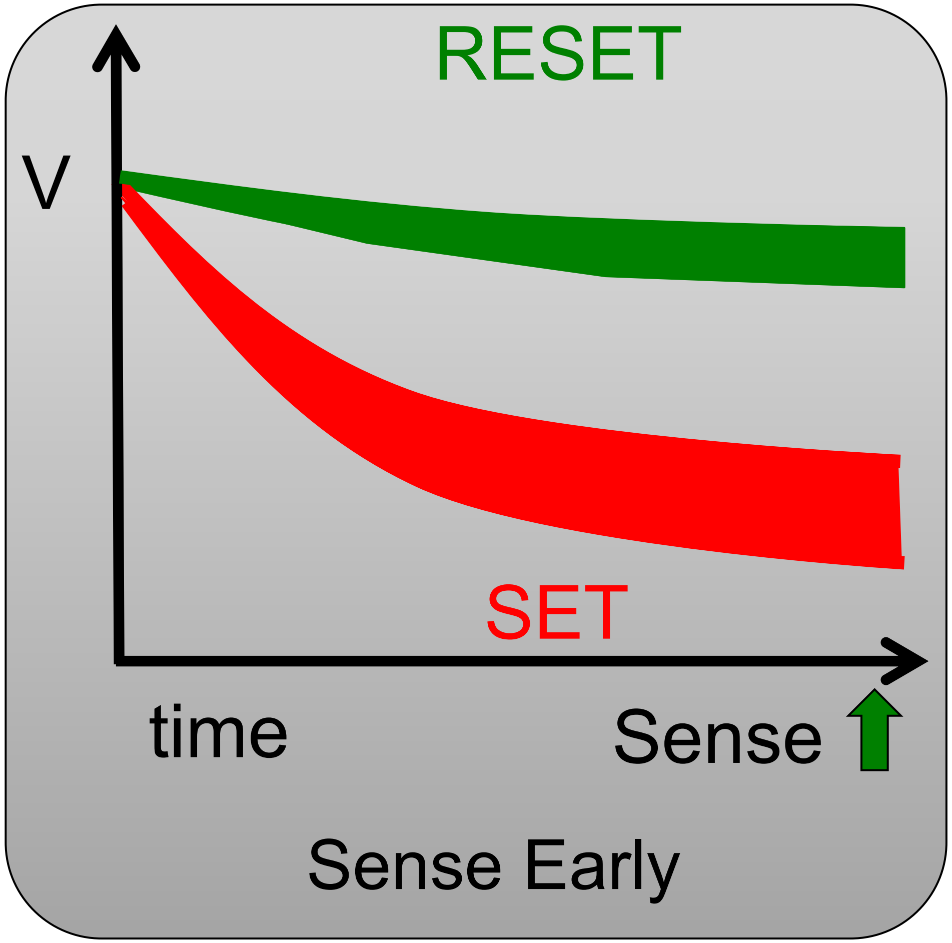


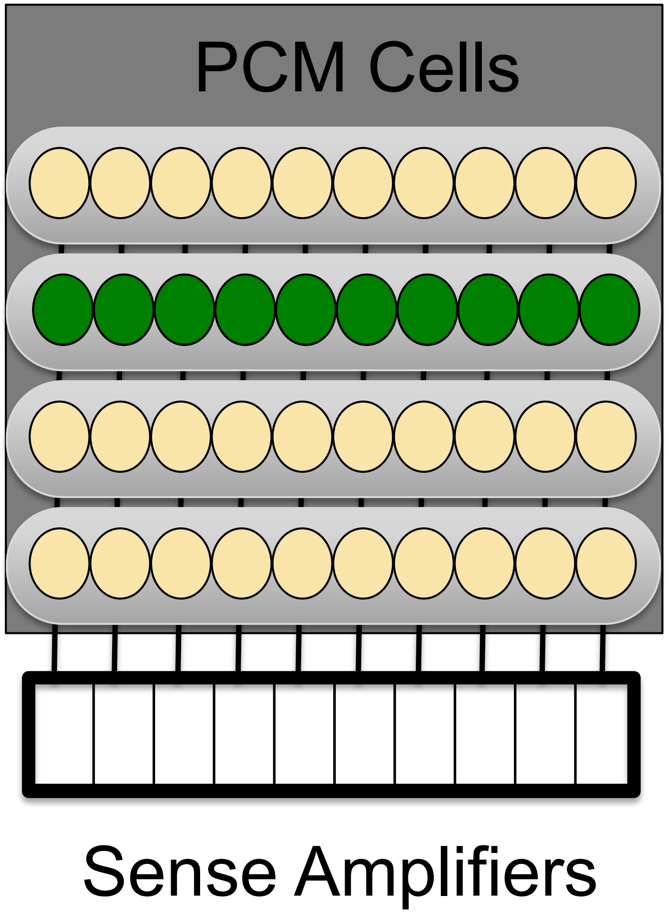
SENSING EARLY: OBSERVATION



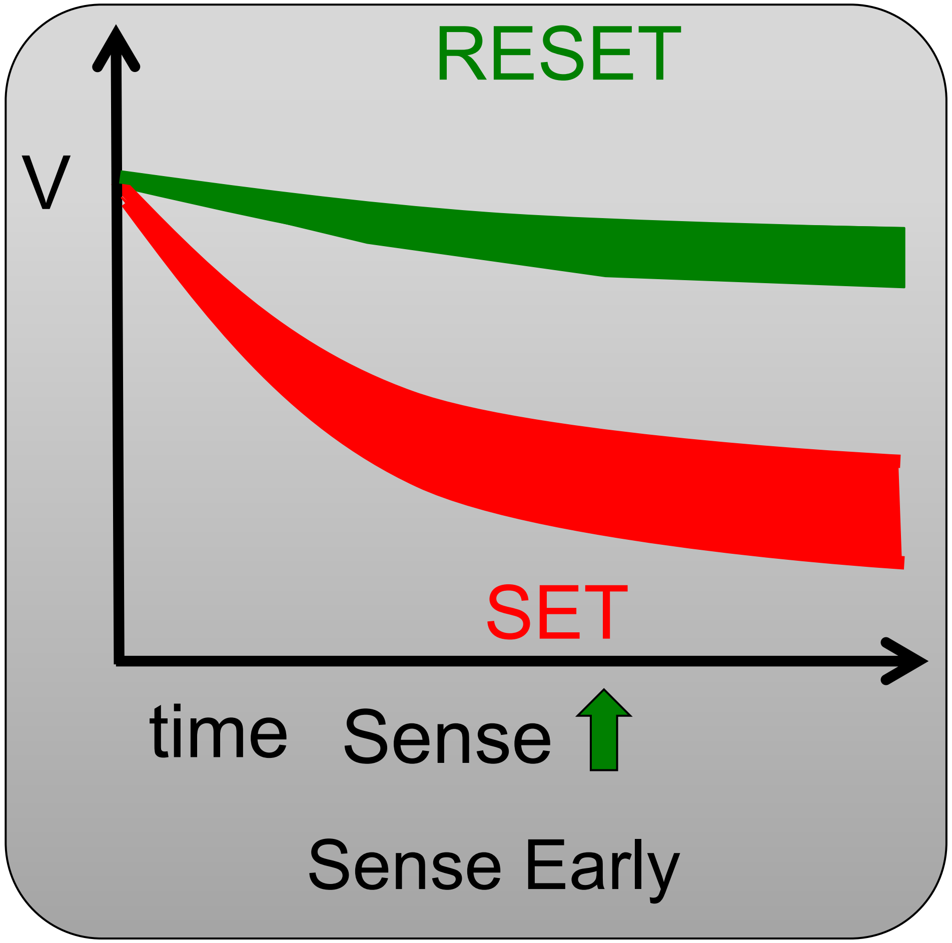


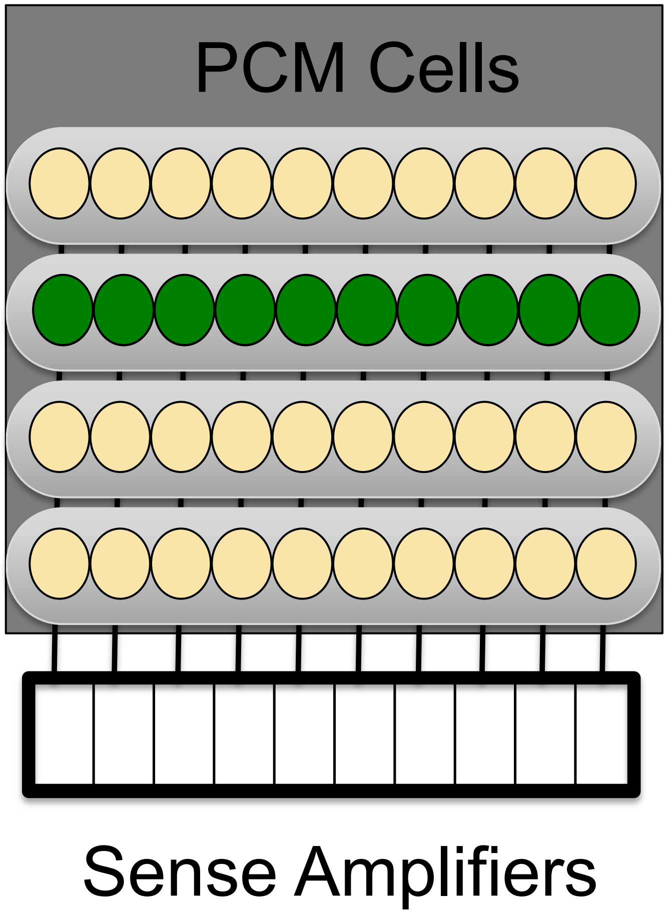
SENSING EARLY: OBSERVATION



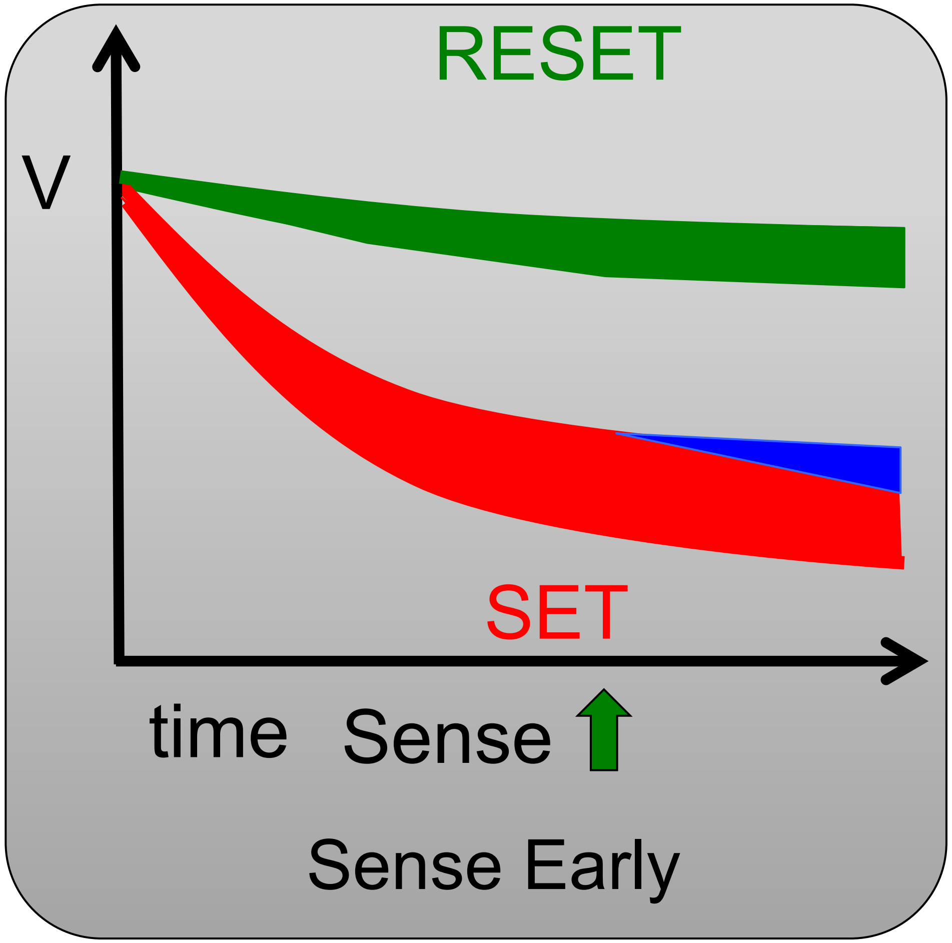


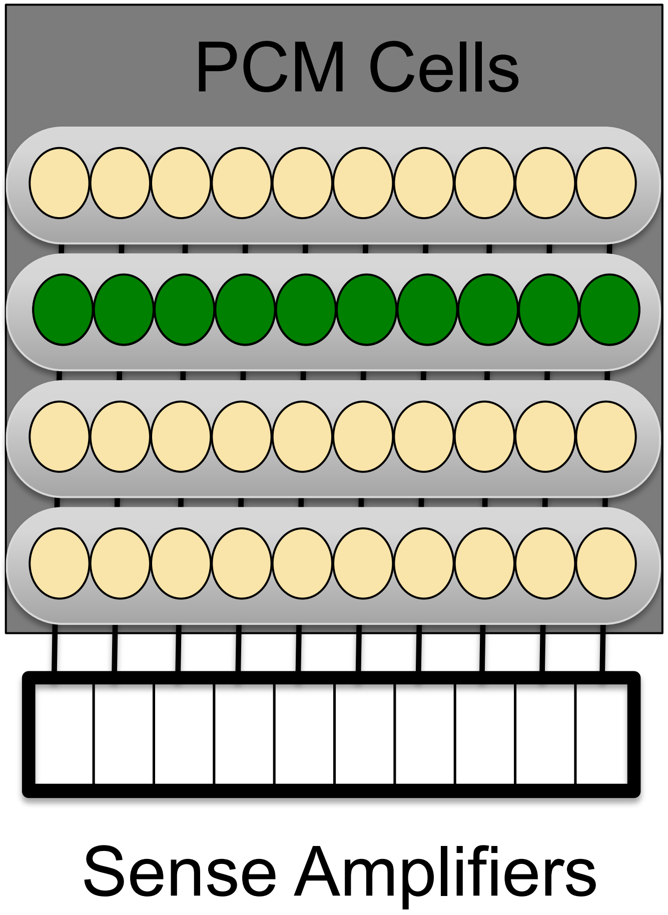
SENSING EARLY: OBSERVATION



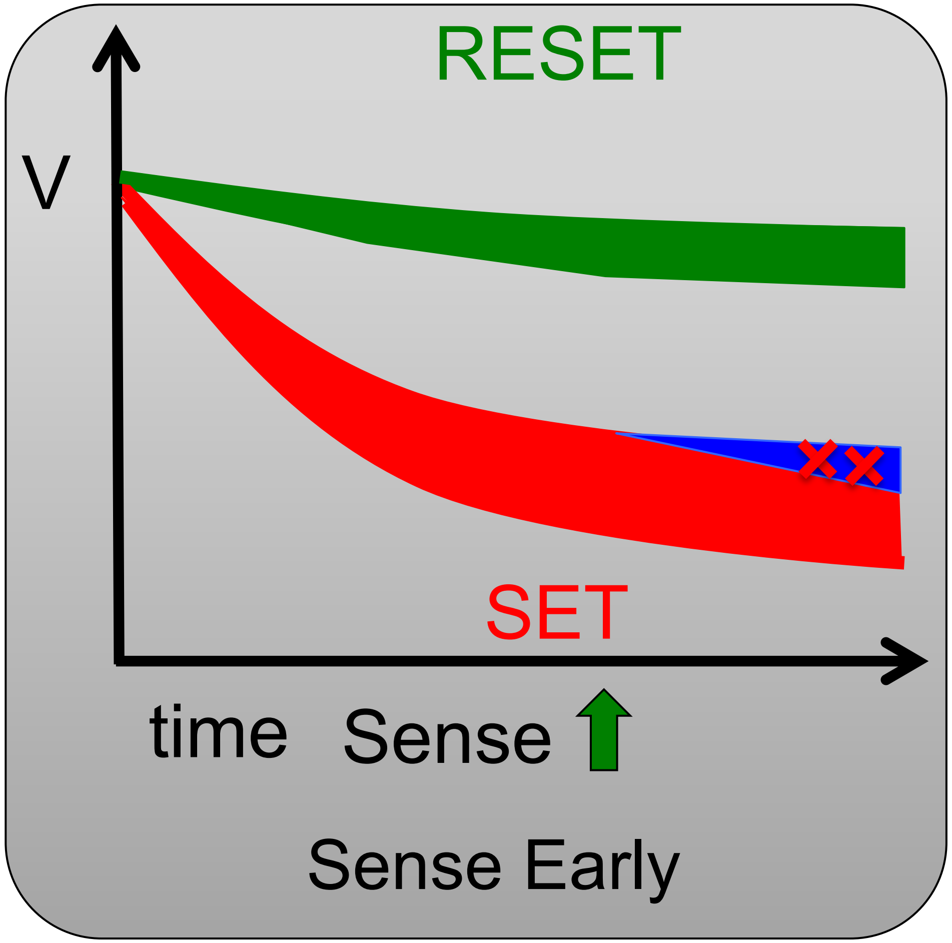


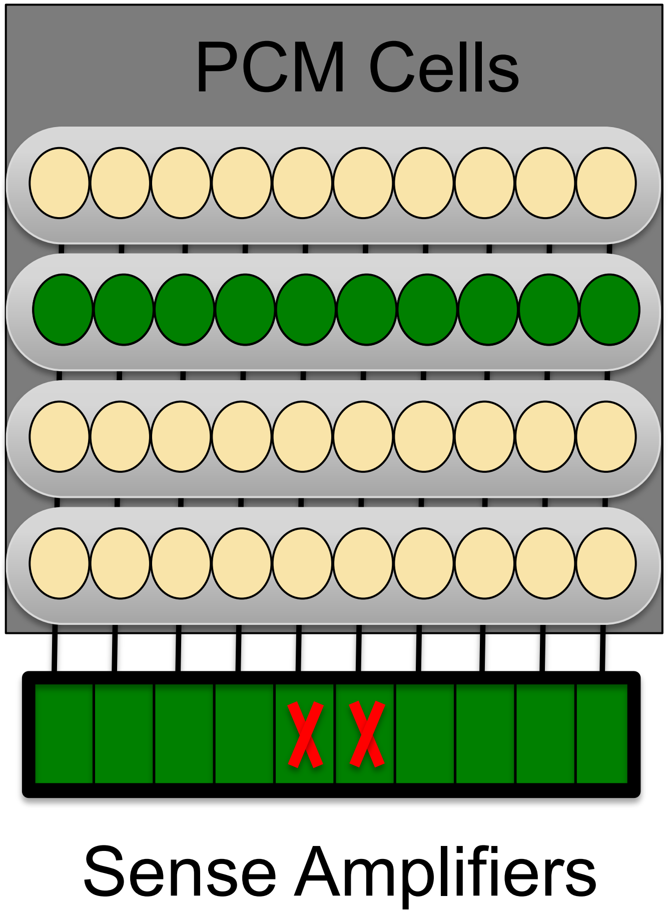
SENSING EARLY: OBSERVATION



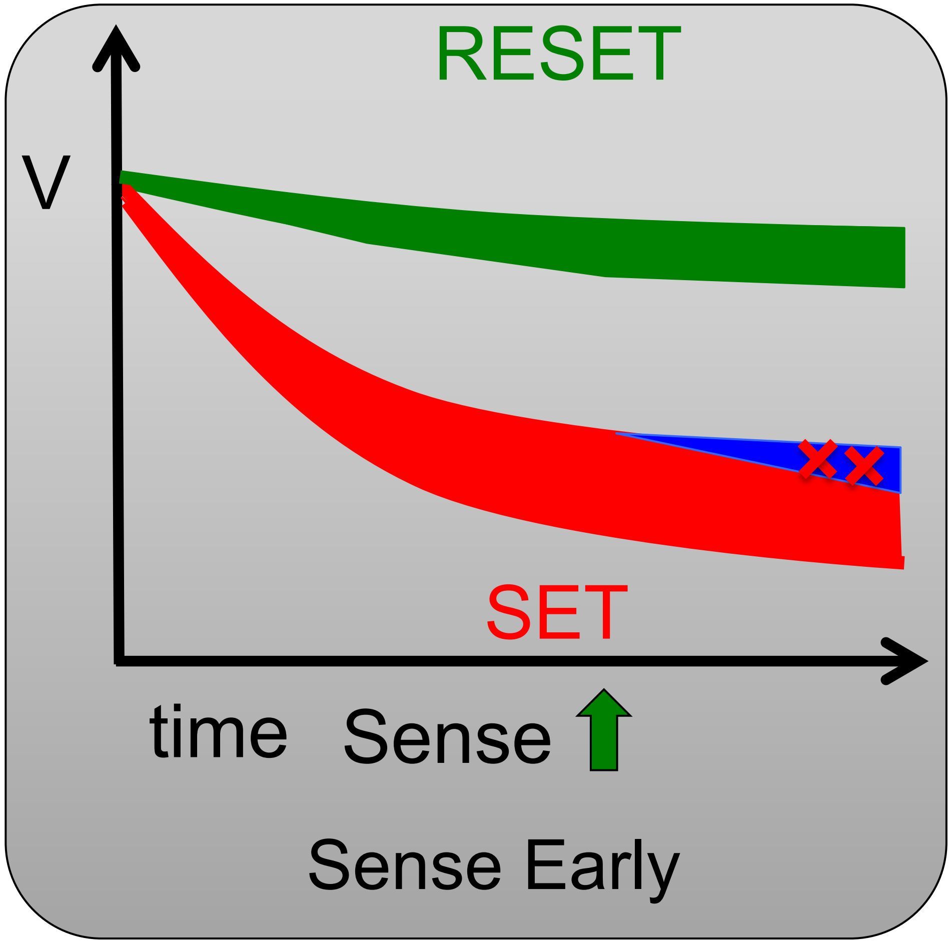


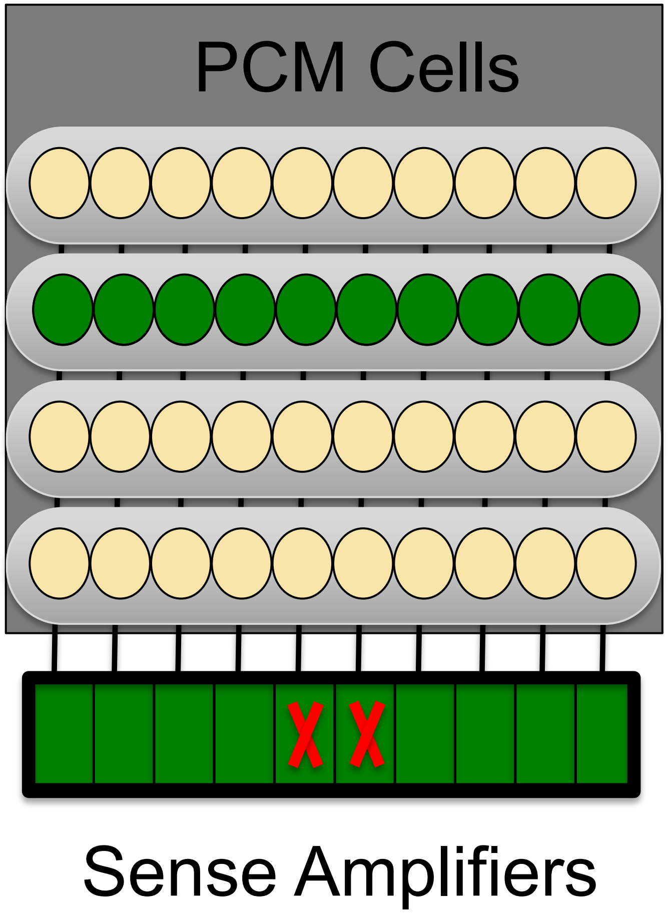
SENSING EARLY: OBSERVATION





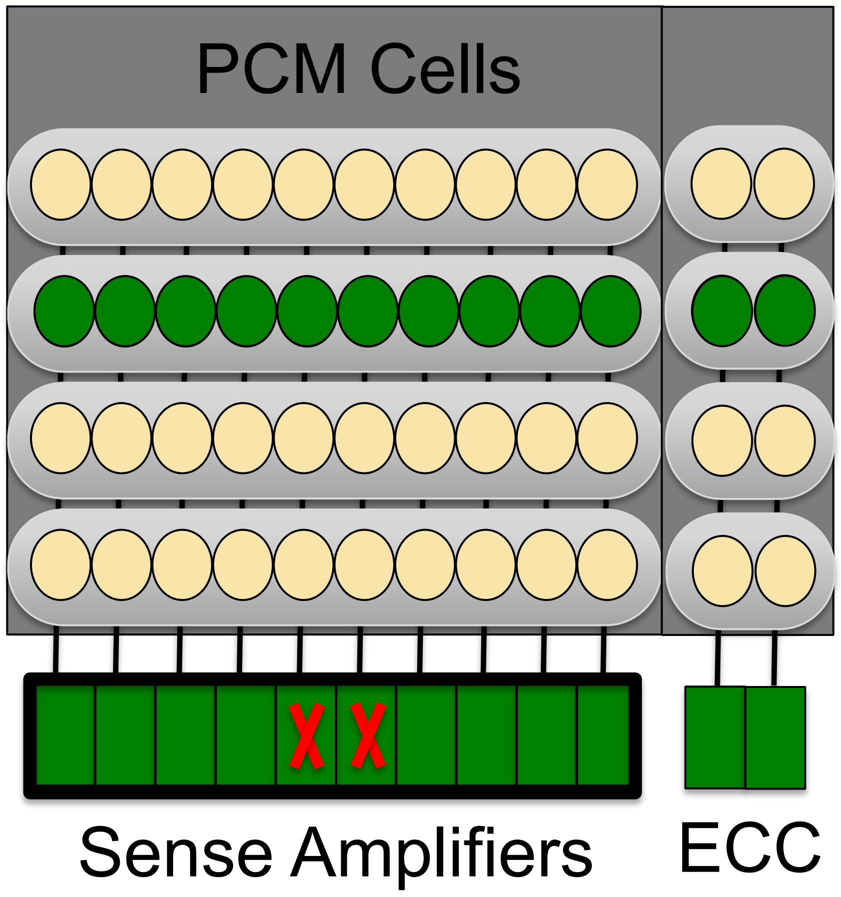
SENSING EARLY: OBSERVATION



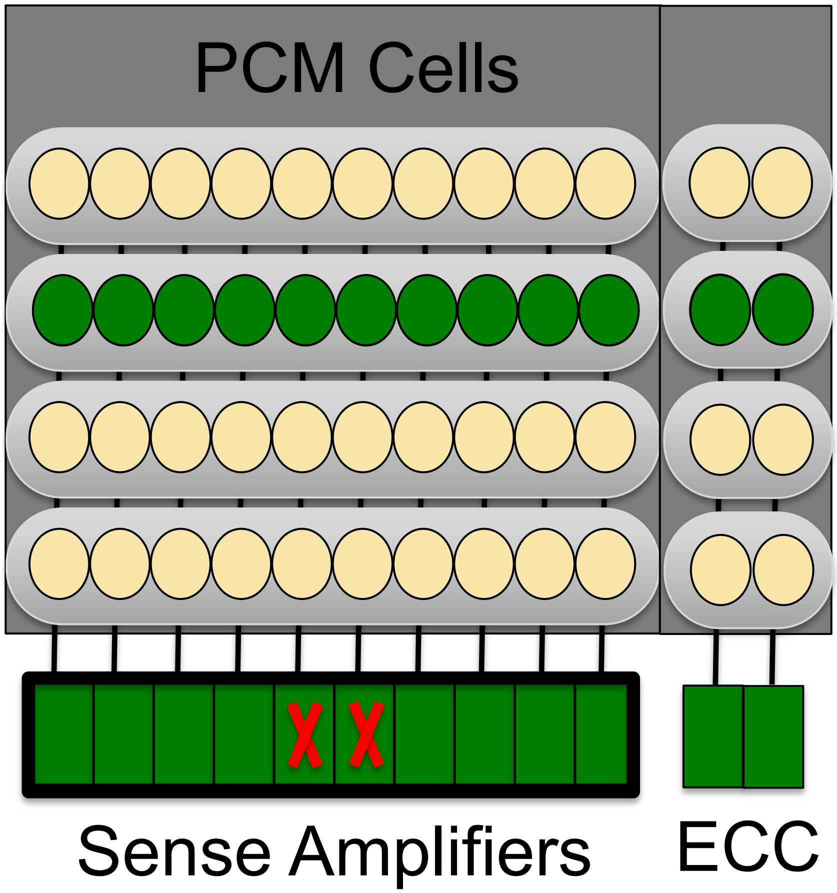


1. Sensing early causes errors in sense amplifiers
2. The cells in PCM substrate have no error

ECC TO CORRECT LATCHING ERRORS

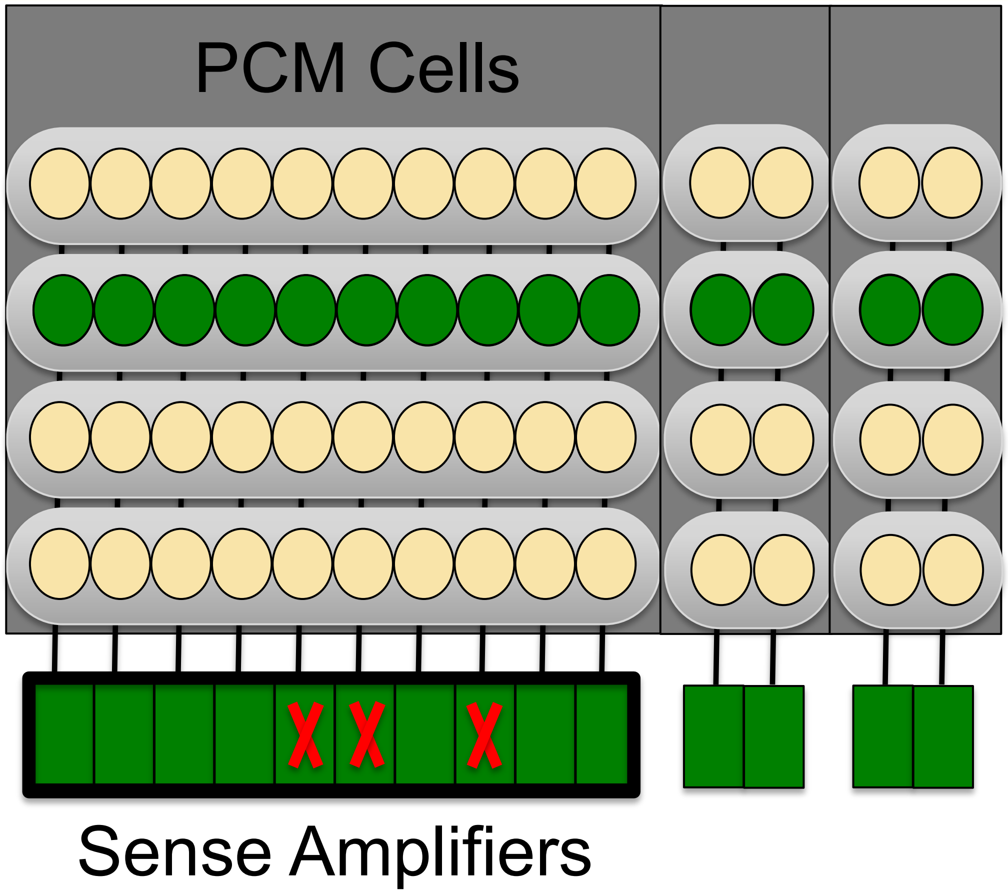


ECC TO CORRECT LATCHING ERRORS



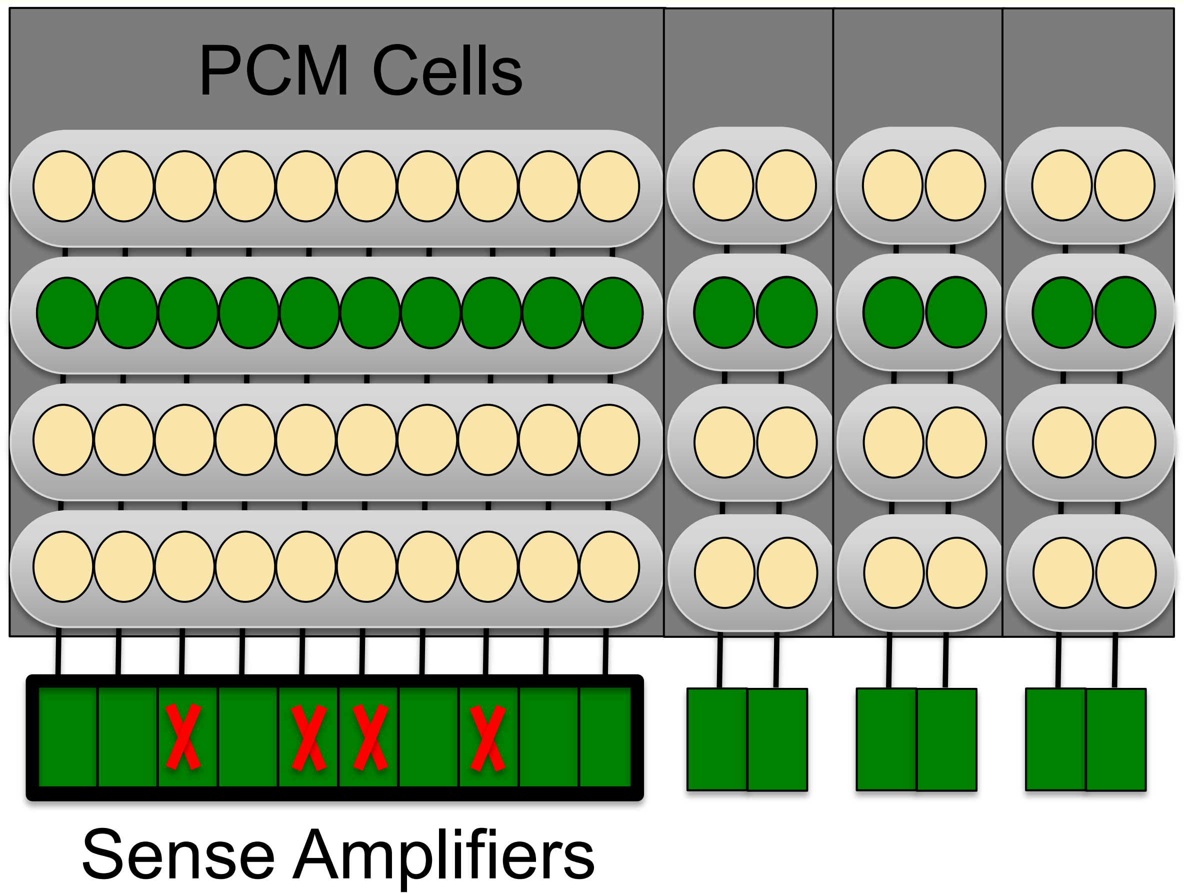
Lower sensing time ➔ more errors ➔ stronger ECC

ECC TO CORRECT LATCHING ERRORS



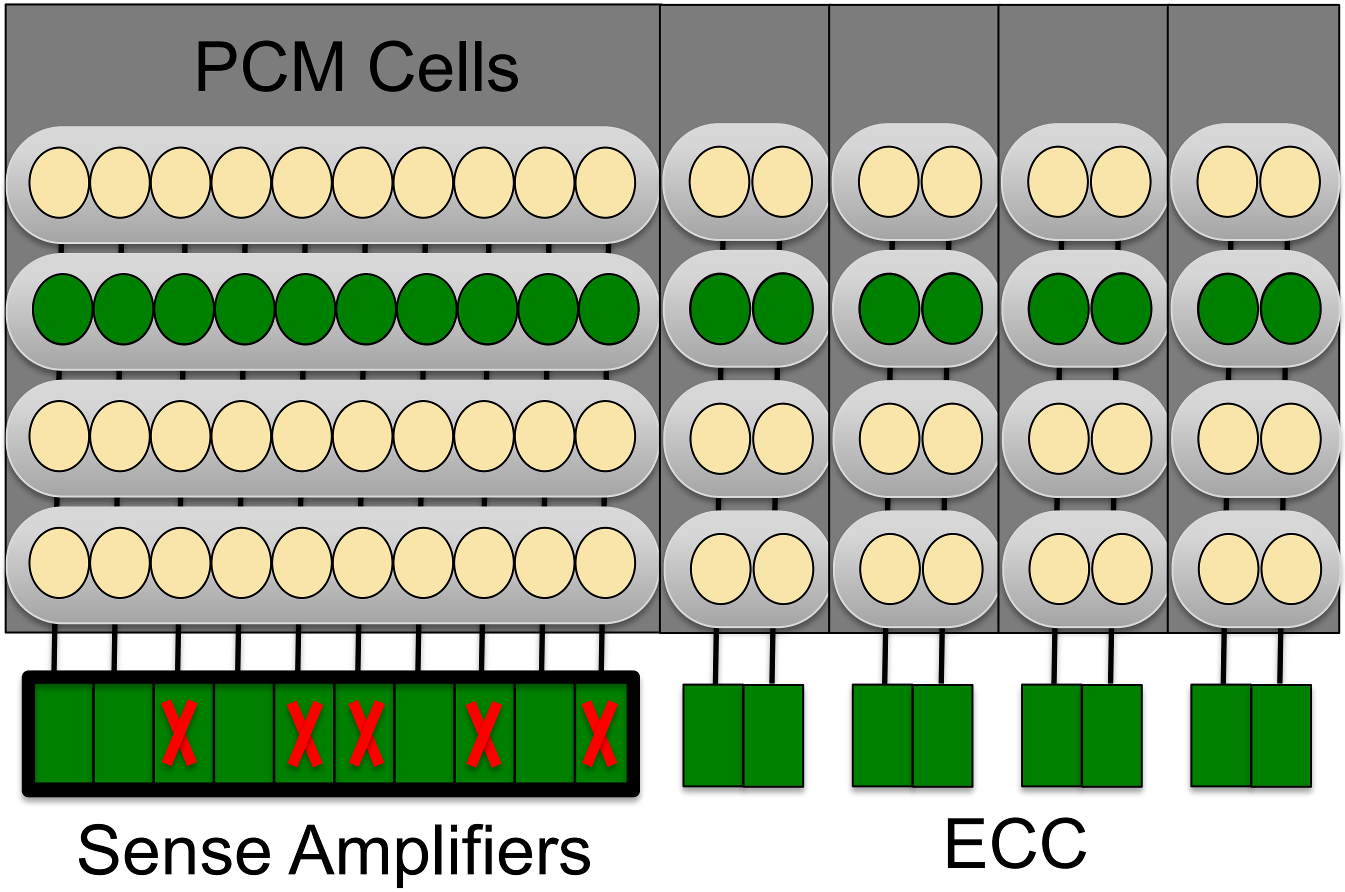
Lower sensing time ➔ more errors ➔ stronger ECC

ECC TO CORRECT LATCHING ERRORS



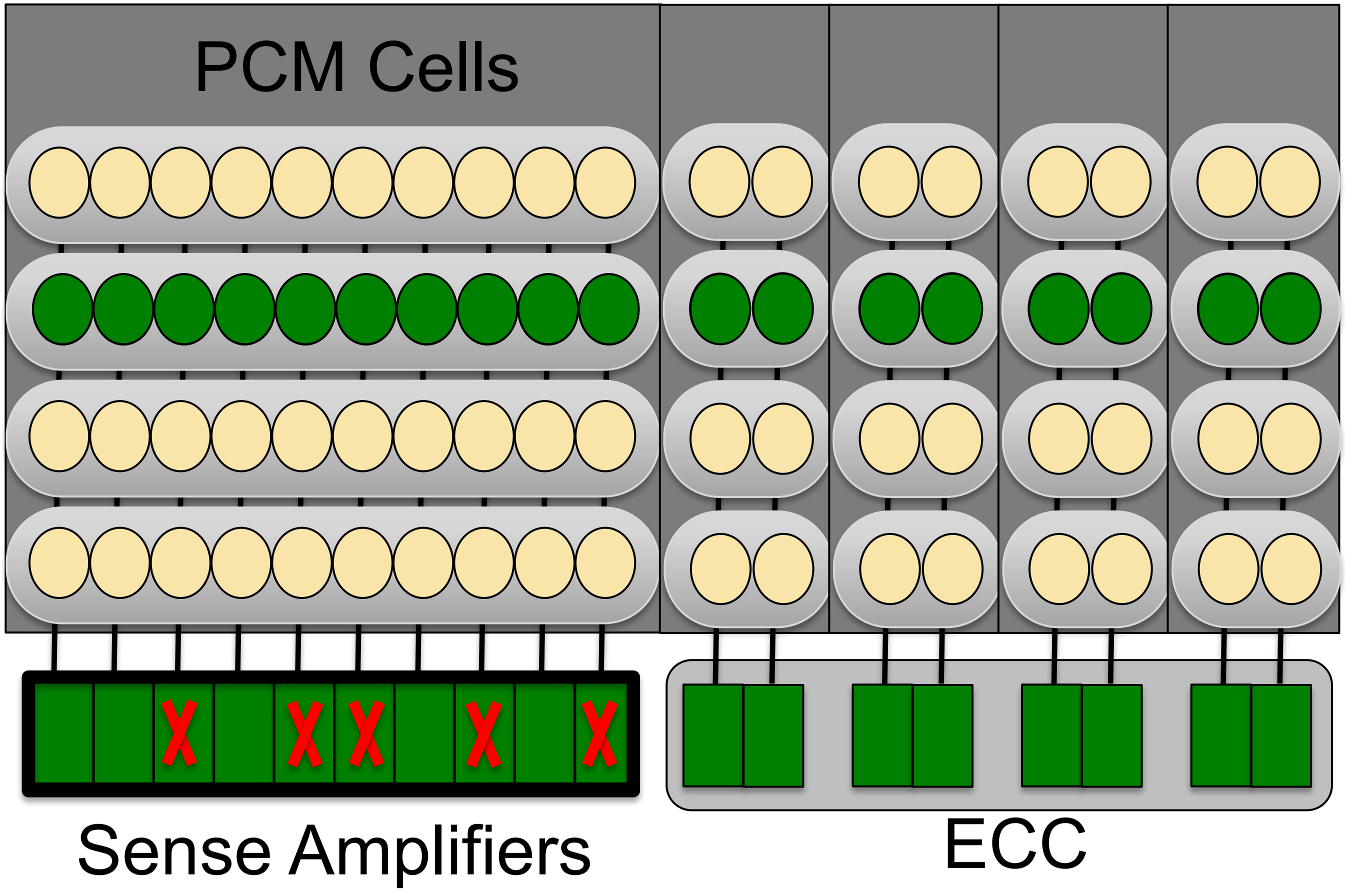
Lower sensing time ➔ more errors ➔ stronger ECC

