Signal Processing and Coding for Non-Volatile Memories

Non-Volatile Memory Workshop

Center for Magnetic Recoding Research (CMRR)

University of California, San Diego

March 3rd, 2013

Part I: Noise Sources in NAND Flash Memory

Jason Bellorado, Ph.D.
SK Hynix Memory Solutions

Part II: Error-Correction and Rewriting Codes for Non-Volatile Memories

Eitan Yaakobi, Ph.D.
California Institute of Technology

Part III: Emerging Coding Methods

Andrew Jiang, Ph.D.
Texas A&M University

Part I: Noise Sources in NAND Flash Memory

Jason Bellorado, Ph.D.
SK Hynix Memory Solutions

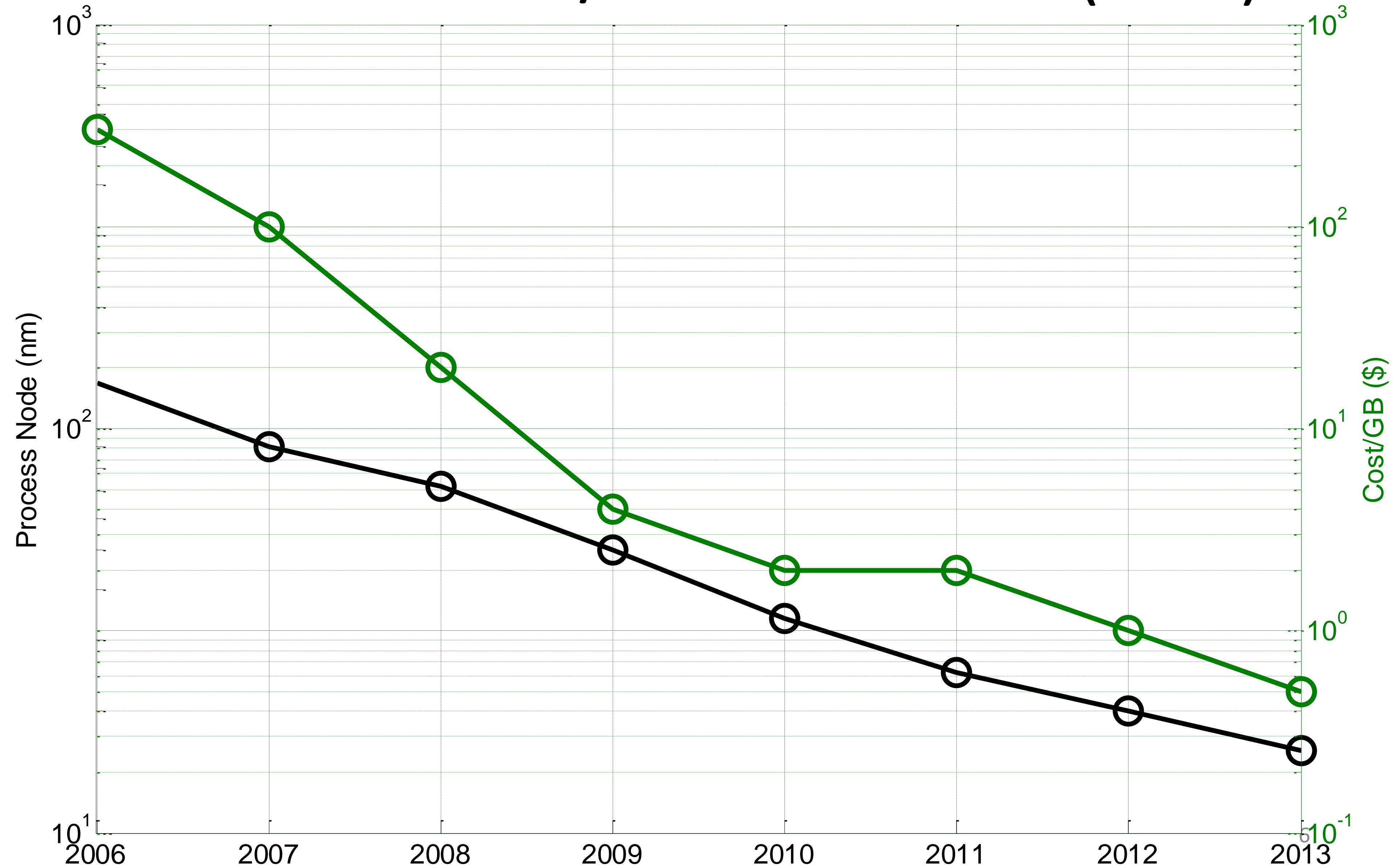
Outline

- NAND Flash Basics
- SLC/MLC Read/Write Processes
- Noise Sources
  - Endurance & Retention
  - Write Induced
  - Read Induced
  - Pattern Induced