ECC TO CORRECT LATCHING ERRORS



Lower sensing time ➔ more errors ➔ stronger ECC

ECC TO CORRECT LATCHING ERRORS

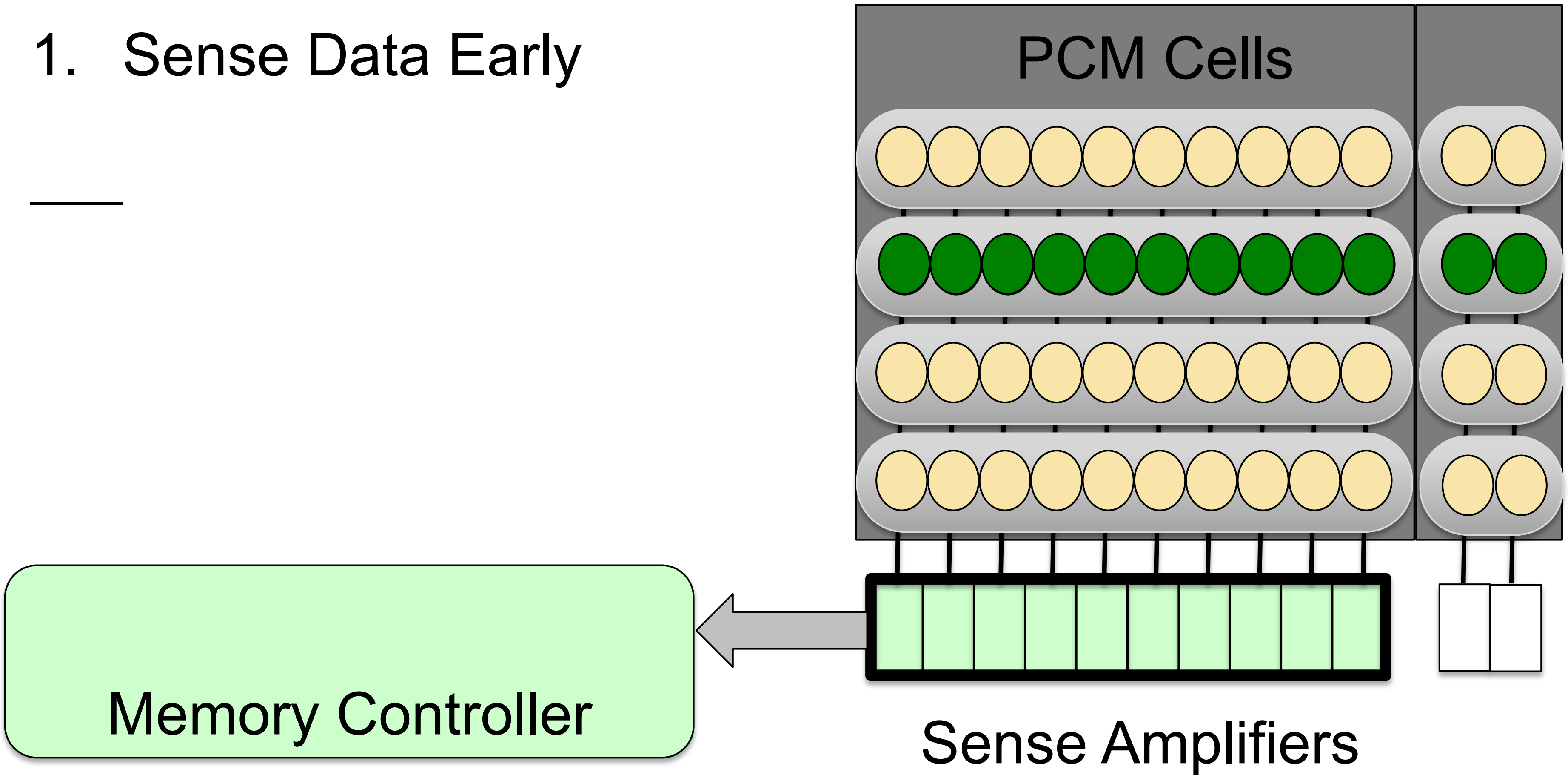


Lower sensing time → more errors → stronger ECC

Strong ECC → Huge area overheads

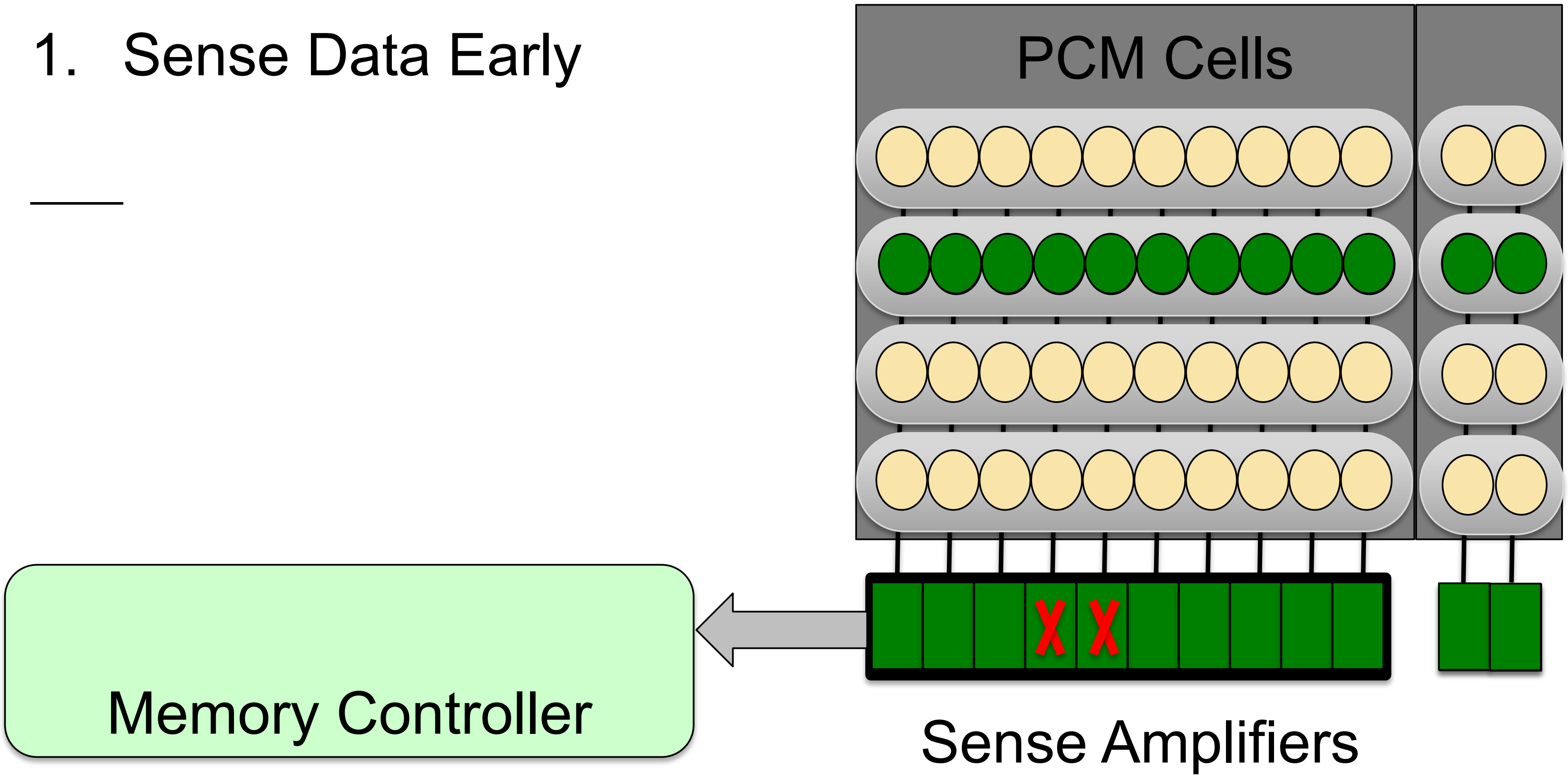
INSIGHT: USE RETRY FOR CORRECTION

1. Sense Data Early



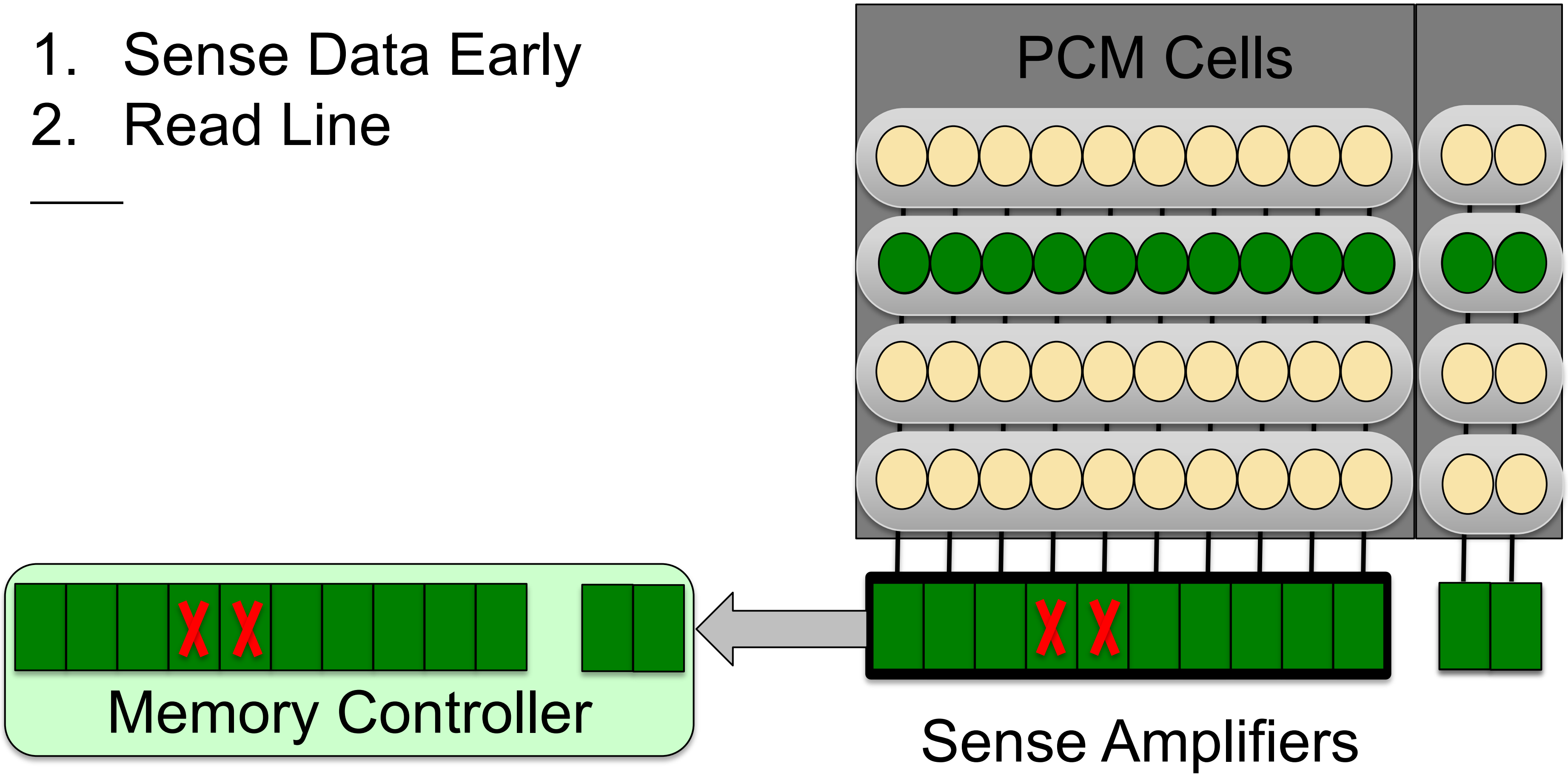
INSIGHT: USE RETRY FOR CORRECTION

1. Sense Data Early



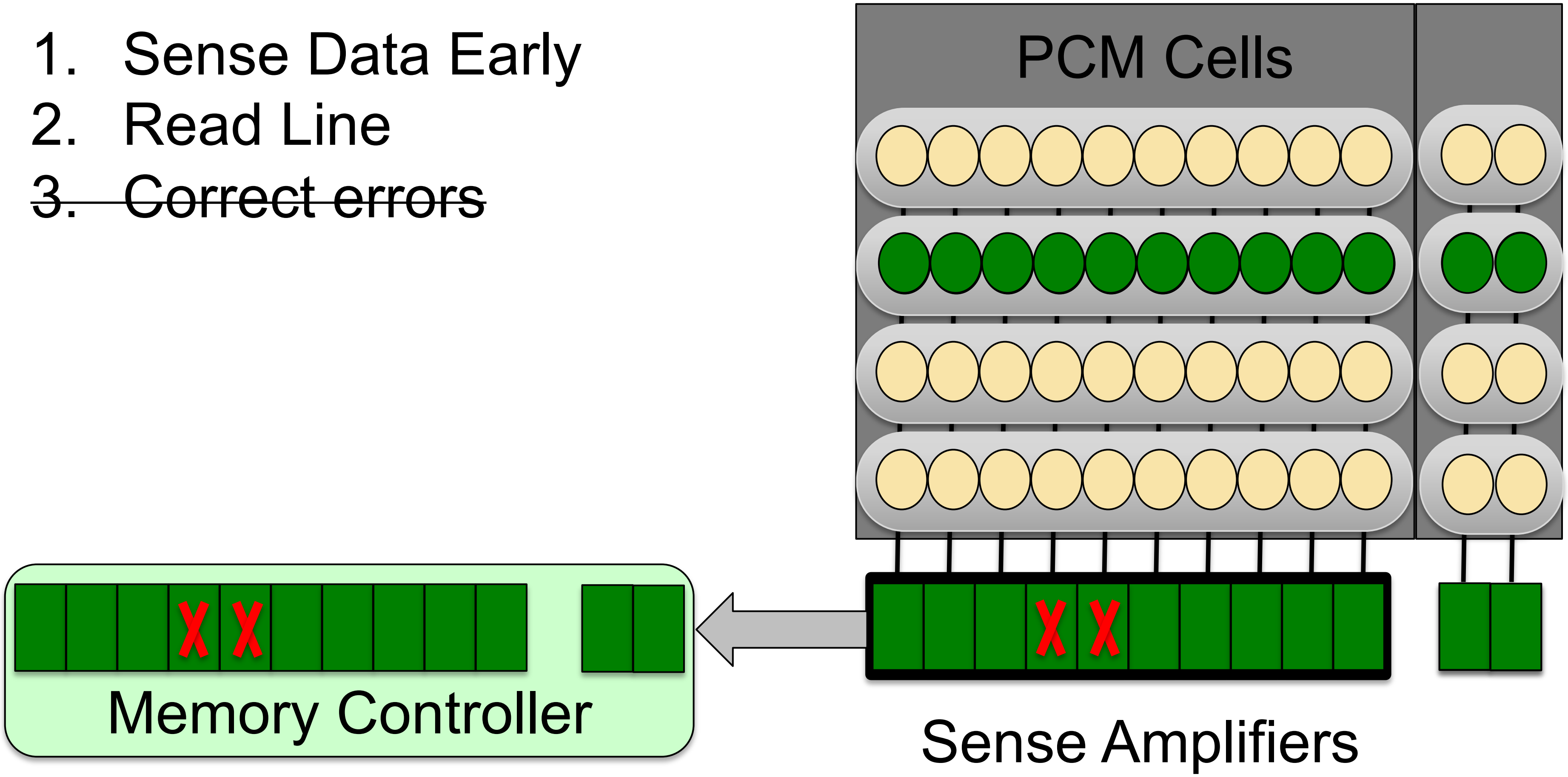
INSIGHT: USE RETRY FOR CORRECTION

1. Sense Data Early
2. Read Line



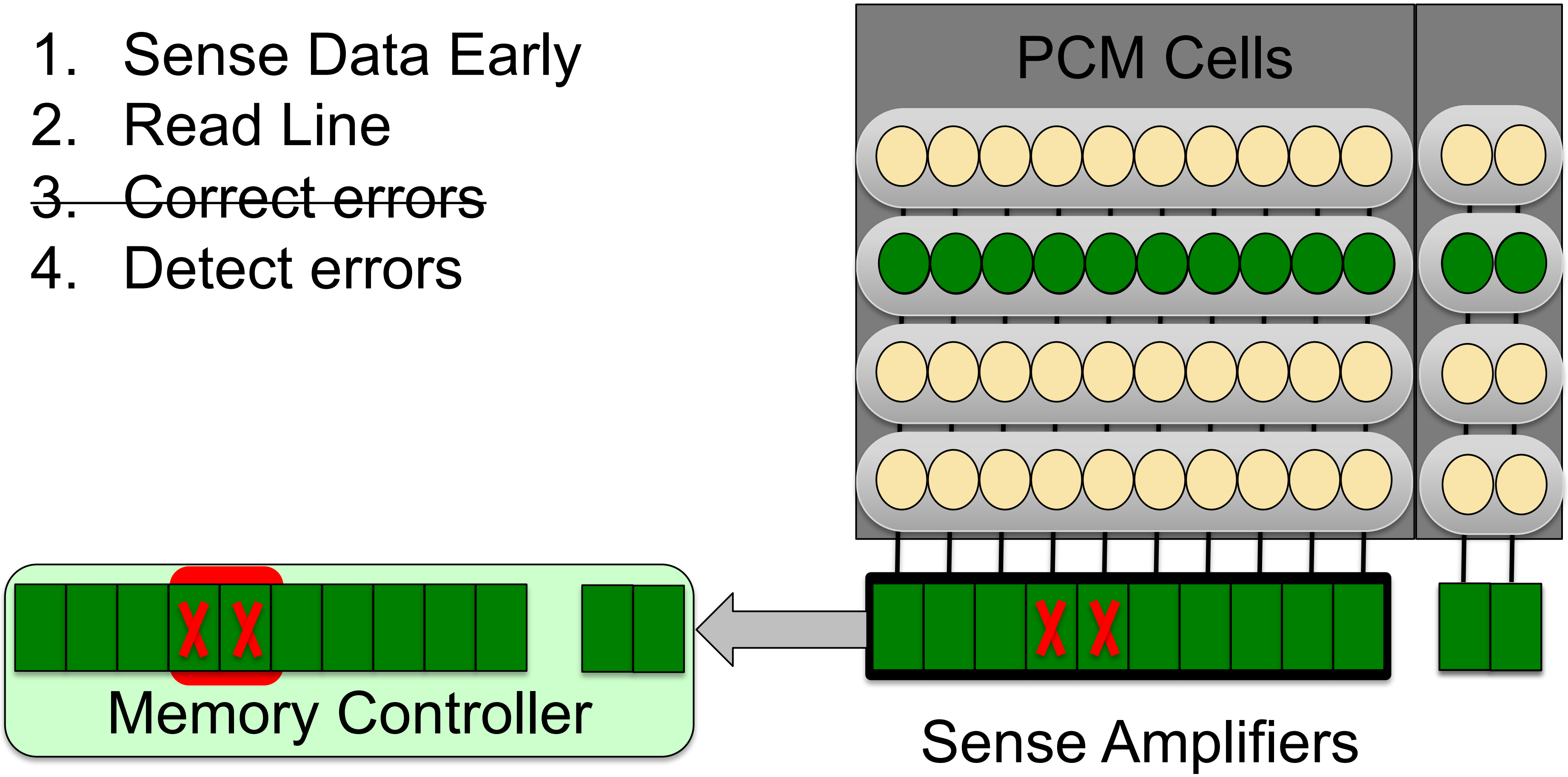
INSIGHT: USE RETRY FOR CORRECTION

1. Sense Data Early
2. Read Line
3. ~~Correct errors~~



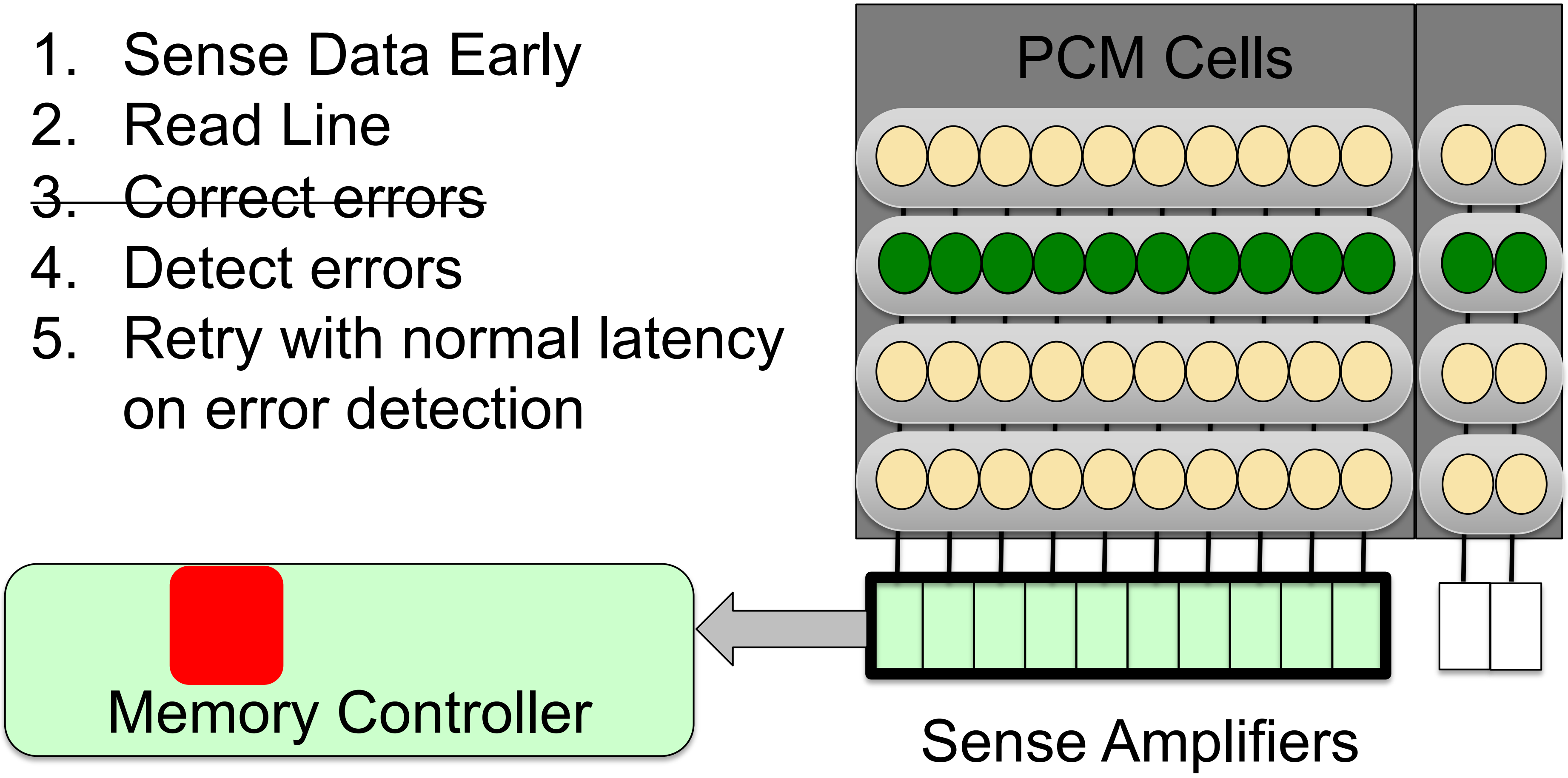
INSIGHT: USE RETRY FOR CORRECTION

1. Sense Data Early
2. Read Line
3. ~~Correct errors~~
4. Detect errors



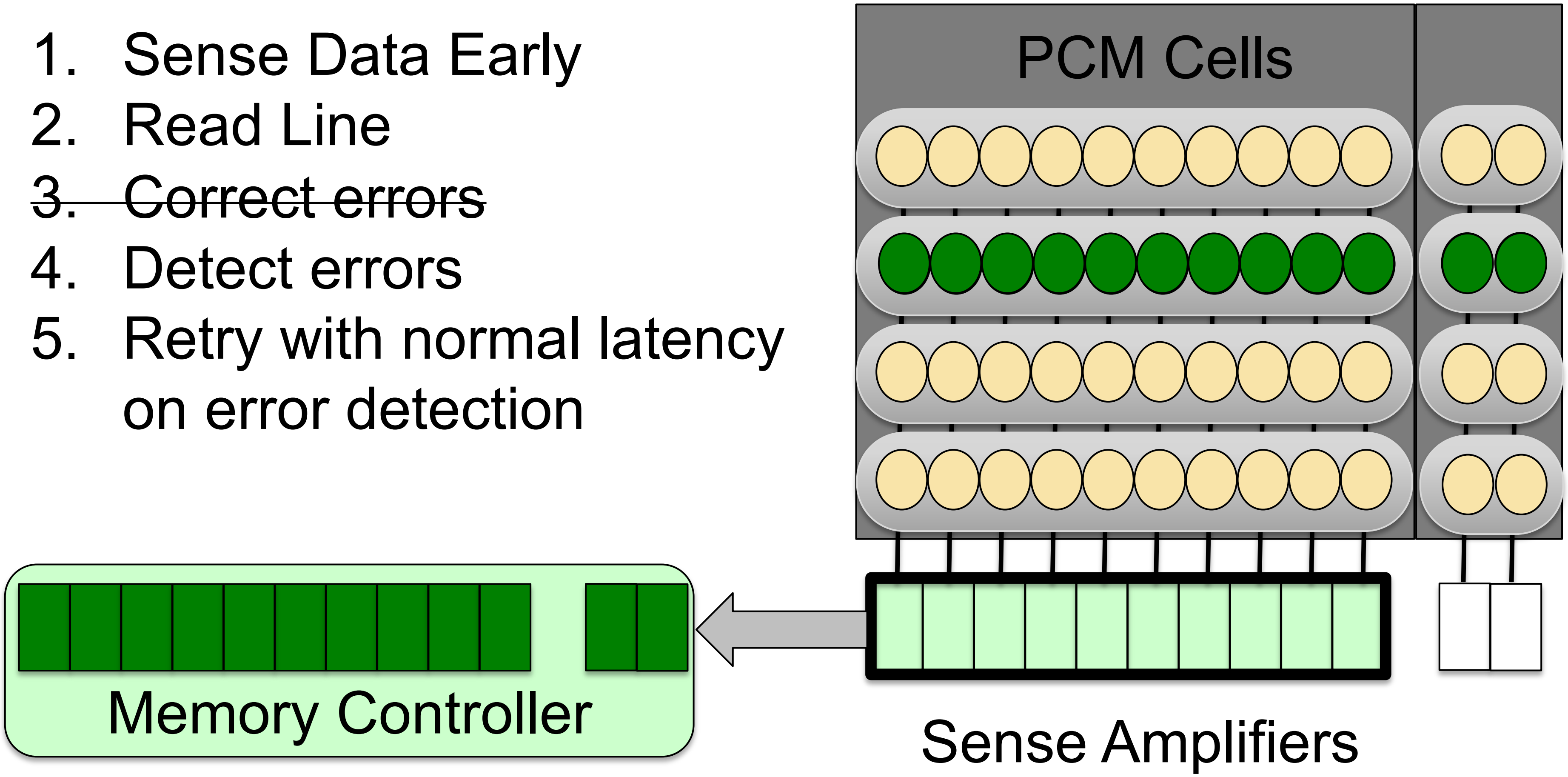
INSIGHT: USE RETRY FOR CORRECTION

1. Sense Data Early
2. Read Line
3. ~~Correct errors~~
4. Detect errors
5. Retry with normal latency on error detection



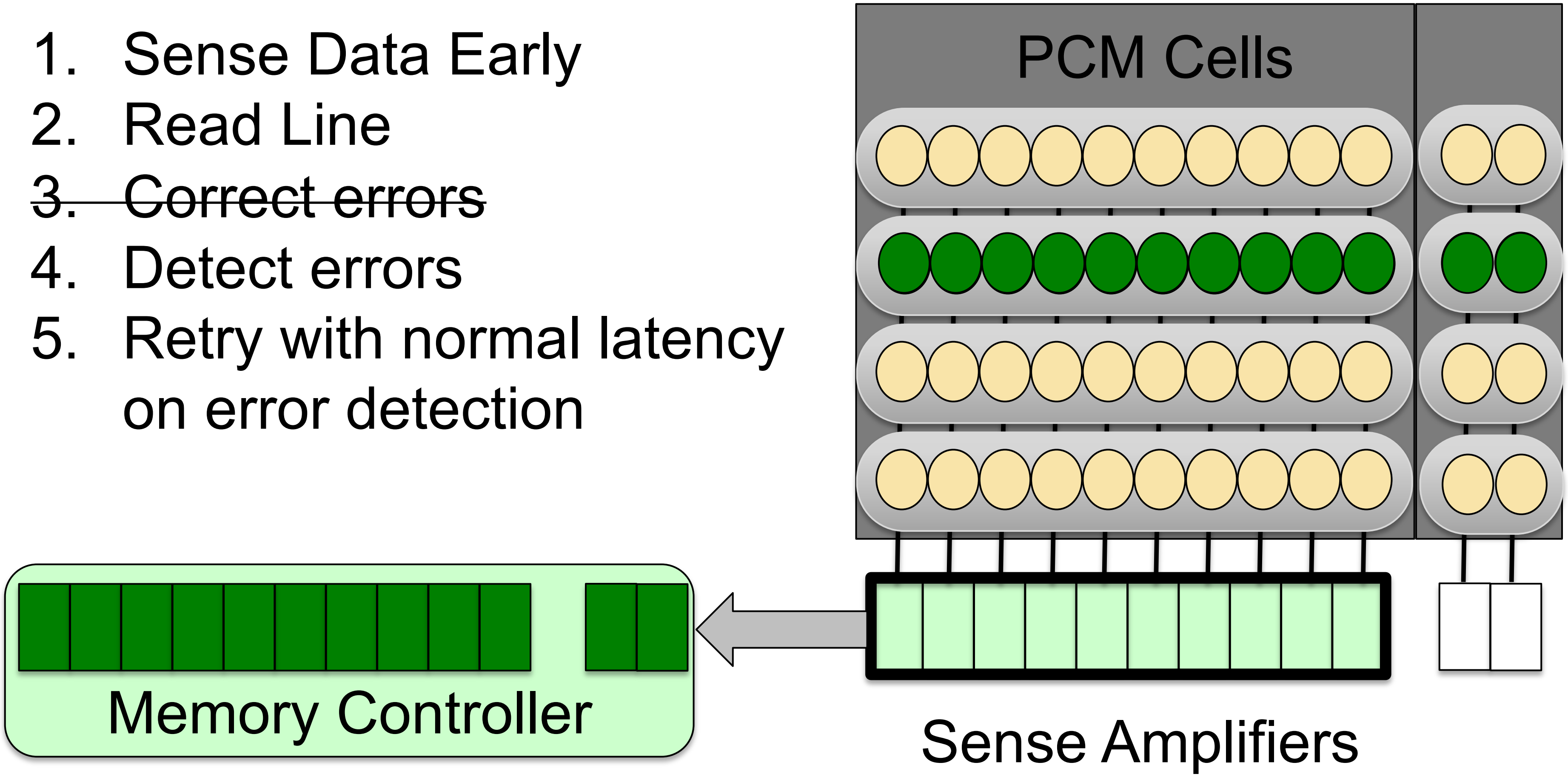
INSIGHT: USE RETRY FOR CORRECTION

1. Sense Data Early
2. Read Line
3. ~~Correct errors~~
4. Detect errors
5. Retry with normal latency on error detection



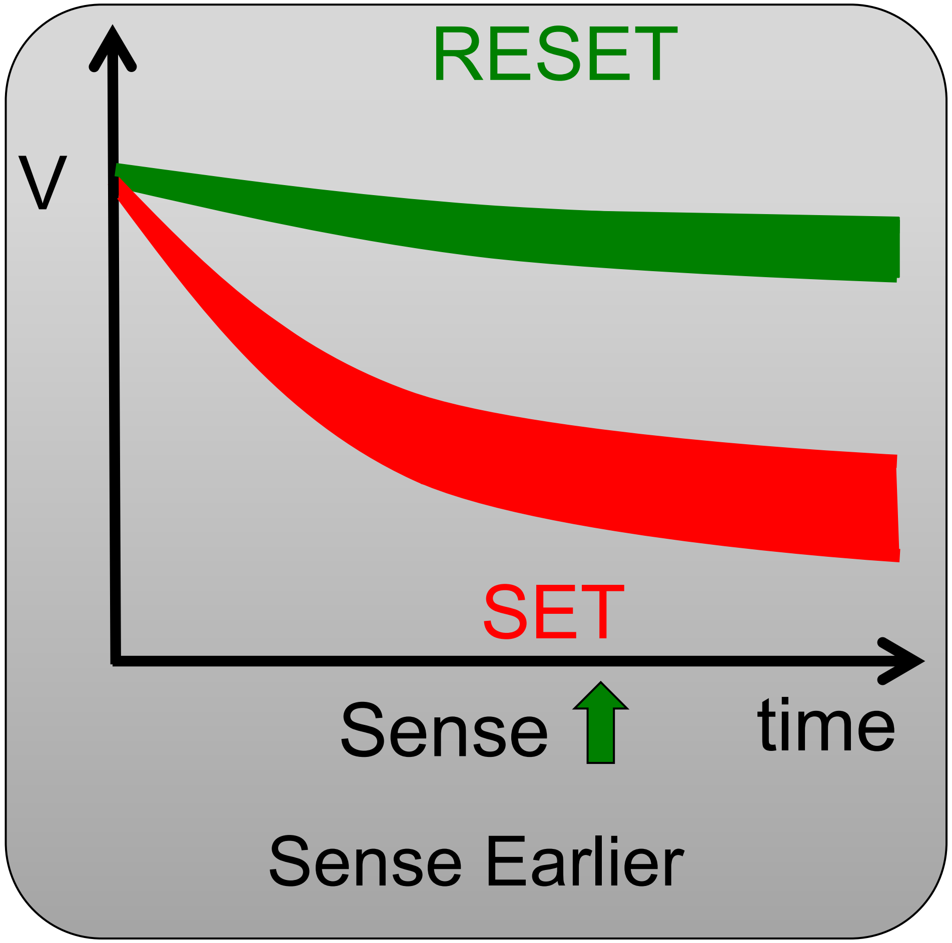
INSIGHT: USE RETRY FOR CORRECTION

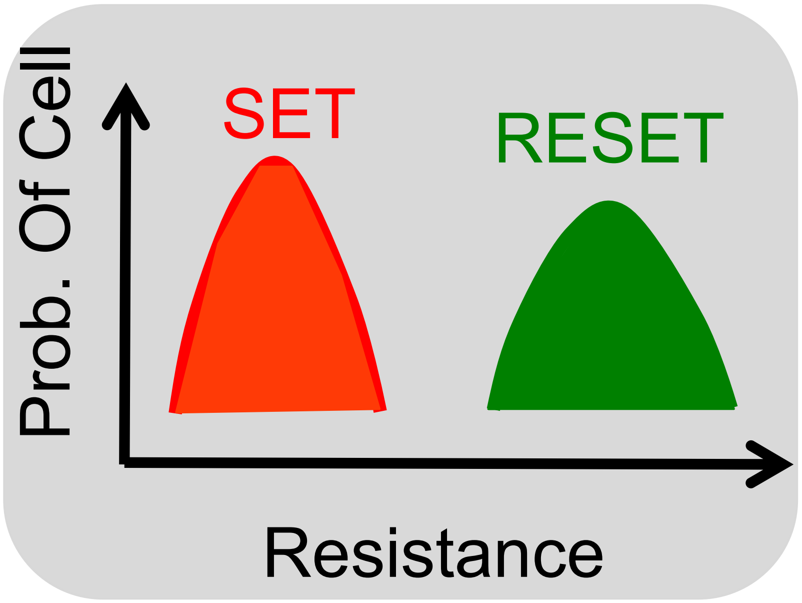
1. Sense Data Early
2. Read Line
3. ~~Correct errors~~
4. Detect errors
5. Retry with normal latency on error detection



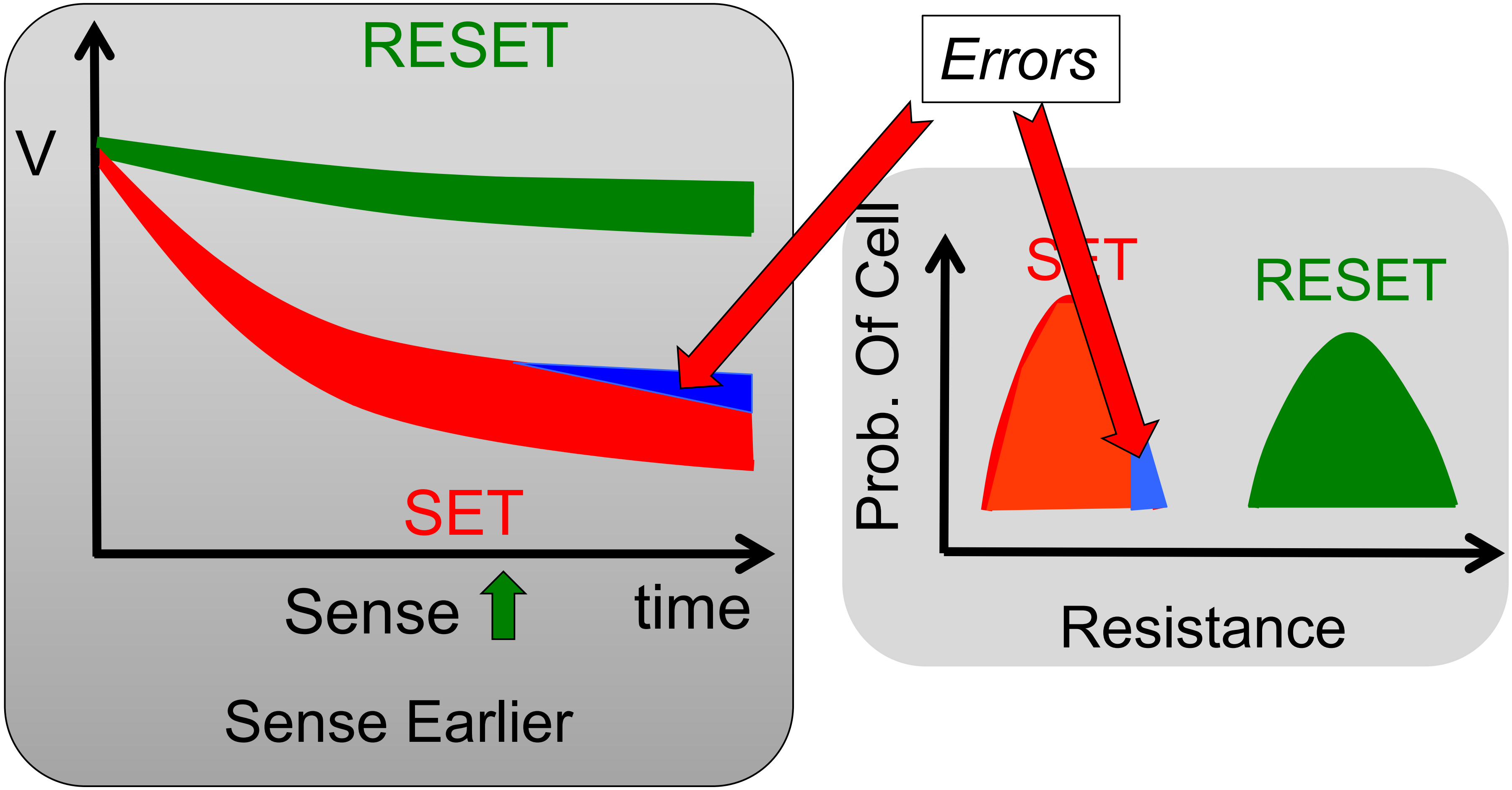
Early Read ➔ detect and retry to read correctly at lower latency

INSIGHT: ERRORS ARE UNIDIRECTIONAL

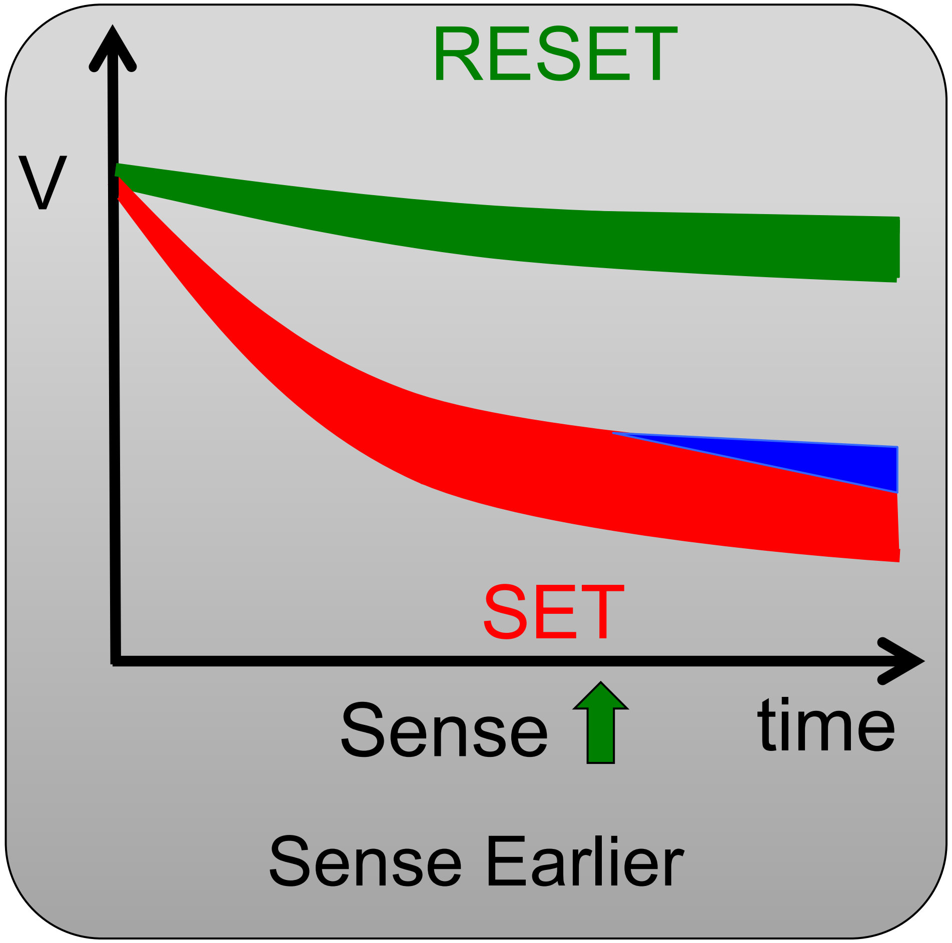


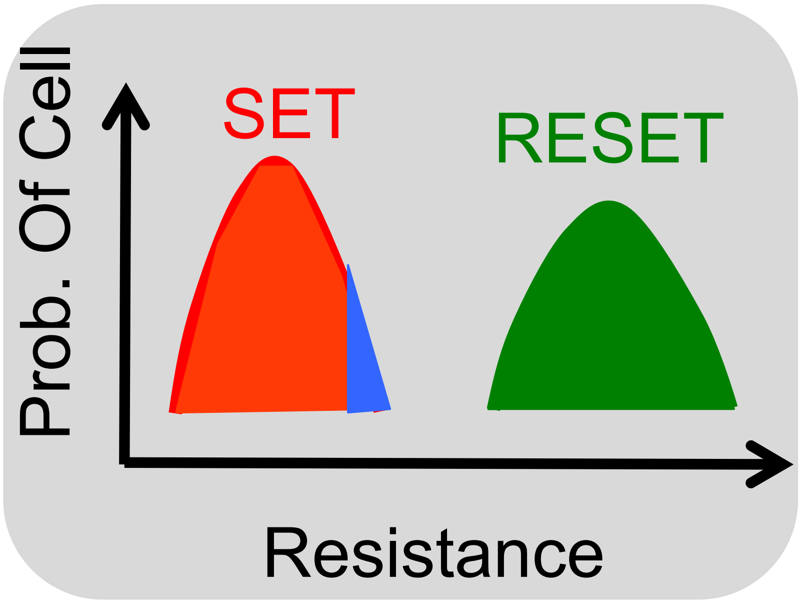


INSIGHT: ERRORS ARE UNIDIRECTIONAL

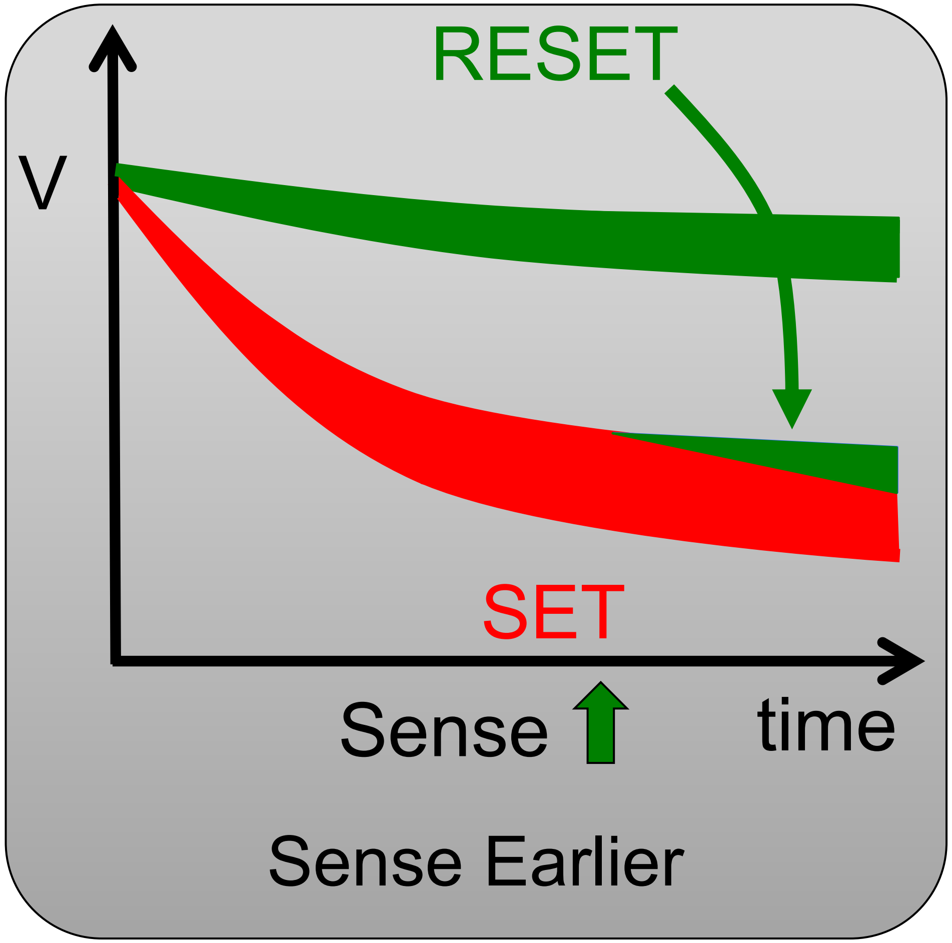


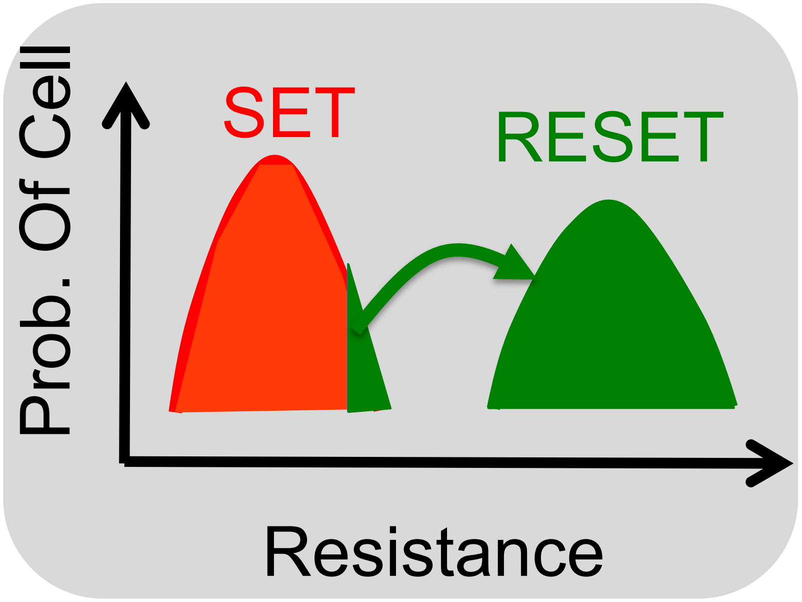
INSIGHT: ERRORS ARE UNIDIRECTIONAL





INSIGHT: ERRORS ARE UNIDIRECTIONAL

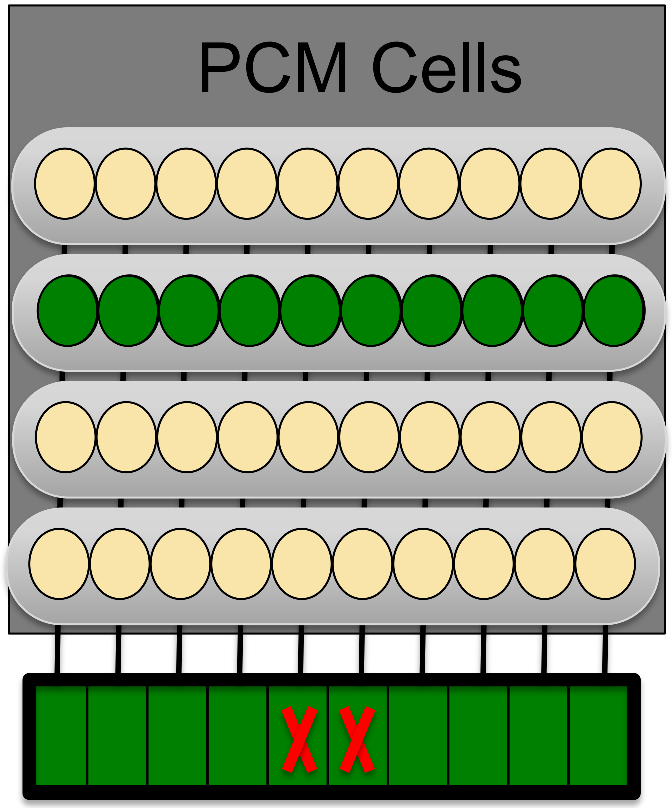




Sensing errors→Unidirectional→SET classified as RESET

UNIDIRECTIONAL ERROR DETECTION

- All unidirectional errors can be detected using Berger Code

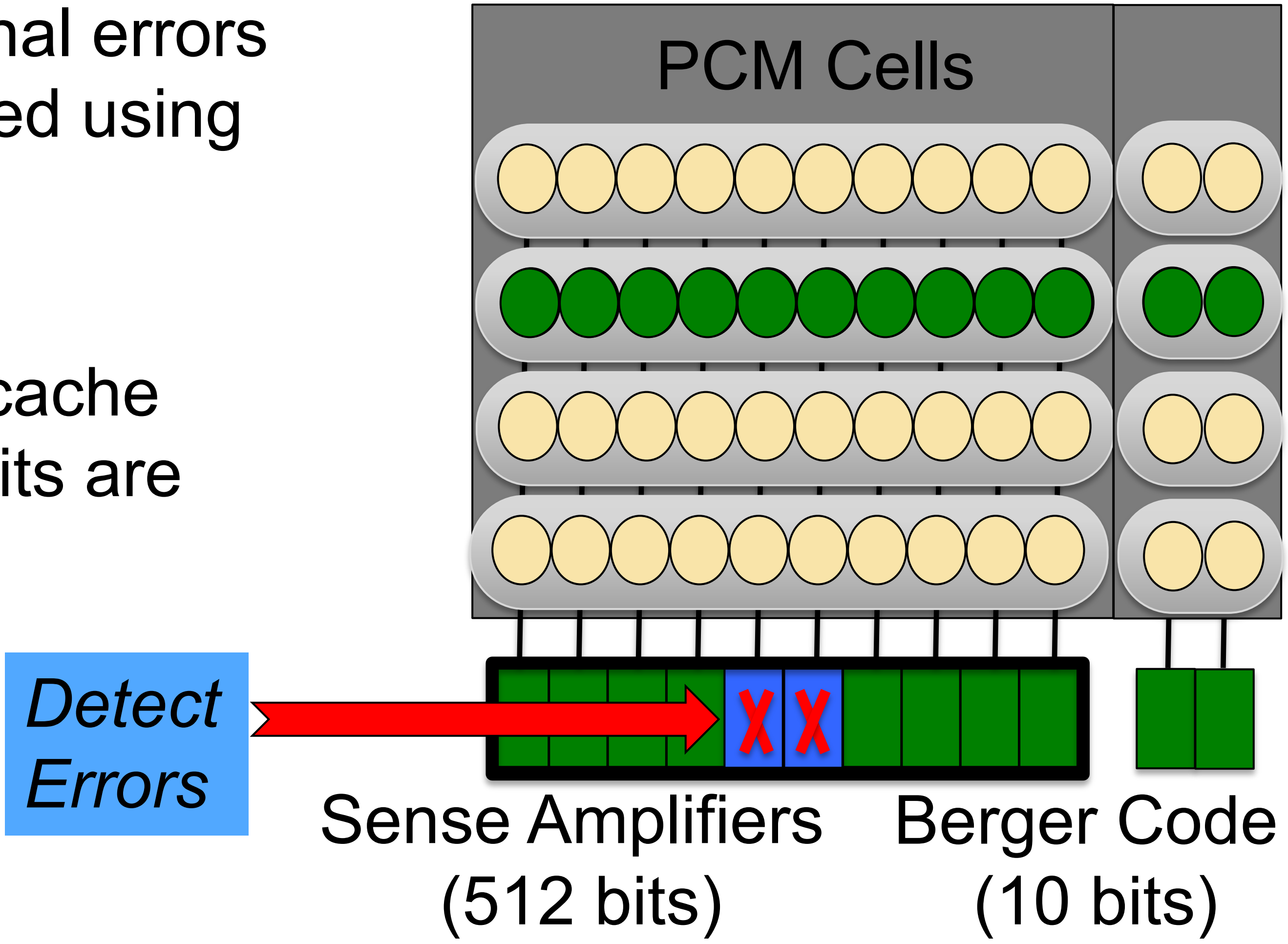


Sense Amplifiers
(512 bits)

UNIDIRECTIONAL ERROR DETECTION

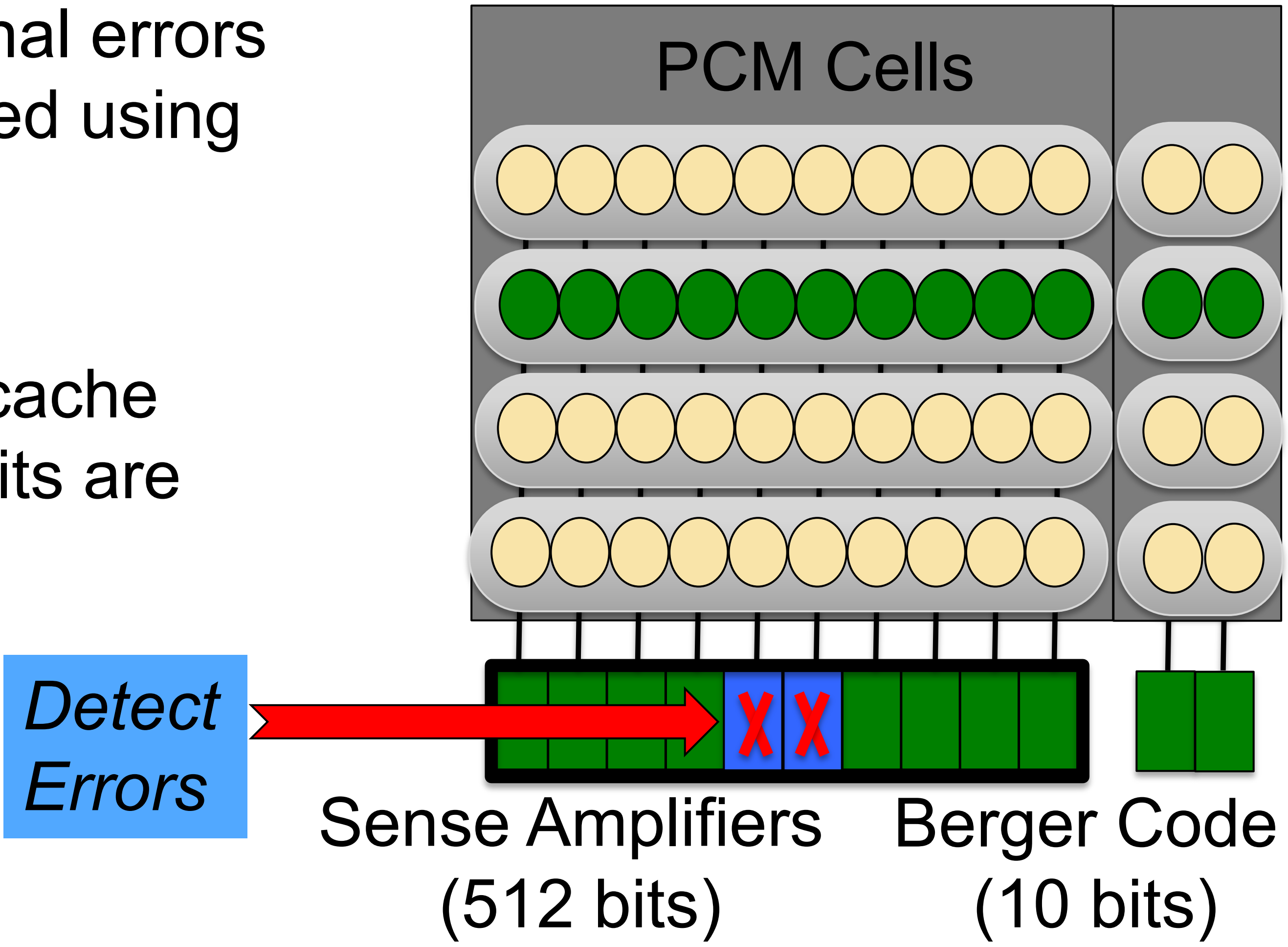
- All unidirectional errors can be detected using Berger Code

- For a 512 bit cache line, only 10 bits are needed



UNIDIRECTIONAL ERROR DETECTION

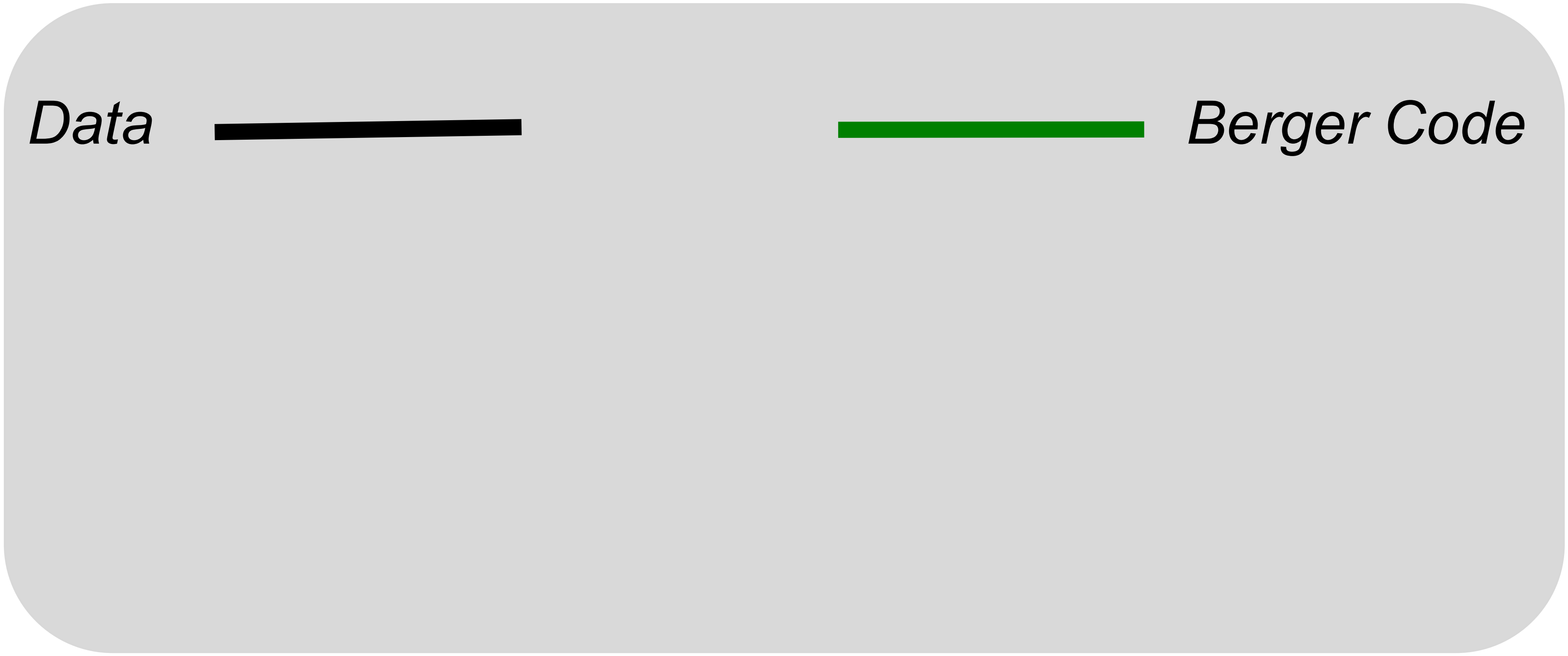
- All unidirectional errors can be detected using Berger Code
- For a 512 bit cache line, only 10 bits are needed



Berger Code detects unidirectional errors with low cost

BERGER CODES: HOW AND WHY

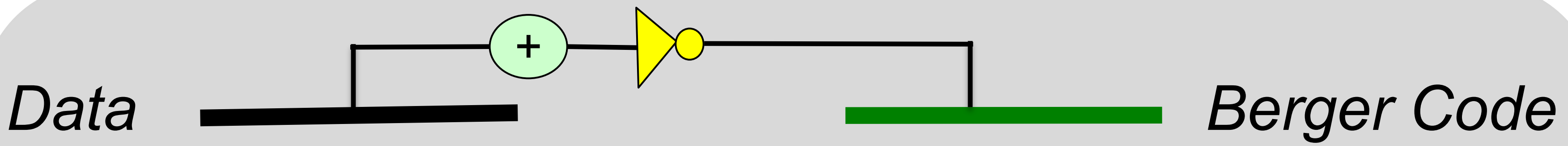
Sum the number of 1’s in data, invert and store



Berger code provides guaranteed detection of all unidirectional errors

BERGER CODES: HOW AND WHY

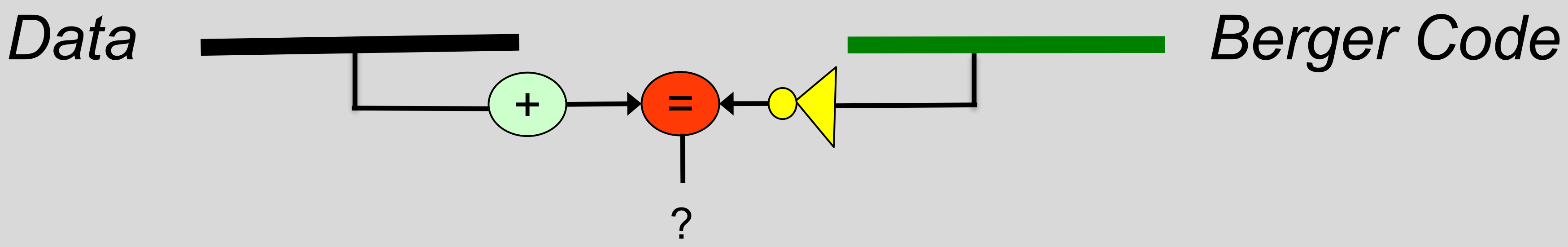
Sum the number of 1's in data, invert and store



Berger code provides guaranteed detection of all unidirectional errors

BERGER CODES: HOW AND WHY

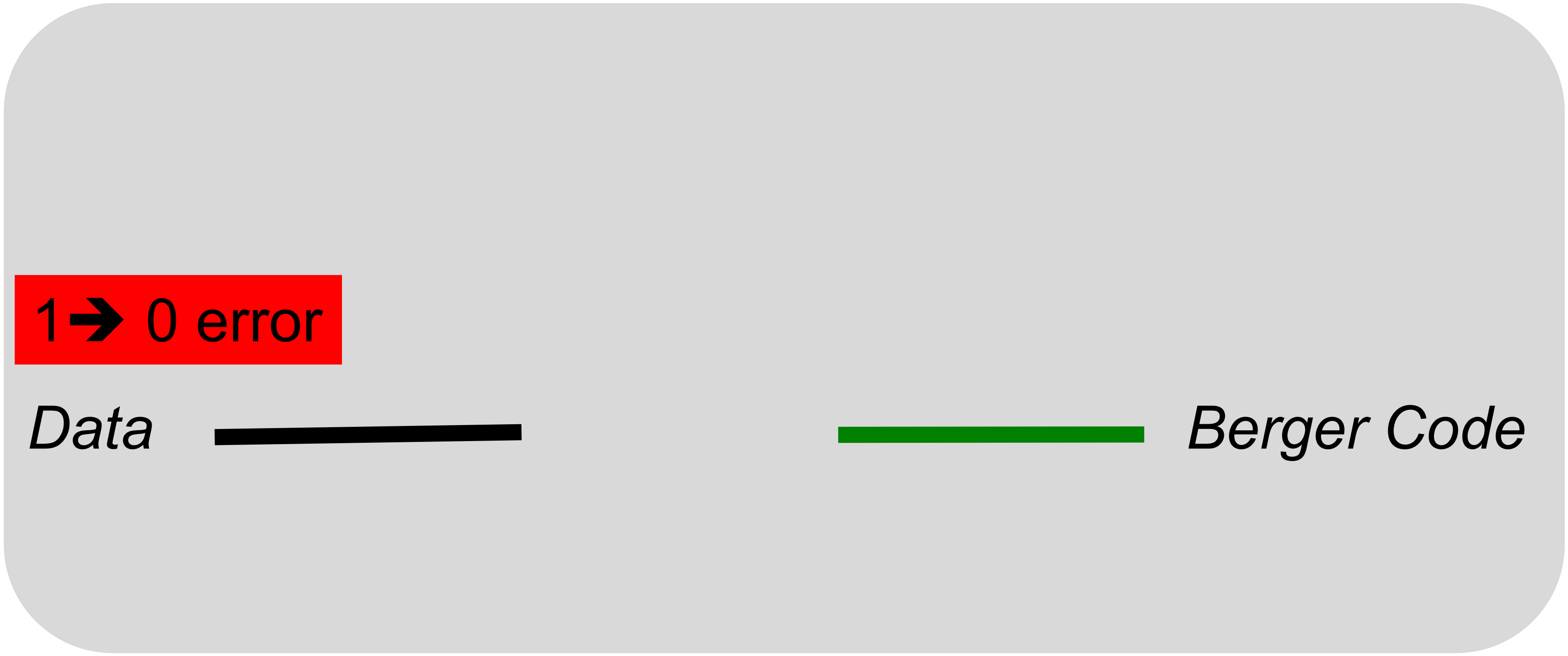
Sum the number of 1's in data, invert and store



Berger code provides guaranteed detection of all unidirectional errors

BERGER CODES: HOW AND WHY

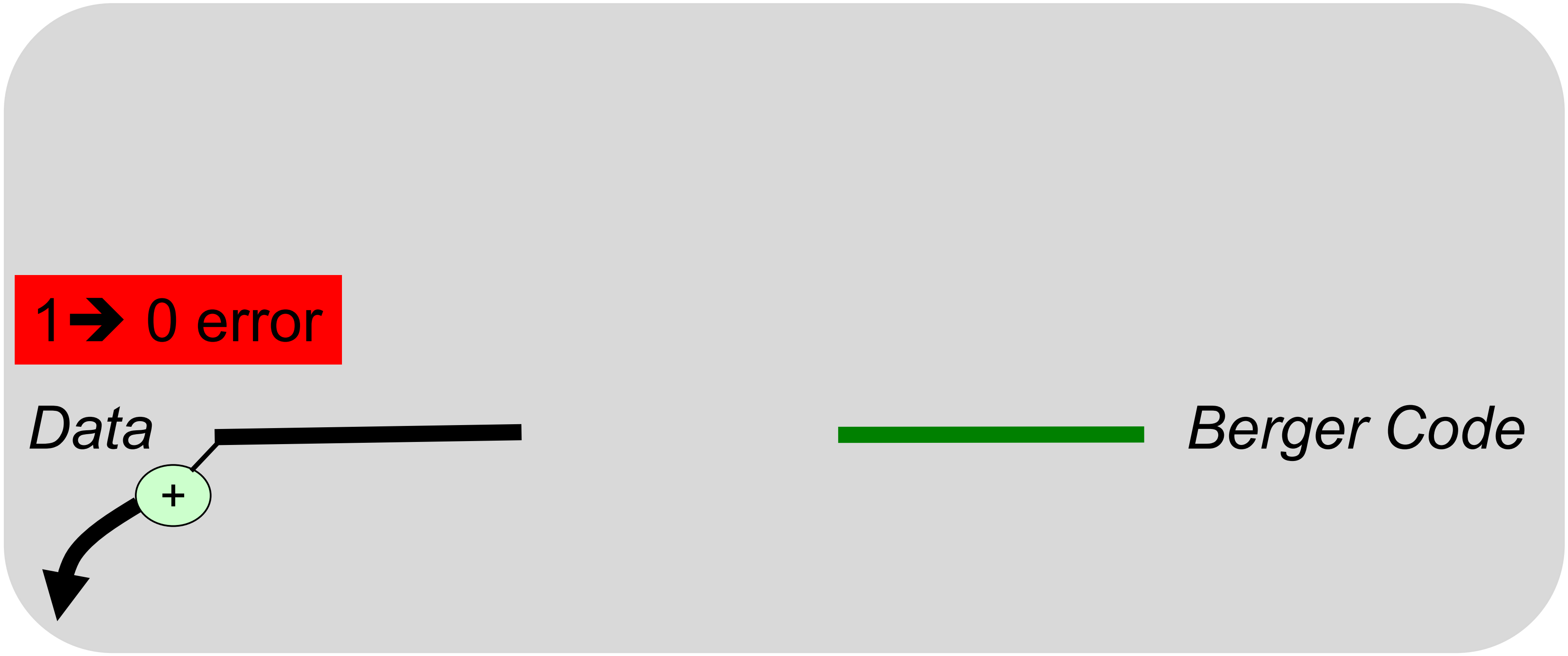
Sum the number of 1's in data, invert and store



Berger code provides guaranteed detection of all unidirectional errors

BERGER CODES: HOW AND WHY

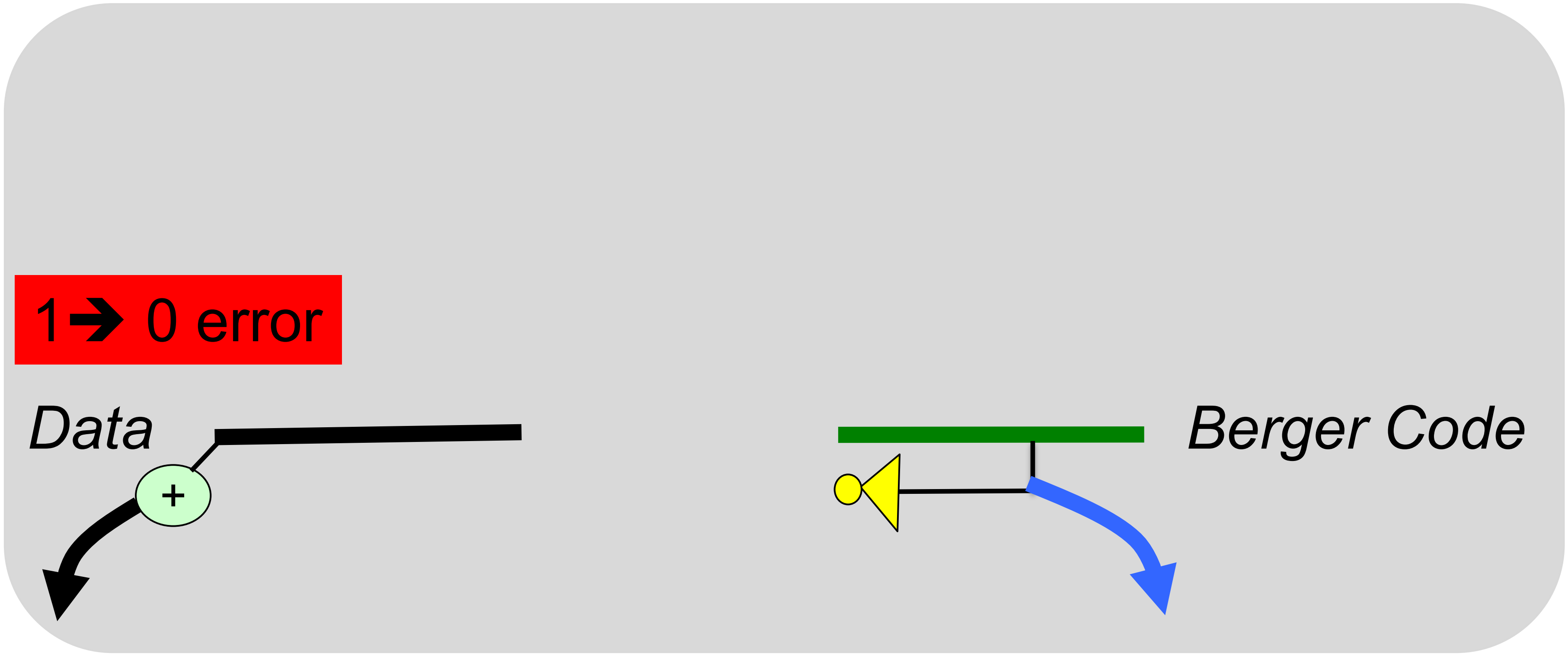
Sum the number of 1's in data, invert and store



Berger code provides guaranteed detection of all unidirectional errors

BERGER CODES: HOW AND WHY

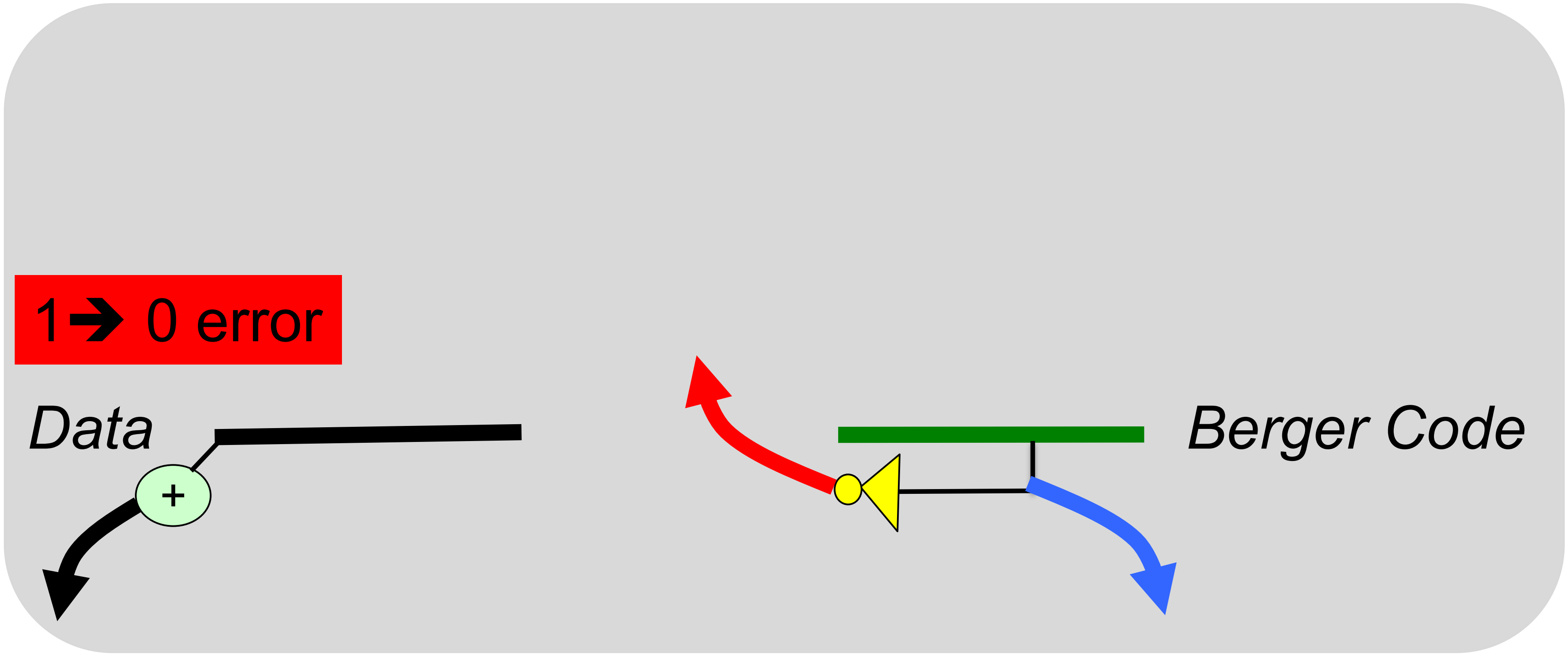
Sum the number of 1's in data, invert and store



Berger code provides guaranteed detection of all unidirectional errors

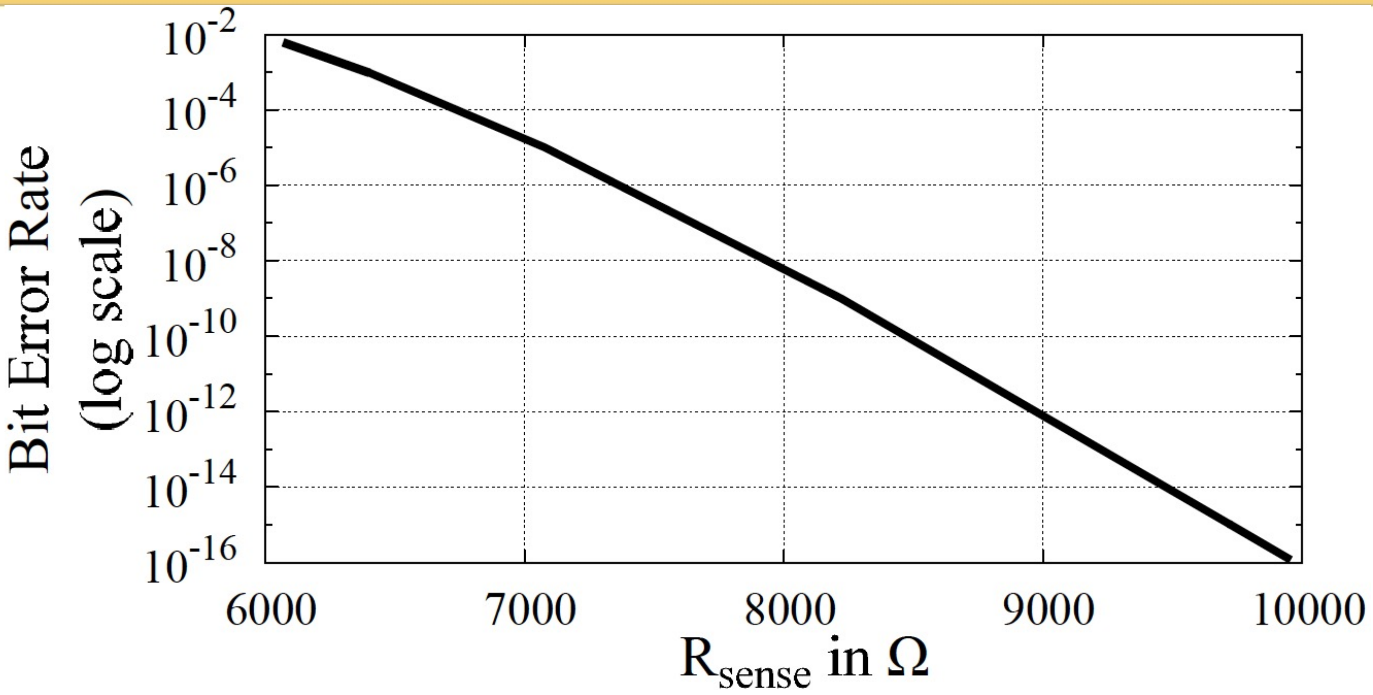
BERGER CODES: HOW AND WHY

Sum the number of 1's in data, invert and store

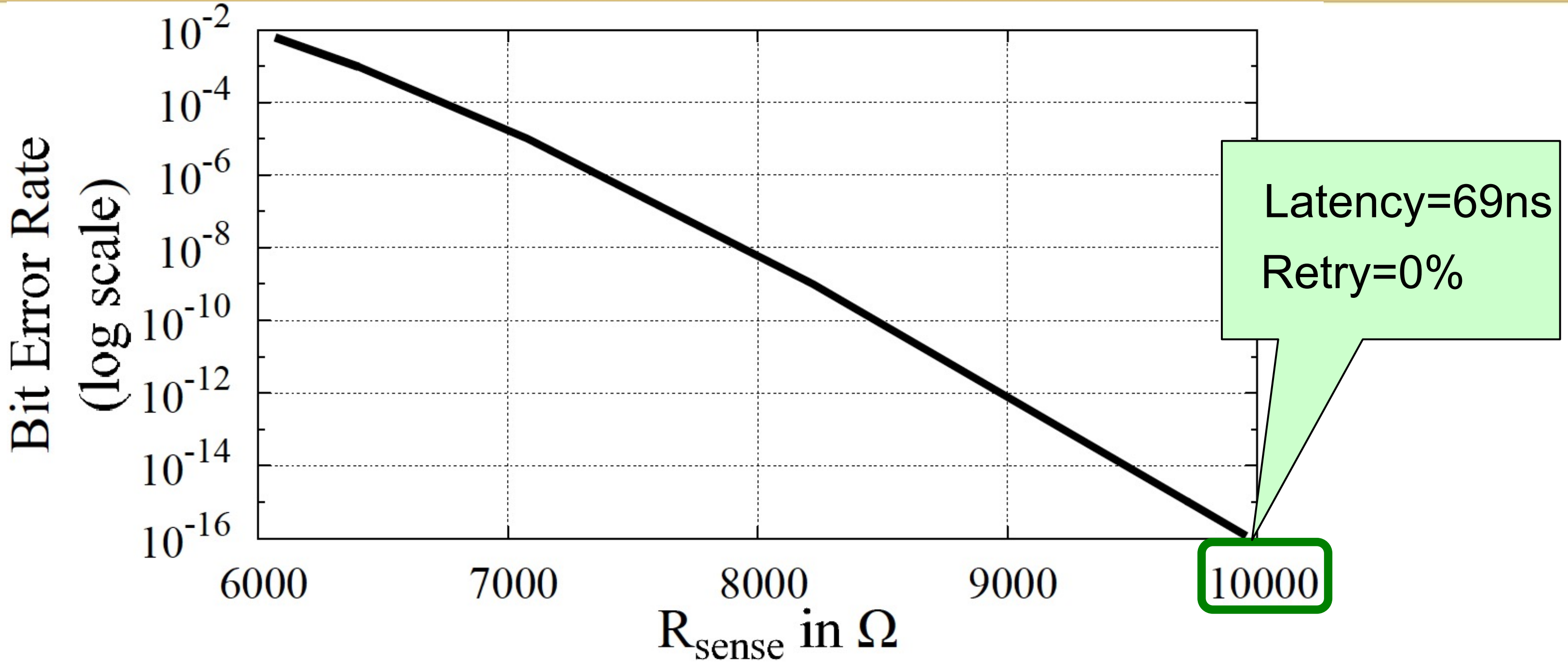


Berger code provides guaranteed detection of all unidirectional errors

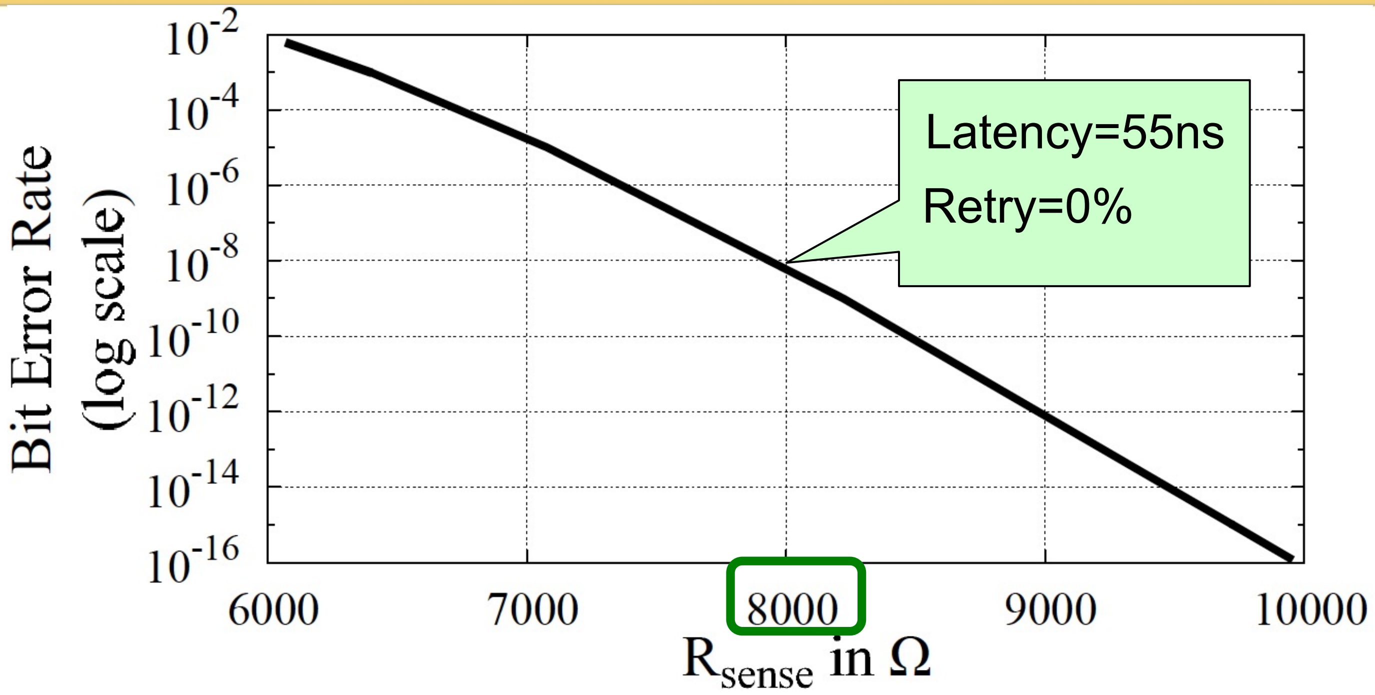
EARLY READ: DESIGN



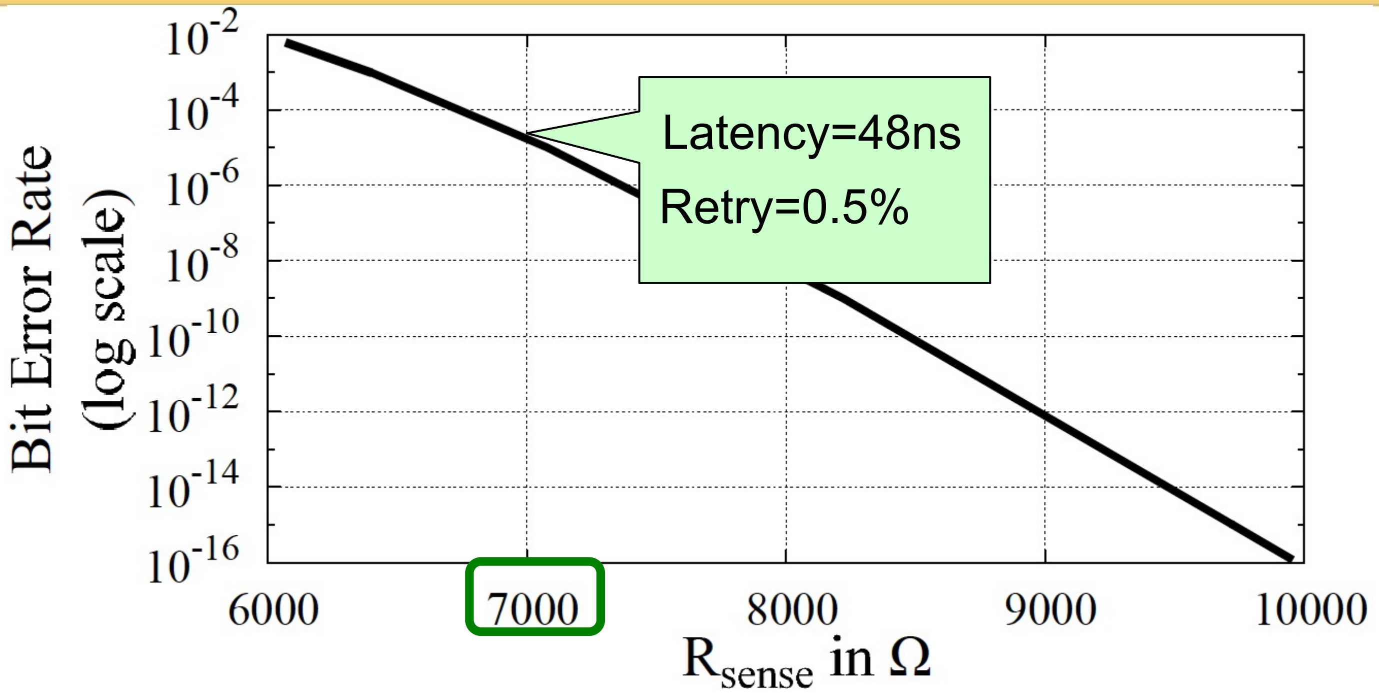
EARLY READ: DESIGN



EARLY READ: DESIGN

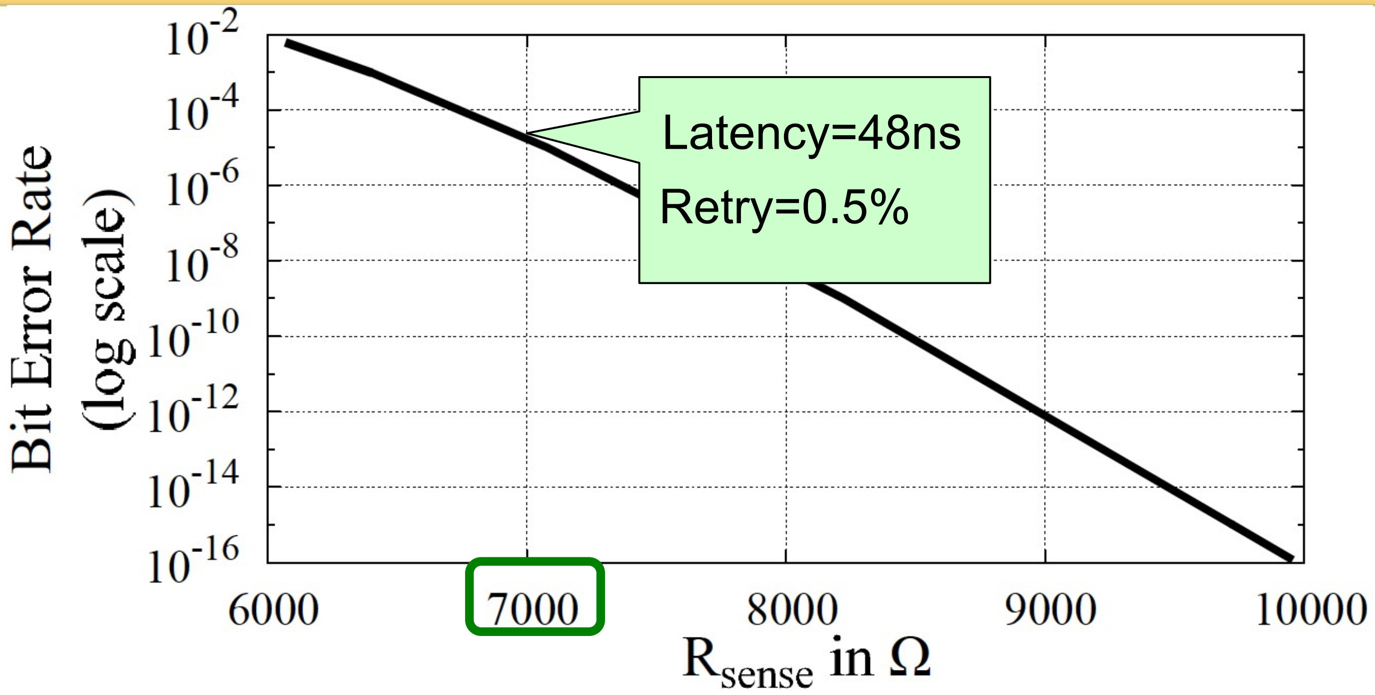


EARLY READ: DESIGN



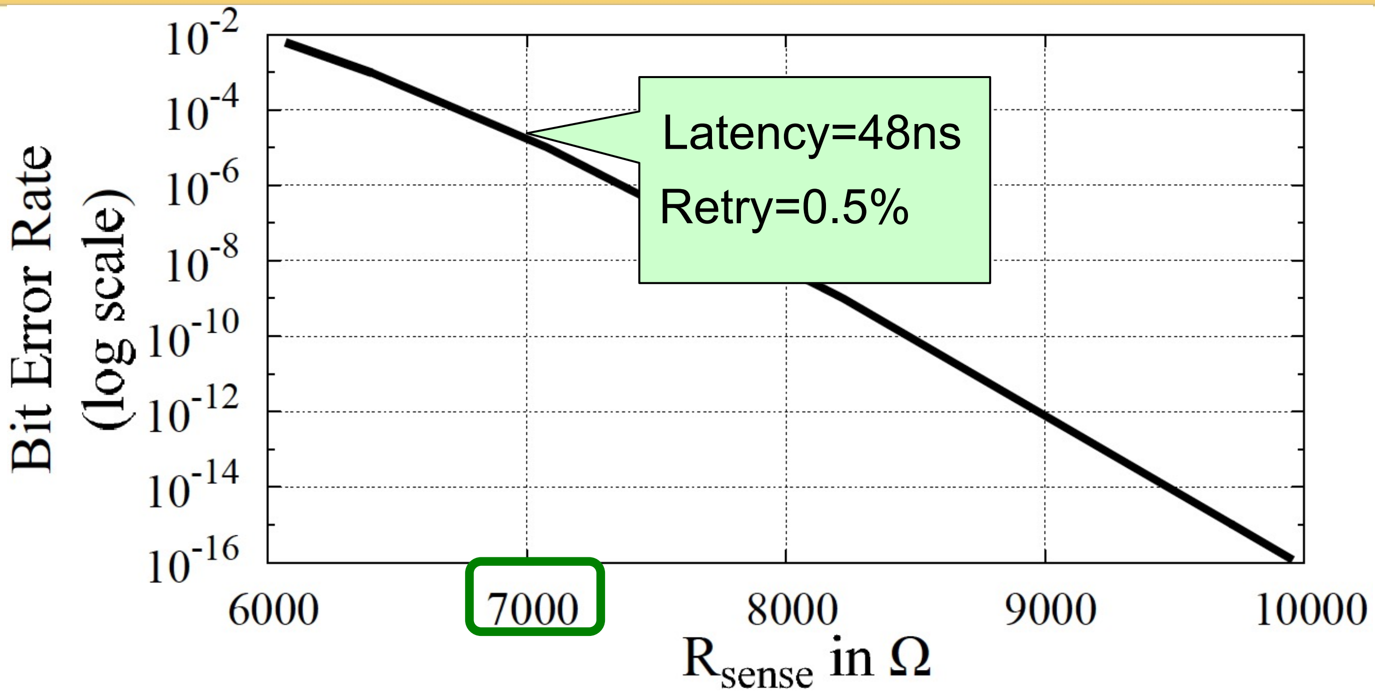
- Early Read reduces R_{sense} from 10KΩ to 7KΩ

EARLY READ: DESIGN



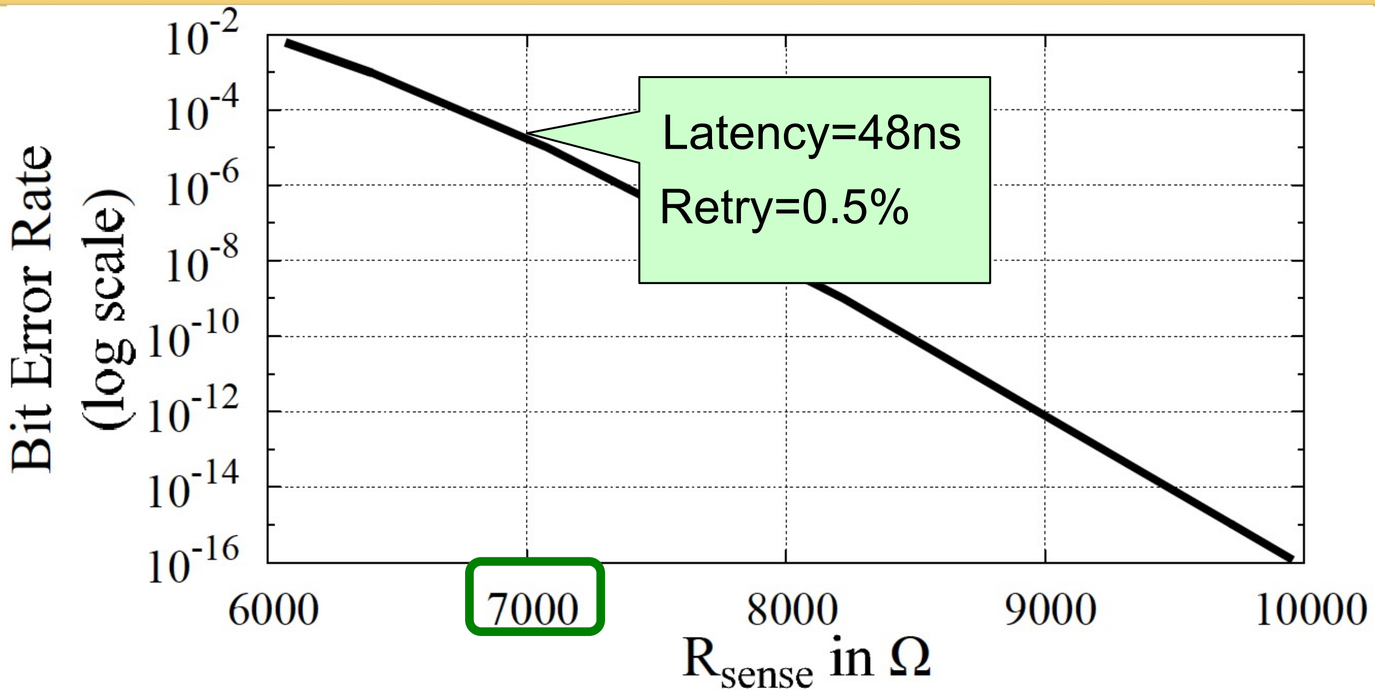
- Early Read reduces R_{sense} from 10KΩ to 7KΩ
- The BER increases from 10^{-16} to 10^{-5}

EARLY READ: DESIGN



- Early Read reduces R_{sense} from 10KΩ to 7KΩ
- The BER increases from 10^{-16} to 10^{-5}
- Detect using Berger Code, retrying 0.5% times

EARLY READ: DESIGN



- Early Read reduces R_{sense} from 10KΩ to 7KΩ
- The BER increases from 10^{-16} to 10^{-5}
- Detect using Berger Code, retrying 0.5% times

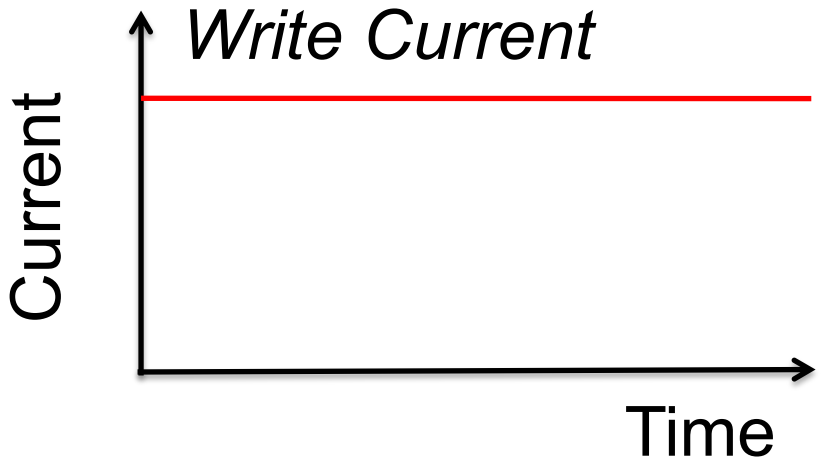
25% reduction in read latency using Early Read

OUTLINE

- Introduction and Background
- Early Read
- Turbo Read
- Early+Turbo Read
- Results
- Summary

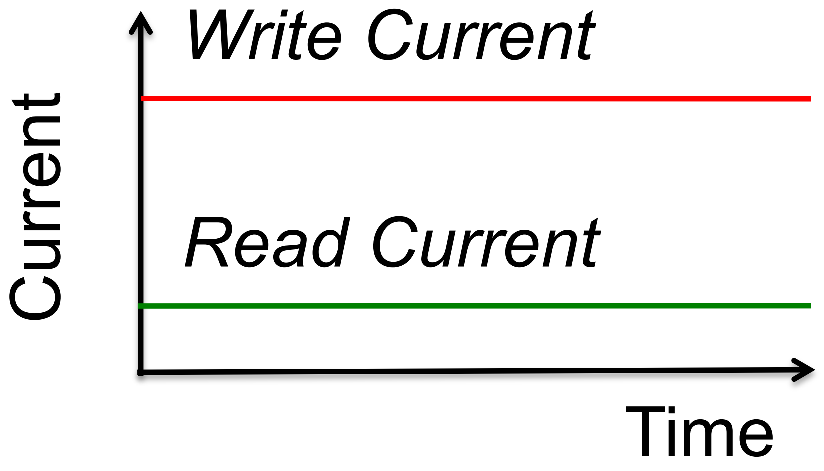
READING WITH HIGHER VOLTAGE

- PCM writes data by passing current through cell



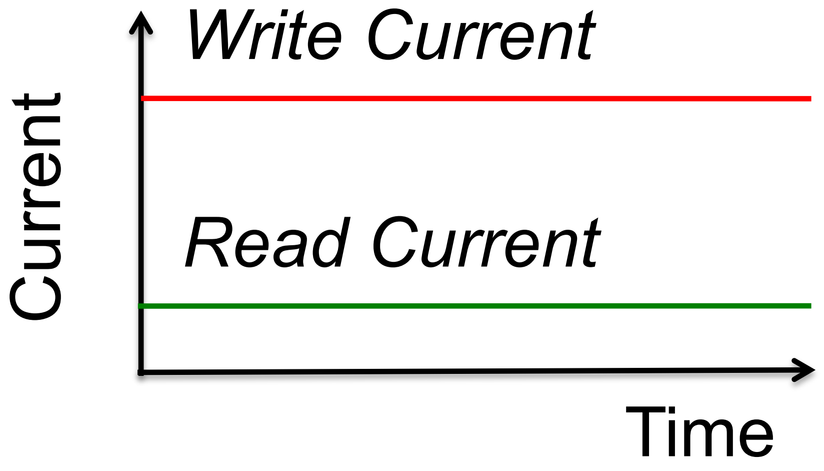
READING WITH HIGHER VOLTAGE

- PCM writes data by passing current through cell
- PCM reads data by passing current through cell



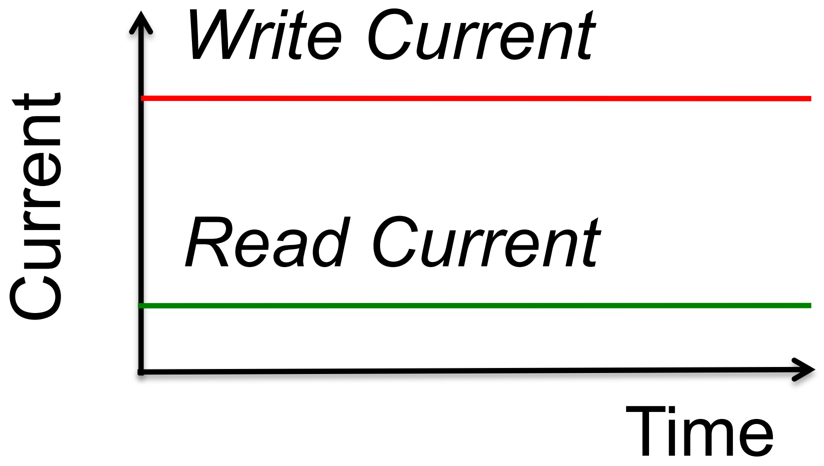
READING WITH HIGHER VOLTAGE

- PCM writes data by passing current through cell
- PCM reads data by passing current through cell
  - – Read current << Write current



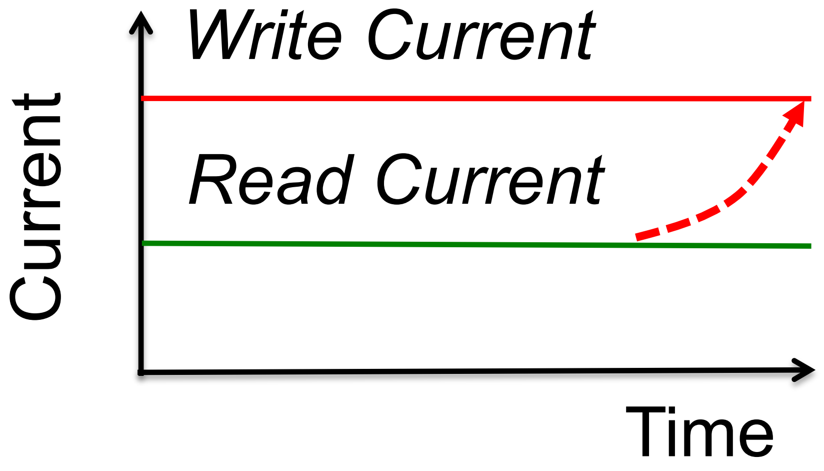
READING WITH HIGHER VOLTAGE

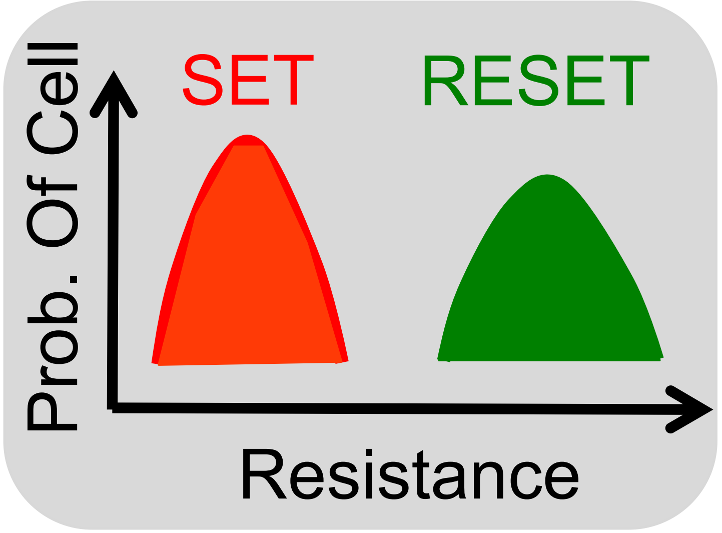
- PCM writes data by passing current through cell
- PCM reads data by passing current through cell
  - – Read current << Write current
- Higher read current can reduce read latency



READING WITH HIGHER VOLTAGE

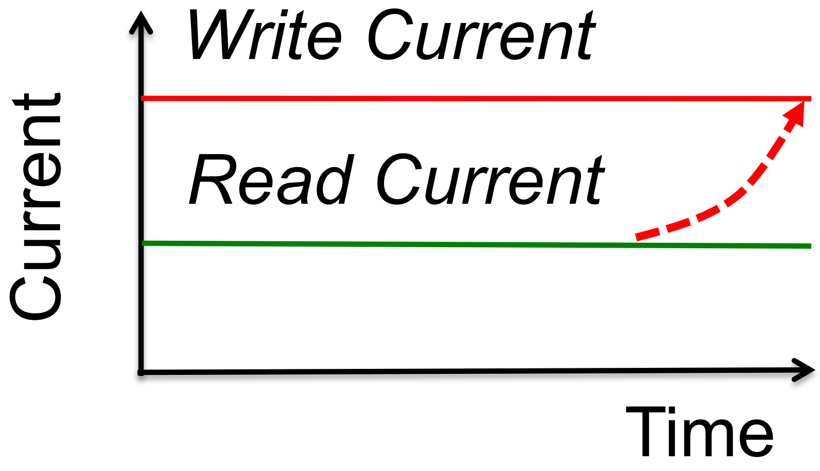
- PCM writes data by passing current through cell
- PCM reads data by passing current through cell
  - Read current << Write current
- Higher read current can reduce read latency
- **Read Disturb ➔** Causes PCM cells to accidently flip

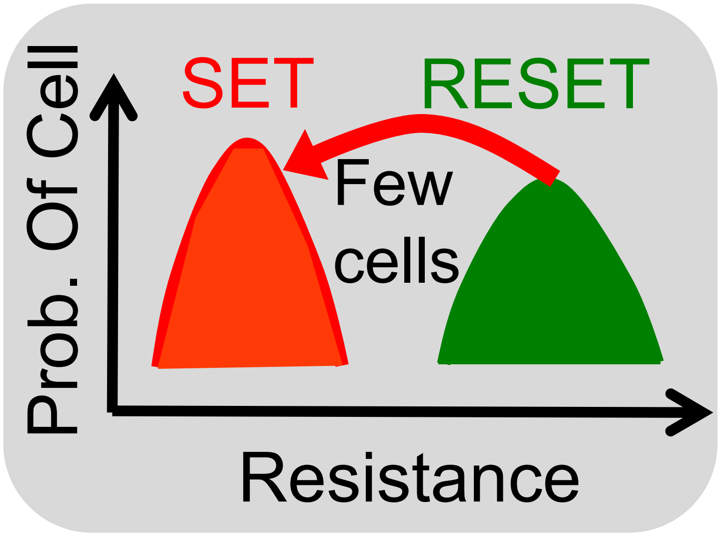




READING WITH HIGHER VOLTAGE

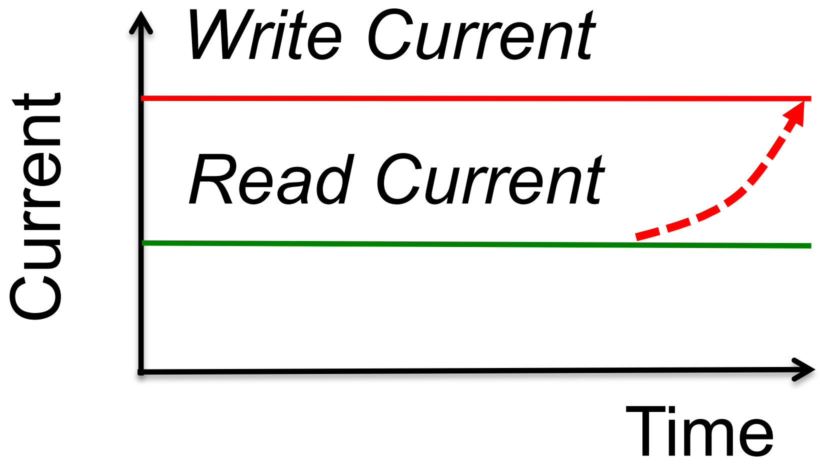
- PCM writes data by passing current through cell
- PCM reads data by passing current through cell
  - Read current << Write current
- Higher read current can reduce read latency
- **Read Disturb ➔** Causes PCM cells to accidently flip

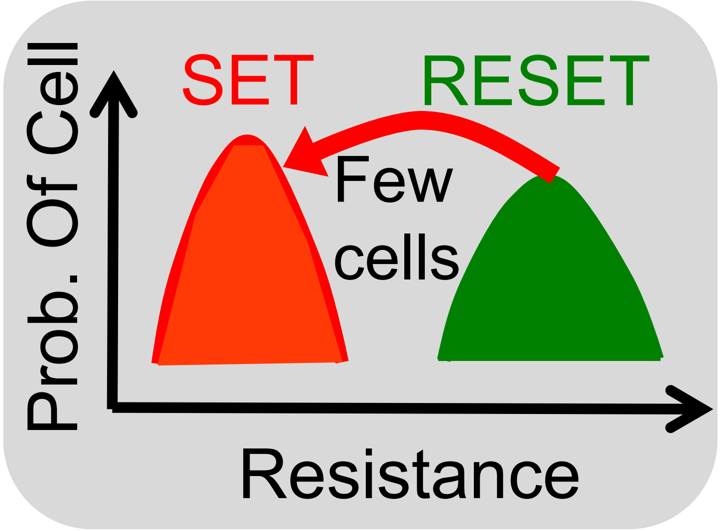




READING WITH HIGHER VOLTAGE

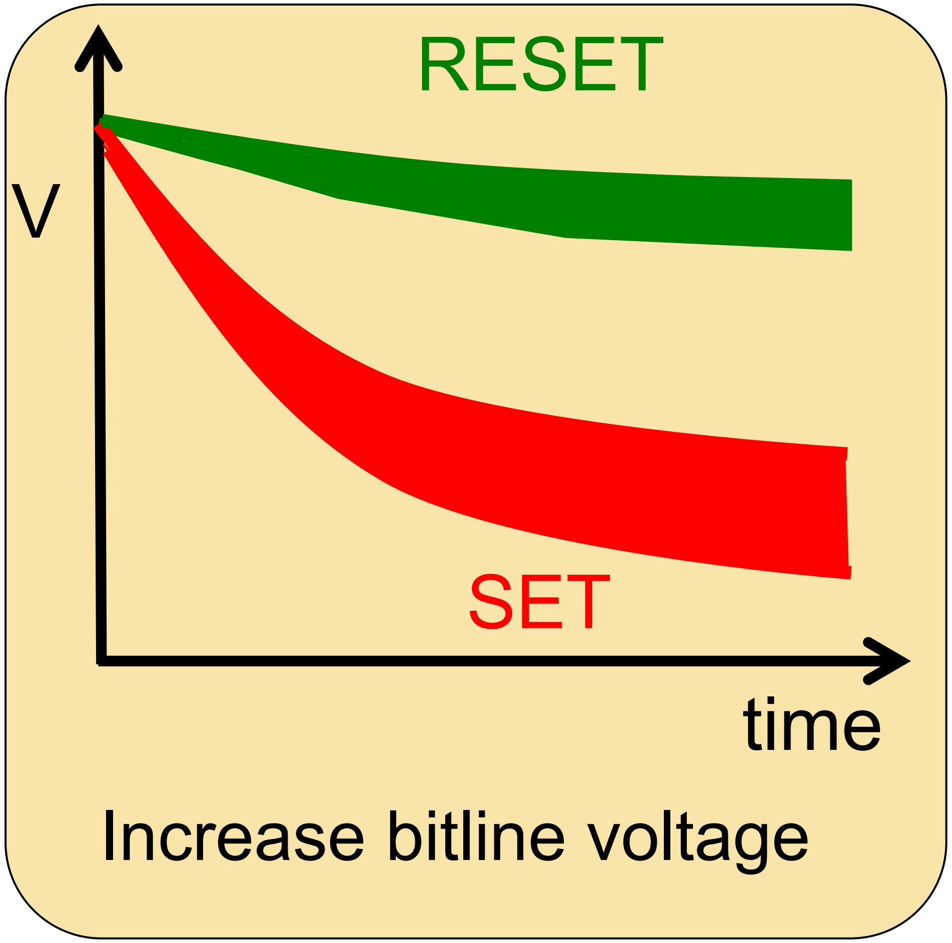
- PCM writes data by passing current through cell
- PCM reads data by passing current through cell
  - Read current << Write current
- Higher read current can reduce read latency
- **Read Disturb ➔** Causes PCM cells to accidently flip