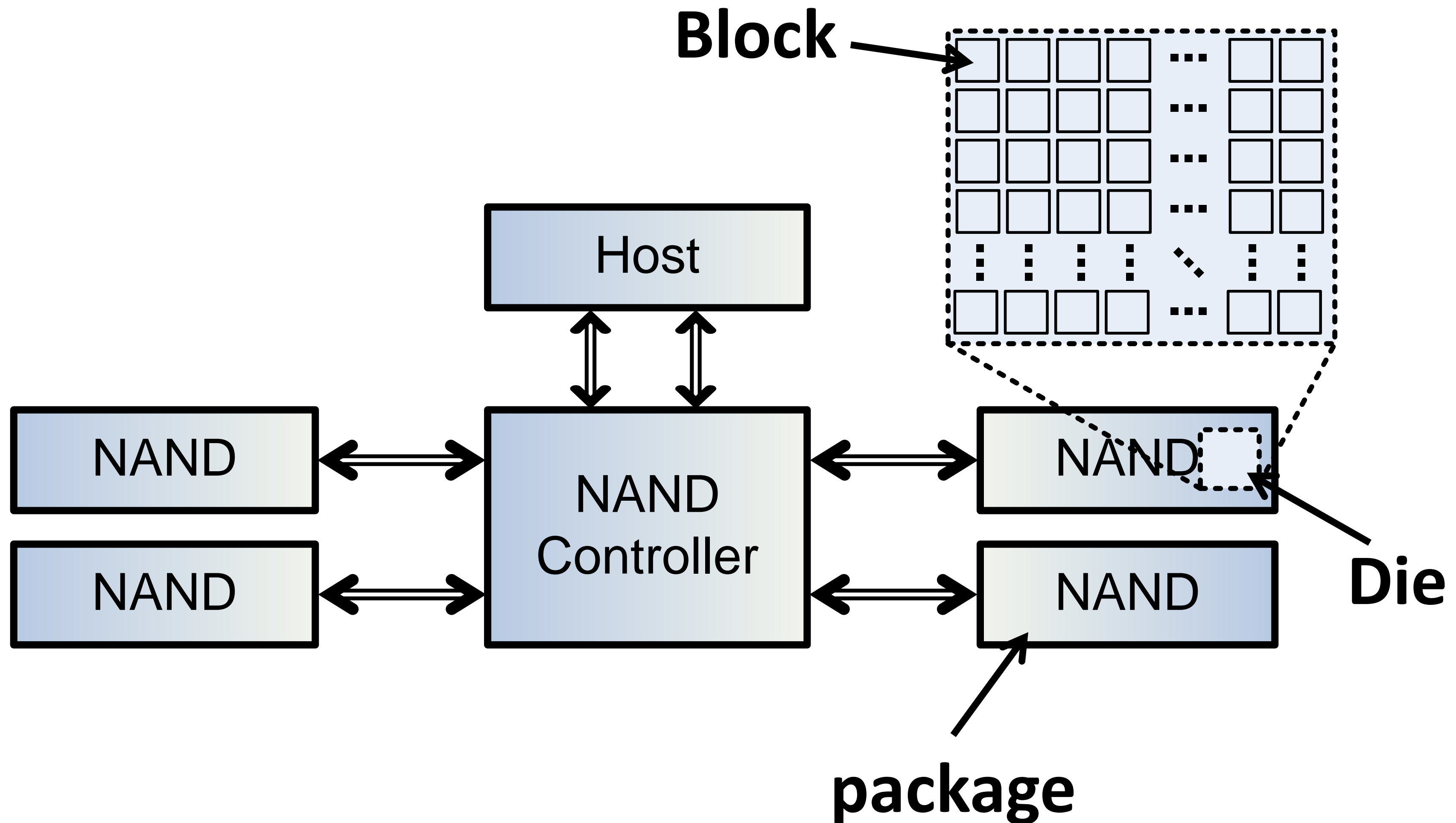
Introduction

- In an effort to reduce the cost of NAND flash-based storage devices, NAND manufacturers have aggressively scaled down their process.
- This scaling has exceeded the rate predicted by Moore's Law and has reduced the price/GB from > $100 in 2008 to < $1 today.
- Unfortunately, scaling down the feature size of NAND flash cells acts to exacerbate many of its noise sources.
- To design reliable NAND-based storage systems, these noise sources must be well-understood.

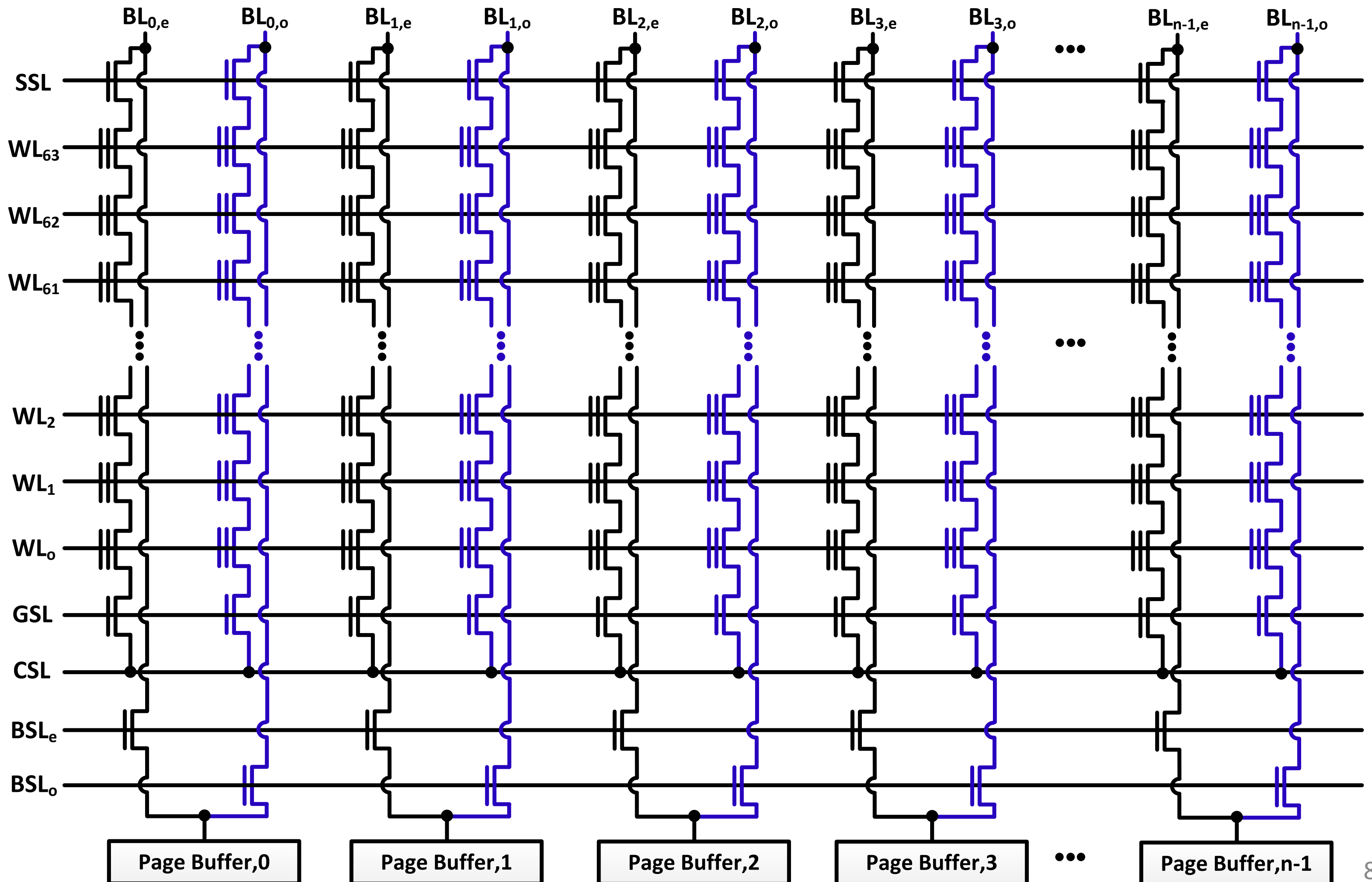
NAND Process/Cost Evolution (MLC)