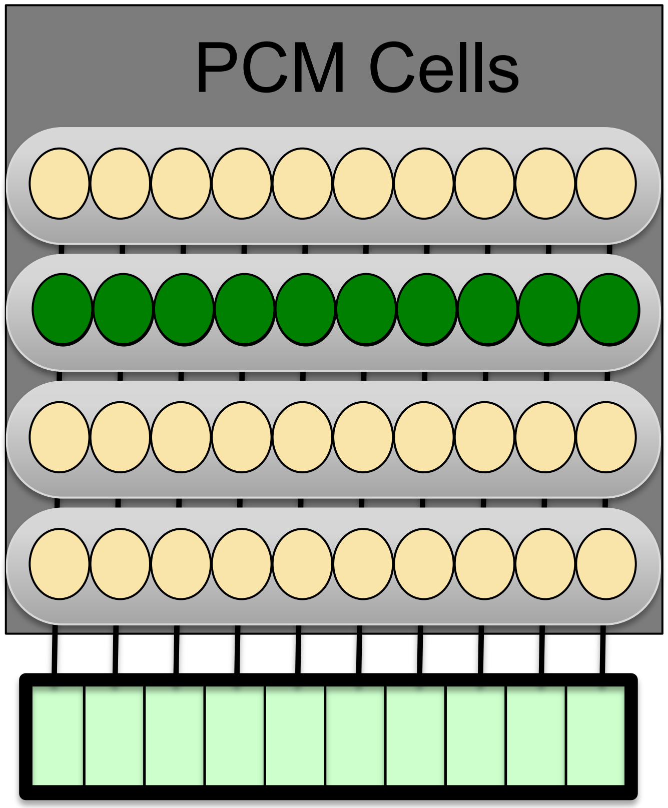


Higher bitline voltage causes Read Disturb

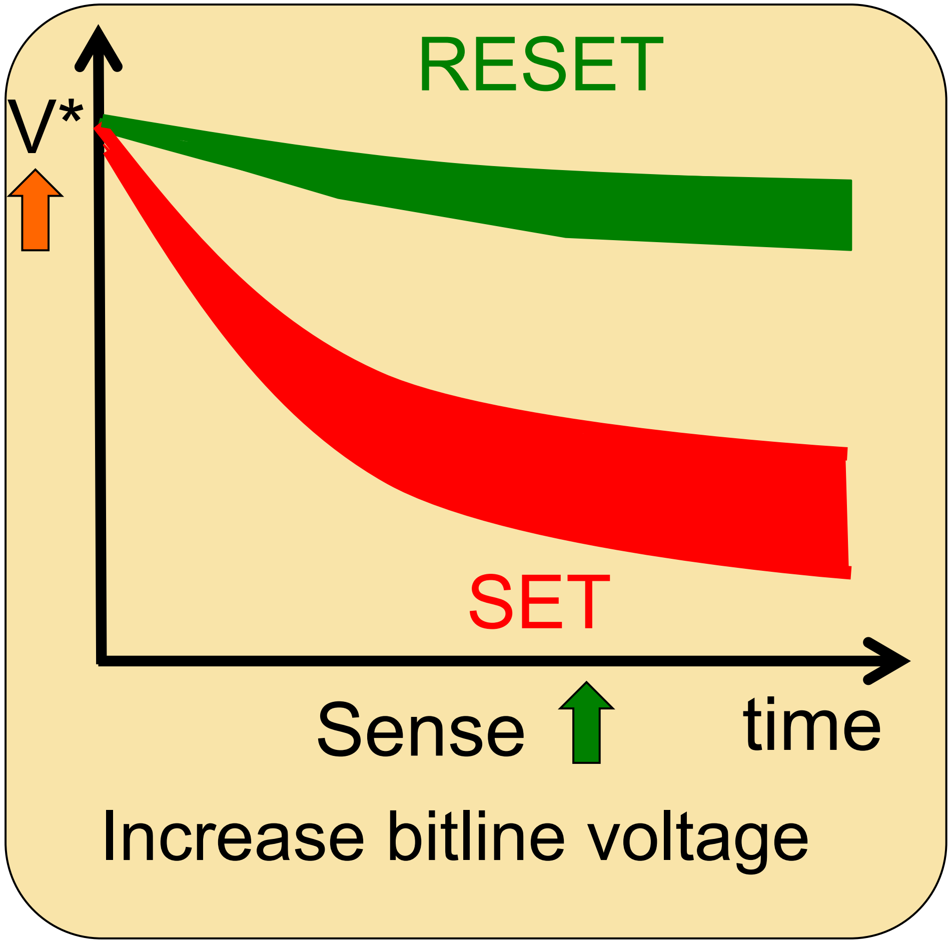
READ DISTURB: OBSERVATION

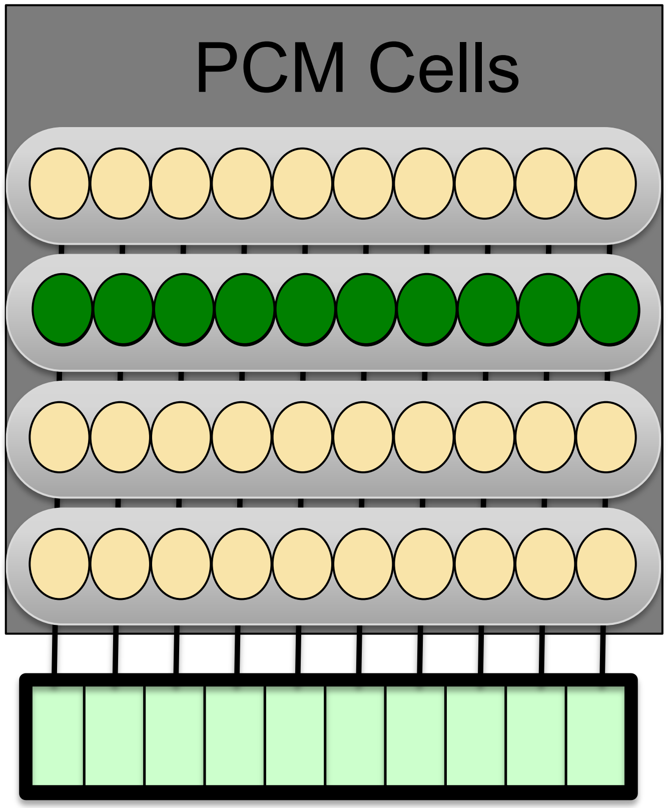


Increase bitline voltage

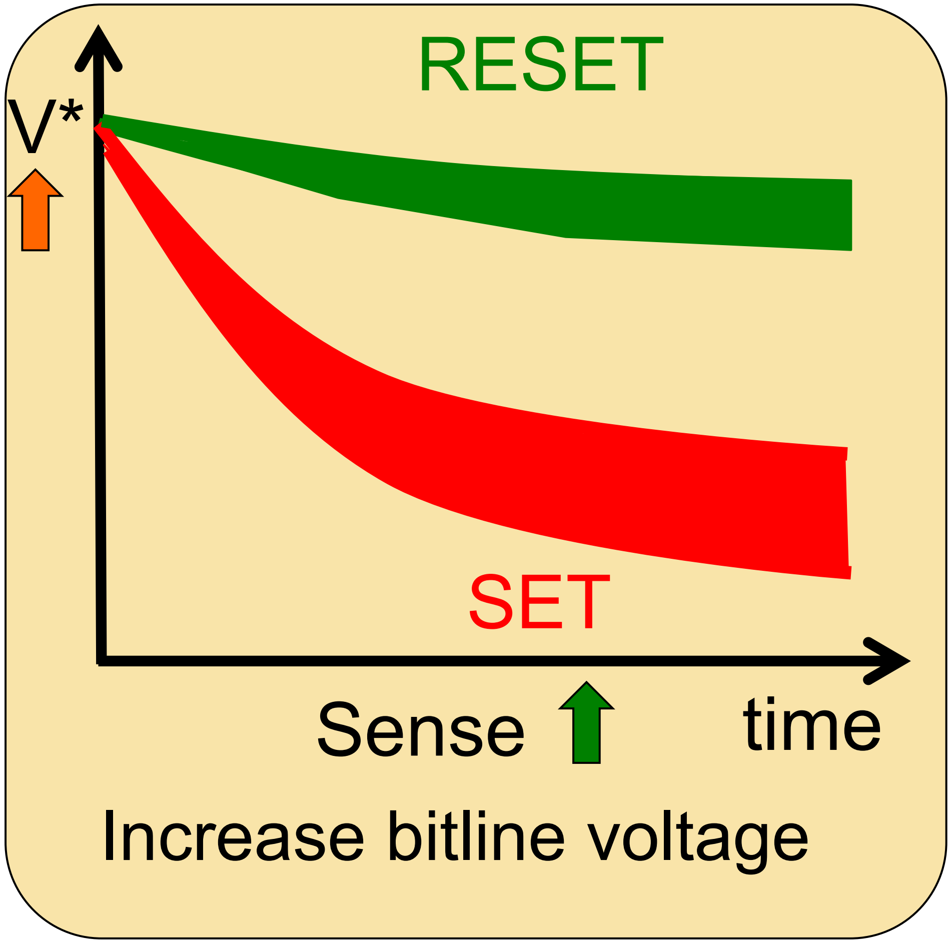


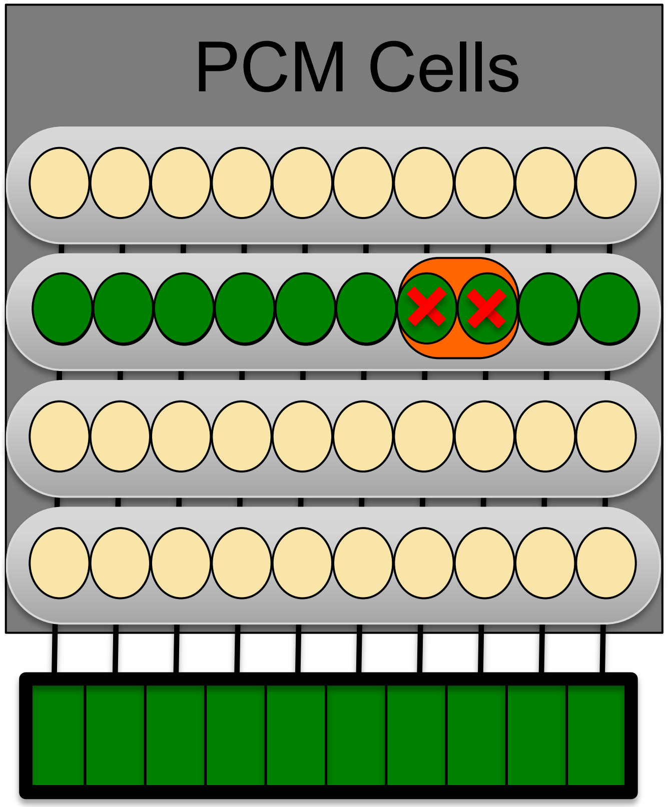
READ DISTURB: OBSERVATION



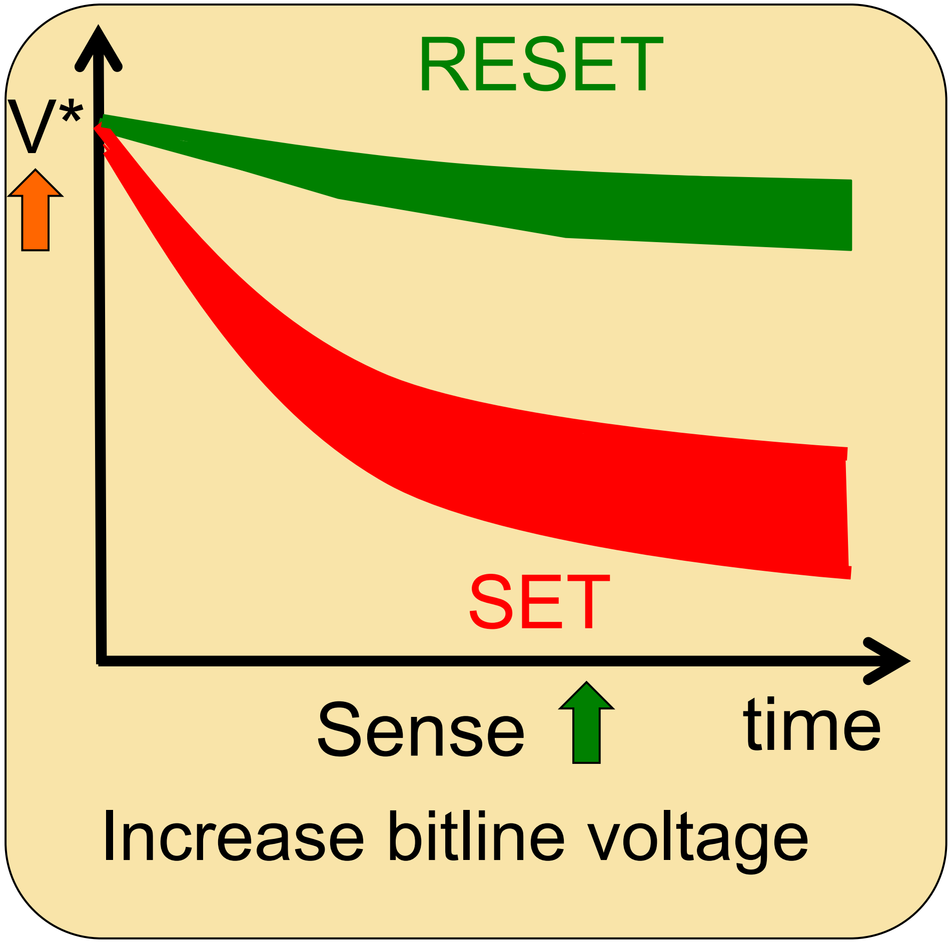


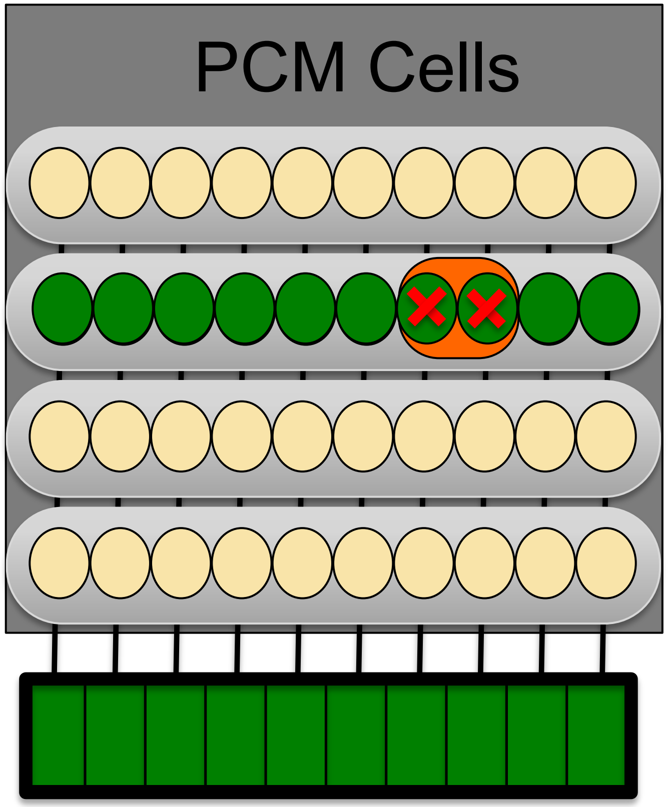
READ DISTURB: OBSERVATION





READ DISTURB: OBSERVATION

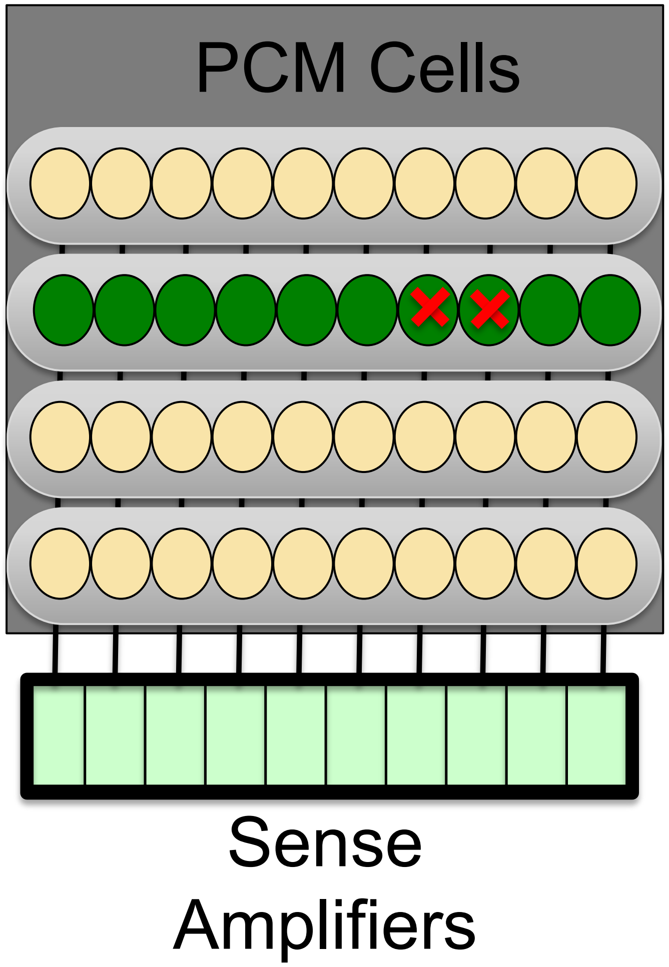




Reading with higher voltage → Read Disturb → causes errors in PCM cells

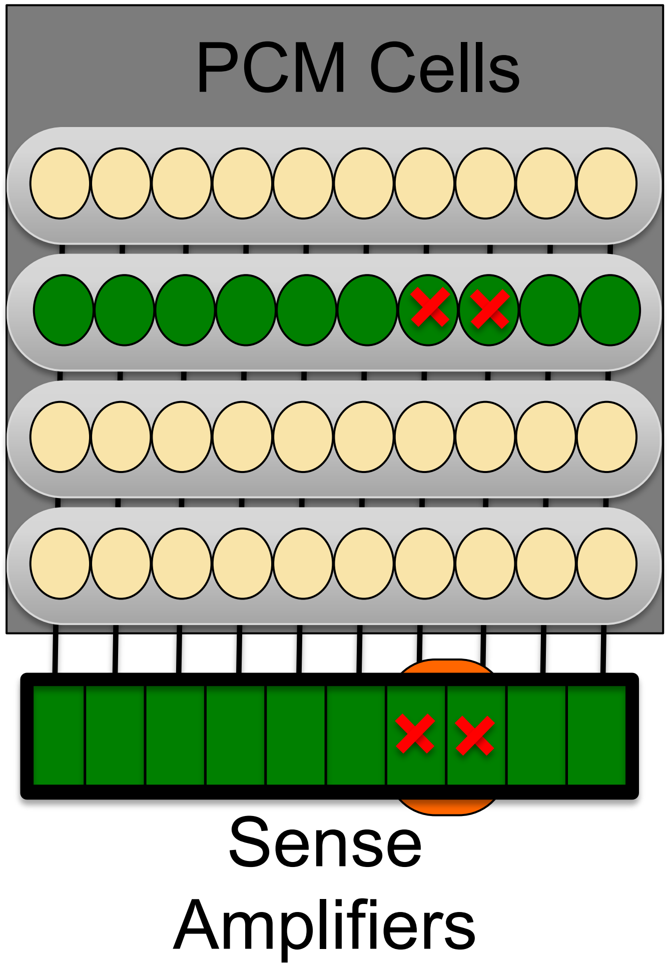
ECC FOR READ DISTURB ERRORS

- Incorrect value may be read



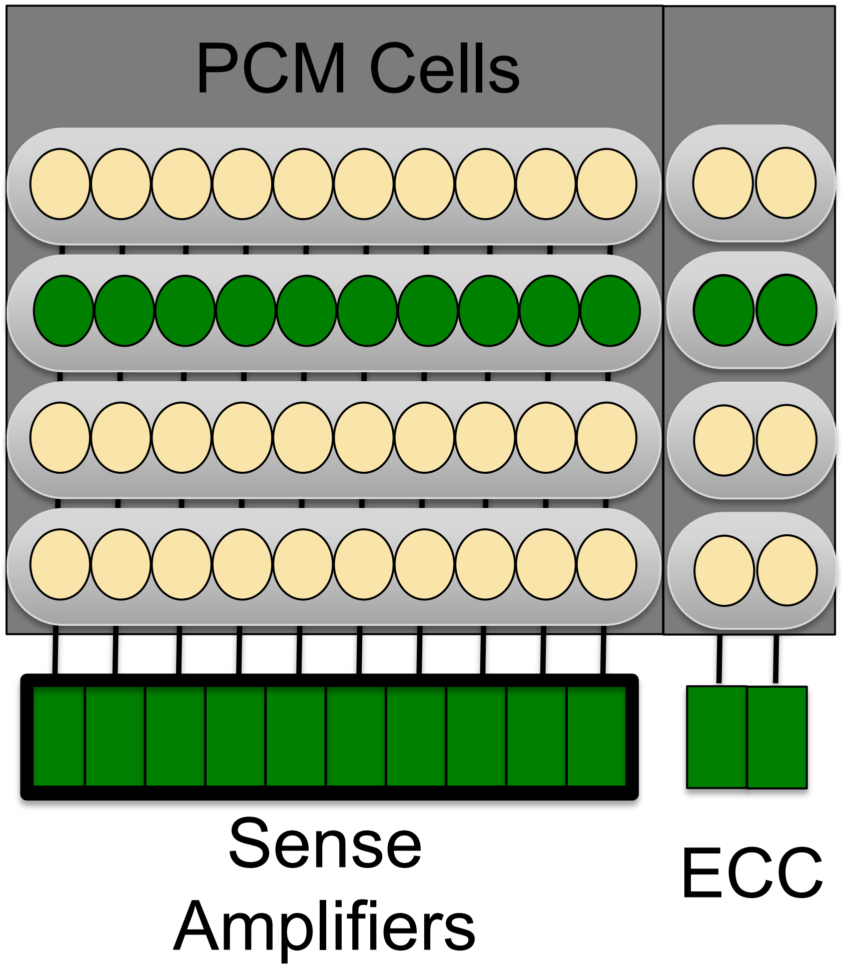
ECC FOR READ DISTURB ERRORS

- Incorrect value may be read



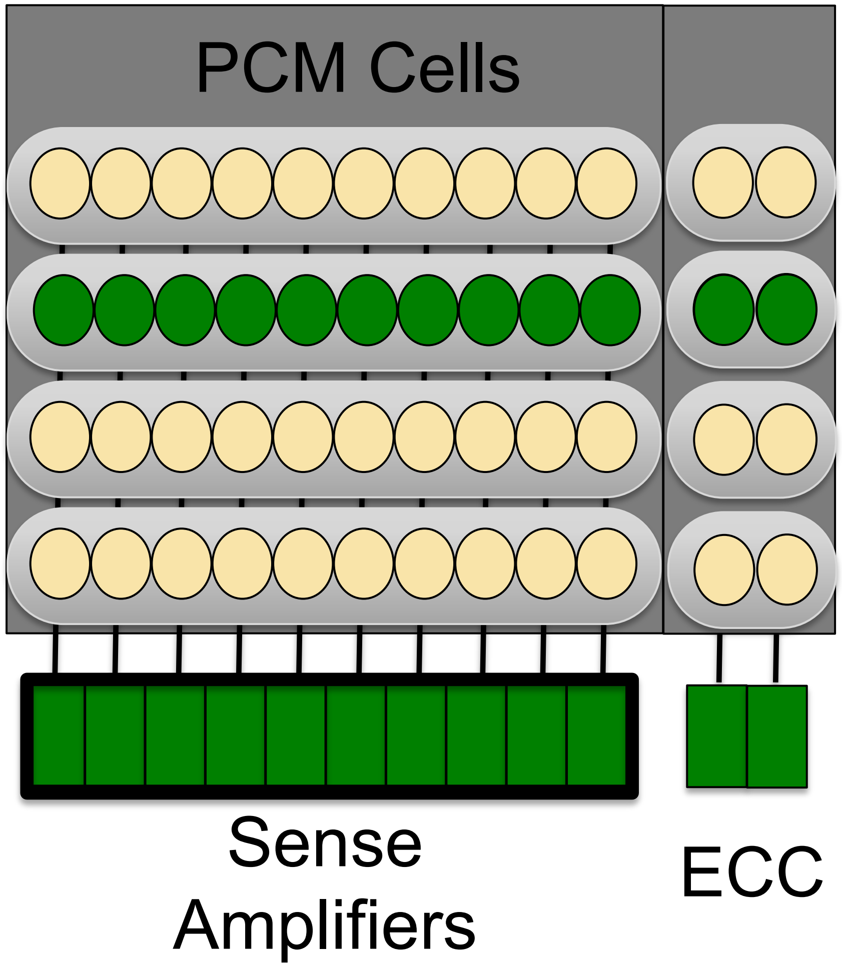
ECC FOR READ DISTURB ERRORS

- Incorrect value may be read
- Read disturb errors can be corrected with Error Correcting Codes (ECC)



ECC FOR READ DISTURB ERRORS

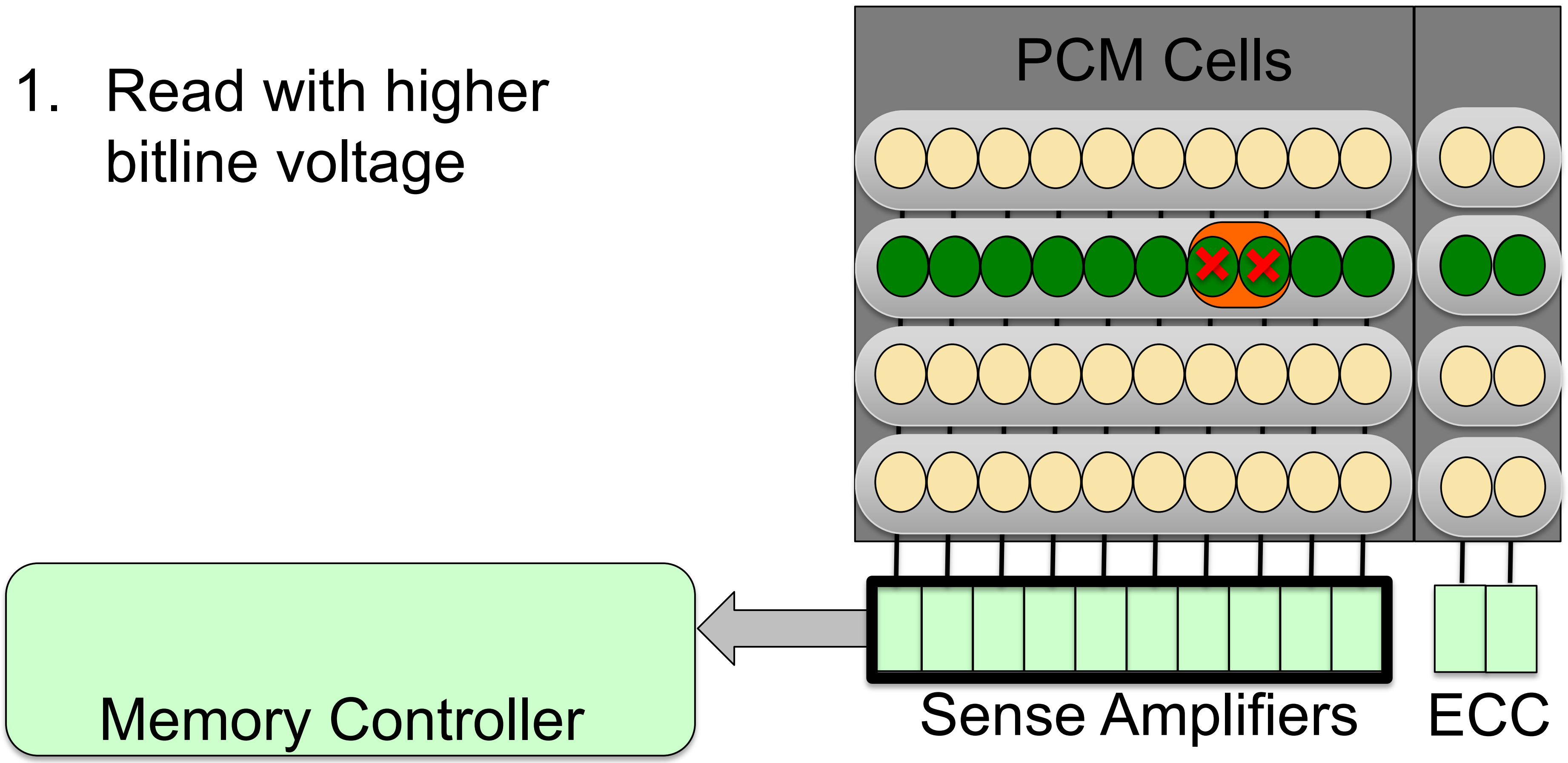
- Incorrect value may be read
- Read disturb errors can be corrected with Error Correcting Codes (ECC)



Correcting read disturb with ECC allows low latency read

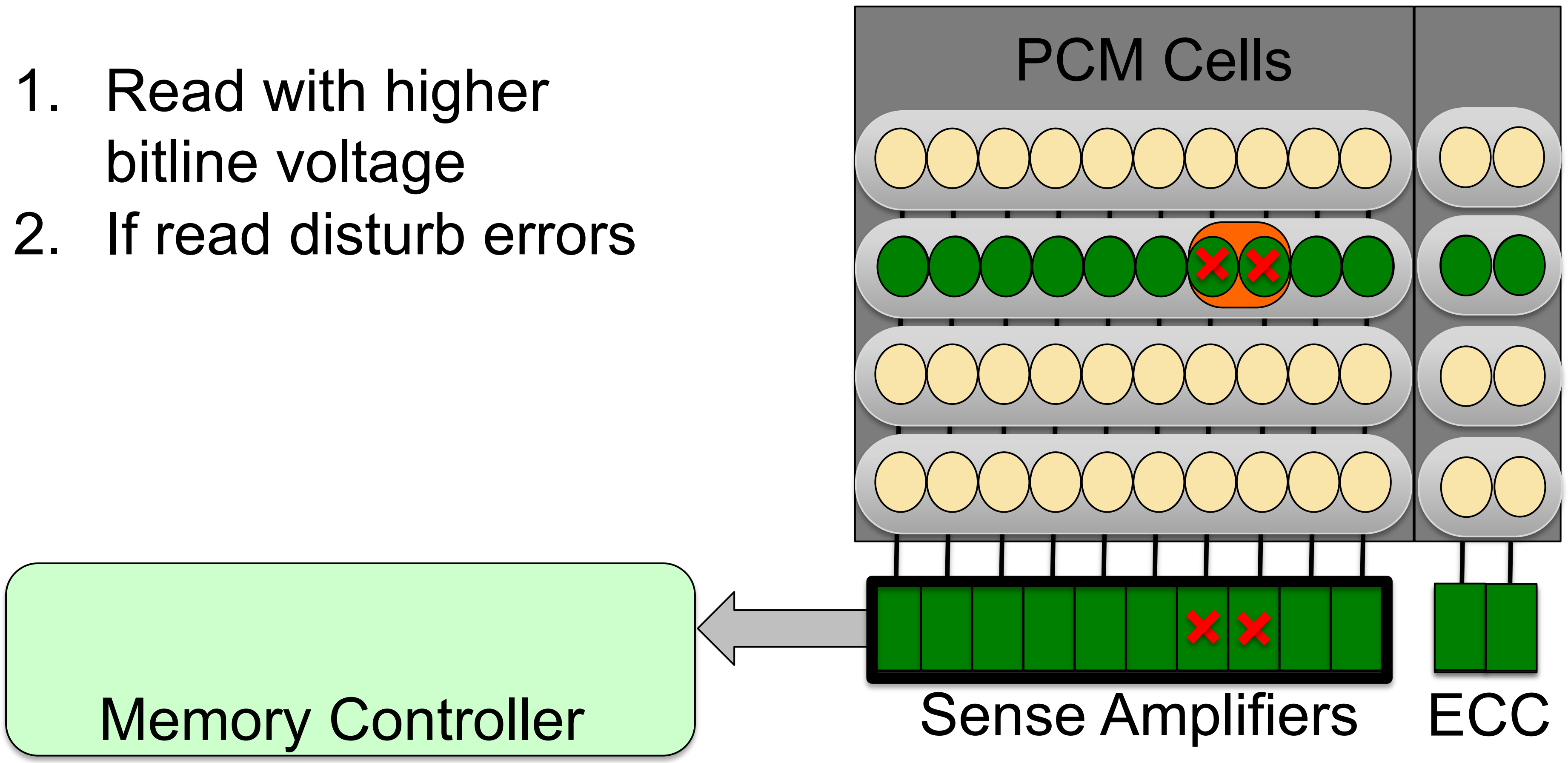
TURBO READ

1. Read with higher bitline voltage



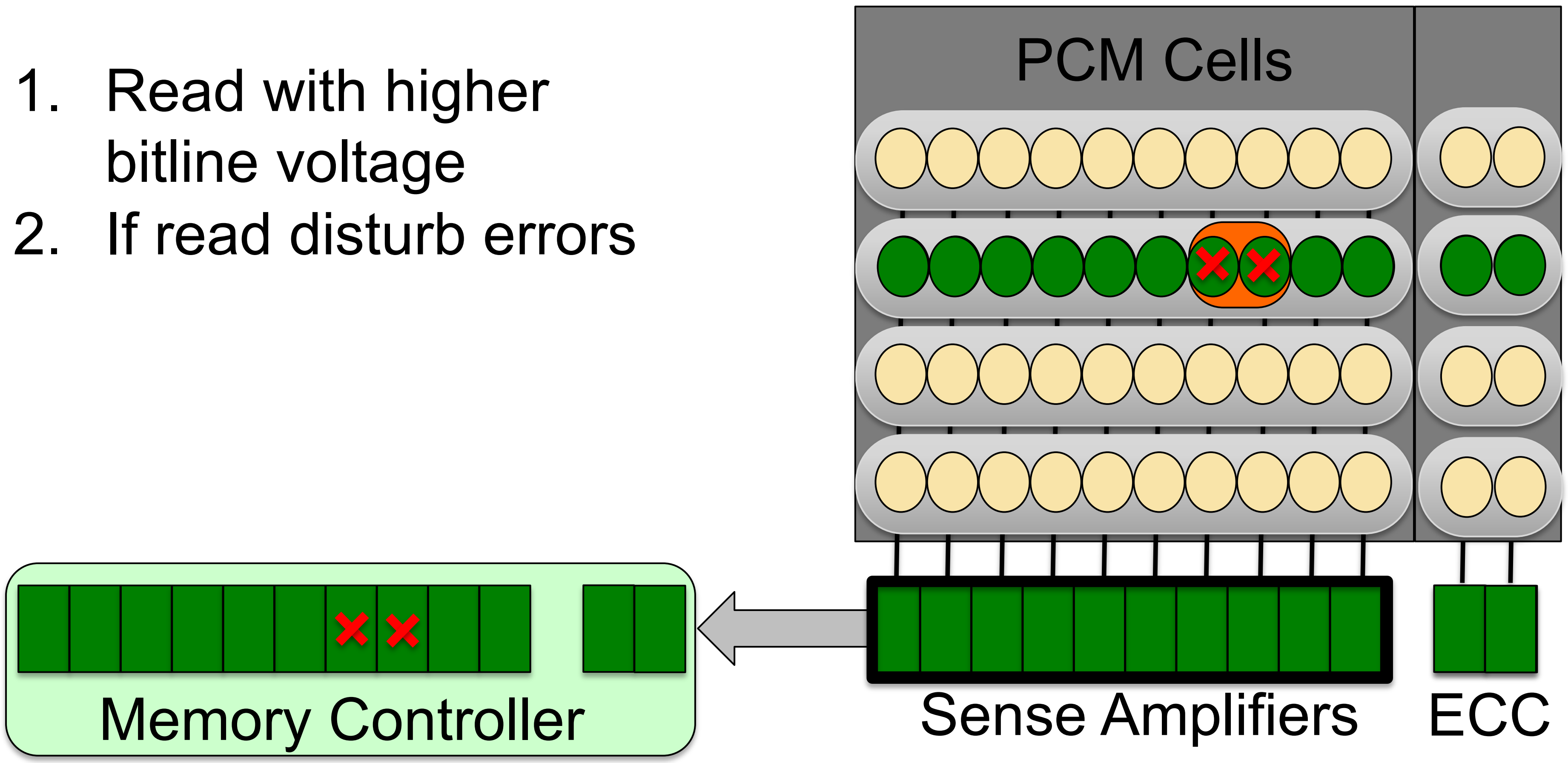
TURBO READ

1. Read with higher bitline voltage
2. If read disturb errors



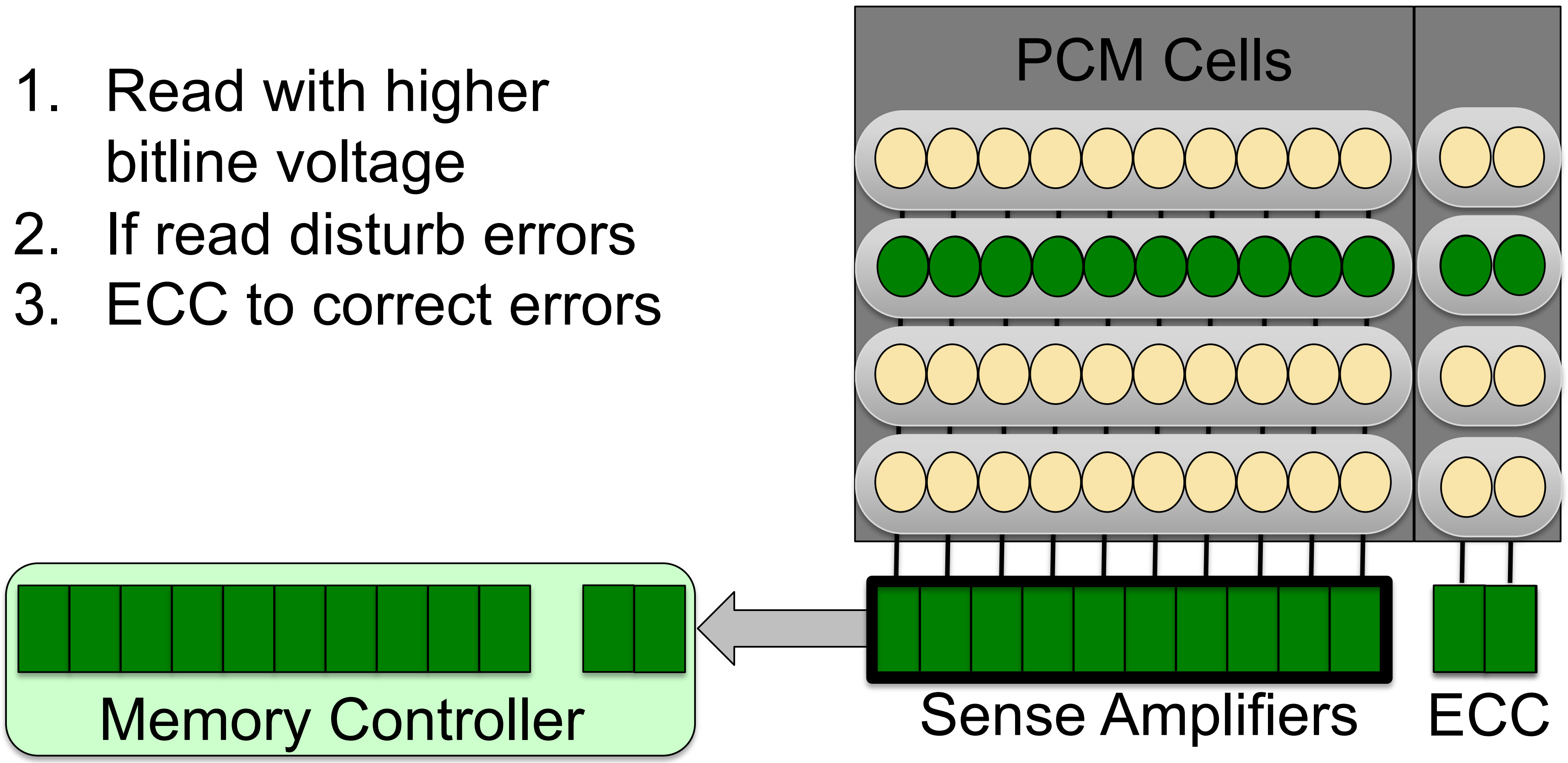
TURBO READ

1. Read with higher bitline voltage
2. If read disturb errors



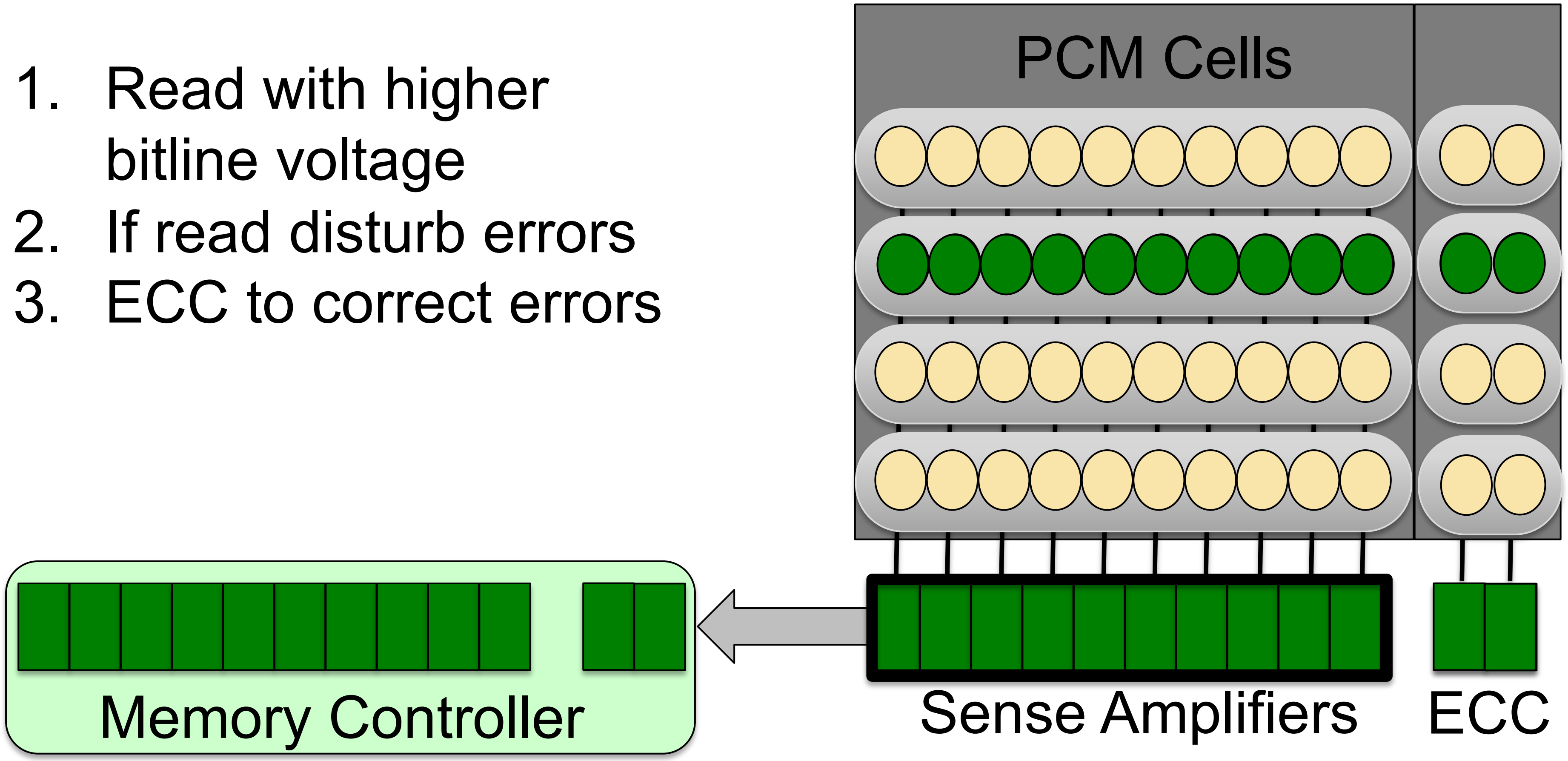
TURBO READ

1. Read with higher bitline voltage
2. If read disturb errors
3. ECC to correct errors



TURBO READ

1. Read with higher bitline voltage
2. If read disturb errors
3. ECC to correct errors



Turbo Read ➔ Read with higher bitline voltage and use ECC to correct read disturb errors

TURBO READ: DESIGN

- Systems are typically designed for failure rate < 10^{-16}
- Fix with a small amount of budget ➔ DECTED

TURBO READ: DESIGN

- Systems are typically designed for failure rate < 10^{-16}
- Fix with a small amount of budget ➔ DECTED

BER Read Disturb	Probability Line has 3 Errors	Latency
10^{-9}	$< 10^{-19}$	57ns

TURBO READ: DESIGN

- Systems are typically designed for failure rate < 10^{-16}
- Fix with a small amount of budget ➔ DECTED

BER Read Disturb	Probability Line has 3 Errors	Latency
10^{-9}	$< 10^{-19}$	57ns

ECC can mitigate read disturb errors in Turbo Read

TURBO READ: DESIGN

- Systems are typically designed for failure rate < 10^{-16}
- Fix with a small amount of budget ➔ DECTED

BER Read Disturb	Probability Line has 3 Errors	Latency
10^{-9}	$< 10^{-19}$	57ns

- Probabilistic Scrub (PRS) to mitigate latent faults

ECC can mitigate read disturb errors in Turbo Read

OUTLINE

WHY COMBINE EARLY AND TURBO READ

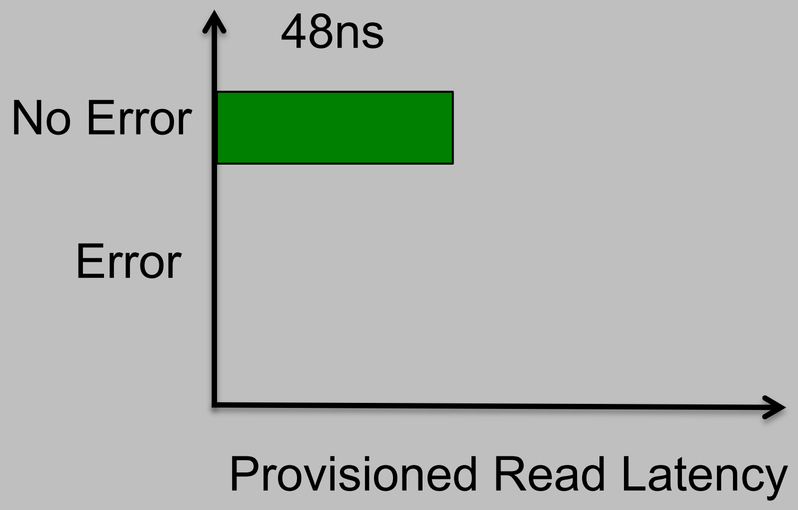
Early read➔Error➔Retry

– Bimodal Read Latency

WHY COMBINE EARLY AND TURBO READ

Early read➔Error➔Retry

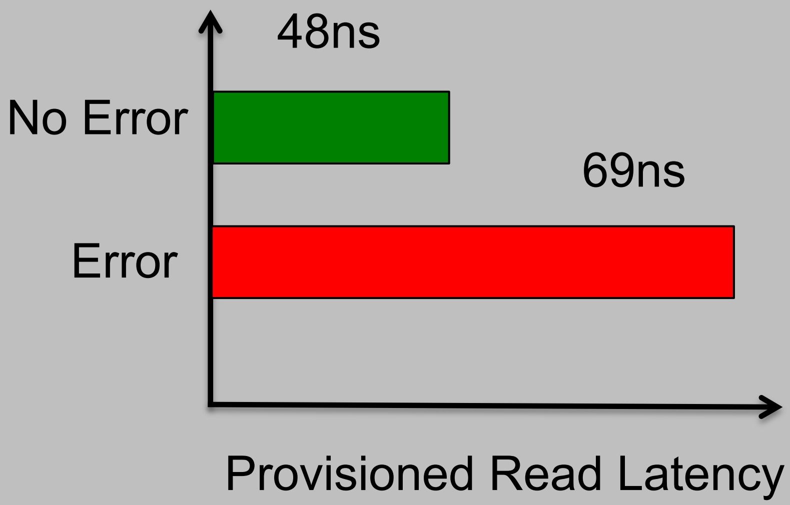
– Bimodal Read Latency



WHY COMBINE EARLY AND TURBO READ

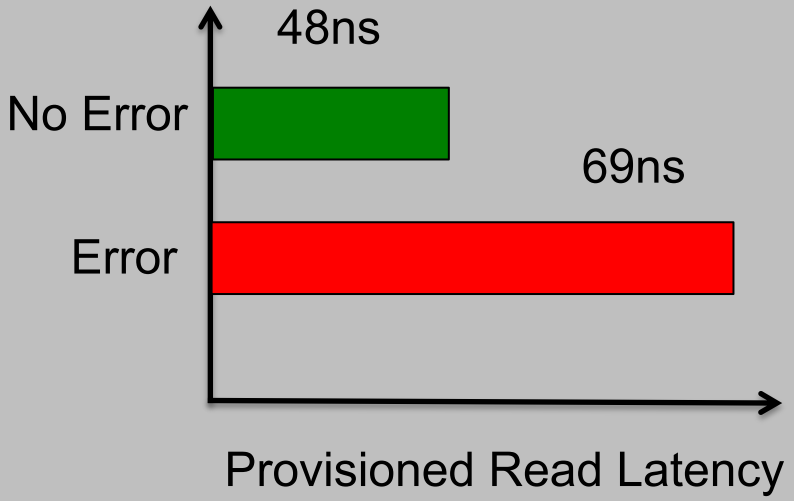
Early read➔Error➔Retry

– Bimodal Read Latency



WHY COMBINE EARLY AND TURBO READ

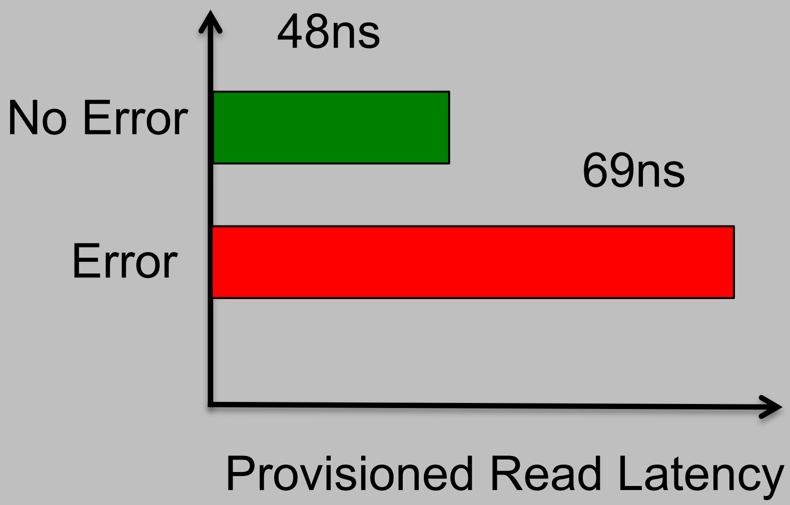
Early read➔Error➔Retry
– Bimodal Read Latency



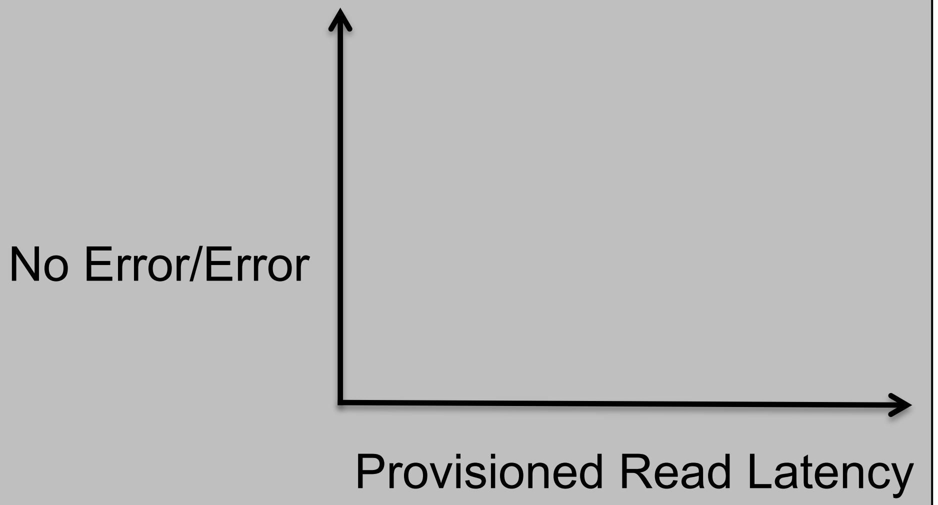
Turbo read➔Error➔No Retry
– Read Latency Fixed

WHY COMBINE EARLY AND TURBO READ

Early read➔Error➔Retry
– Bimodal Read Latency

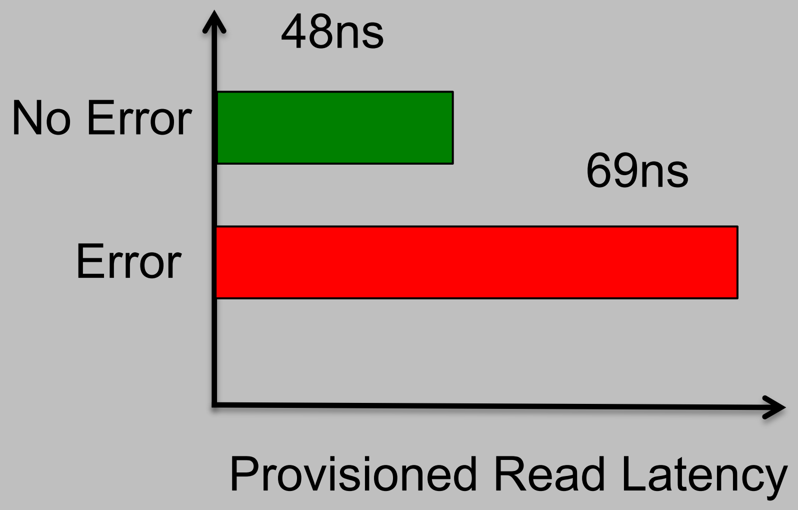


Turbo read➔Error➔No Retry
– Read Latency Fixed

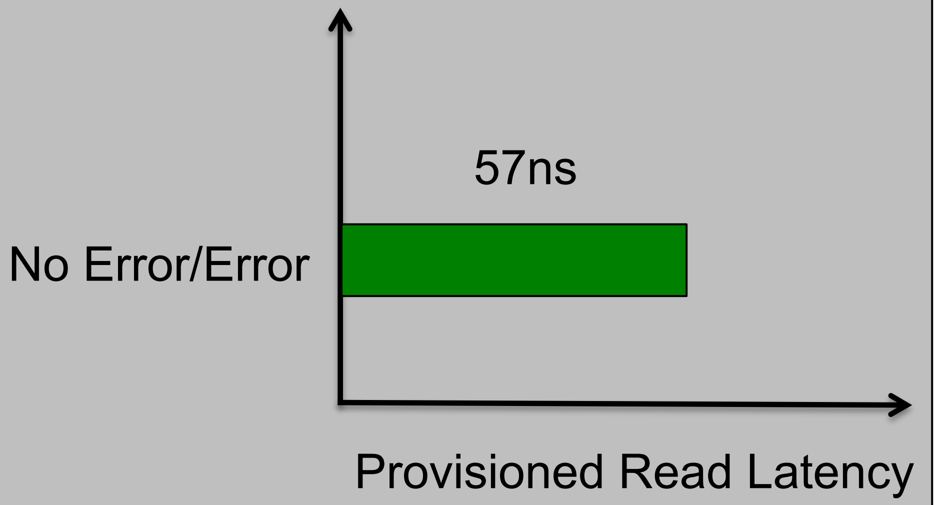


WHY COMBINE EARLY AND TURBO READ

Early read➔Error➔Retry
– Bimodal Read Latency

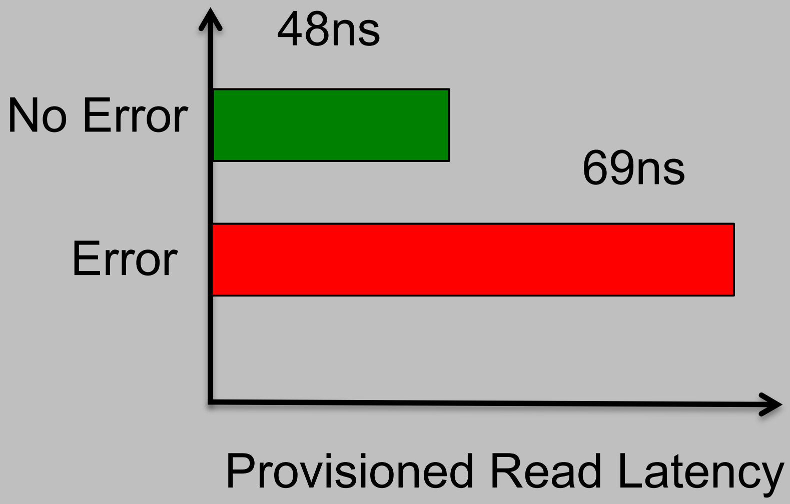


Turbo read➔Error➔No Retry
– Read Latency Fixed

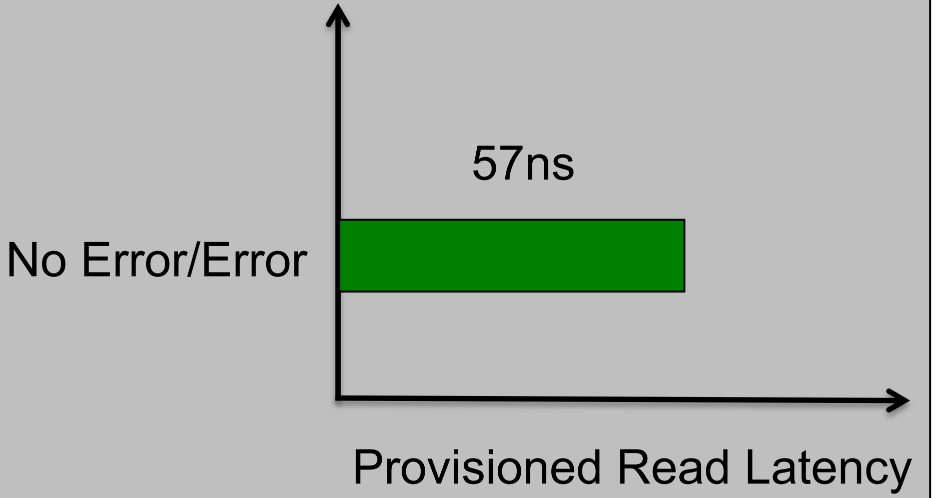


WHY COMBINE EARLY AND TURBO READ

Early read ➔ Error ➔ Retry – Bimodal Read Latency

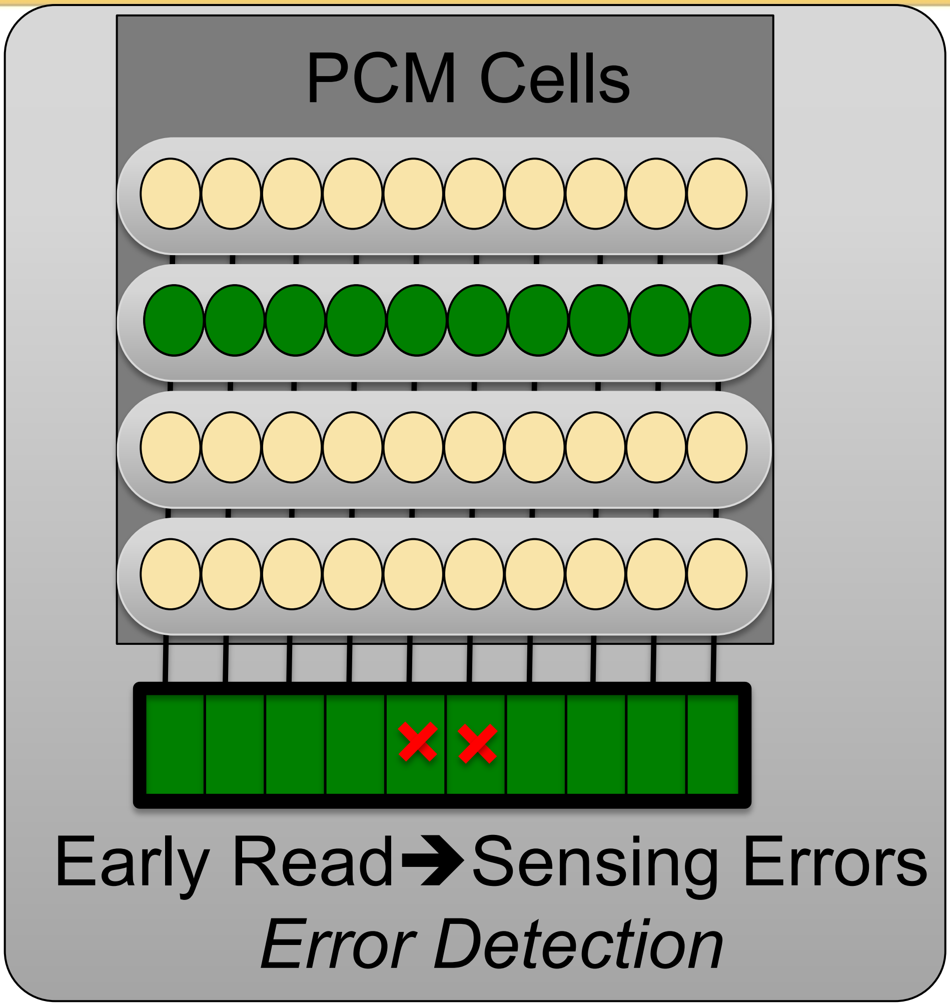


Turbo read ➔ Error ➔ No Retry – Read Latency Fixed

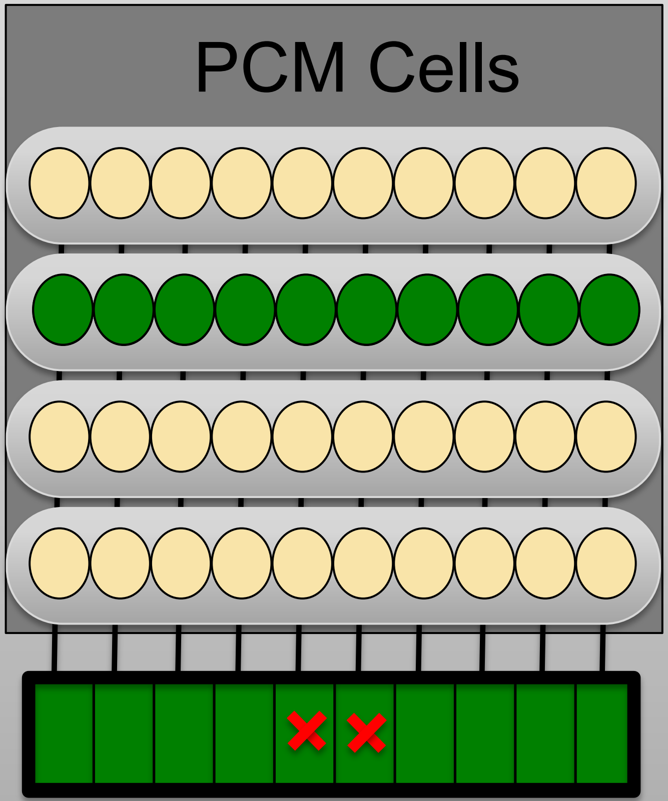


Combine Early and Turbo Reads ➔ Get benefits of both without bimodal latency

CHALLENGES IN EARLY+TURBO READ

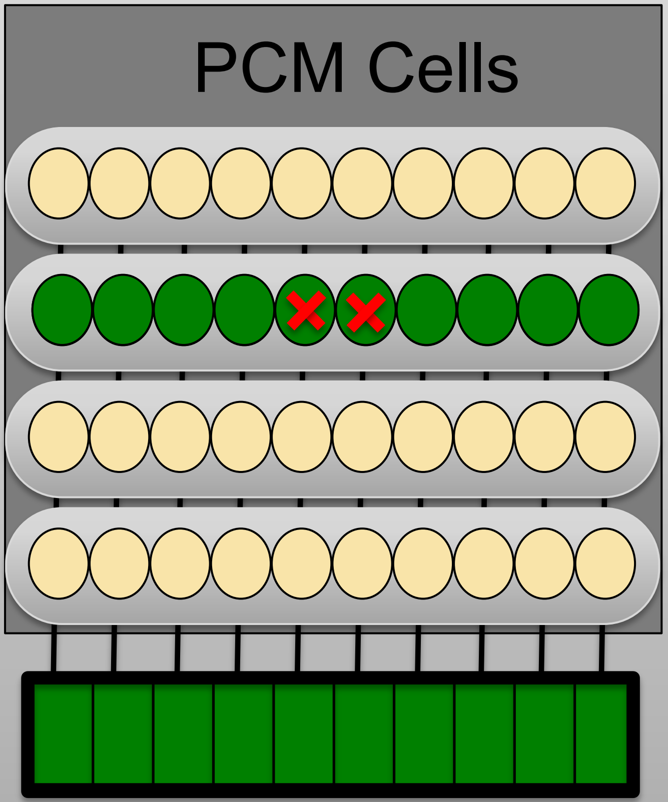


CHALLENGES IN EARLY+TURBO READ



Early Read→Sensing Errors

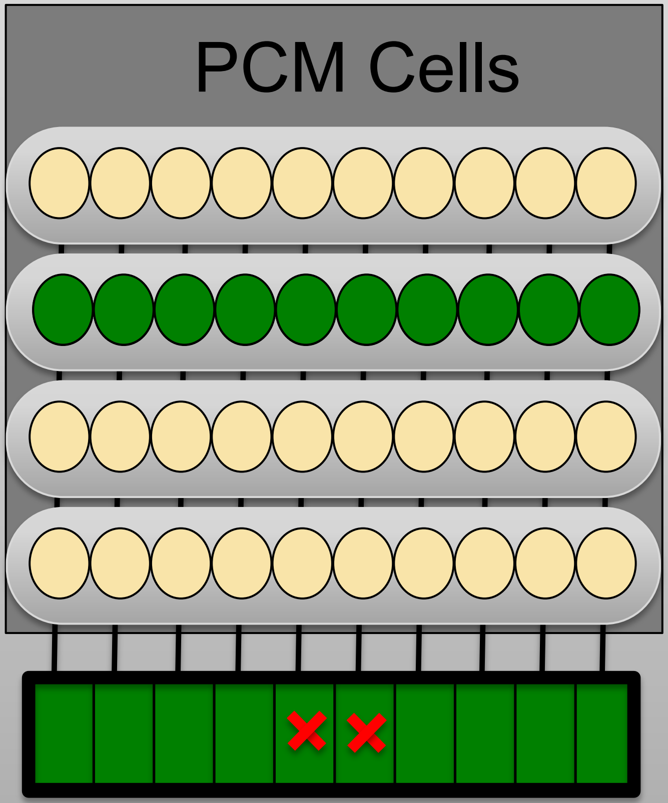
Error Detection



Turbo Read→PCM Cell Errors

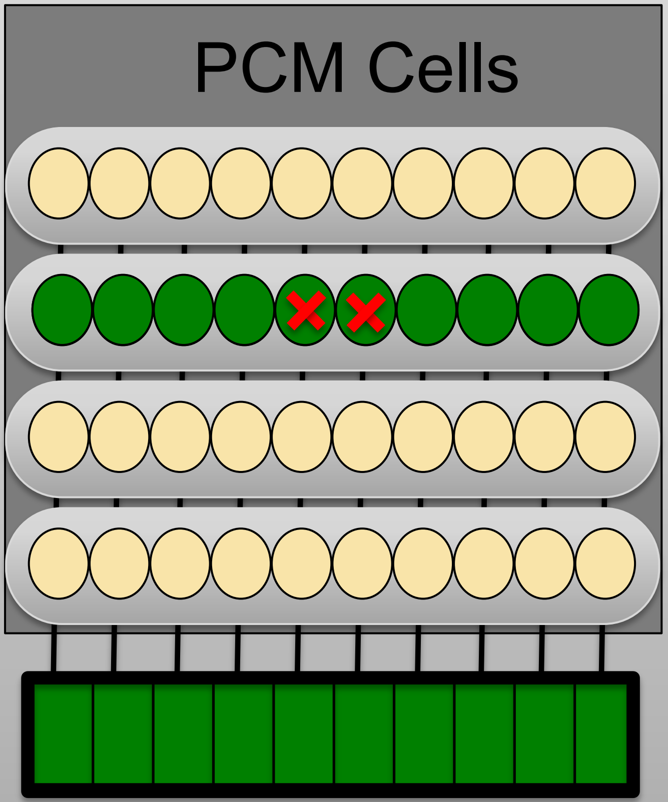
Error Correction

CHALLENGES IN EARLY+TURBO READ



Early Read➔Sensing Errors

Error Detection



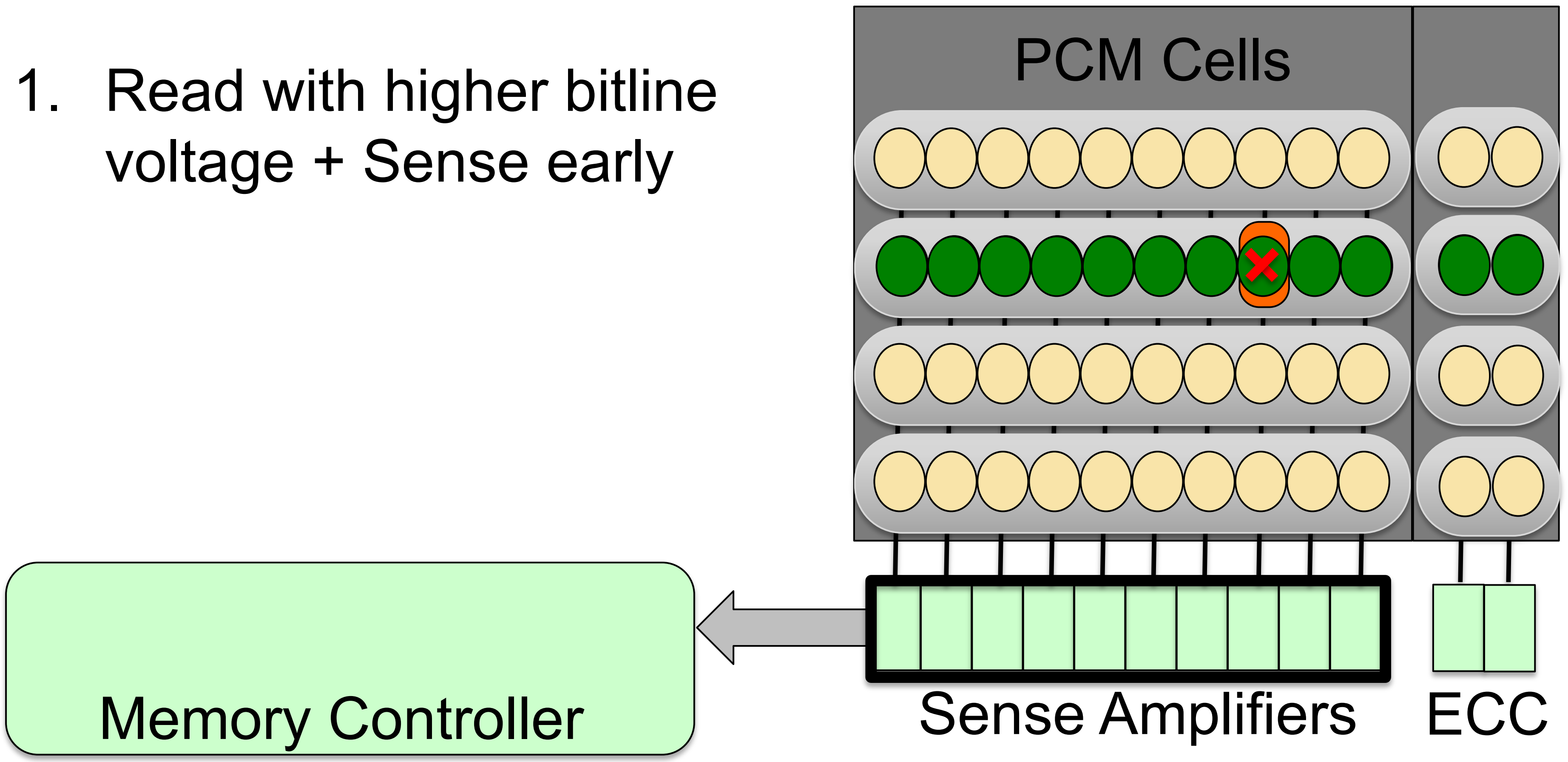
Turbo Read➔PCM Cell Errors

Error Correction

Early+Turbo Read will have PCM cell errors and require Error Correcting Codes

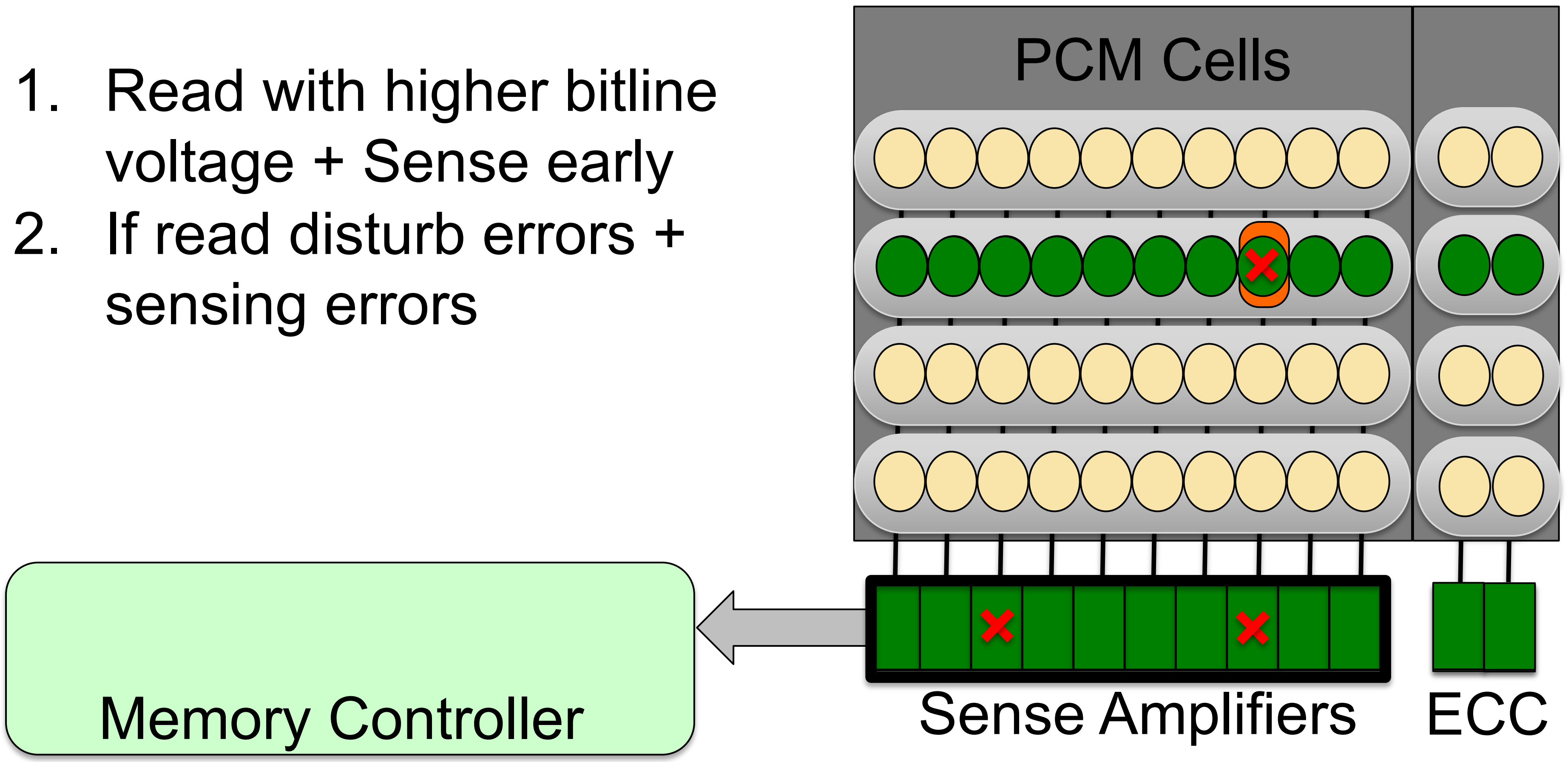
EARLY+TURBO READ

1. Read with higher bitline voltage + Sense early



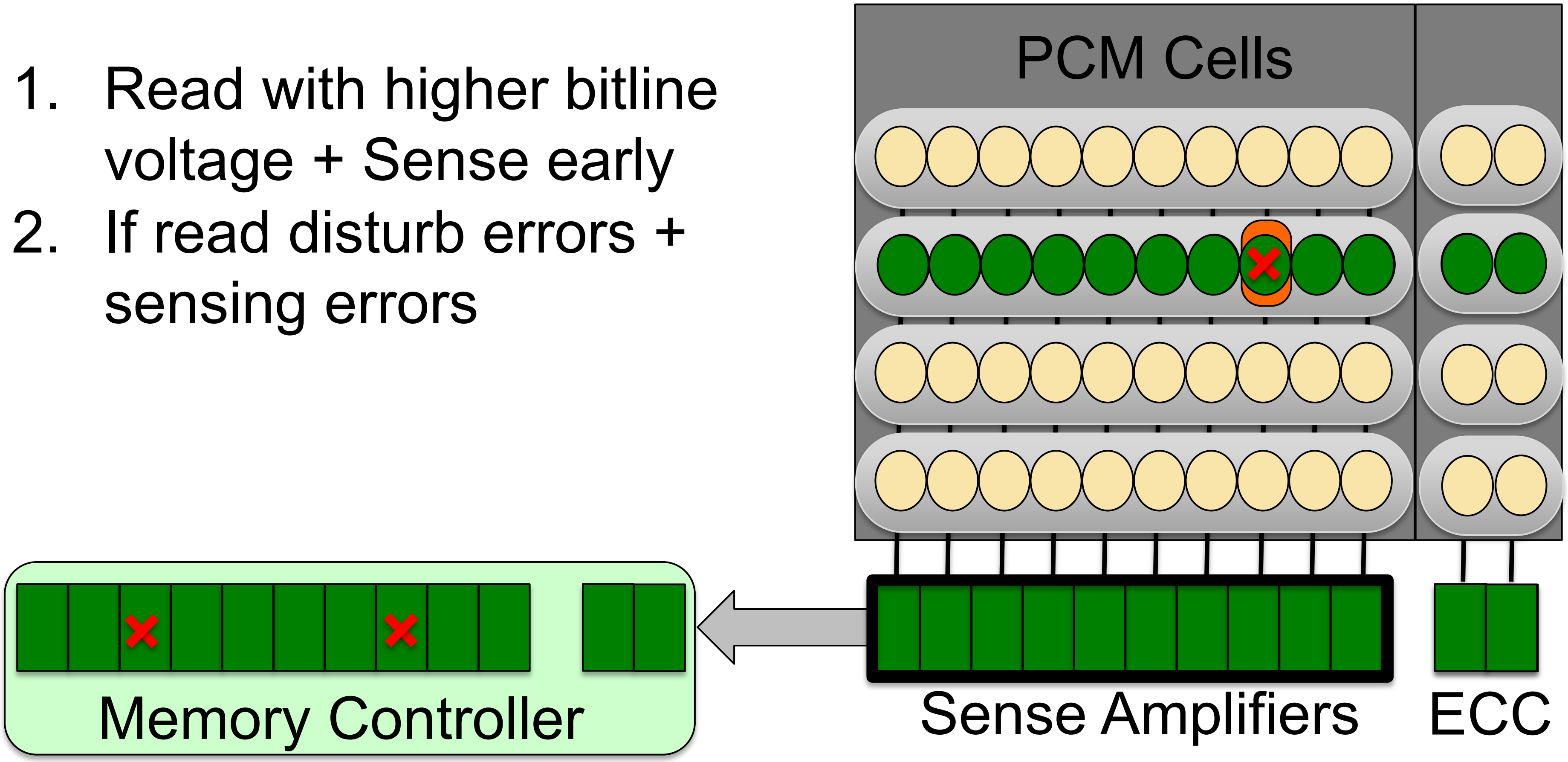
EARLY+TURBO READ

1. Read with higher bitline voltage + Sense early
2. If read disturb errors + sensing errors



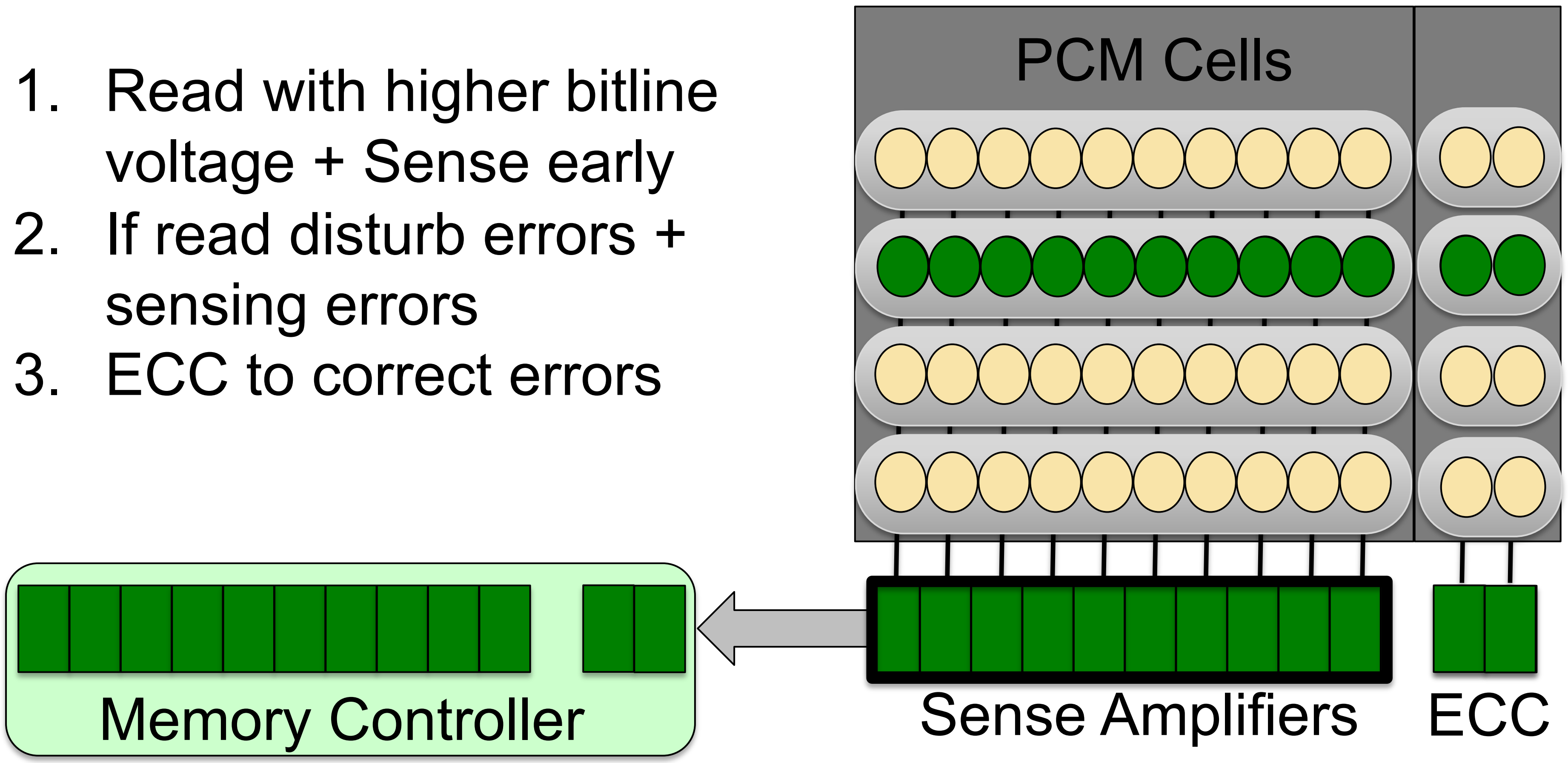
EARLY+TURBO READ

1. Read with higher bitline voltage + Sense early
2. If read disturb errors + sensing errors



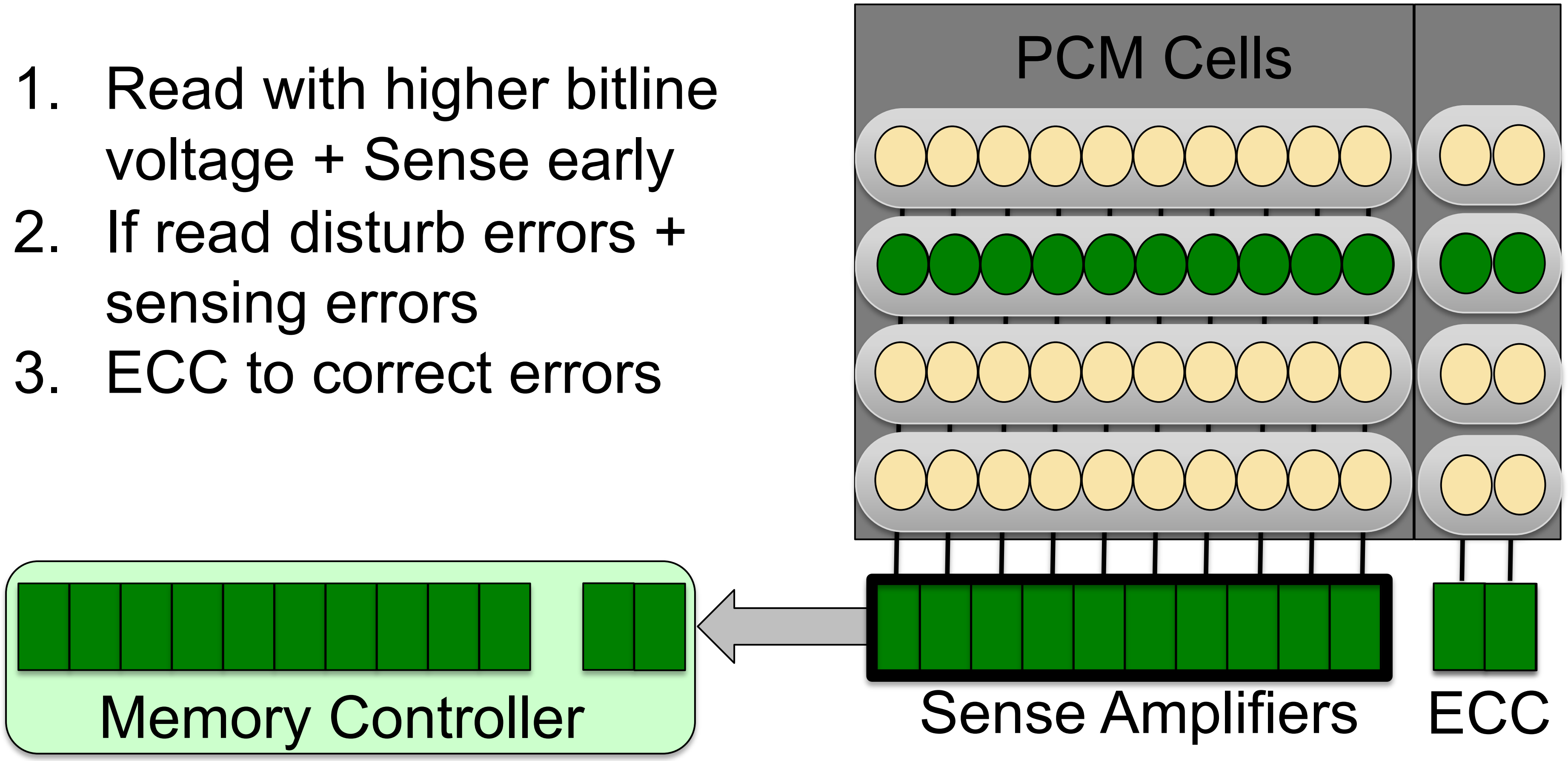
EARLY+TURBO READ

1. Read with higher bitline voltage + Sense early
2. If read disturb errors + sensing errors
3. ECC to correct errors



EARLY+TURBO READ

1. Read with higher bitline voltage + Sense early
2. If read disturb errors + sensing errors
3. ECC to correct errors



Early+Turbo Read ➔ Read with higher bitline voltage and sense early ➔ Use ECC to correct errors

EARLY+TURBO READ: DESIGN

	Early Read	Turbo Read	Early+Turbo Read
BER	10^{-5}	10^{-9}	$2x10^{-9}$
Sensing Latency	48ns or 69ns (Bimodal)	57ns (Fixed)	45ns (Fixed)
Storage Overhead	10 bits/line	20 bits/line	20 bits/line

- $2x10^{-9}$ BER ➔ DECTED ➔ System Failure Rate < 10^{-19}

EARLY+TURBO READ: DESIGN

	Early Read	Turbo Read	Early+Turbo Read
BER	10^{-5}	10^{-9}	$2x10^{-9}$
Sensing Latency	48ns or 69ns (Bimodal)	57ns (Fixed)	45ns (Fixed)
Storage Overhead	10 bits/line	20 bits/line	20 bits/line

- $2x10^{-9}$ BER → DECTED → System Failure Rate < 10^{-19}
- Sensing Latency Fixed → 45ns

EARLY+TURBO READ: DESIGN

	Early Read	Turbo Read	Early+Turbo Read
BER	10^{-5}	10^{-9}	$2\text{x}10^{-9}$
Sensing Latency	48ns or 69ns (Bimodal)	57ns (Fixed)	45ns (Fixed)
Storage Overhead	10 bits/line	20 bits/line	20 bits/line

- $2\text{x}10^{-9}$ BER ➔ DECTED ➔ System Failure Rate < 10^{-19}
- Sensing Latency Fixed ➔ 45ns

Early+Turbo Read reduces read latency by 30%

OUTLINE

- Background
- Early Read
- Turbo Read
- Early+Turbo Read
- Results
- Summary

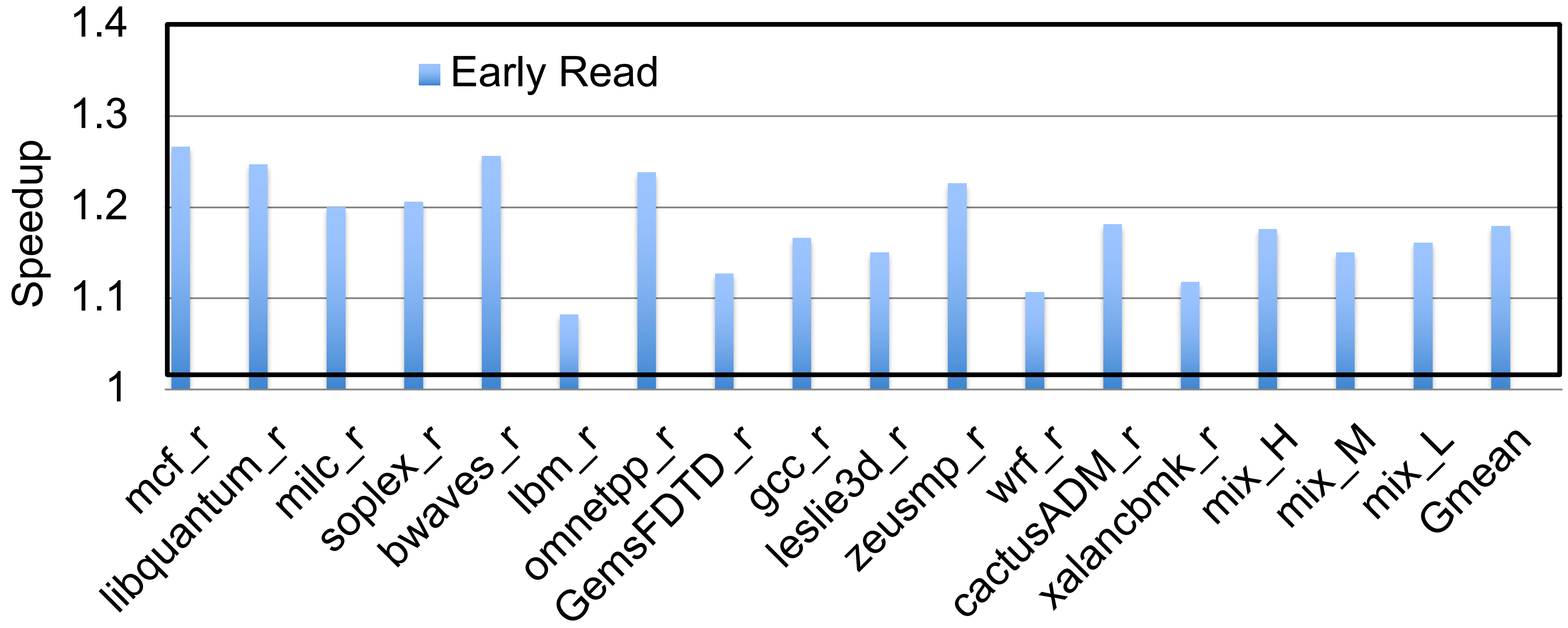
SYSTEM CONFIGURATION

Parameter	Configuration
Cores	8 cores @ 3Ghz
L1-L2-L3 Cache	32KB-256KB-1MB (Private)
L4 Cache	128MB (Shared) @ 15ns latency
PCM System	
Channels	4 Channels @ 8GB/Channel
Read Latency	80ns → 69ns sensing time*
Write Latency	250ns*

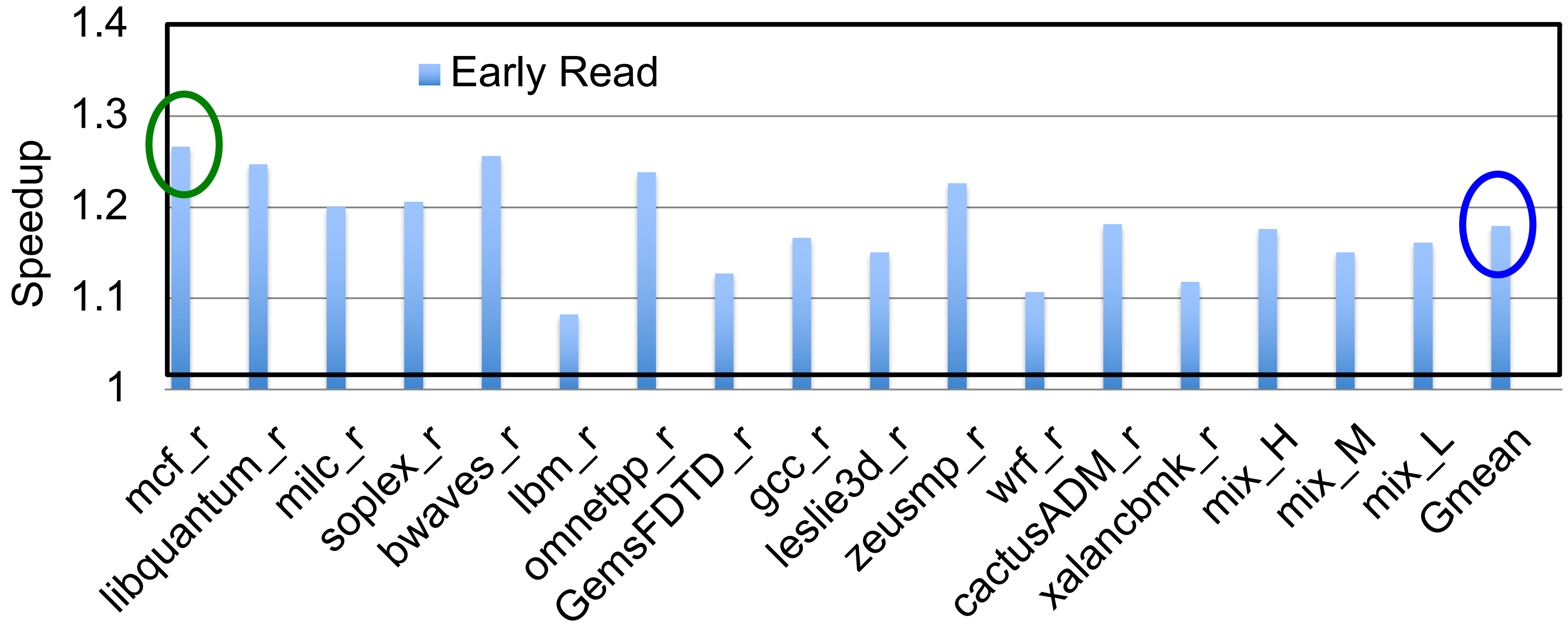
Spec Benchmarks with read MPKI from DRAM Cache > 1

* ISSCC 2012 [PCM-Samsung]