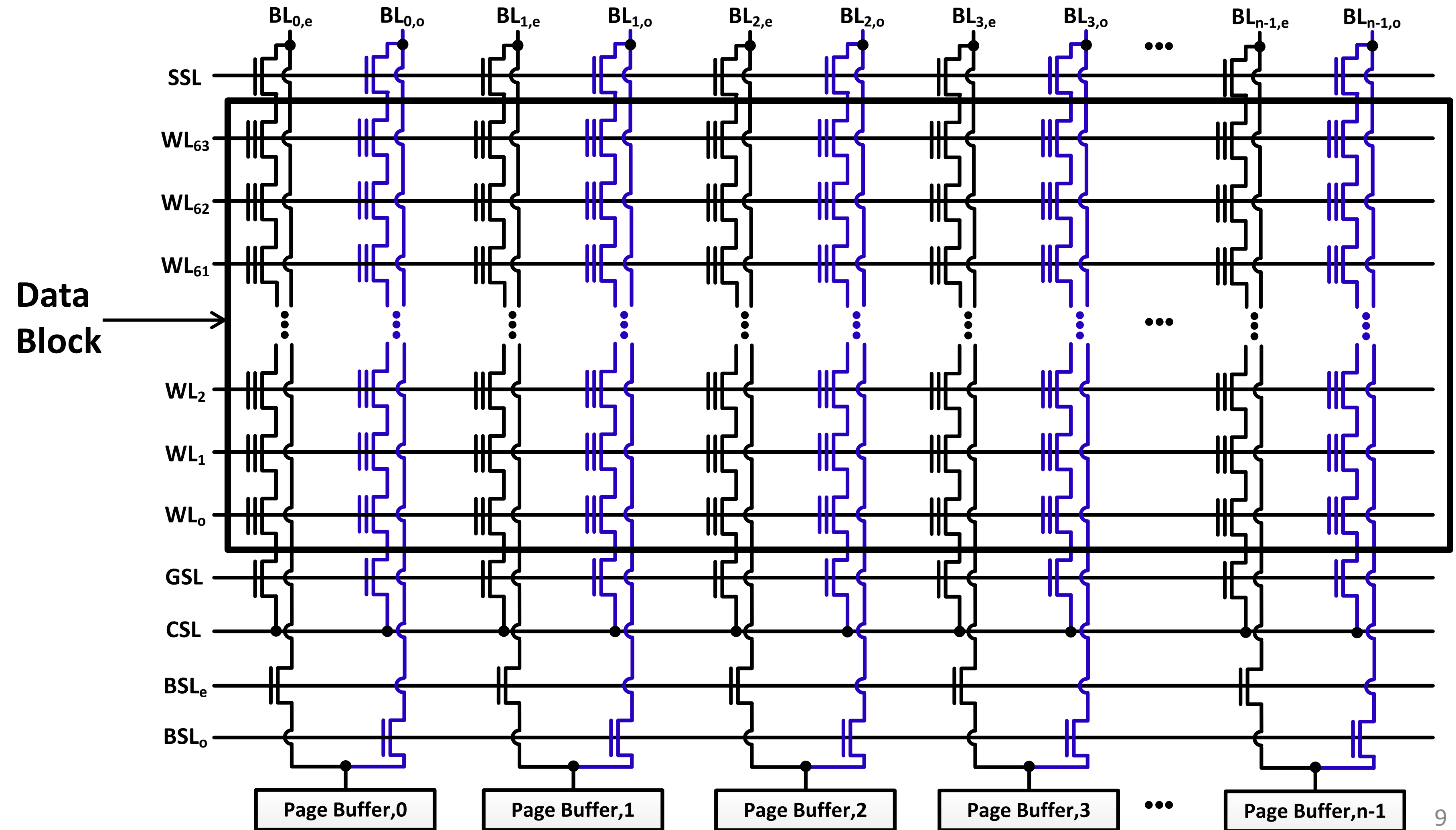
Storage Device Architecture



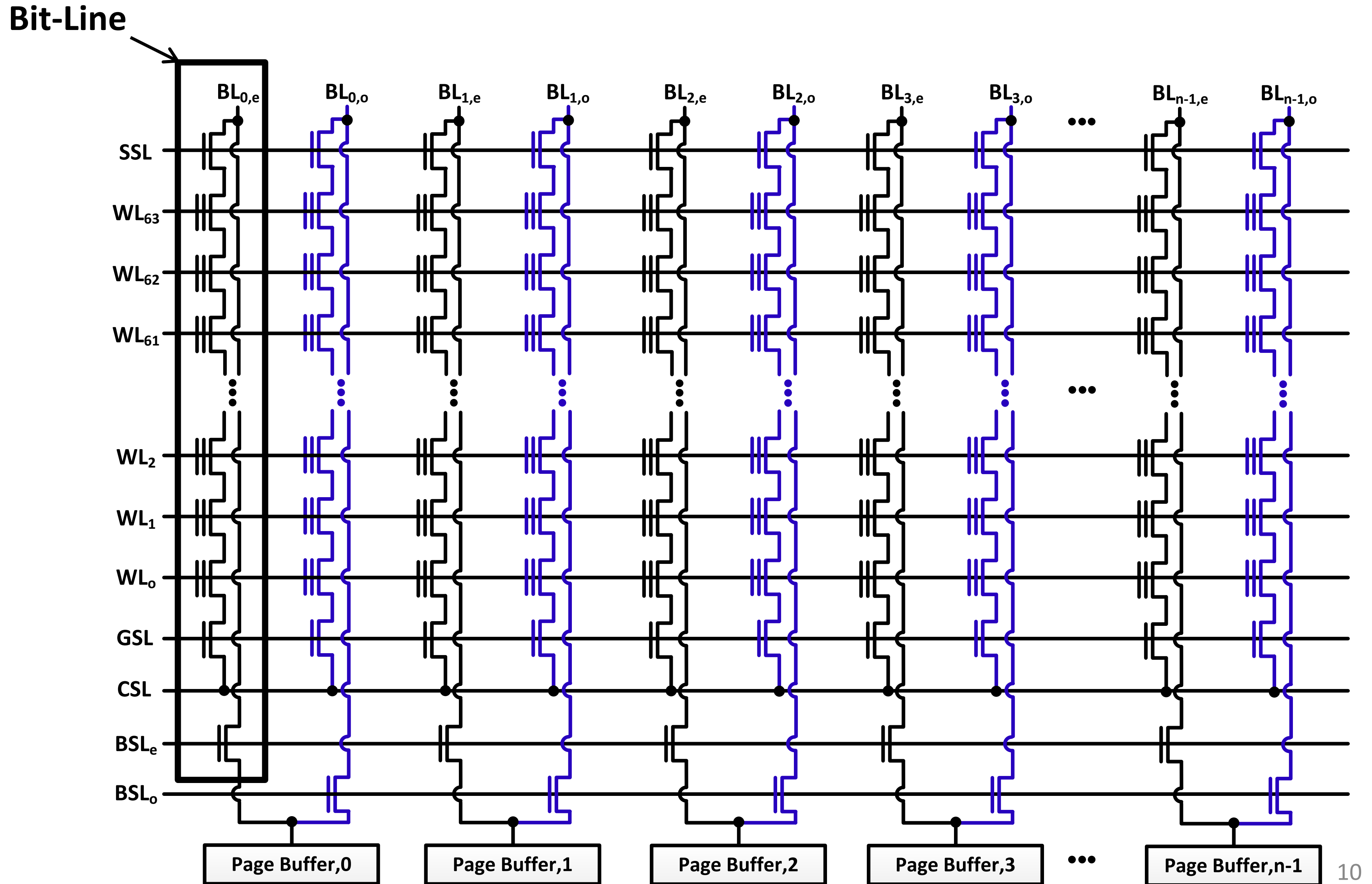
NAND Flash Block