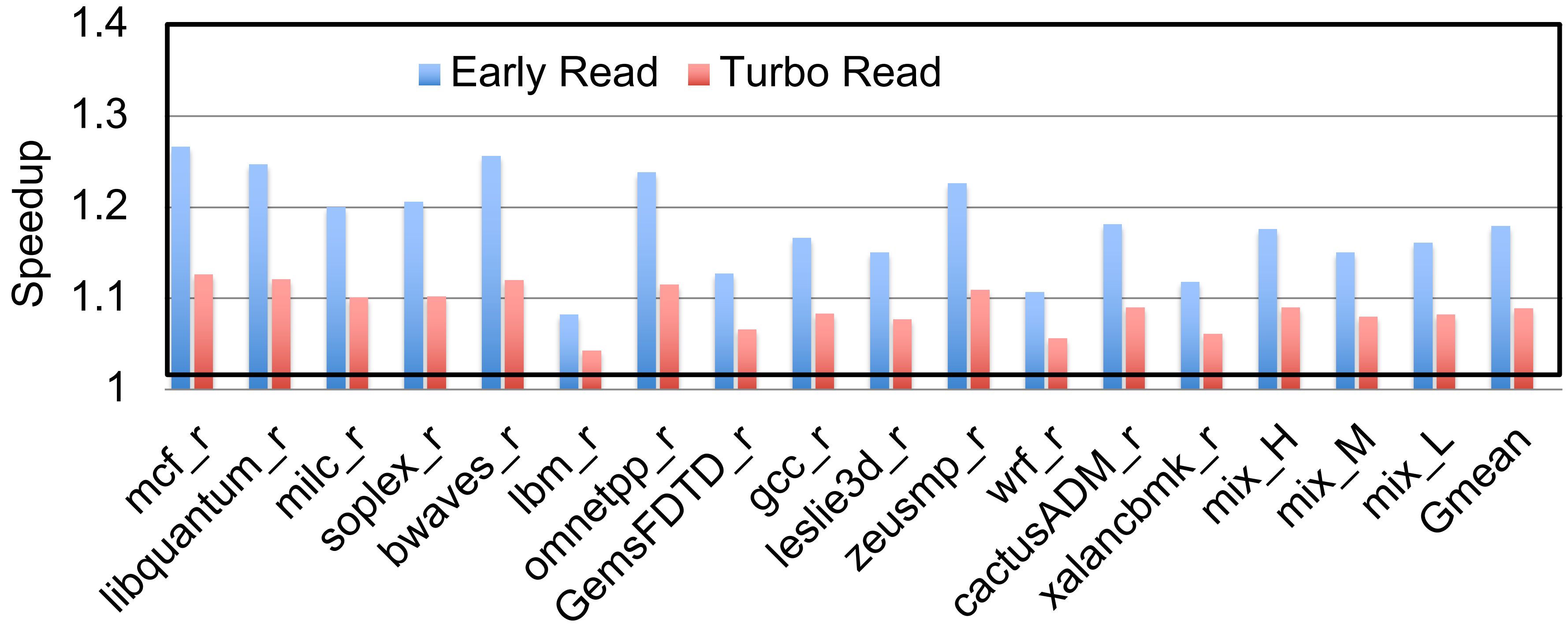
PERFORMANCE



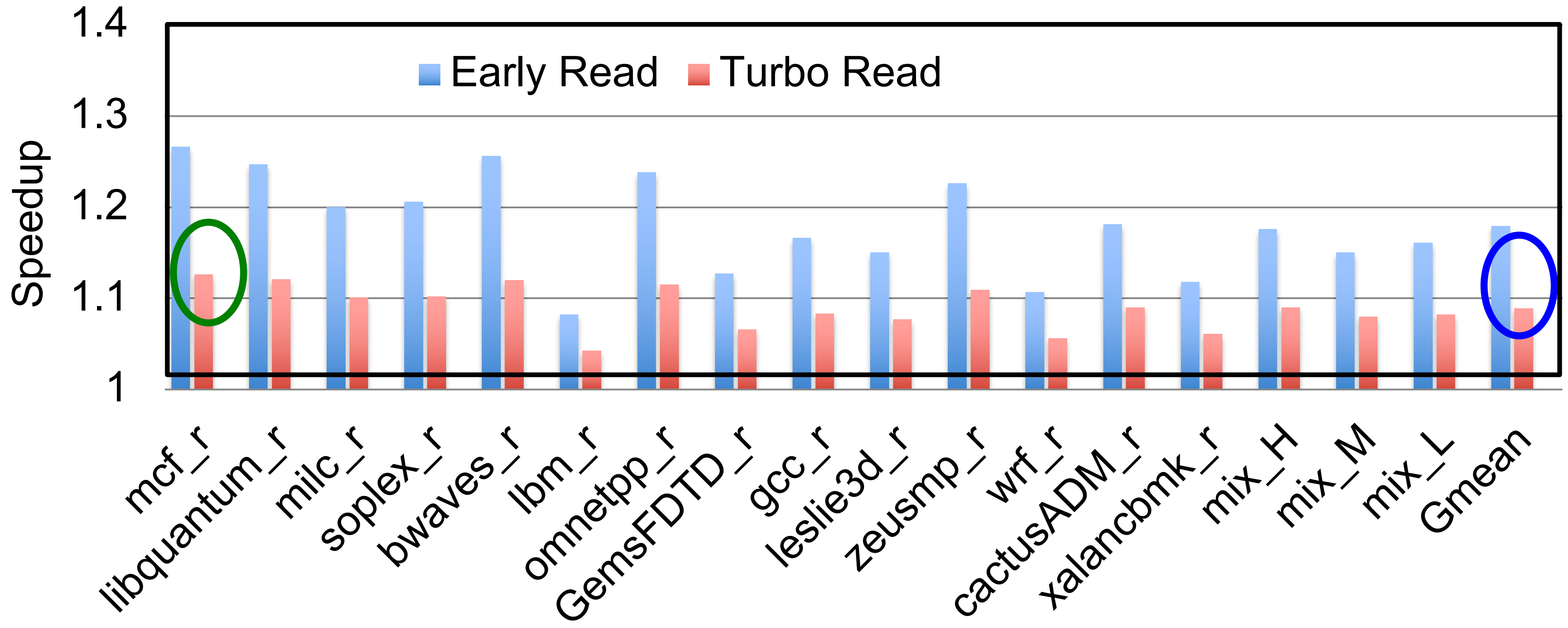
PERFORMANCE



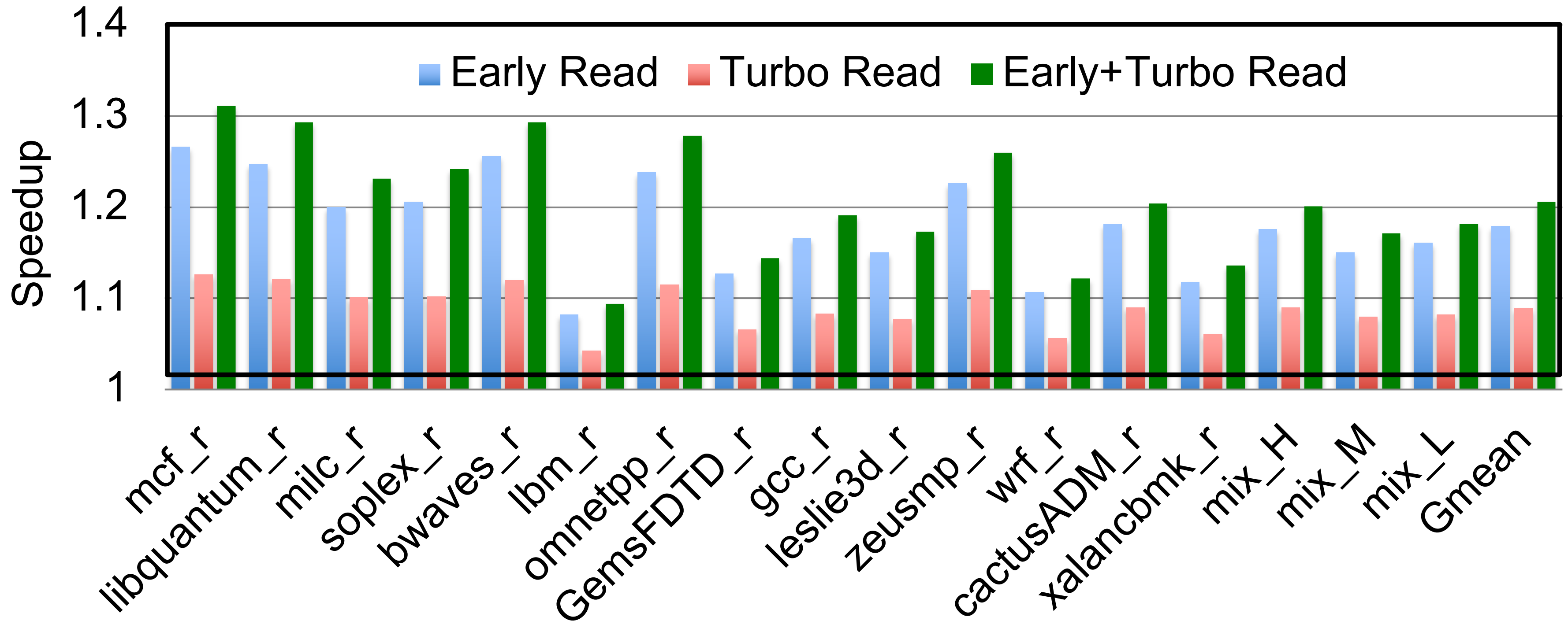
PERFORMANCE



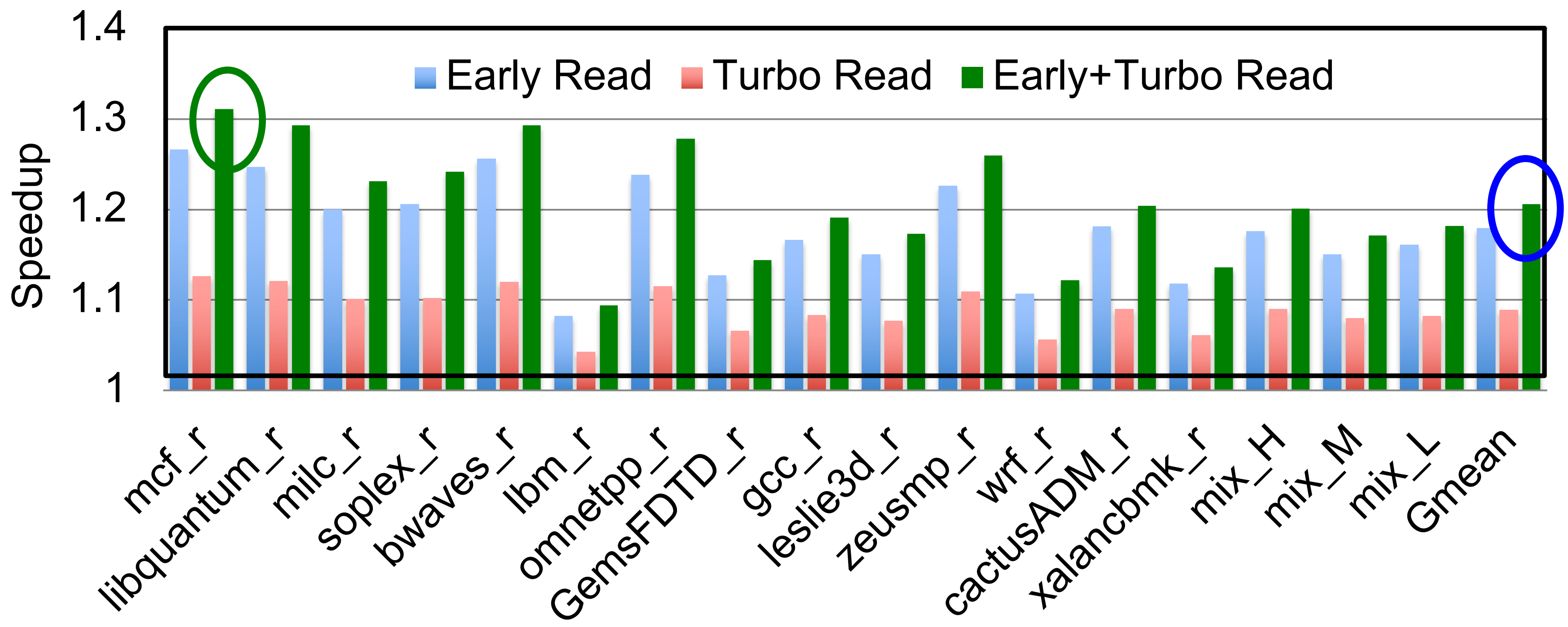
PERFORMANCE



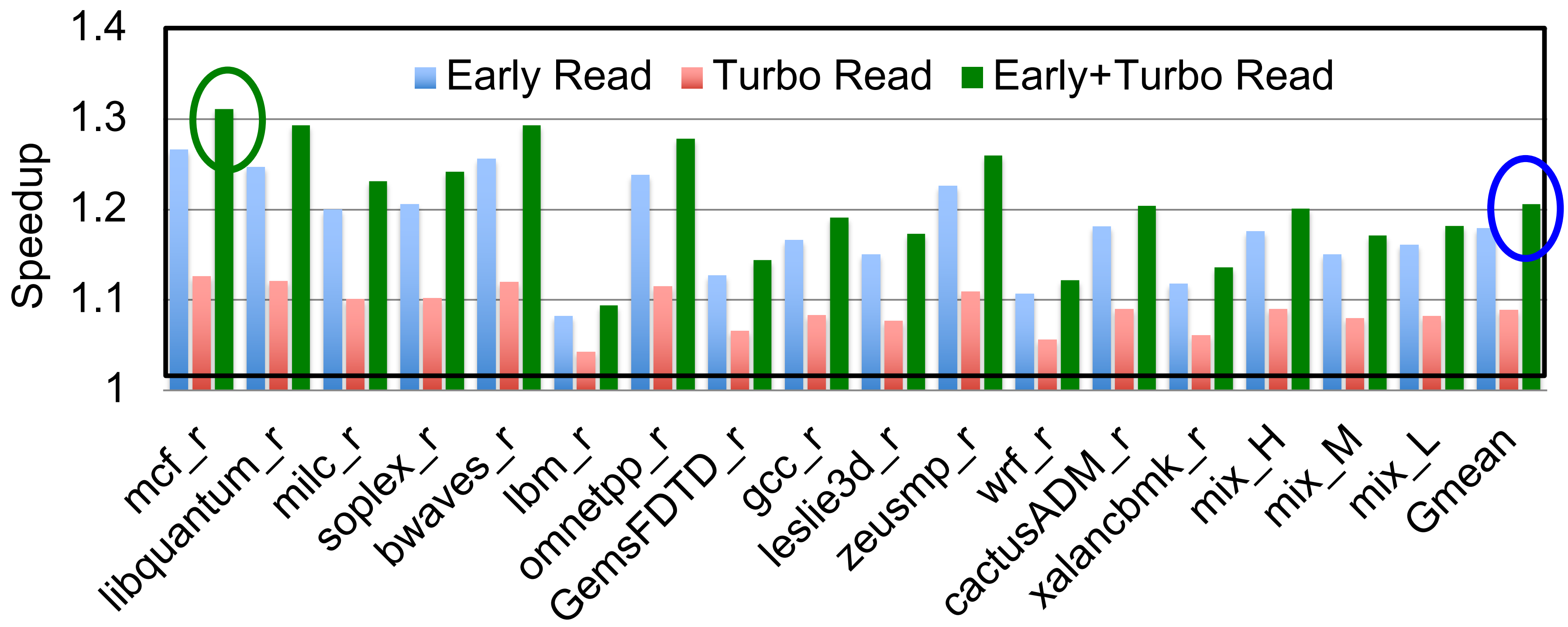
PERFORMANCE



PERFORMANCE

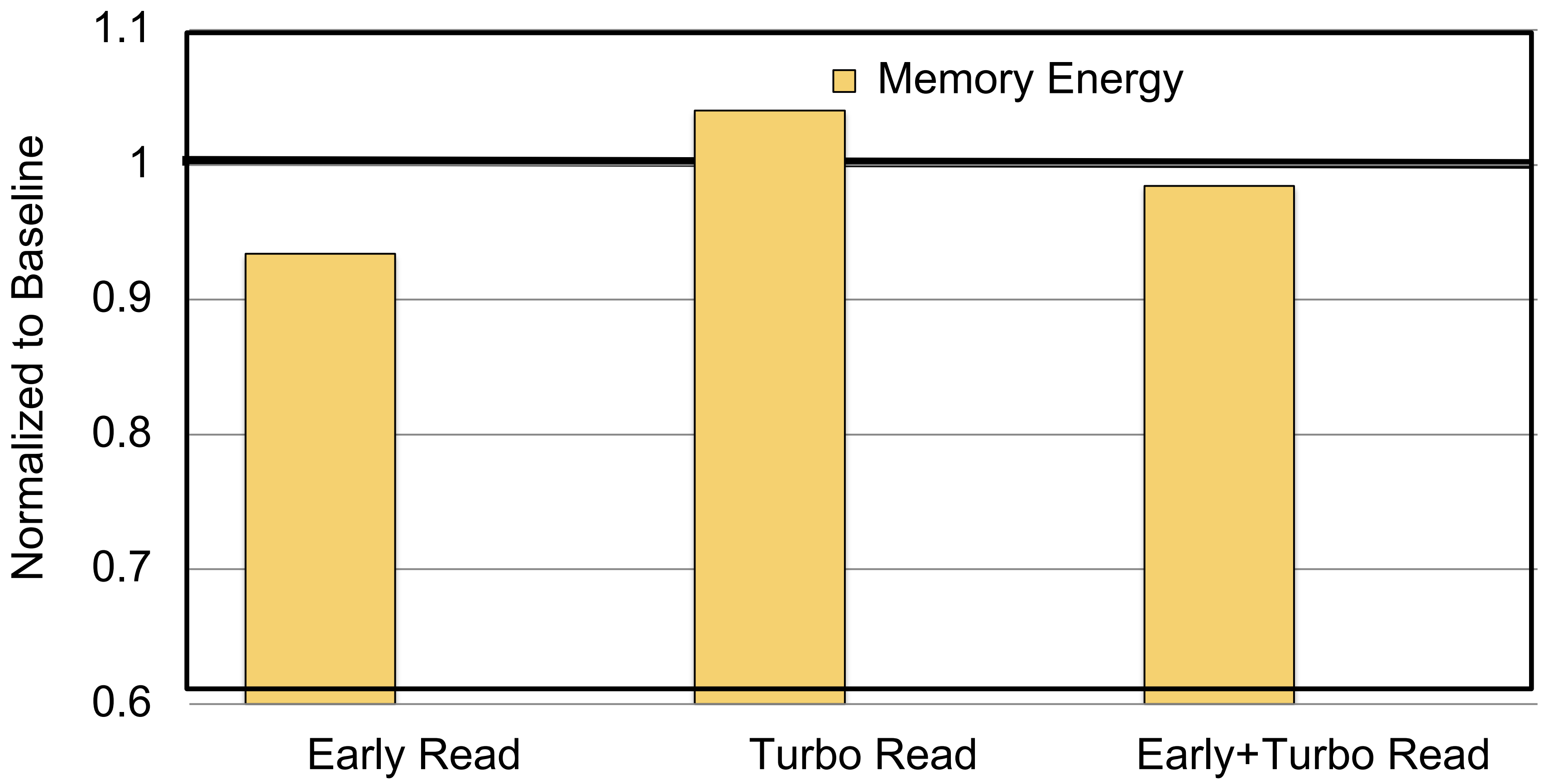


PERFORMANCE

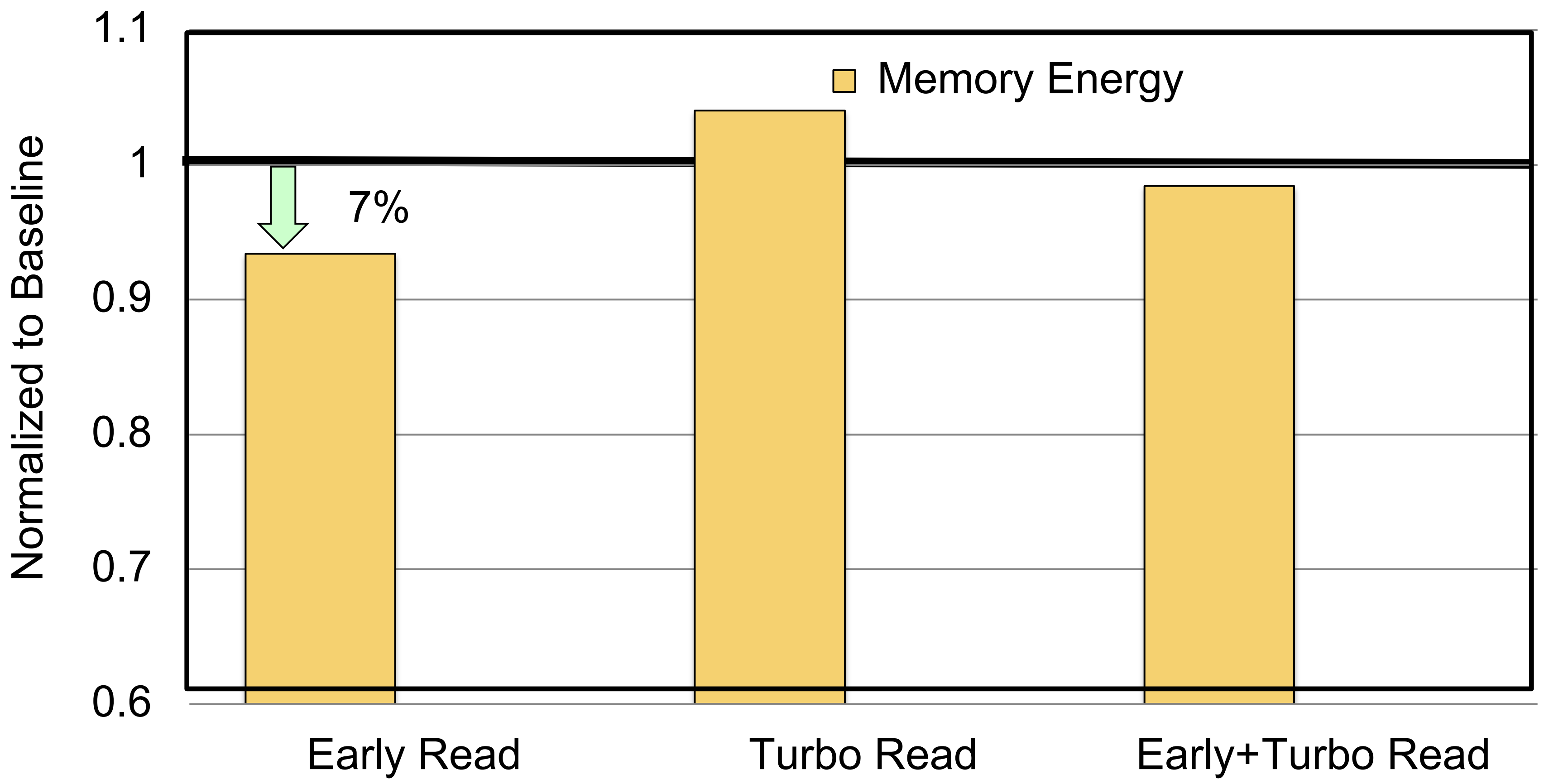


Our proposals improve performance by upto 21%

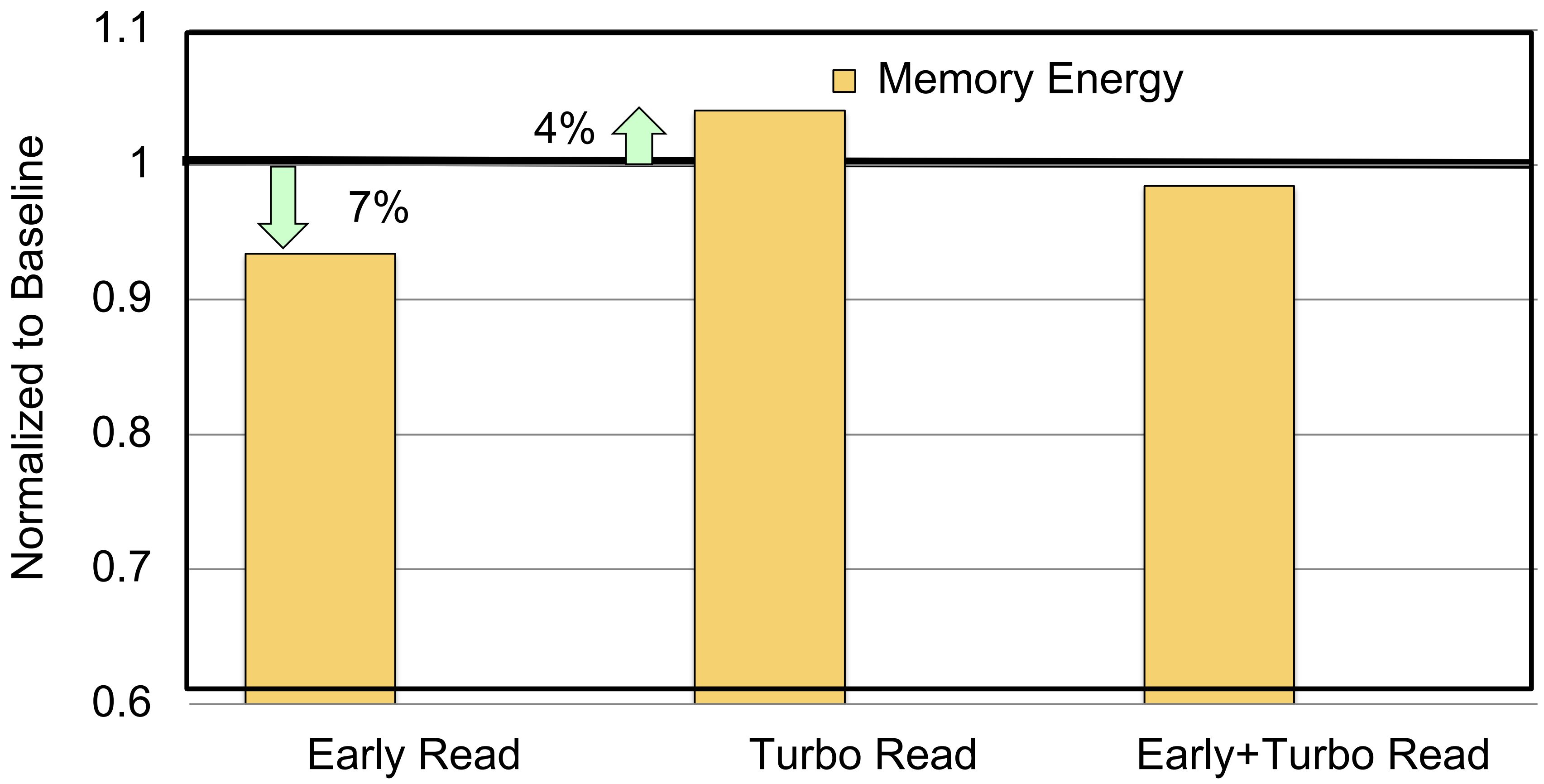
ENERGY AND EDP



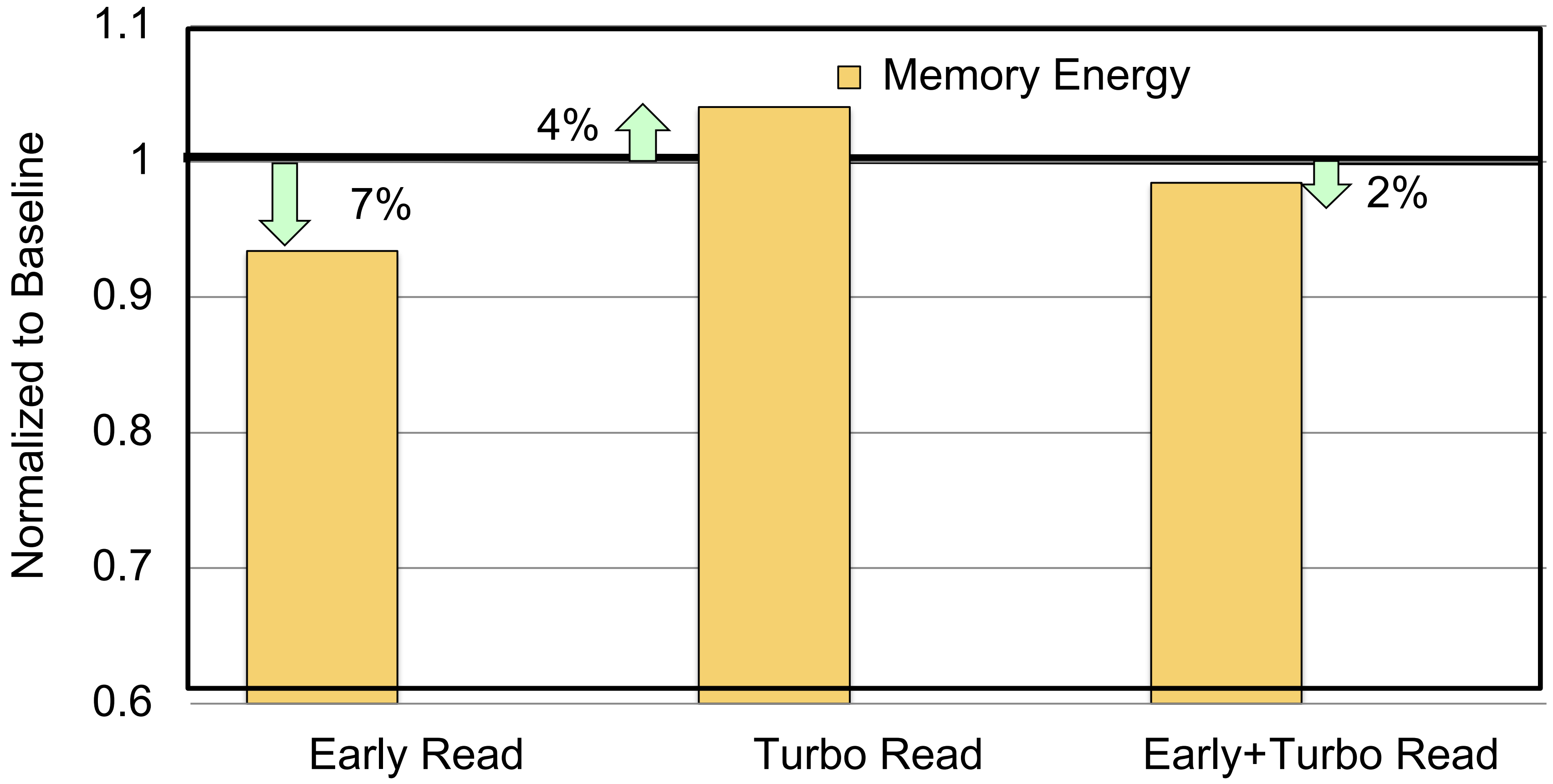
ENERGY AND EDP



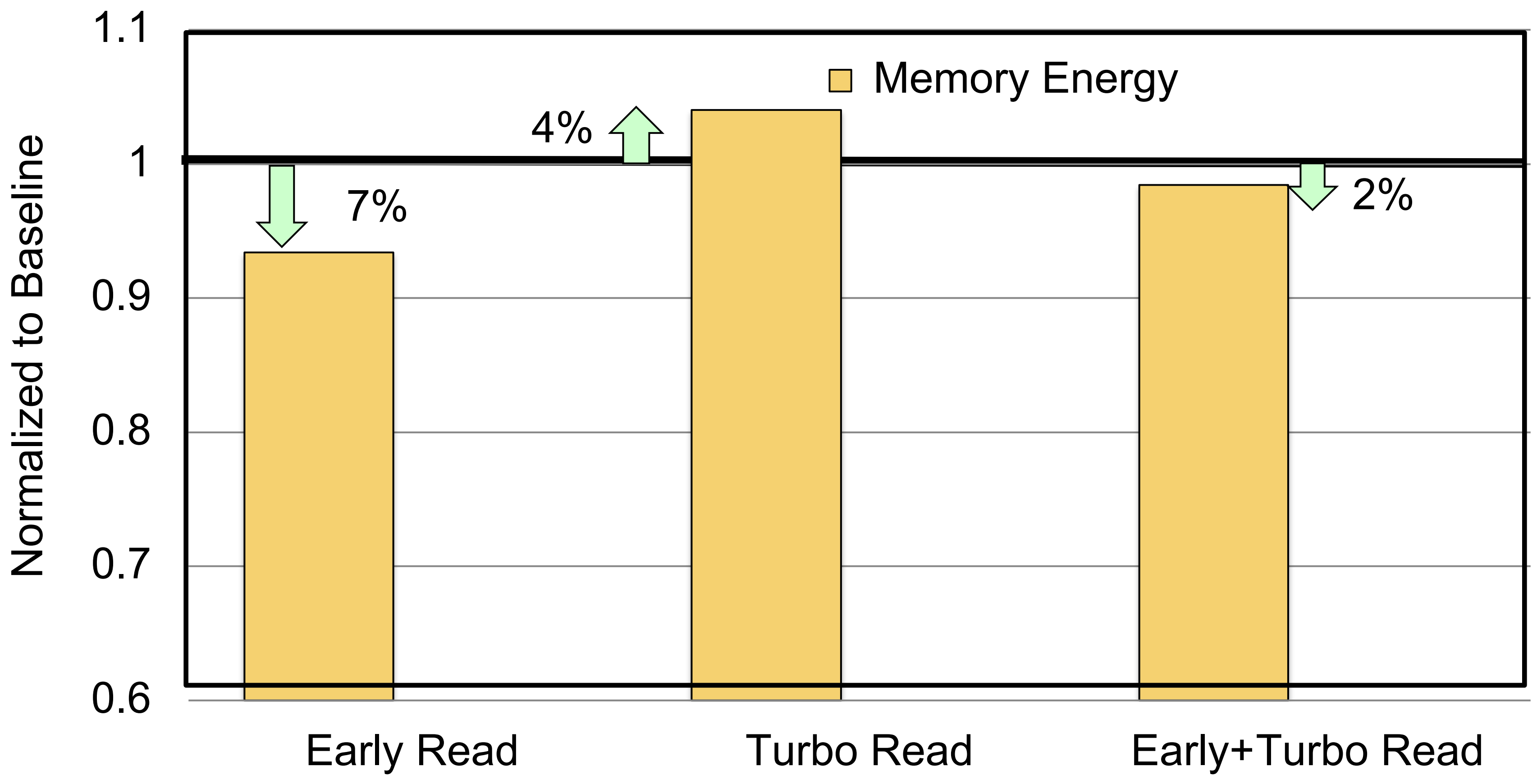
ENERGY AND EDP



ENERGY AND EDP

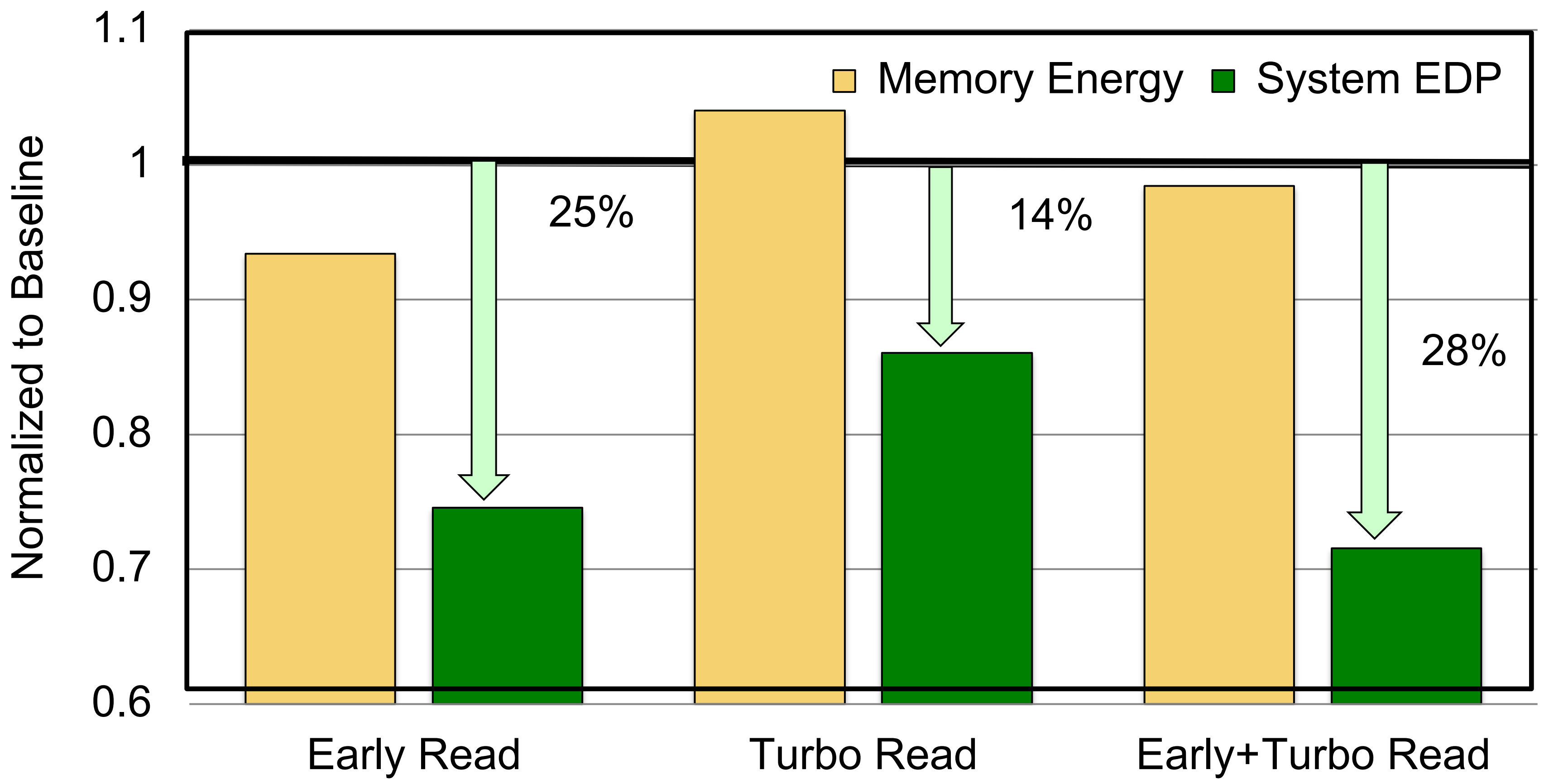


ENERGY AND EDP



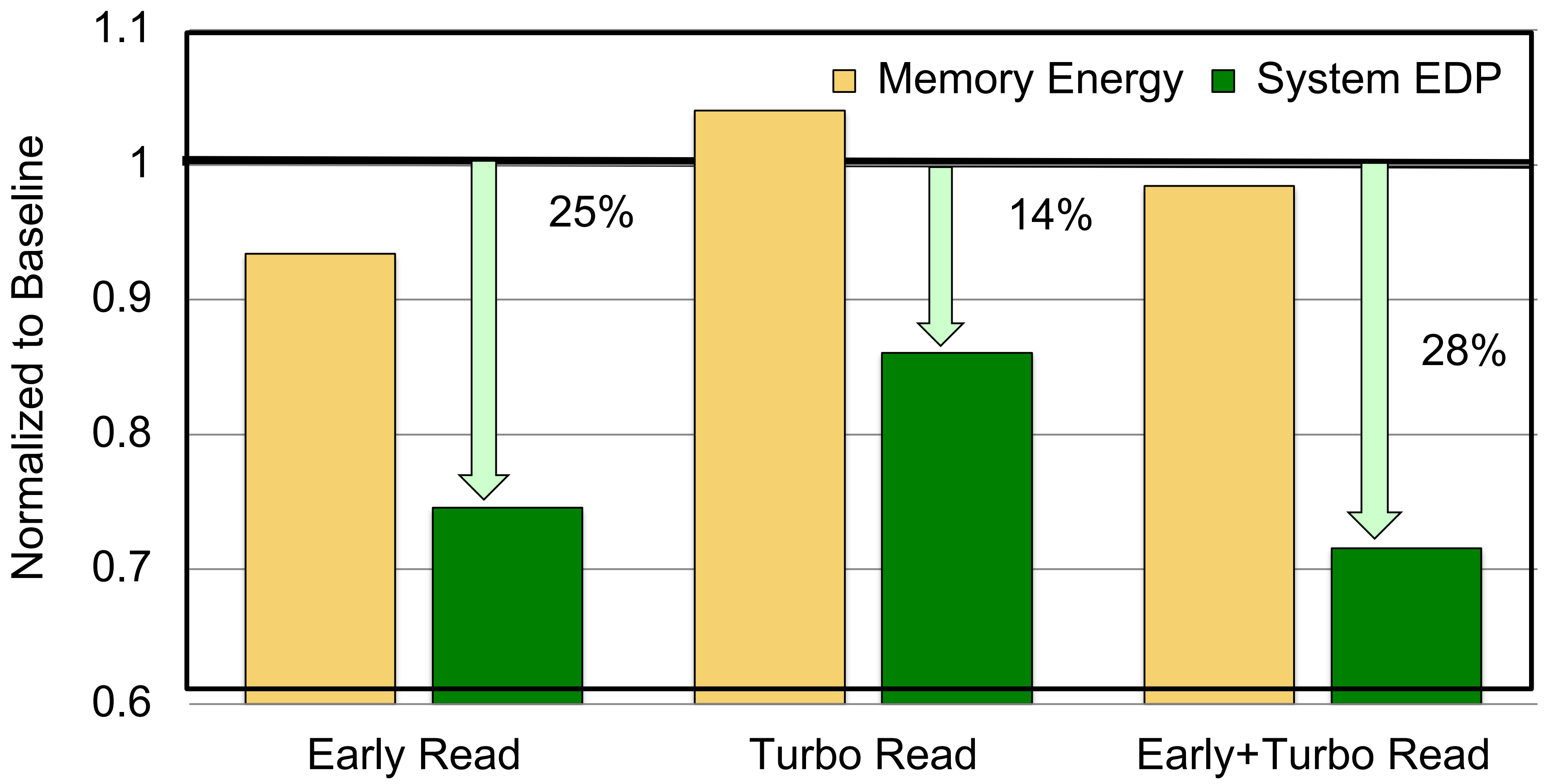
Our proposals reduce energy by 7%

ENERGY AND EDP



Our proposals reduce energy by 7%

ENERGY AND EDP



Our proposals reduce energy by 7%

Our proposals reduce EDP by 28%

OUTLINE

- Background
- Early Read
- Turbo Read
- Early+Turbo Read
- Results
- Summary ⇦

Summary

- Goal ➔ Reduce the read latency of PCM

Summary

- Goal ➔ Reduce the read latency of PCM
- Two low cost solutions

Summary

- Goal → Reduce the read latency of PCM

- Two low cost solutions
- Early Read: Better-than-worst-case sensing using Berger Codes to detect errors and retry

Summary

- Goal → Reduce the read latency of PCM

- Two low cost solutions
- Early Read: Better-than-worst-case sensing using Berger Codes to detect errors and retry
- Turbo Read: Read with higher current and fix read disturb errors with ECC

Summary

- Goal ➔ Reduce the read latency of PCM

- Two low cost solutions
- Early Read: Better-than-worst-case sensing using Berger Codes to detect errors and retry
- Turbo Read: Read with higher current and fix read disturb errors with ECC

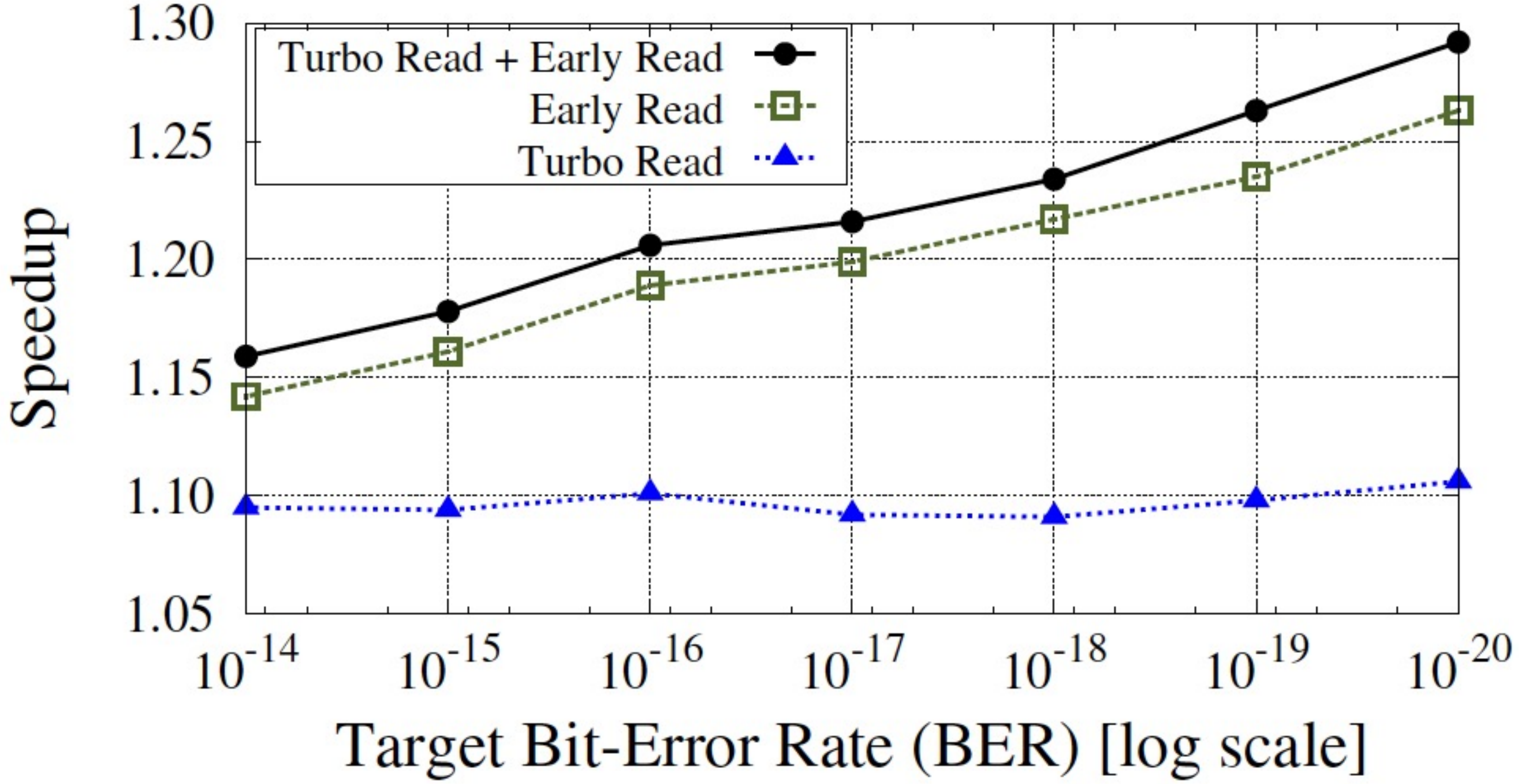
- Proposed solutions reduce read latency by 30% ➔ Performance improves by 21%, EDP by 28%

Thank You

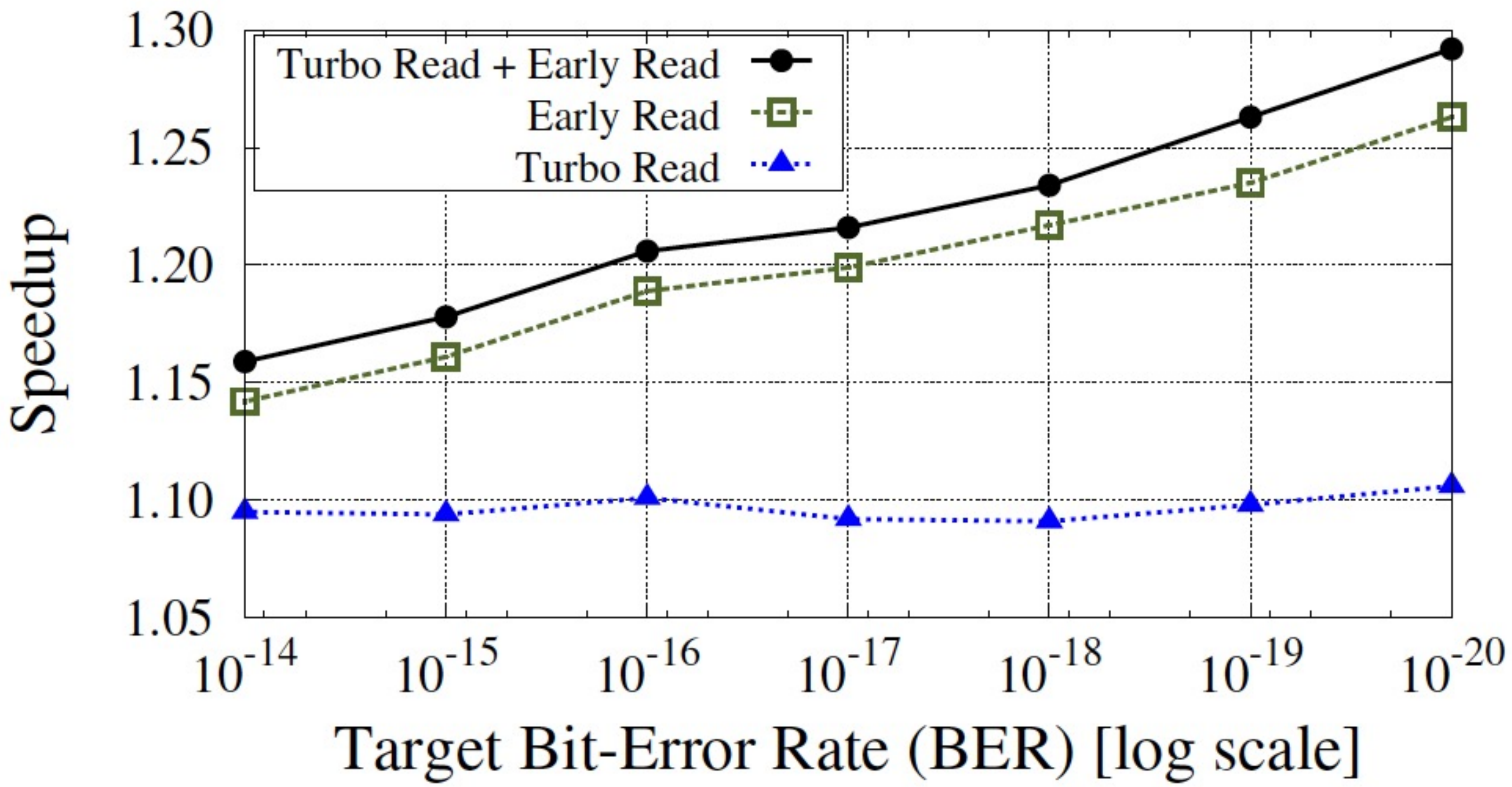
BEWARE OF THE WORST-CASE SCENARIO
You could do so much more by thinking of "Better-Than-the-Worst-Case"!

BACKUP

SENSITIVITY TO TARGET ERROR RATES

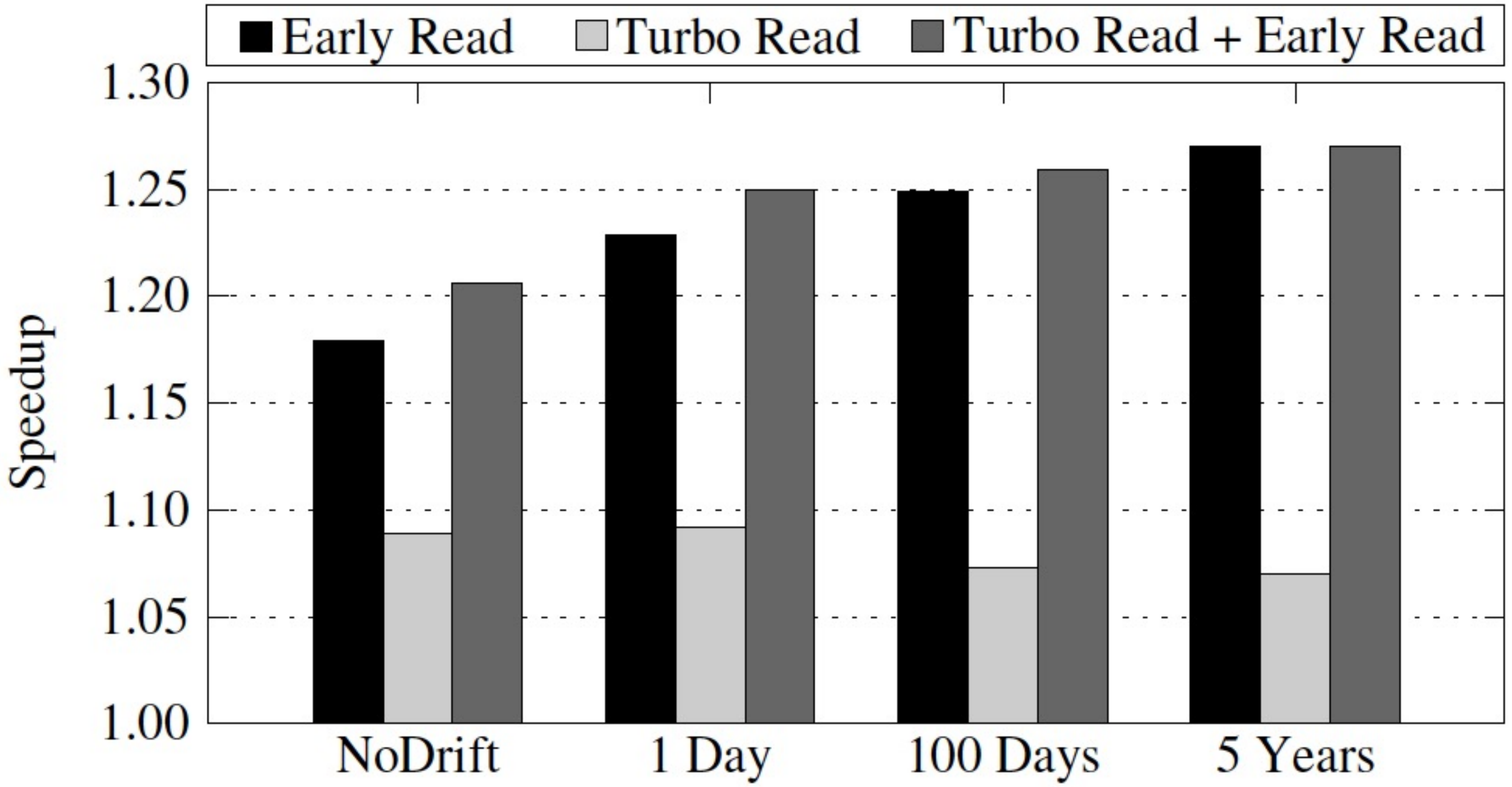


SENSITIVITY TO TARGET ERROR RATES

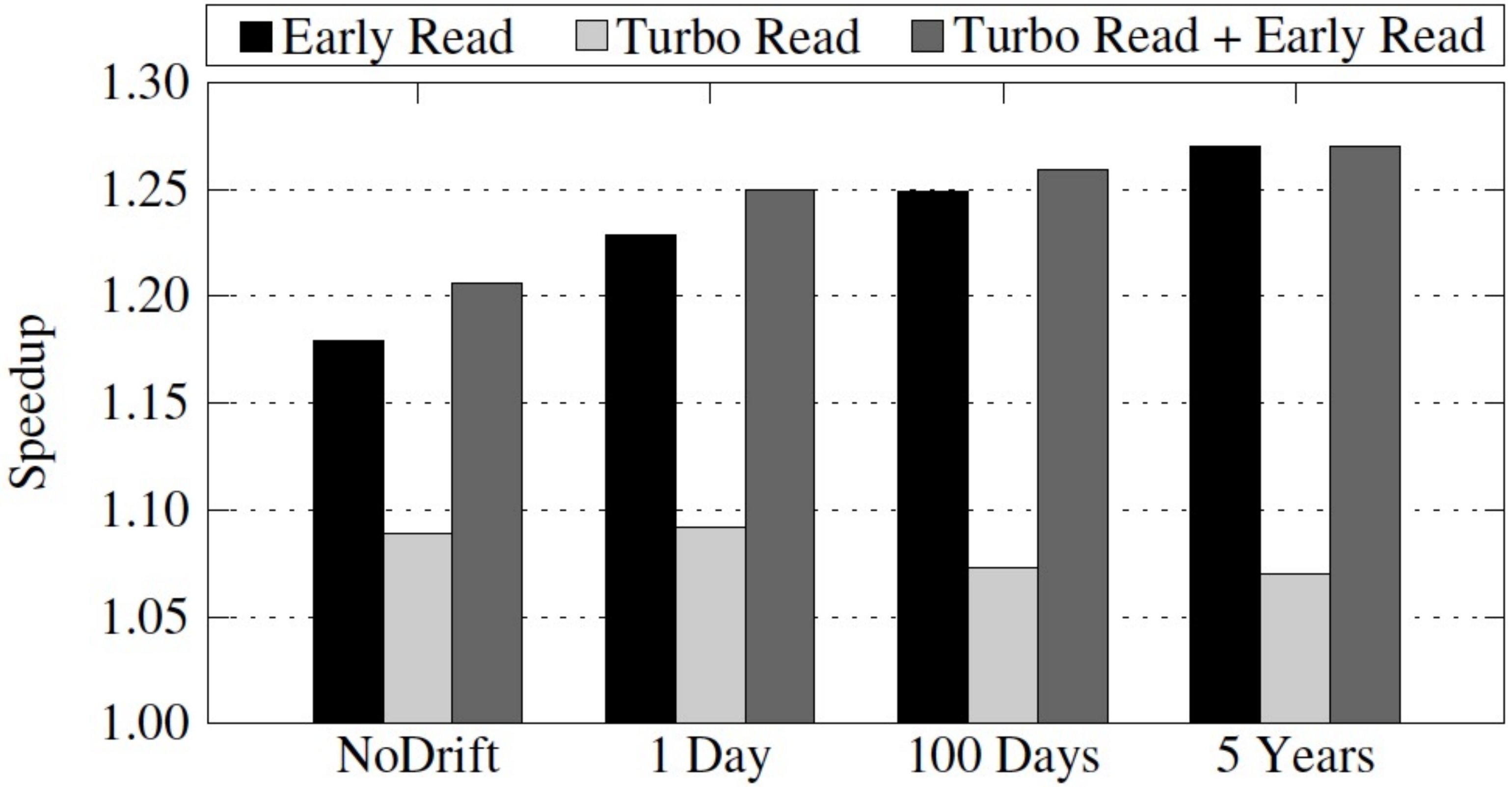


Our proposals become even more effective at higher target design error rates

SENSITIVITY TO DRIFT

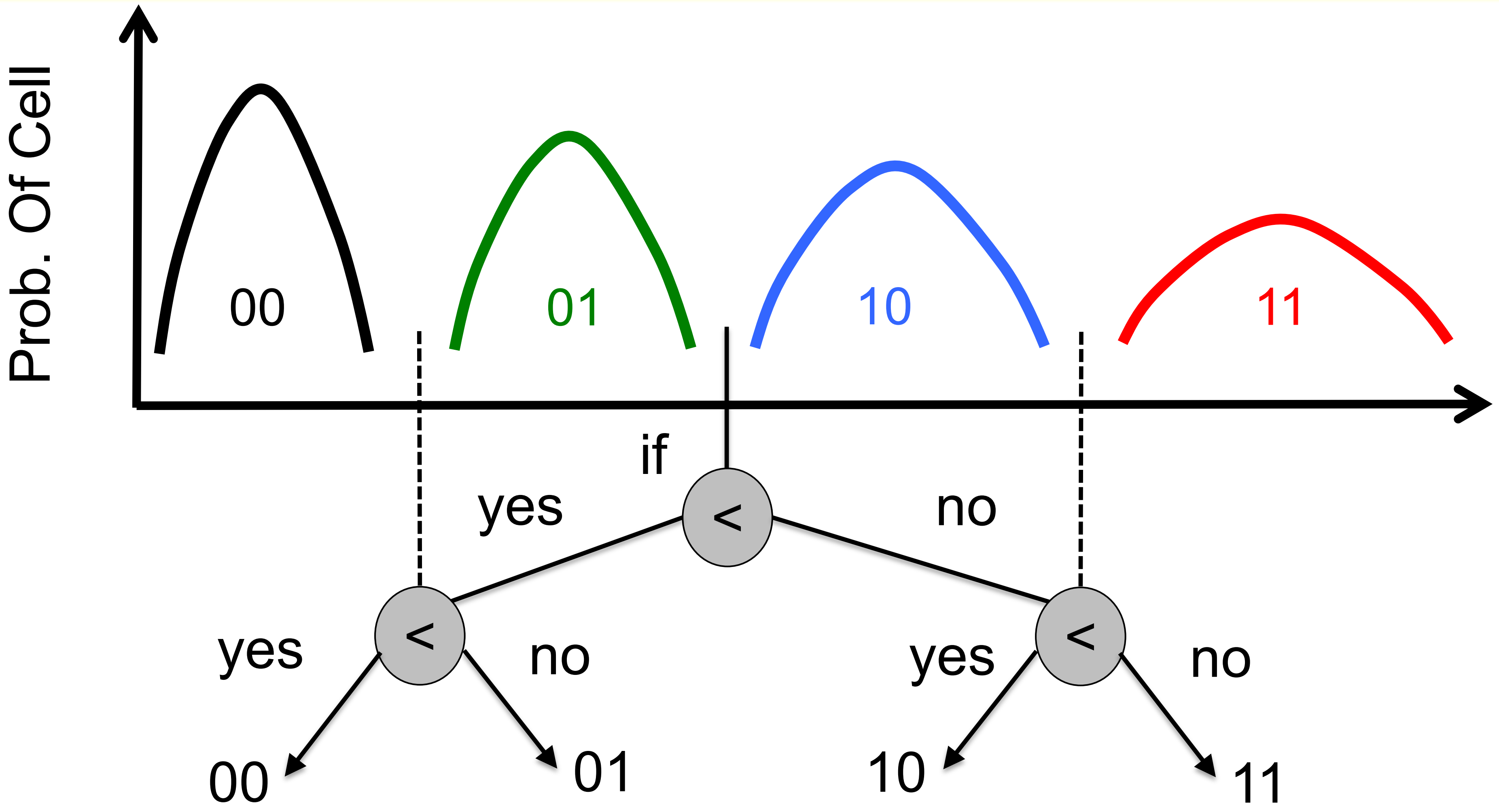


SENSITIVITY TO DRIFT

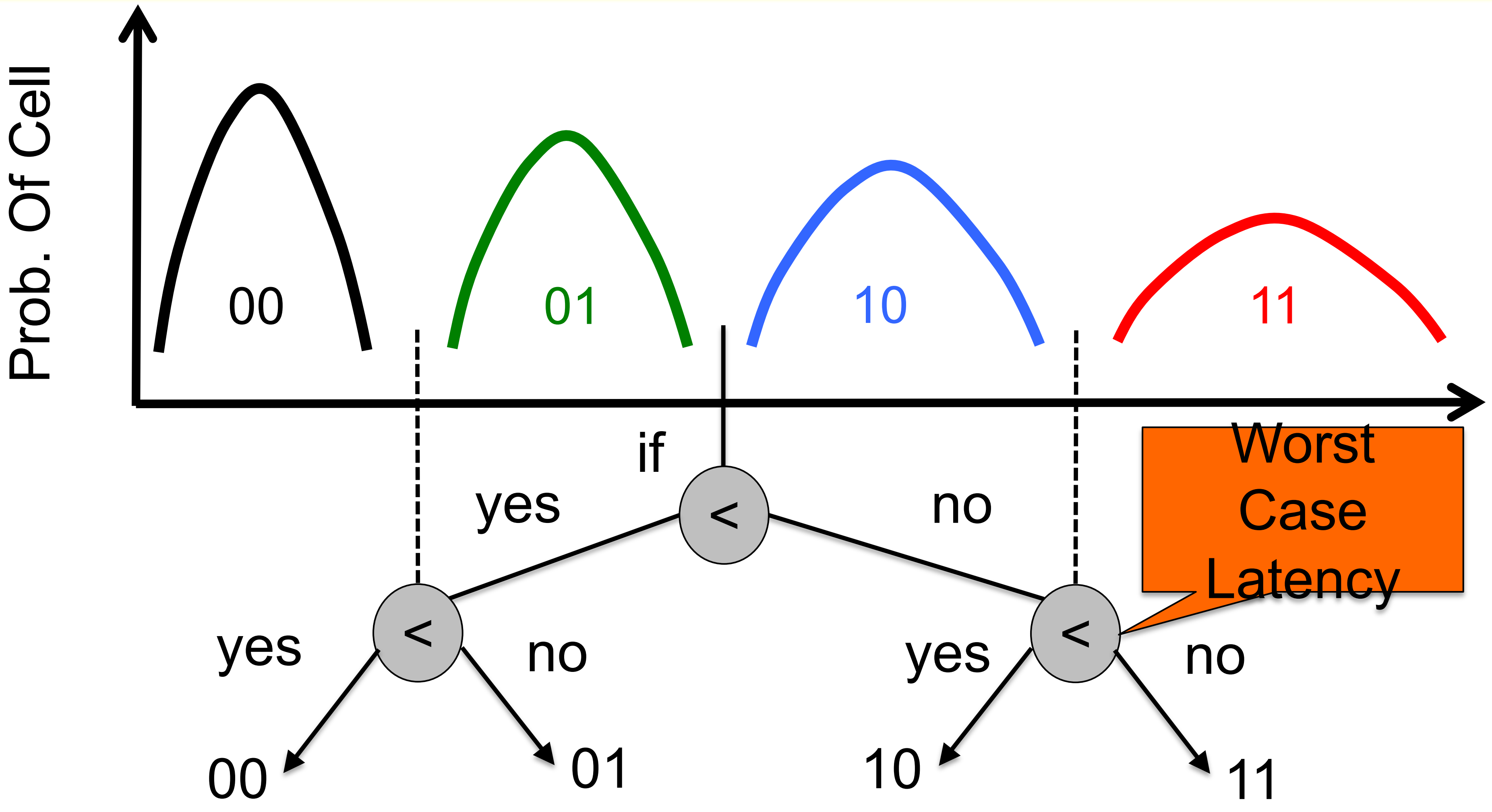


Our proposals become even more effective when drift margins are taken into account

MLC PCM LATENCY



MLC PCM LATENCY



Latency determined by highest resistance states

LATENCY TREND WITH SCALING

- PCM stores values by varying resistance
- Higher resistance causes more read latency

LATENCY TREND WITH SCALING

- PCM stores values by varying resistance
- Higher resistance causes more read latency

Resistance = (Resistivity x Length)/Area

LATENCY TREND WITH SCALING

- PCM stores values by varying resistance
- Higher resistance causes more read latency
- With Technology Scaling:

  Resistance = (Resistivity x Length ↓)/Area

LATENCY TREND WITH SCALING

- PCM stores values by varying resistance
- Higher resistance causes more read latency
- With Technology Scaling:

  Resistance = (Resistivity x Length ↓)/Area ↓↓

LATENCY TREND WITH SCALING

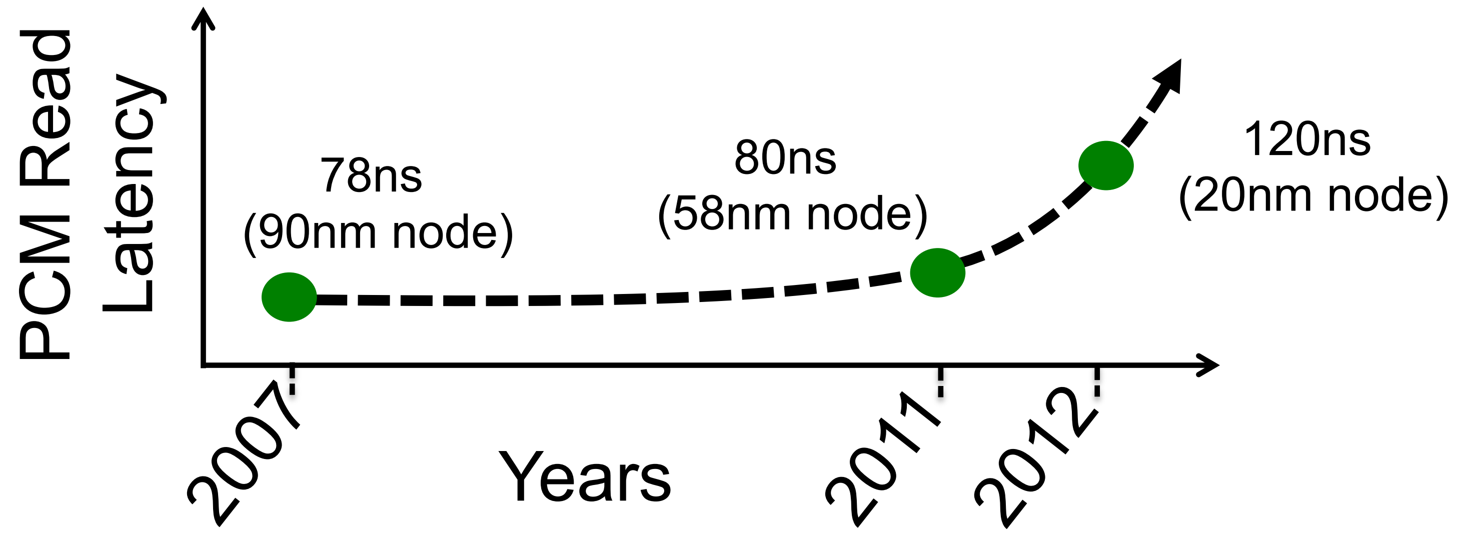
- PCM stores values by varying resistance
- Higher resistance causes more read latency
- With Technology Scaling:

  Resistance ⬆ = (Resistivity x Length ⬇)/Area ⬇⬇

LATENCY TREND WITH SCALING

- PCM stores values by varying resistance
- Higher resistance causes more read latency
- With Technology Scaling:

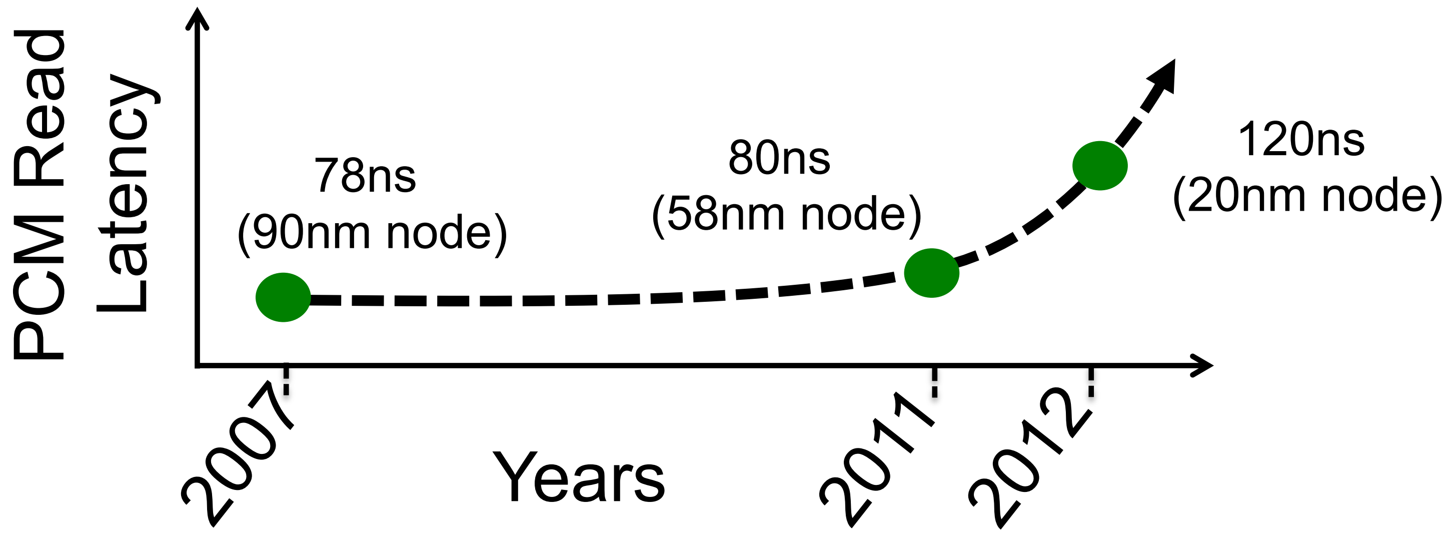
  Resistance ⬆ = (Resistivity x Length ⬇)/Area ⬇⬇



LATENCY TREND WITH SCALING

- PCM stores values by varying resistance
- Higher resistance causes more read latency
- With Technology Scaling:

  Resistance ⬆ = (Resistivity x Length ⬇)/Area ⬇⬇



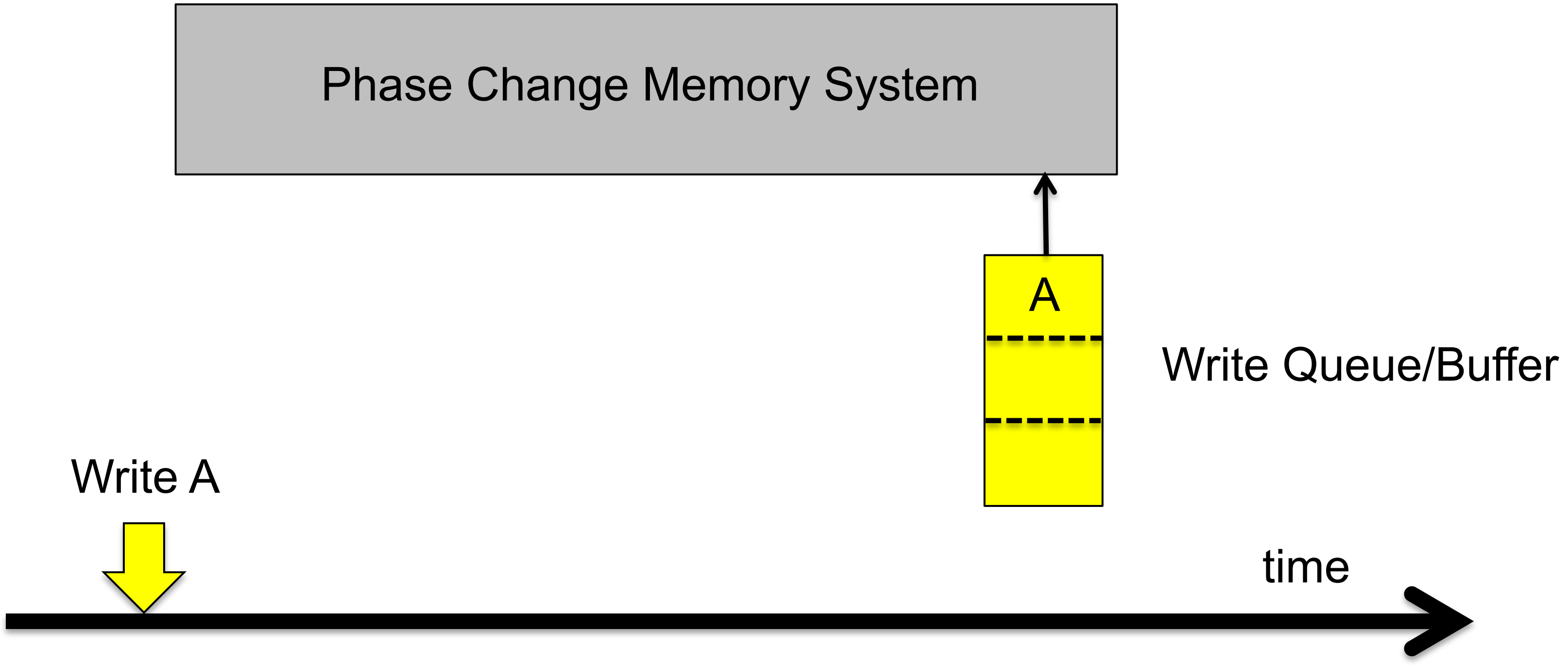
PCM read latency increases with scaling

REDUCING READ LATENCY MATTERS

- Read requests tend to halt execution
- Write requests can be buffered/paused/cancelled

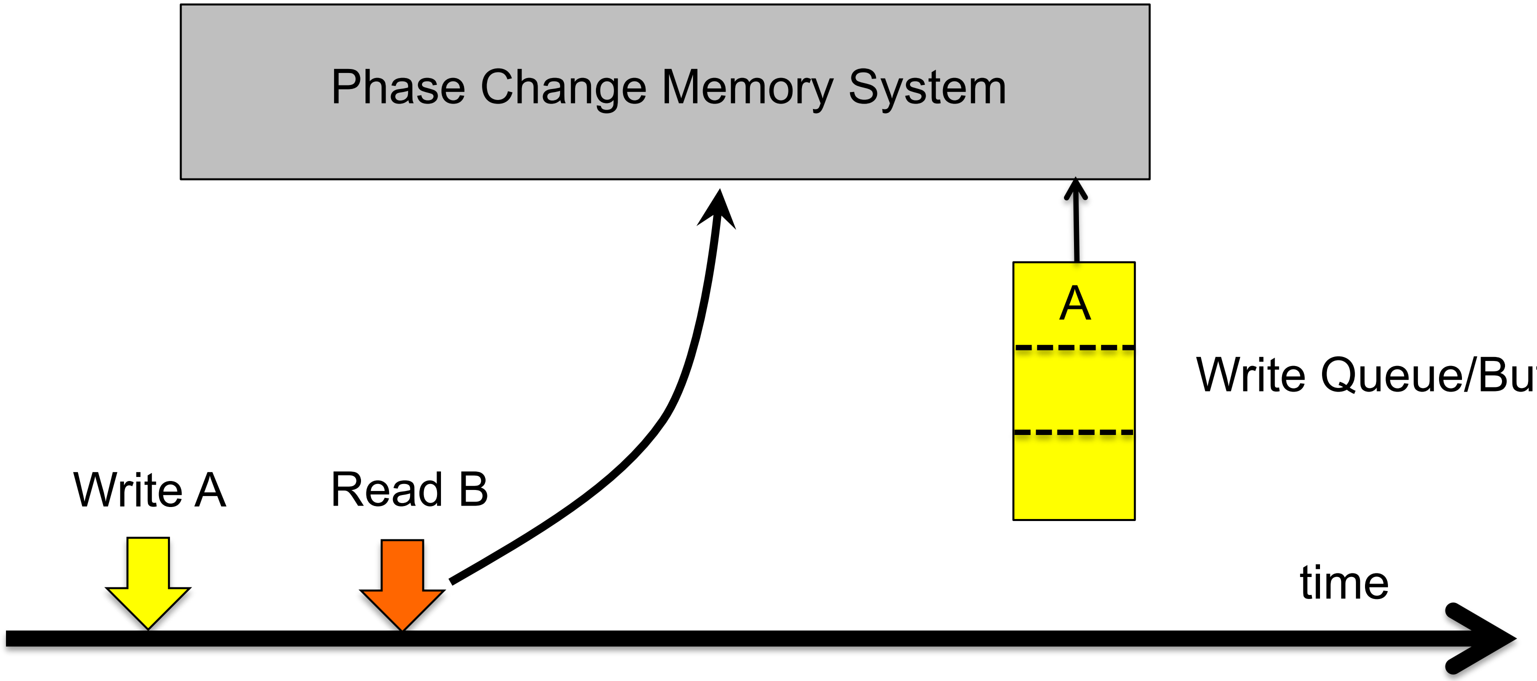
REDUCING READ LATENCY MATTERS

- Read requests tend to halt execution
- Write requests can be buffered/paused/cancelled



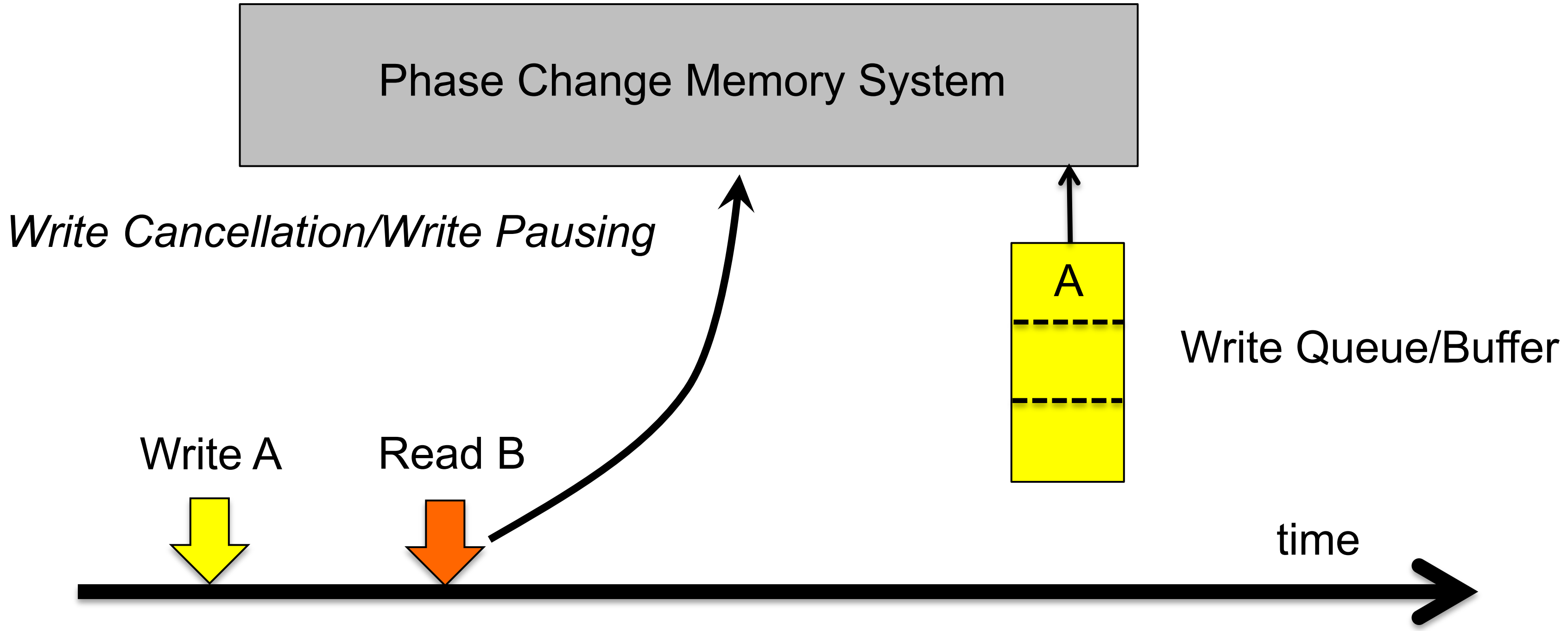
REDUCING READ LATENCY MATTERS

- Read requests tend to halt execution
- Write requests can be buffered/paused/cancelled



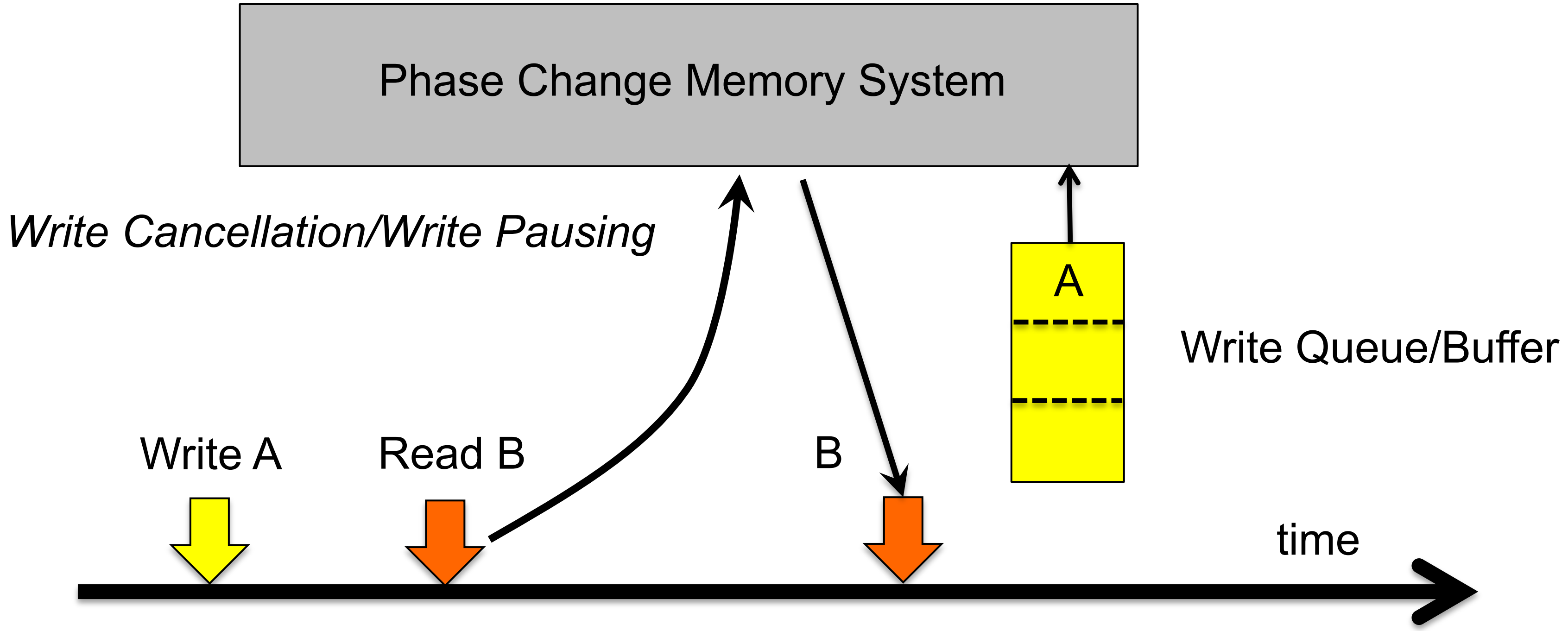
REDUCING READ LATENCY MATTERS

- Read requests tend to halt execution
- Write requests can be buffered/paused/cancelled


Phase Change Memory System
Write Cancellation/Write Pausing
A
Write Queue/Buffer
Write A
Read B
time

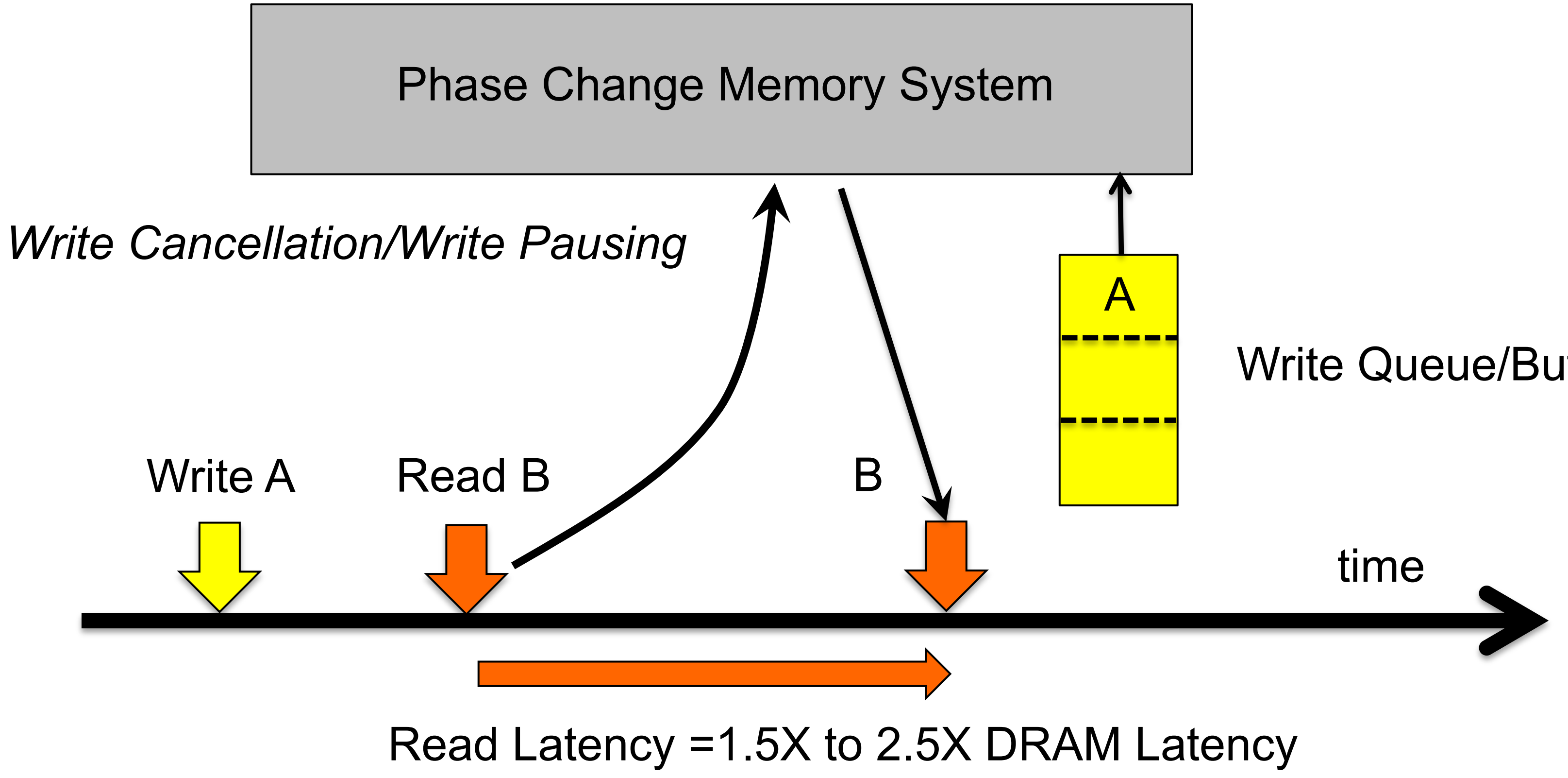
REDUCING READ LATENCY MATTERS

- Read requests tend to halt execution
- Write requests can be buffered/paused/cancelled



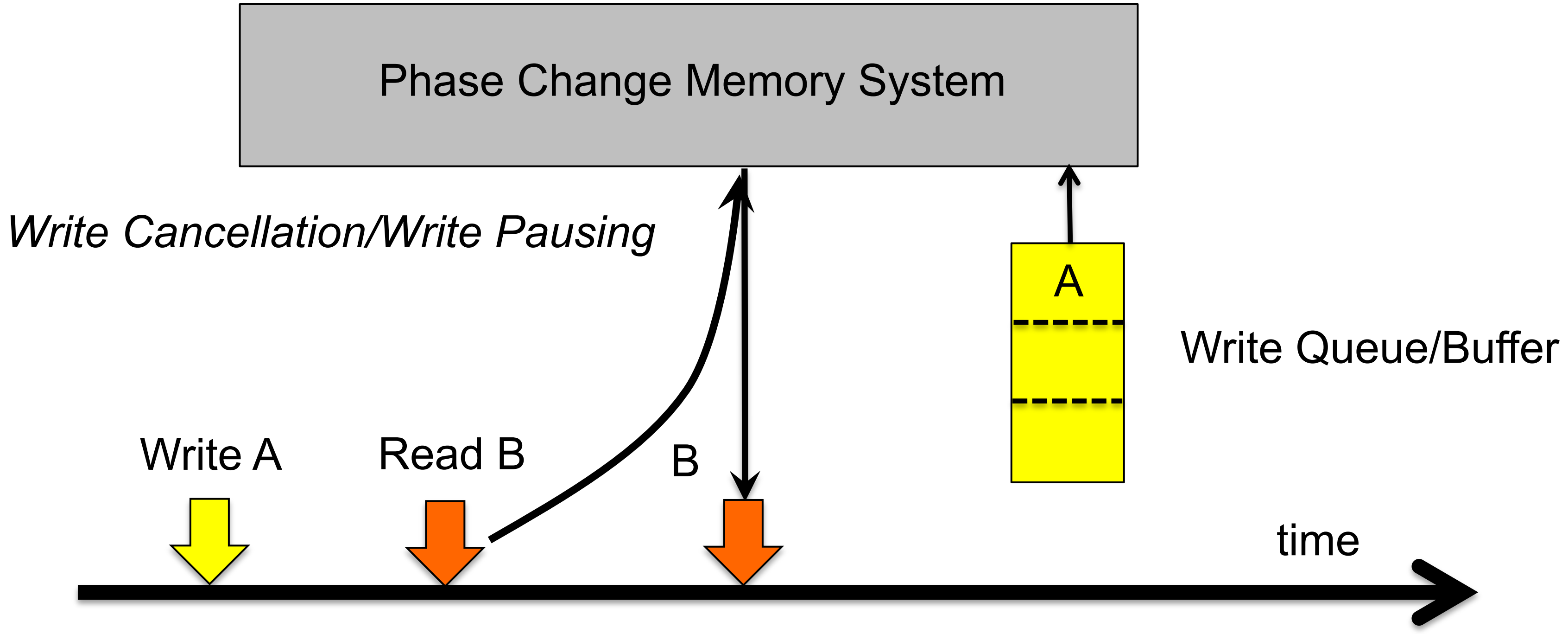
REDUCING READ LATENCY MATTERS

- Read requests tend to halt execution
- Write requests can be buffered/paused/cancelled


Phase Change Memory System
Write Cancellation/Write Pausing
A
Write Queue/Buffer
Write A
Read B
B
time
Read Latency =1.5X to 2.5X DRAM Latency

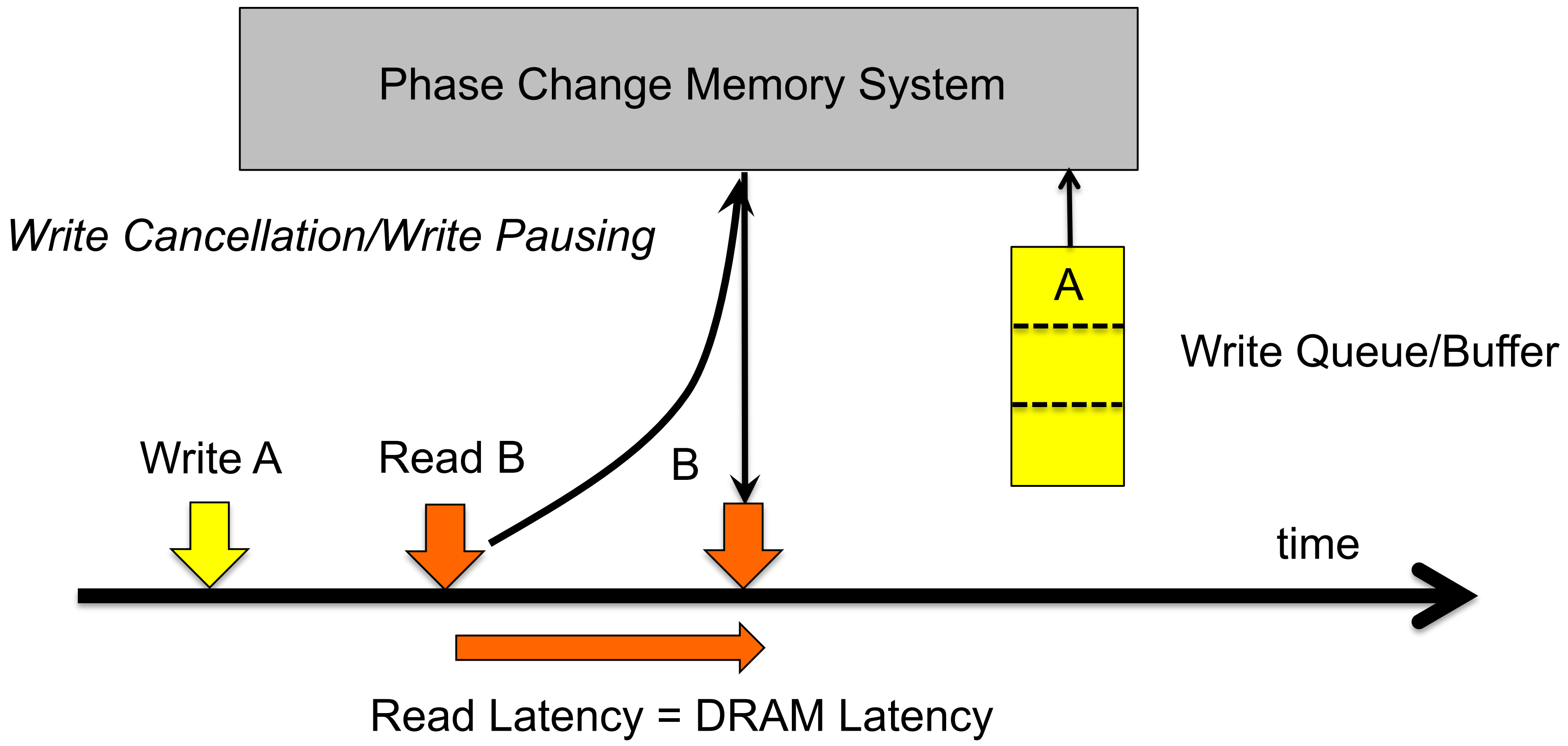
REDUCING READ LATENCY MATTERS

- Read requests tend to halt execution
- Write requests can be buffered/paused/cancelled



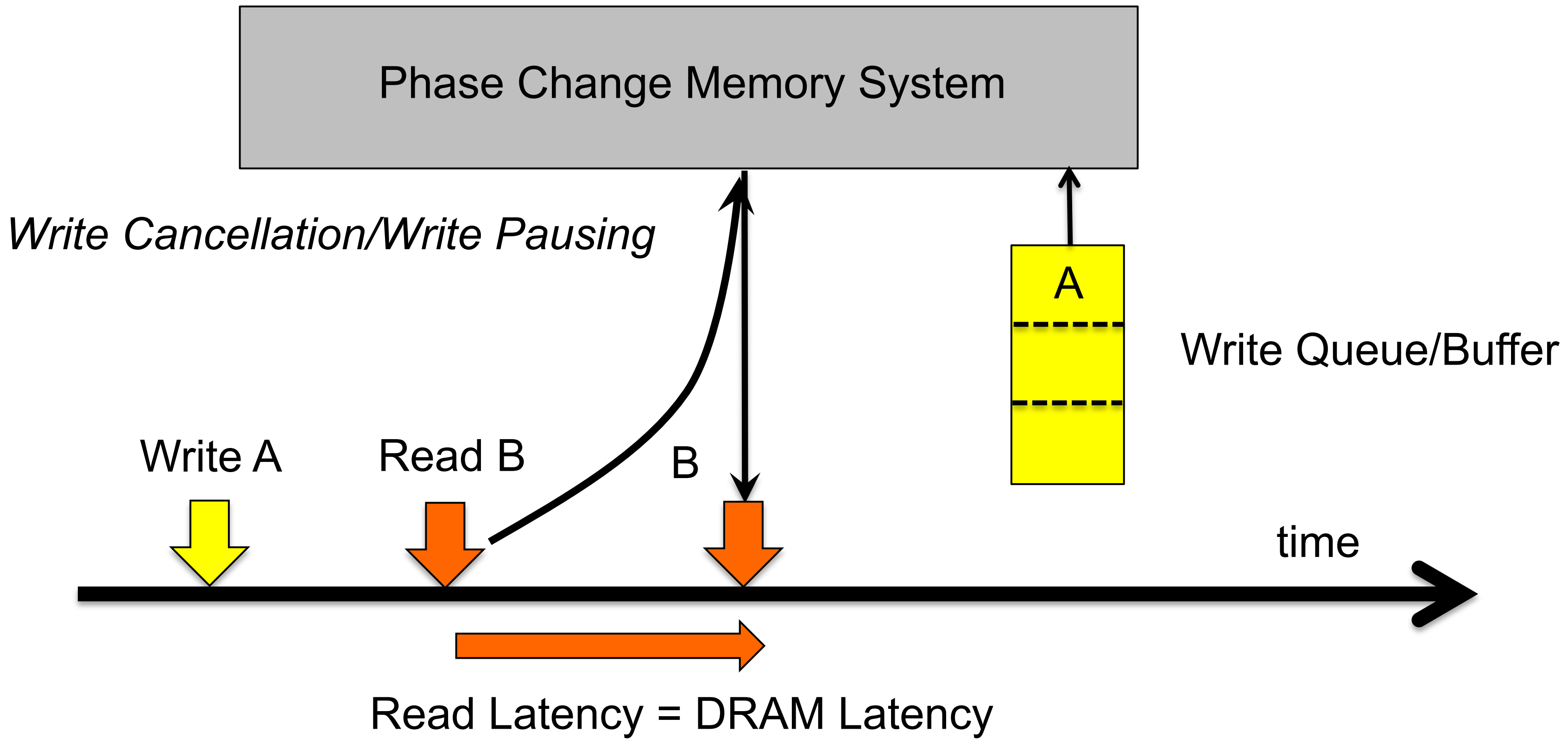
REDUCING READ LATENCY MATTERS

- Read requests tend to halt execution
- Write requests can be buffered/paused/cancelled


Phase Change Memory System
Write Cancellation/Write Pausing
A
Write Queue/Buffer
Write A
Read B
B
time
Read Latency = DRAM Latency

REDUCING READ LATENCY MATTERS

- Read requests tend to halt execution
- Write requests can be buffered/paused/cancelled

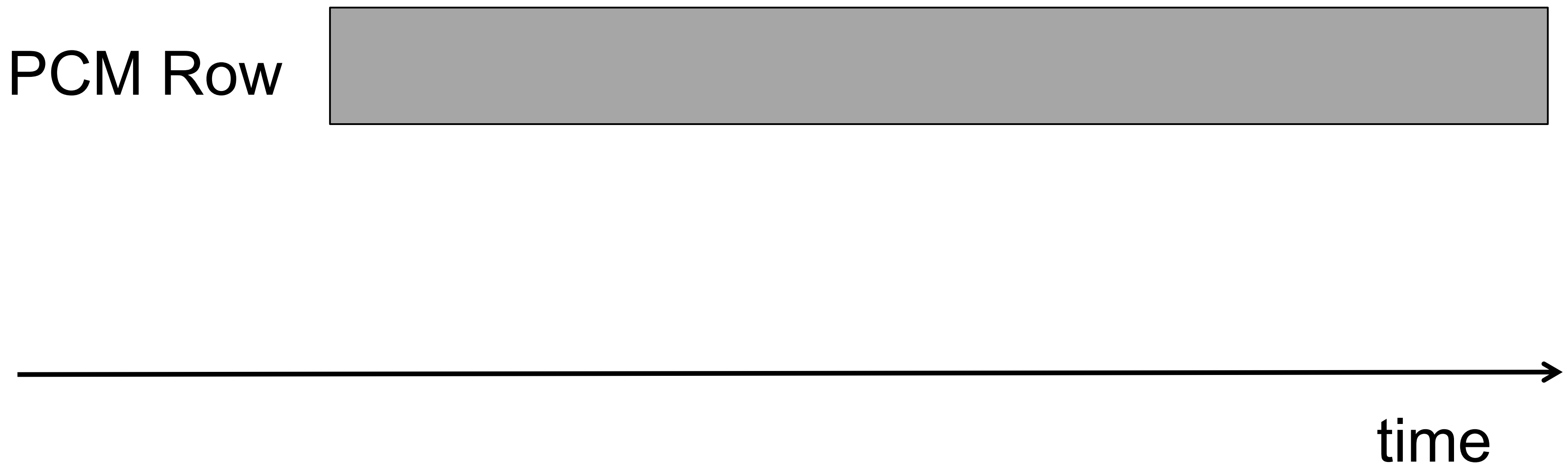


Phase Change Memory System
Write Cancellation/Write Pausing
A
Write Queue/Buffer
Write A
Read B
B
time
Read Latency = DRAM Latency

Low read latency improves performance directly

LATENT FAULTS FROM READ DISTURB

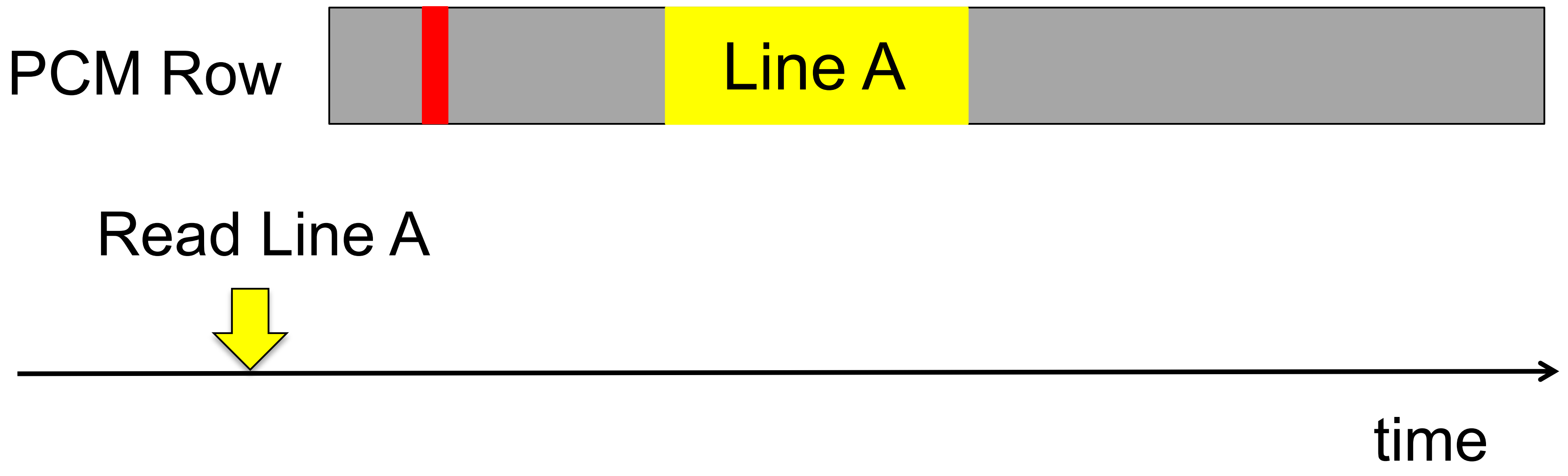
- Adversarial read sequences can cause latent faults



Need a low cost solution to mitigate latent faults

LATENT FAULTS FROM READ DISTURB

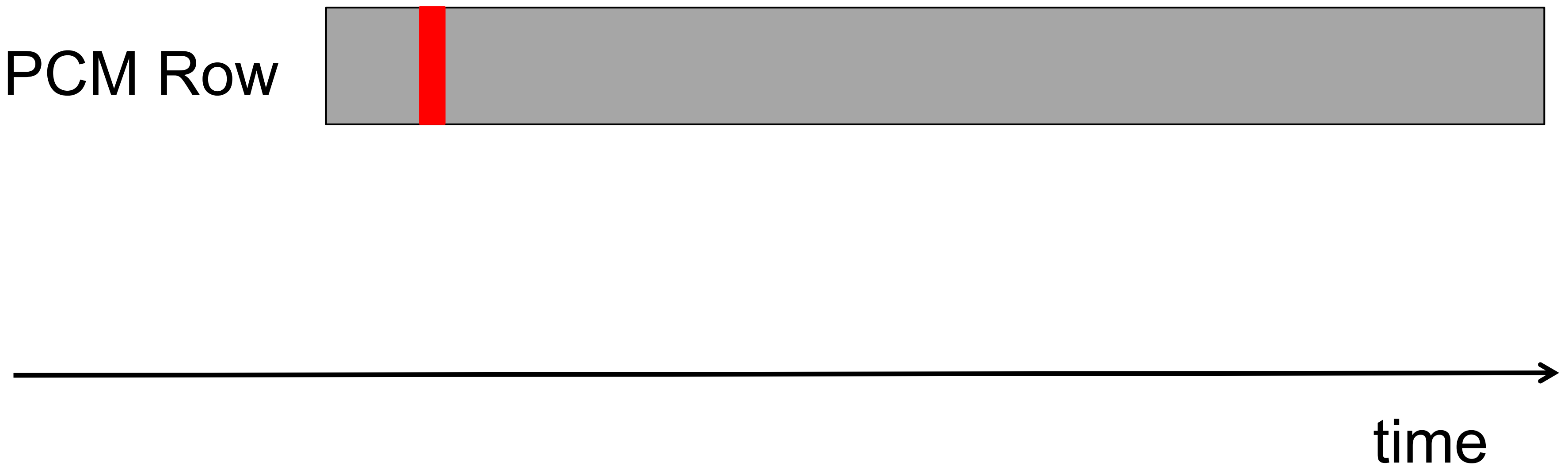
- Adversarial read sequences can cause latent faults



Need a low cost solution to mitigate latent faults

LATENT FAULTS FROM READ DISTURB

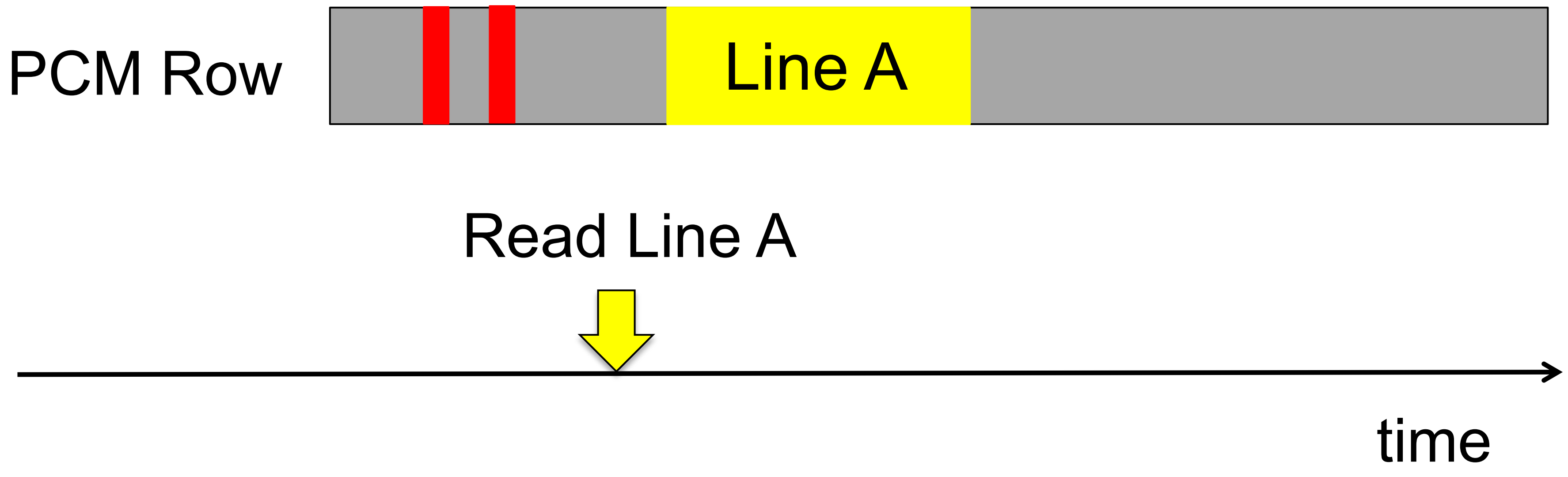
- Adversarial read sequences can cause latent faults