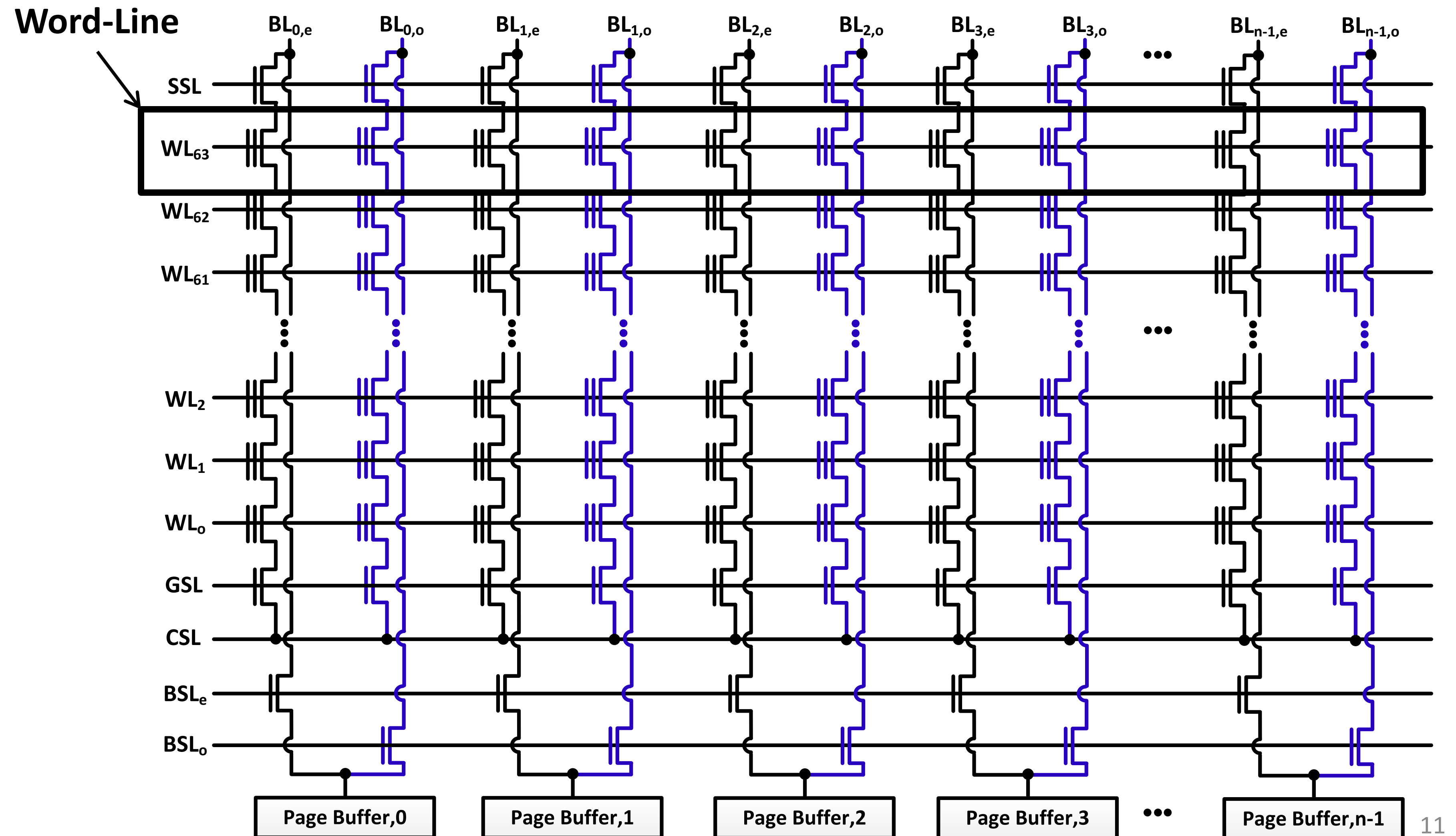
NAND Flash Block



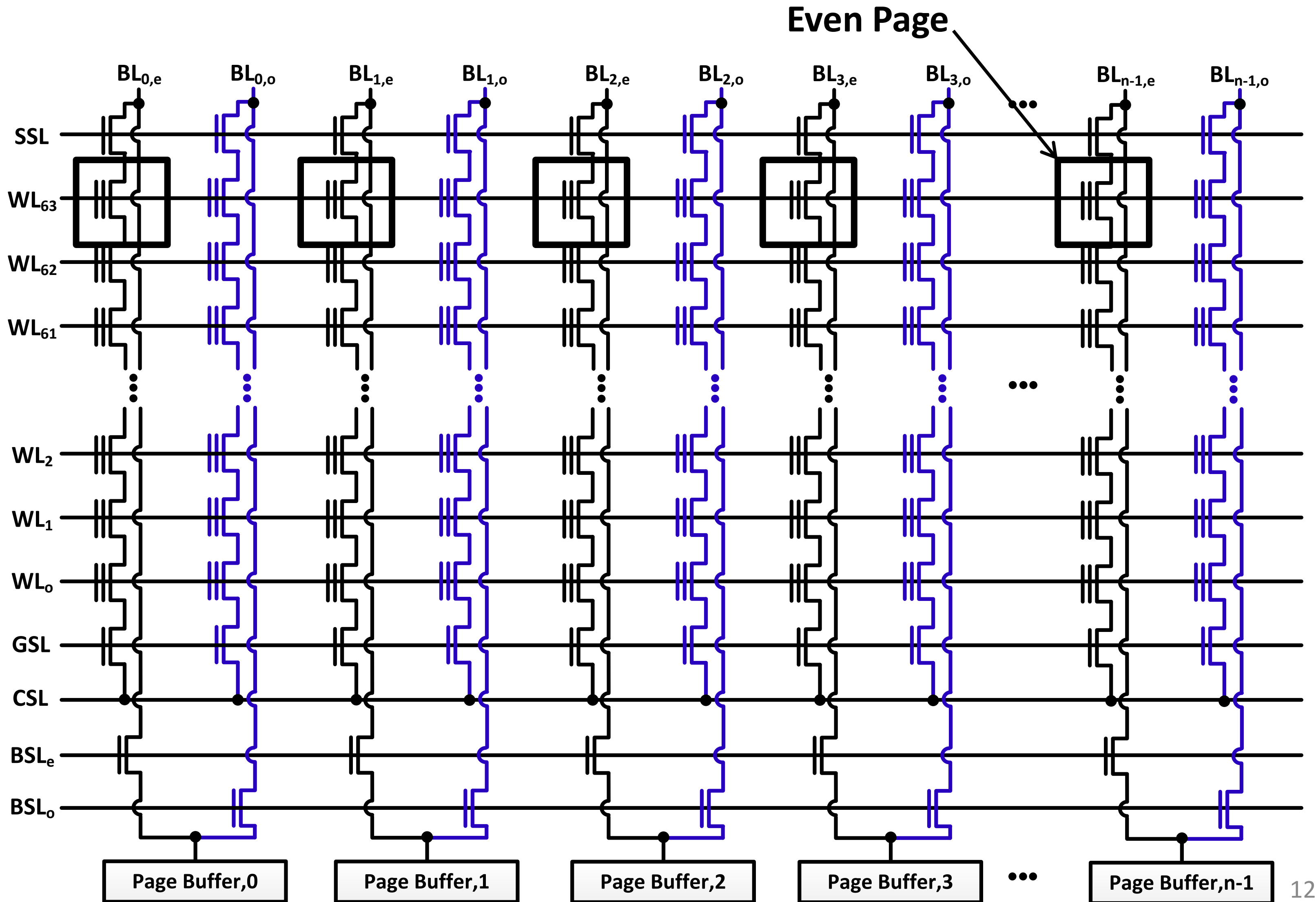
NAND Flash Block



NAND Flash Block



NAND Flash Block



NAND Flash Block

Odd Page

$BL_{0,e}$ $BL_{0,o}$ $BL_{1,e}$ $BL_{1,o}$ $BL_{2,e}$ $BL_{2,o}$ $BL_{3,e}$ $BL_{3,o}$ … $BL_{n-1,e}$ $BL_{n-1,o}$

SSL

WL_{63}

WL_{62}

WL_{61}

WL_2

WL_1

WL_0

GSL

CSL

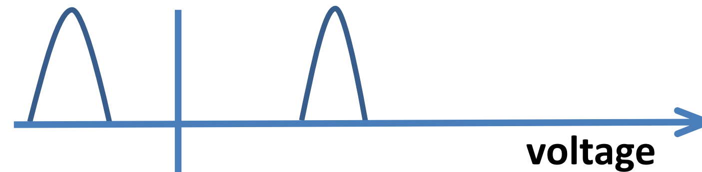
BSL_e

BSL_o

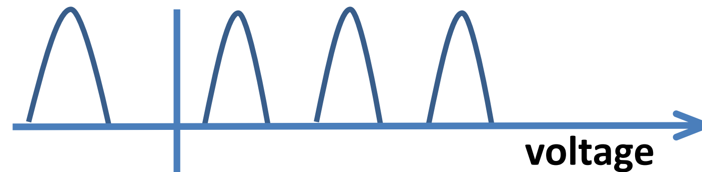
Page Buffer,0 Page Buffer,1 Page Buffer,2 Page Buffer,3 … Page Buffer,n-1

NAND Flash Basics

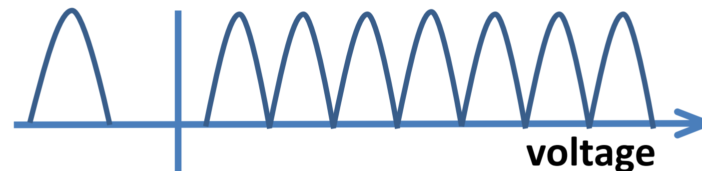
- Information is stored in a NAND flash cell by raising its floating-gate voltage to one of a discrete set of values.
- SLC: 1 bit/cell