Need a low cost solution to mitigate latent faults

LATENT FAULTS FROM READ DISTURB

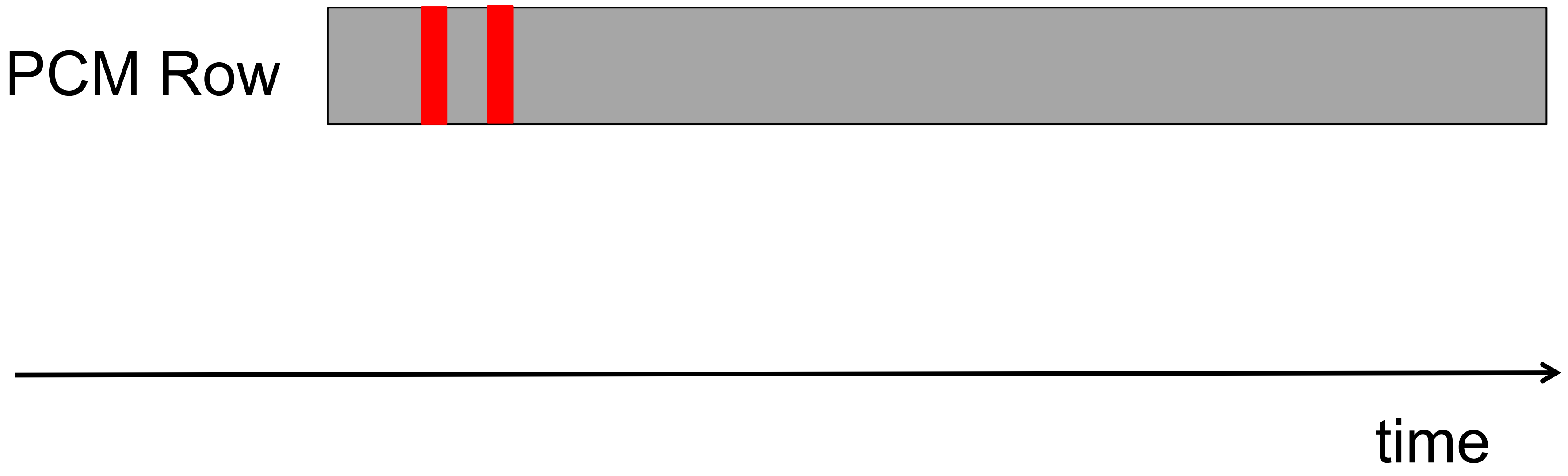
- Adversarial read sequences can cause latent faults



Need a low cost solution to mitigate latent faults

LATENT FAULTS FROM READ DISTURB

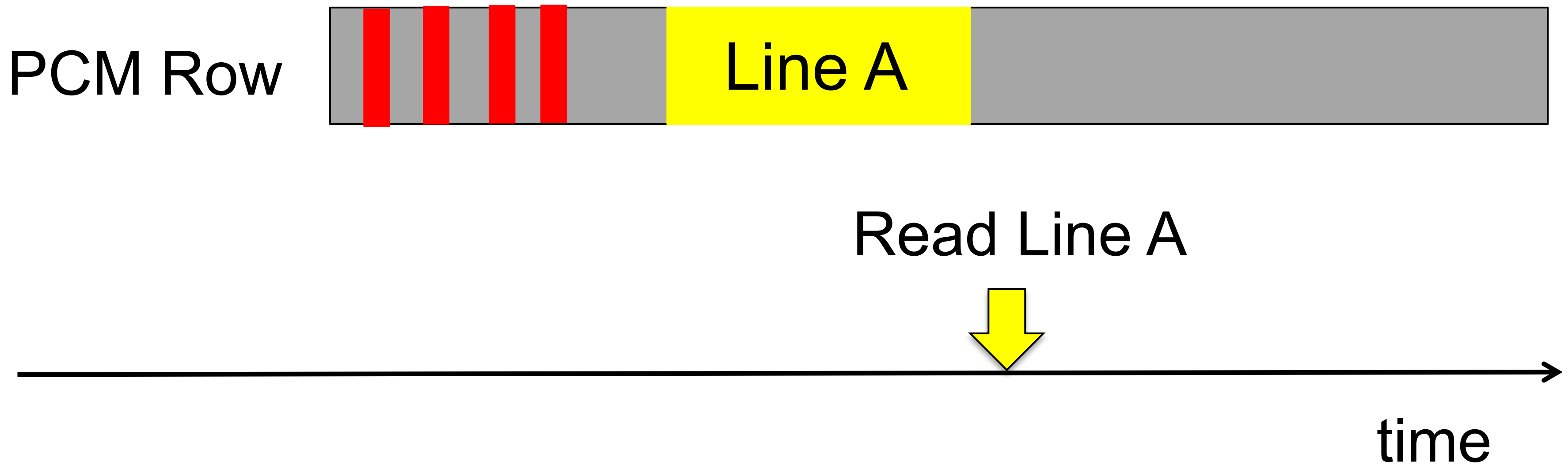
- Adversarial read sequences can cause latent faults



Need a low cost solution to mitigate latent faults

LATENT FAULTS FROM READ DISTURB

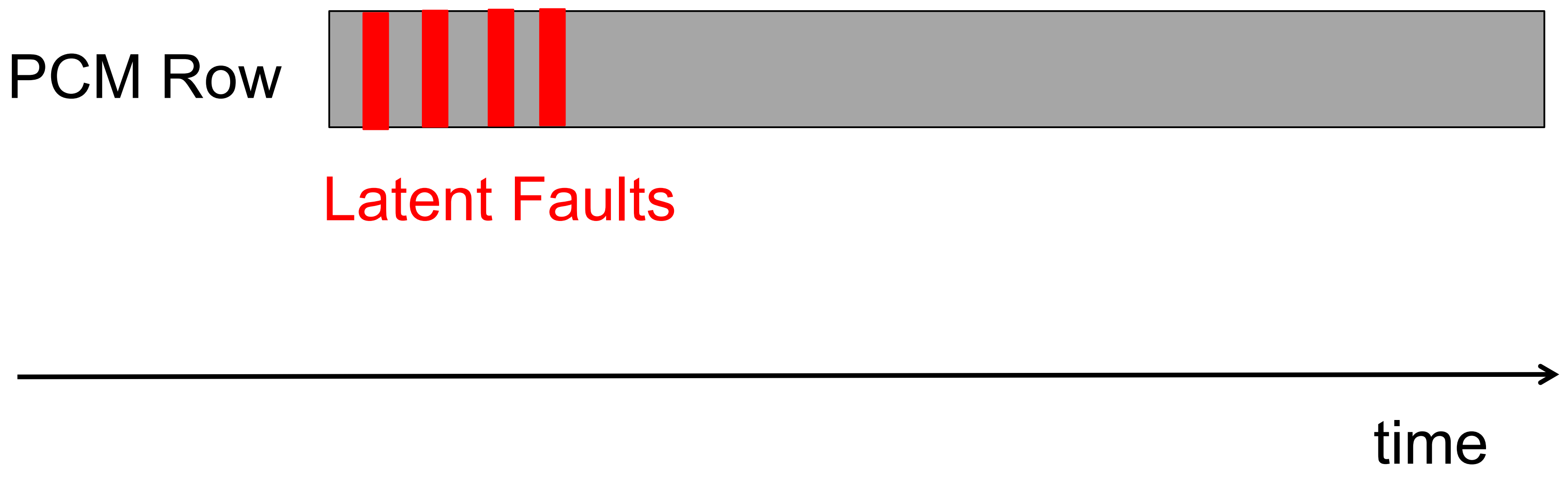
- Adversarial read sequences can cause latent faults



Need a low cost solution to mitigate latent faults

LATENT FAULTS FROM READ DISTURB

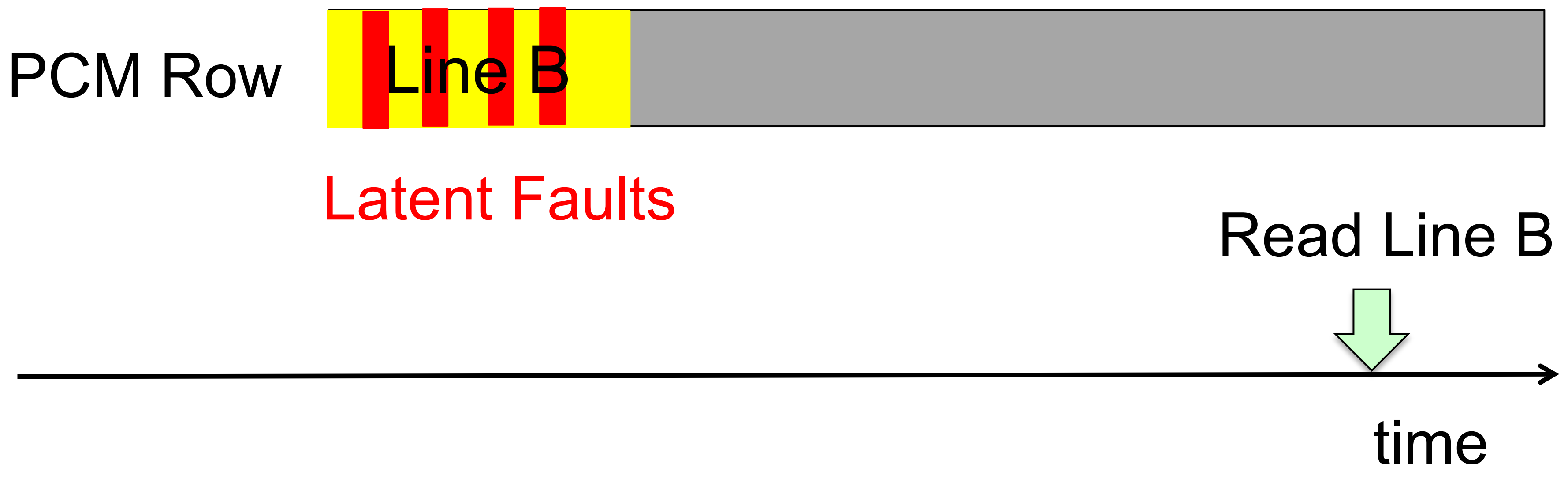
- Adversarial read sequences can cause latent faults



Need a low cost solution to mitigate latent faults

LATENT FAULTS FROM READ DISTURB

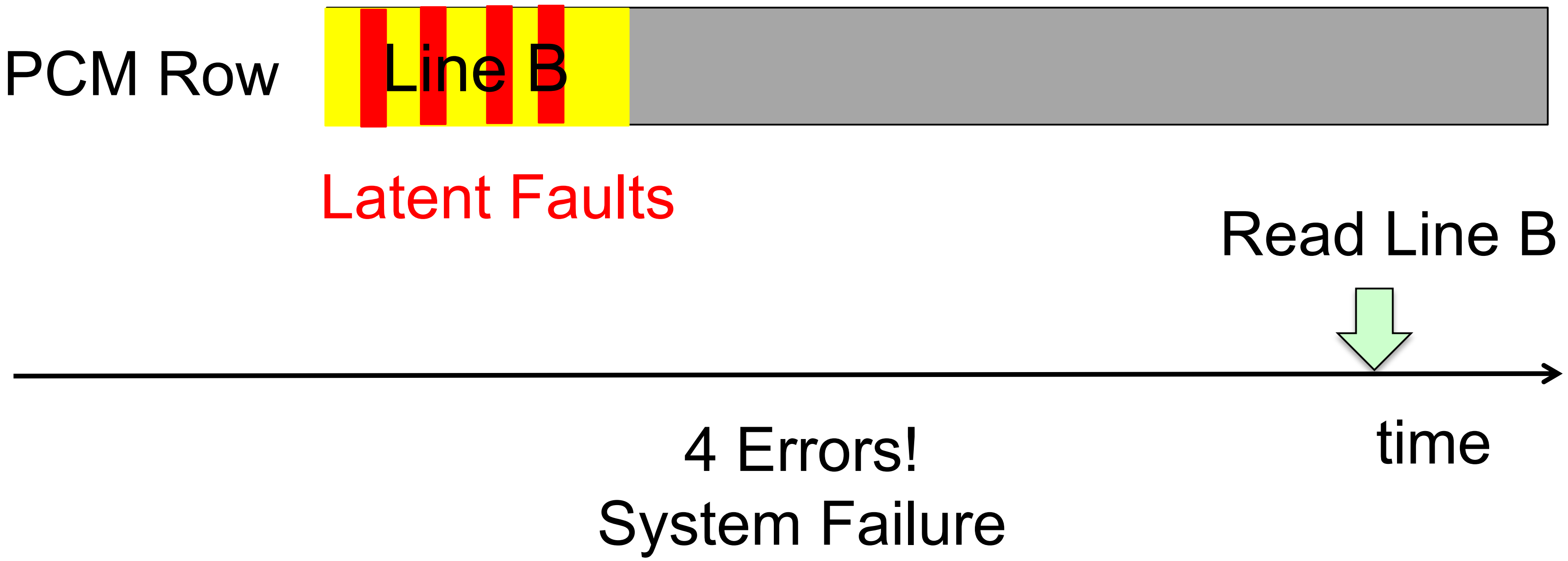
- Adversarial read sequences can cause latent faults



Need a low cost solution to mitigate latent faults

LATENT FAULTS FROM READ DISTURB

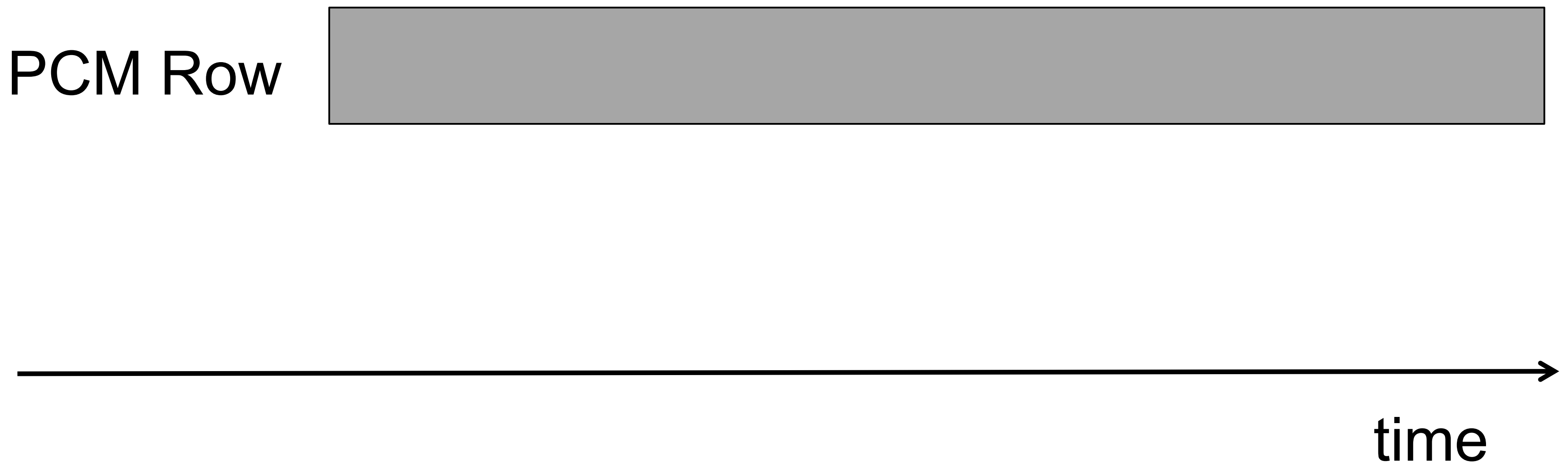
- Adversarial read sequences can cause latent faults



Need a low cost solution to mitigate latent faults

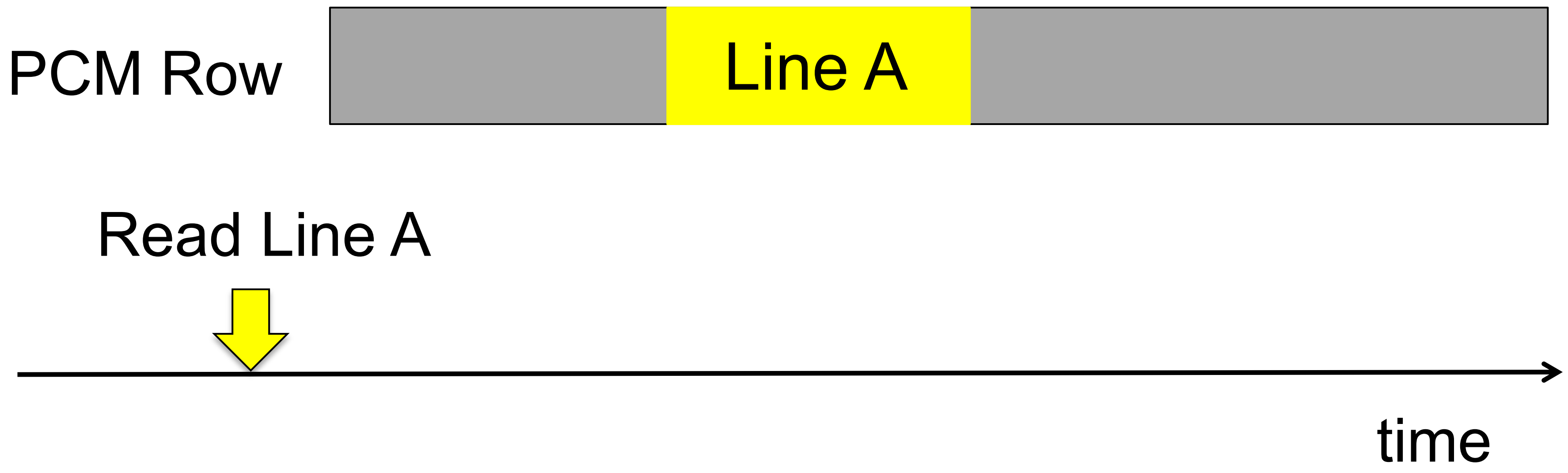
OUR SOLUTION: PROBABILISTIC SCRUB

- Scrub the entire row with low probability (say 1%)



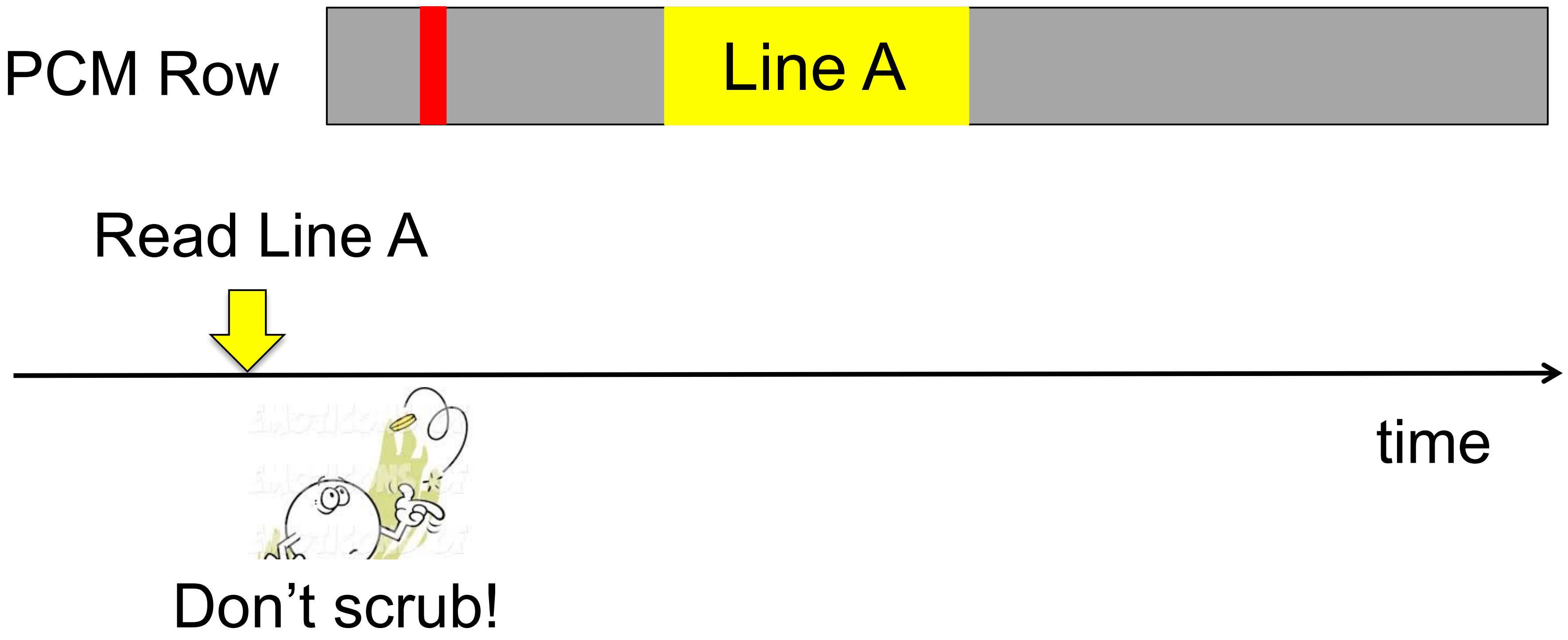
OUR SOLUTION: PROBABILISTIC SCRUB

- Scrub the entire row with low probability (say 1%)



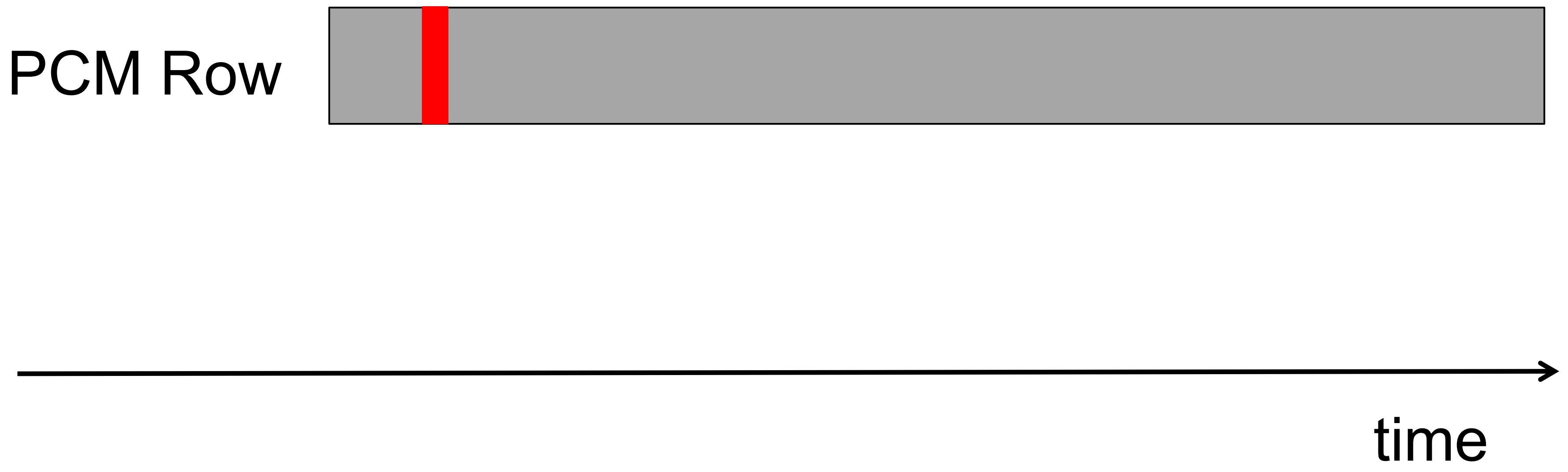
OUR SOLUTION: PROBABILISTIC SCRUB

- Scrub the entire row with low probability (say 1%)



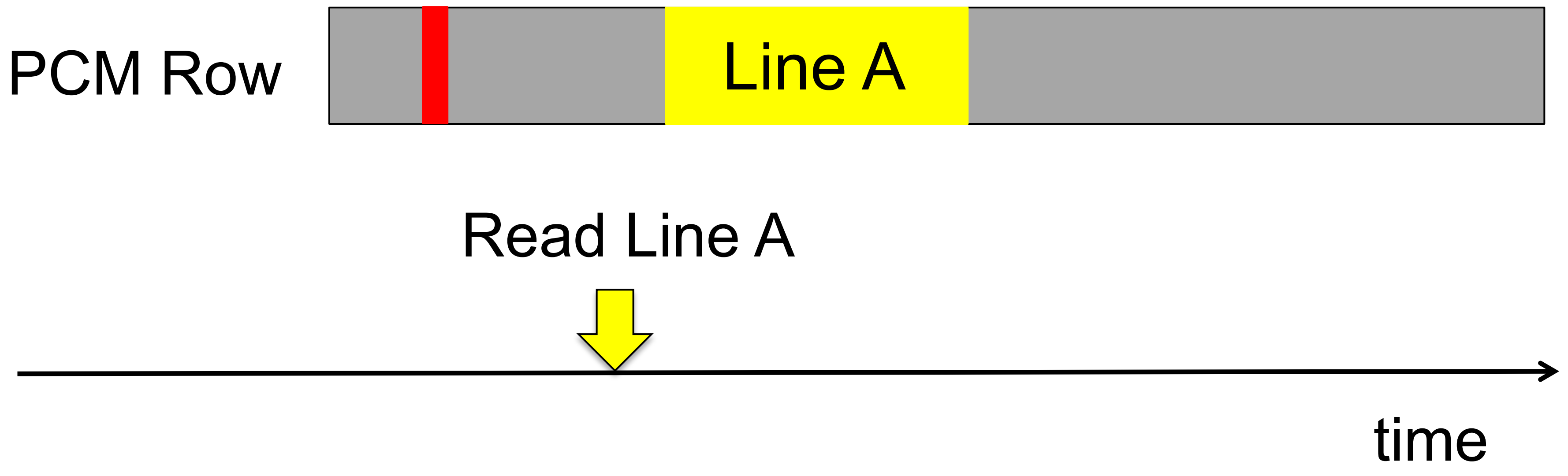
OUR SOLUTION: PROBABILISTIC SCRUB

- Scrub the entire row with low probability (say 1%)



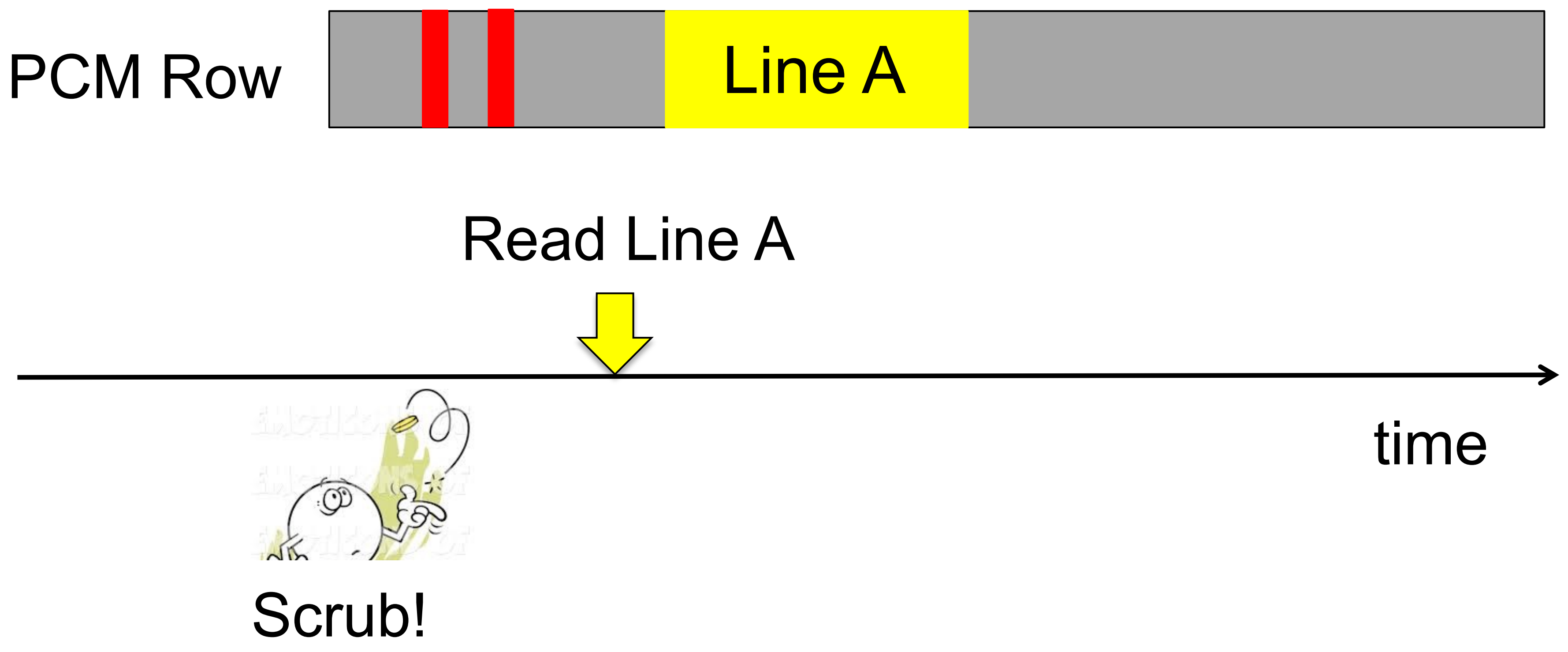
OUR SOLUTION: PROBABILISTIC SCRUB

- Scrub the entire row with low probability (say 1%)



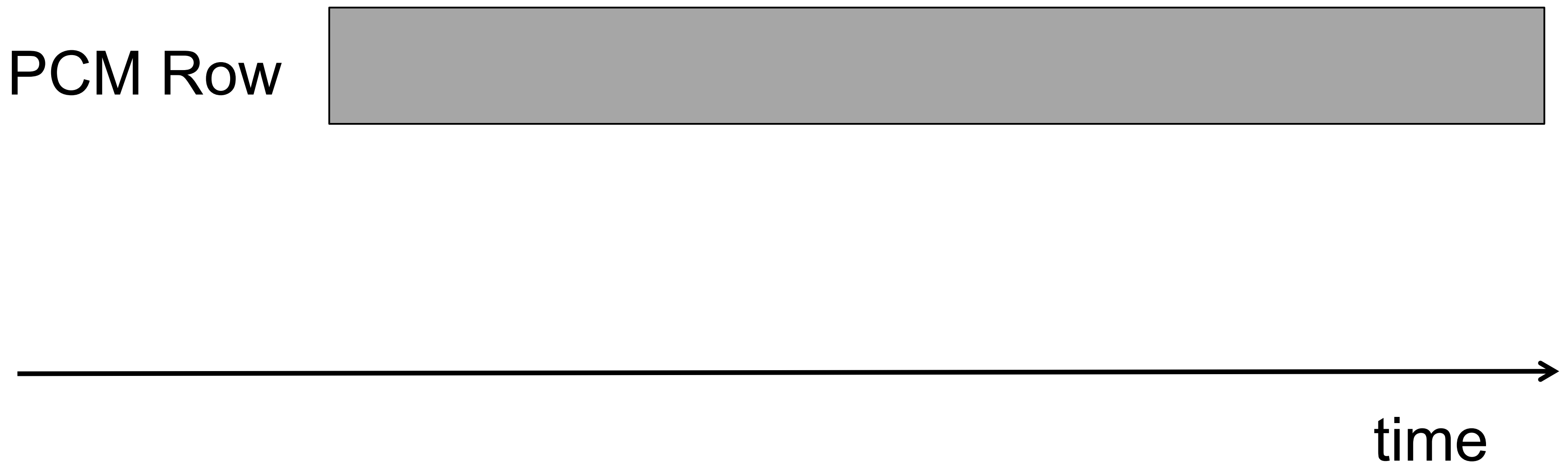
OUR SOLUTION: PROBABILISTIC SCRUB

- Scrub the entire row with low probability (say 1%)



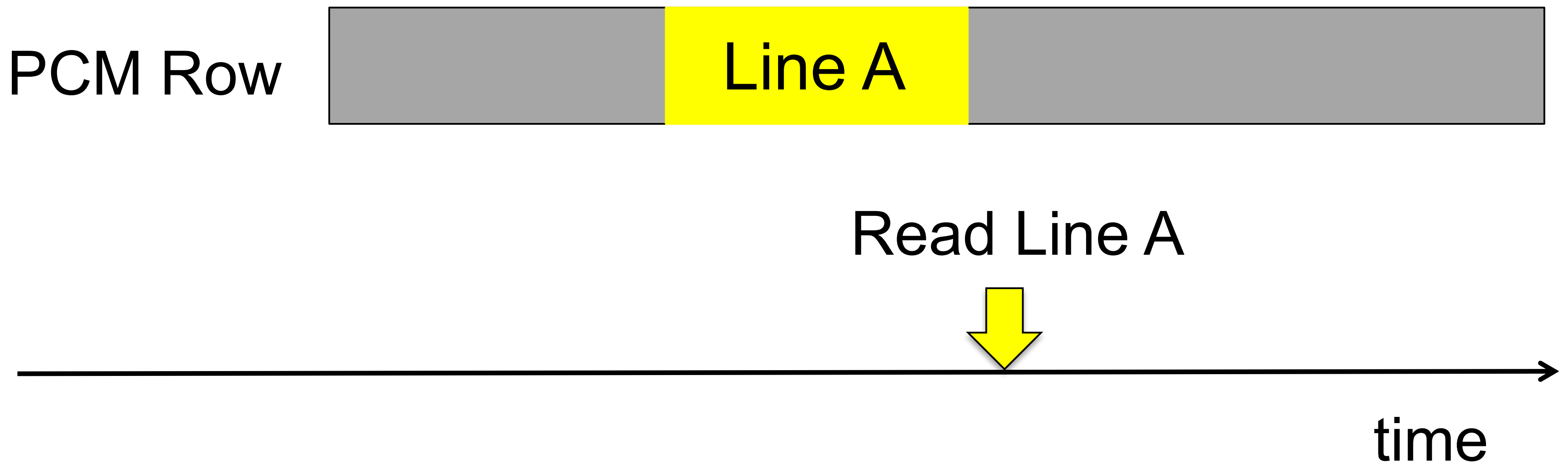
OUR SOLUTION: PROBABILISTIC SCRUB

- Scrub the entire row with low probability (say 1%)



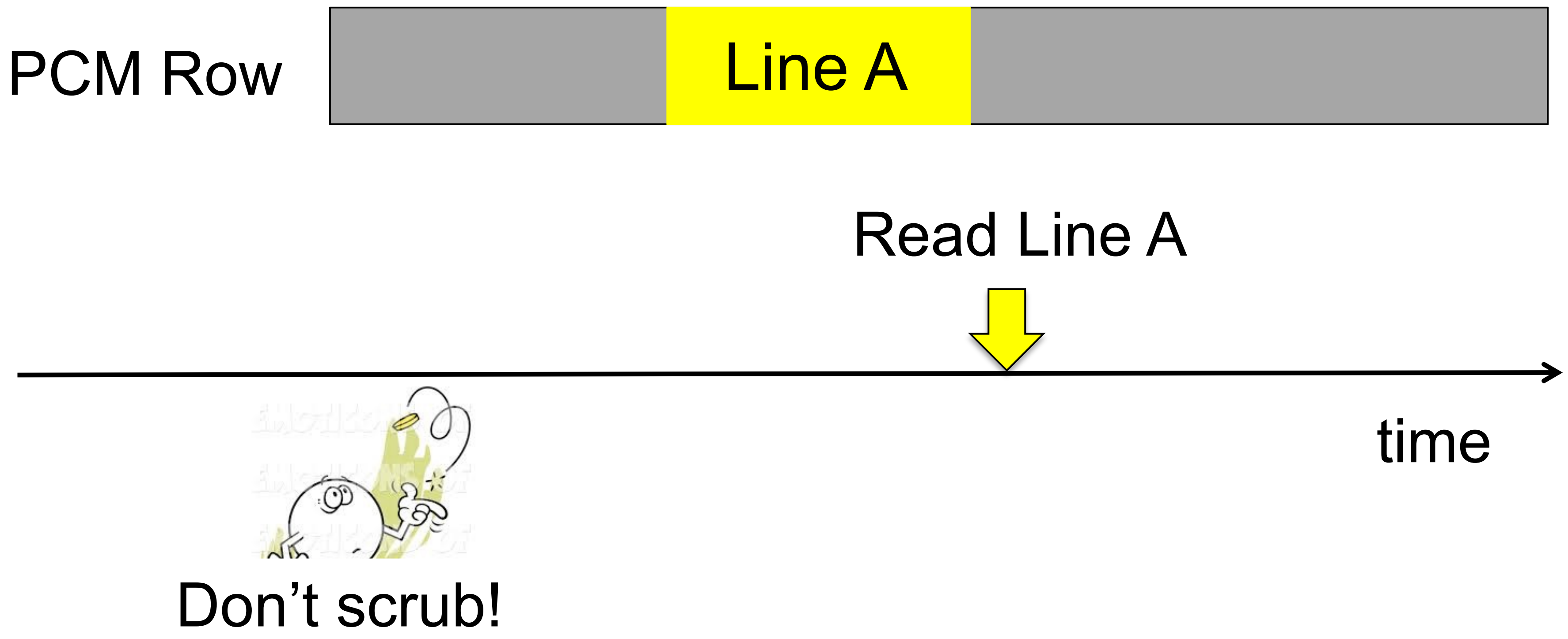
OUR SOLUTION: PROBABILISTIC SCRUB

- Scrub the entire row with low probability (say 1%)



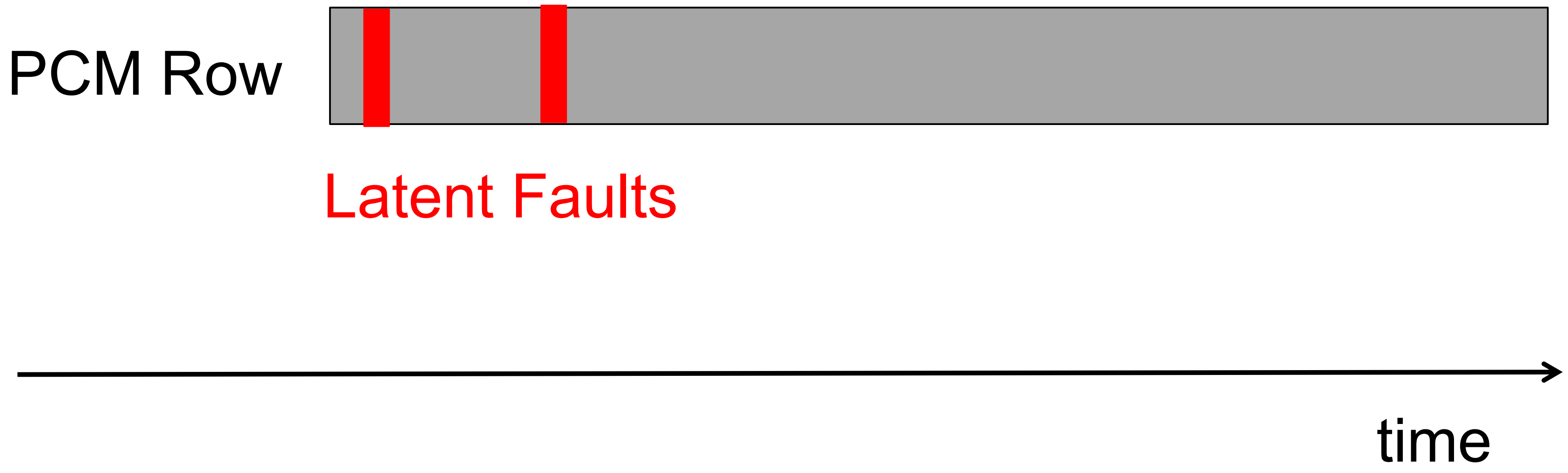
OUR SOLUTION: PROBABILISTIC SCRUB

- Scrub the entire row with low probability (say 1%)



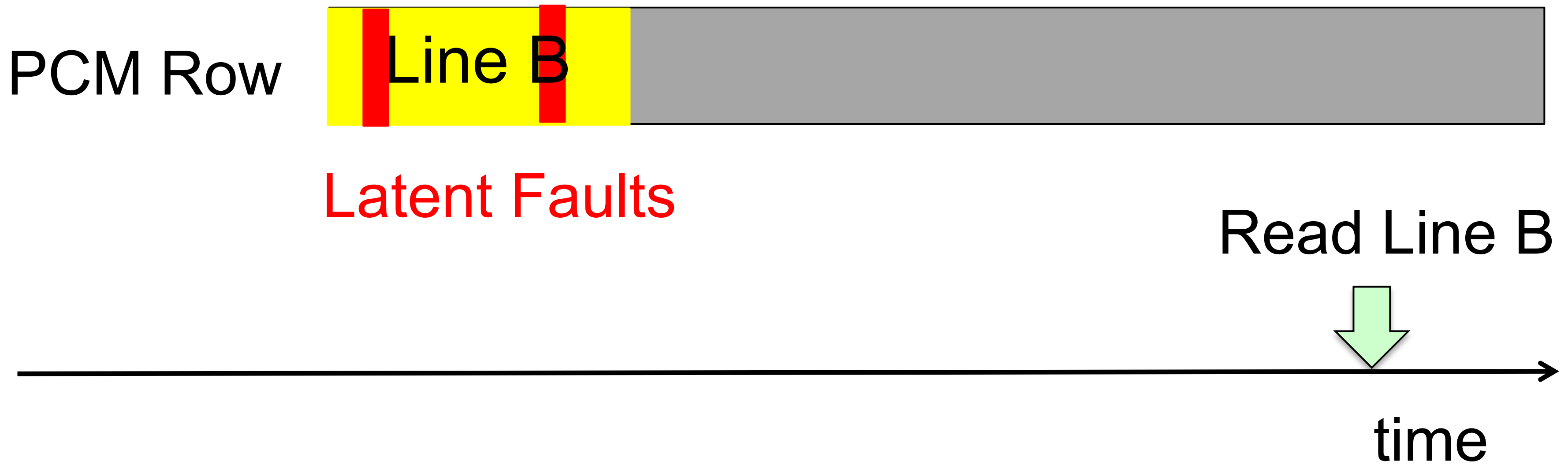
OUR SOLUTION: PROBABILISTIC SCRUB

- Scrub the entire row with low probability (say 1%)



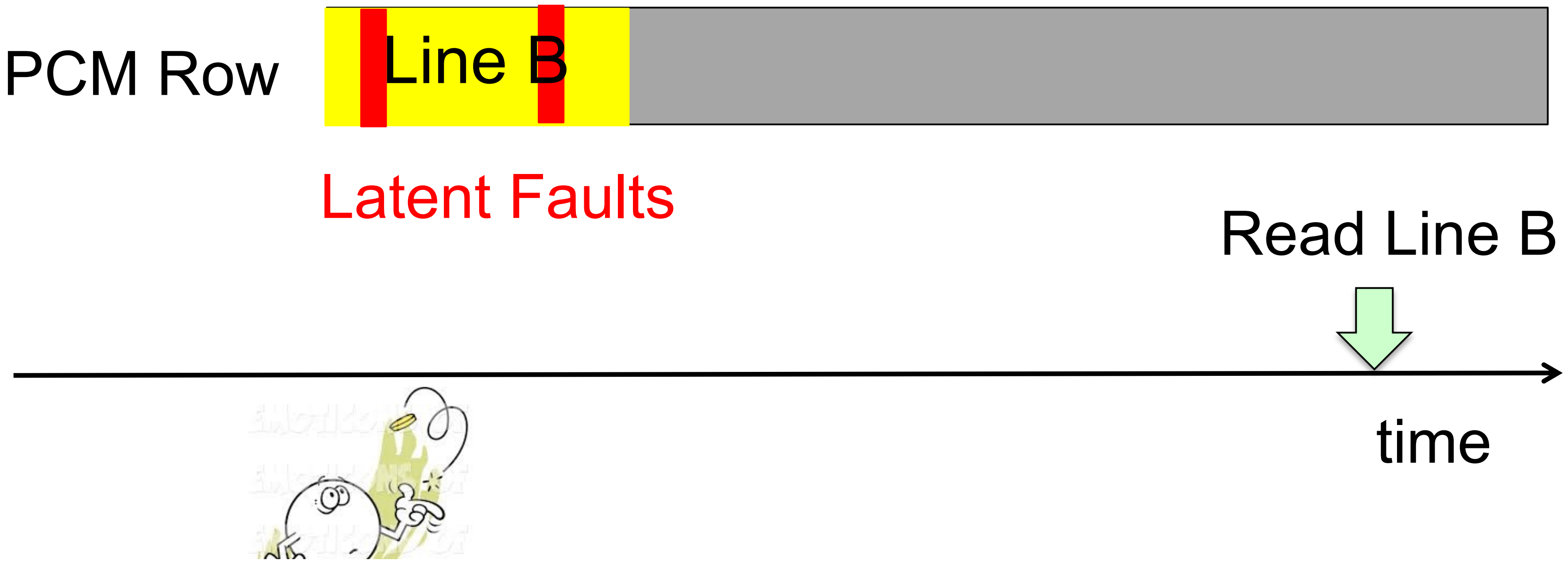
OUR SOLUTION: PROBABILISTIC SCRUB

- Scrub the entire row with low probability (say 1%)



OUR SOLUTION: PROBABILISTIC SCRUB

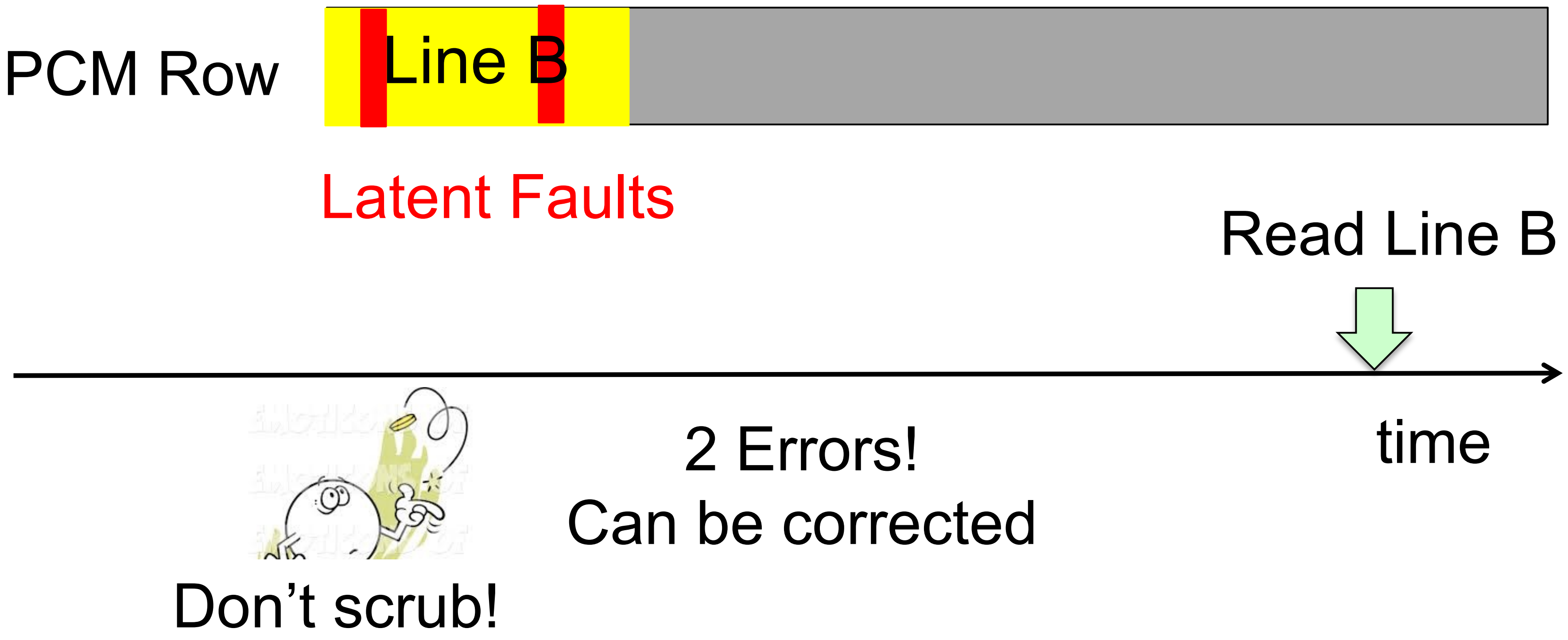
- Scrub the entire row with low probability (say 1%)



Don’t scrub!

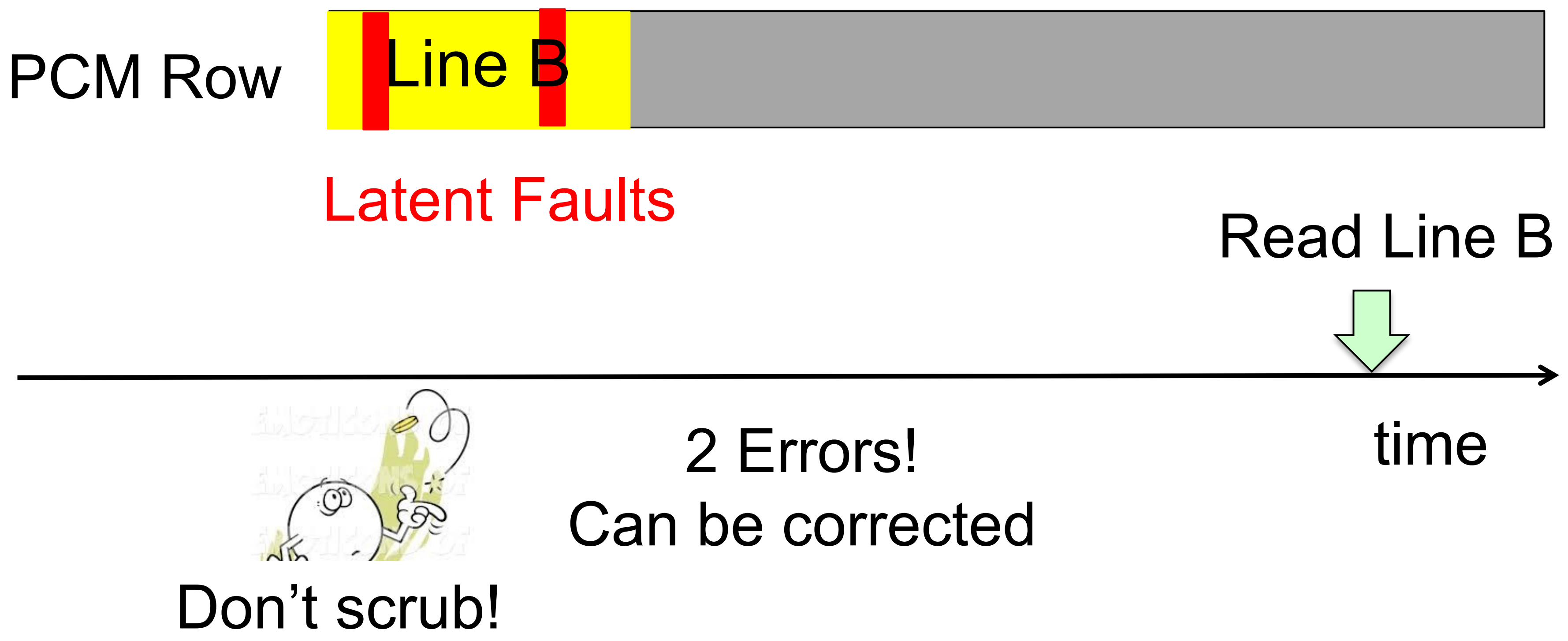
OUR SOLUTION: PROBABILISTIC SCRUB

- Scrub the entire row with low probability (say 1%)



OUR SOLUTION: PROBABILISTIC SCRUB

- Scrub the entire row with low probability (say 1%)



Probabilistic Scrub improves reliability by 10^5 times with negligible impact on performance

END OF BACKUP