- MLC: 2 bits/cell



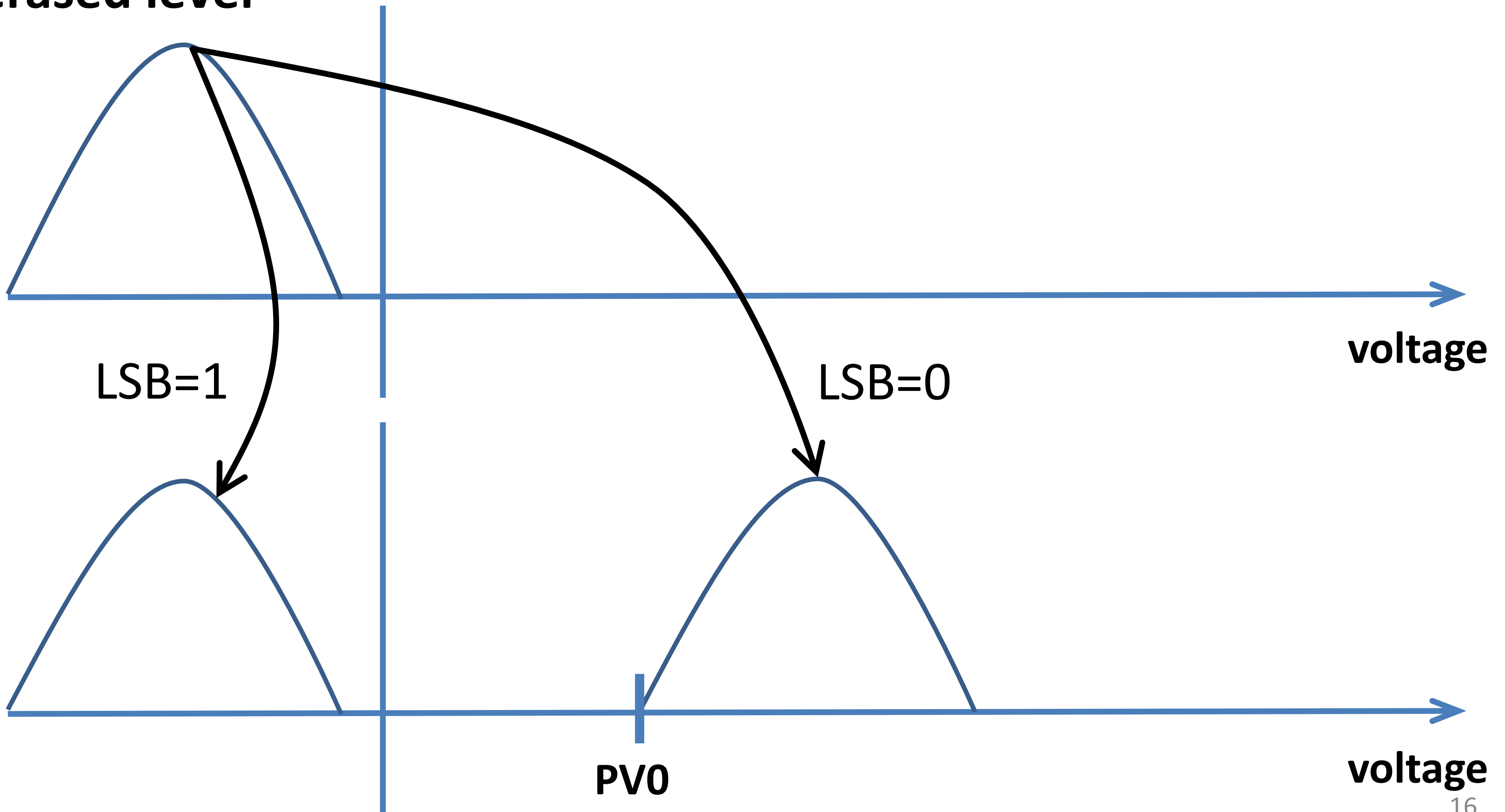
- TLC: 3 bits/cell



SLC/LSB Write Process

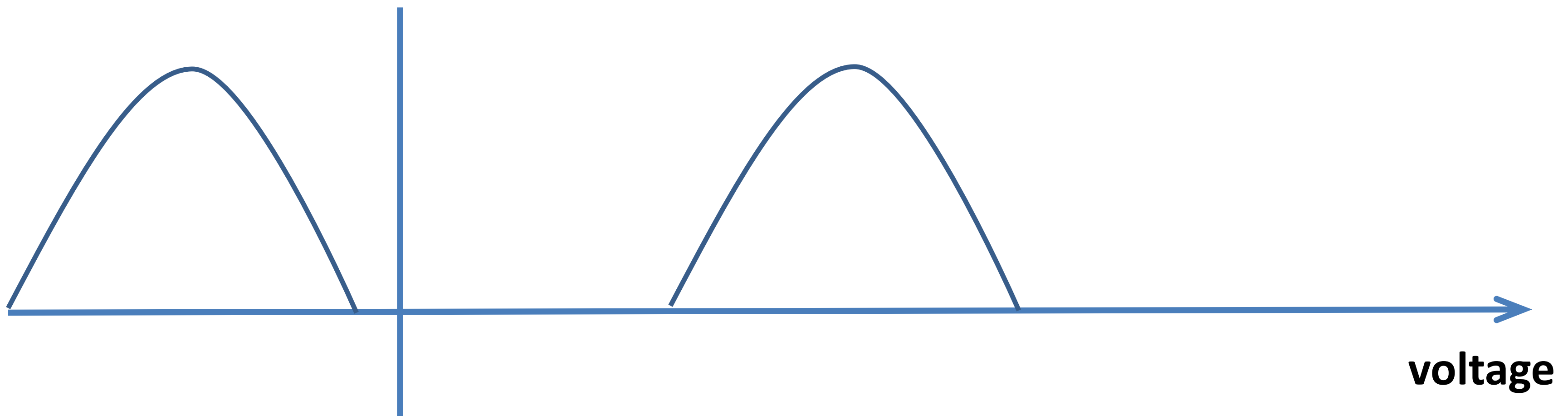
All cells start in erased level

SLC/LSB Write Process



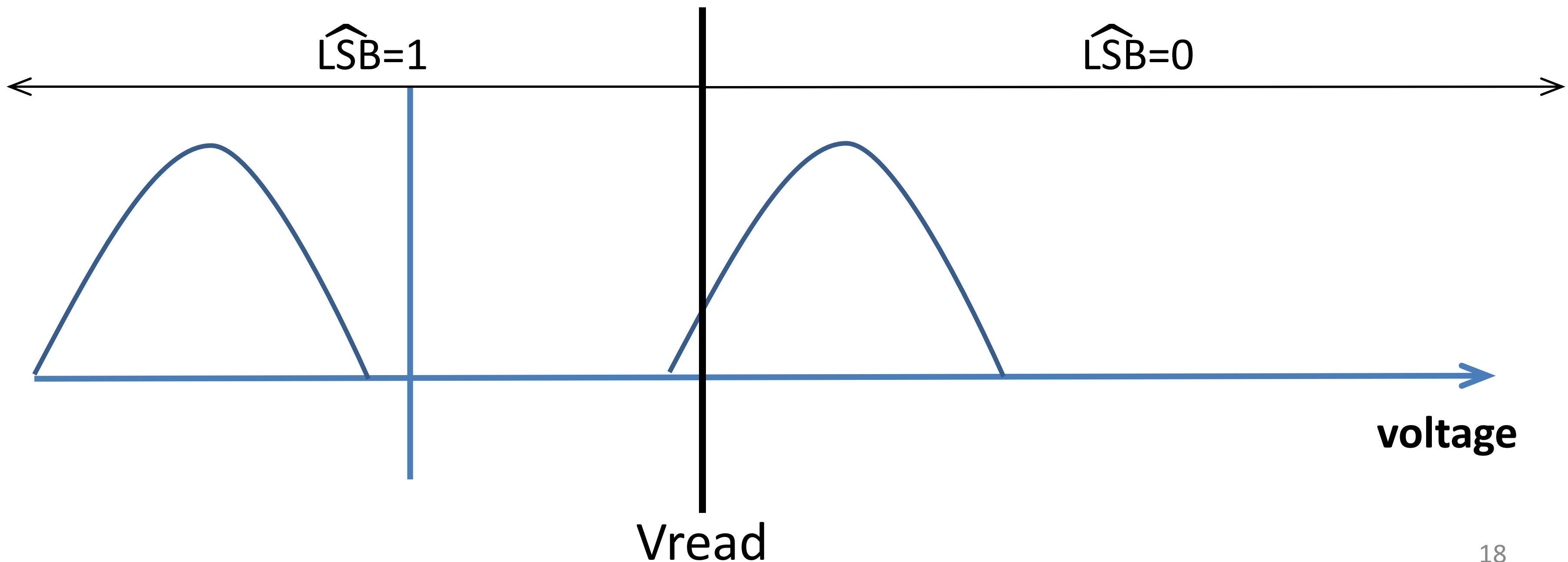
SLC/LSB Read Process

- A reference voltage (Vread) is specified by a NAND register.
  - Cells w/ threshold voltages < (>) Vread read 1 (0).

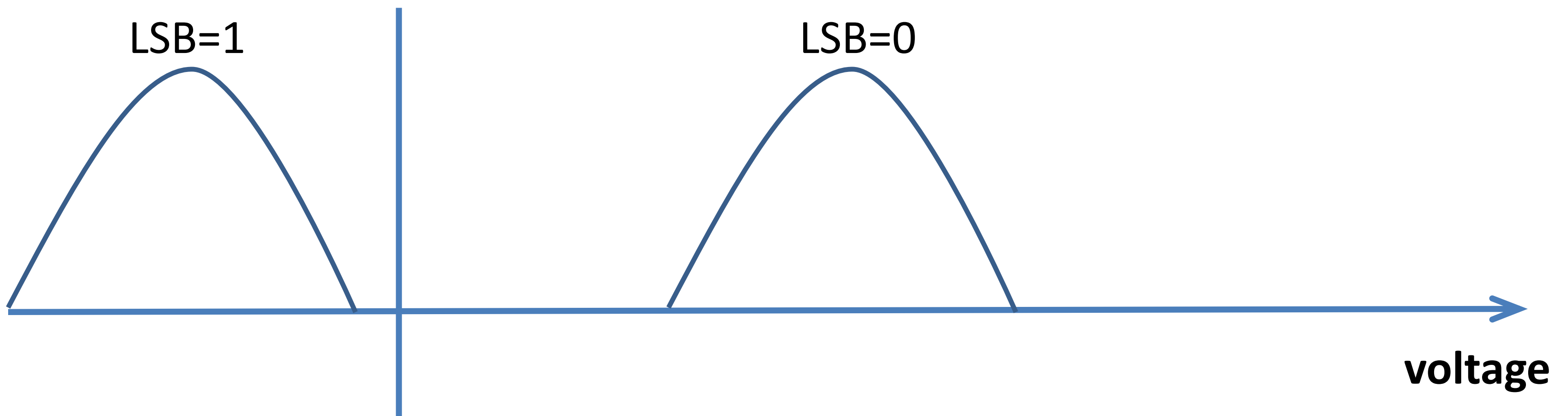


SLC/LSB Read Process

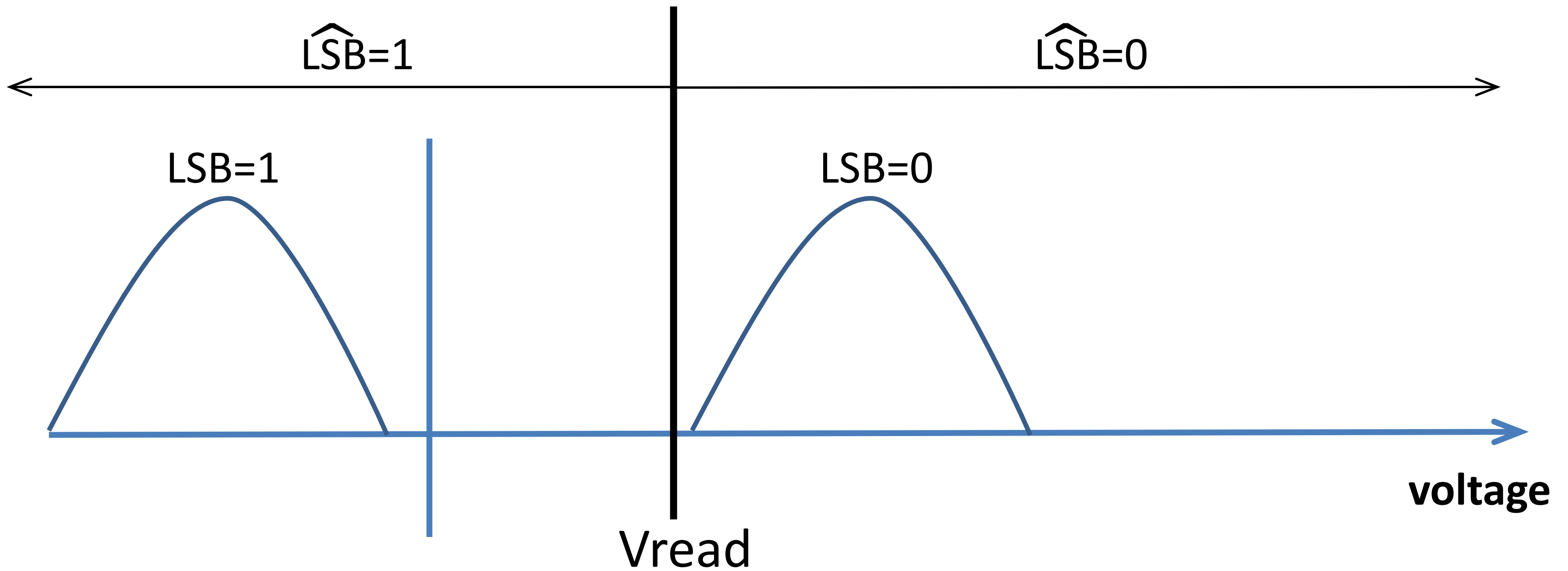
- A reference voltage (Vread) is specified by a NAND register.
  - Cells w/ threshold voltages < (>) Vread read 1 (0).



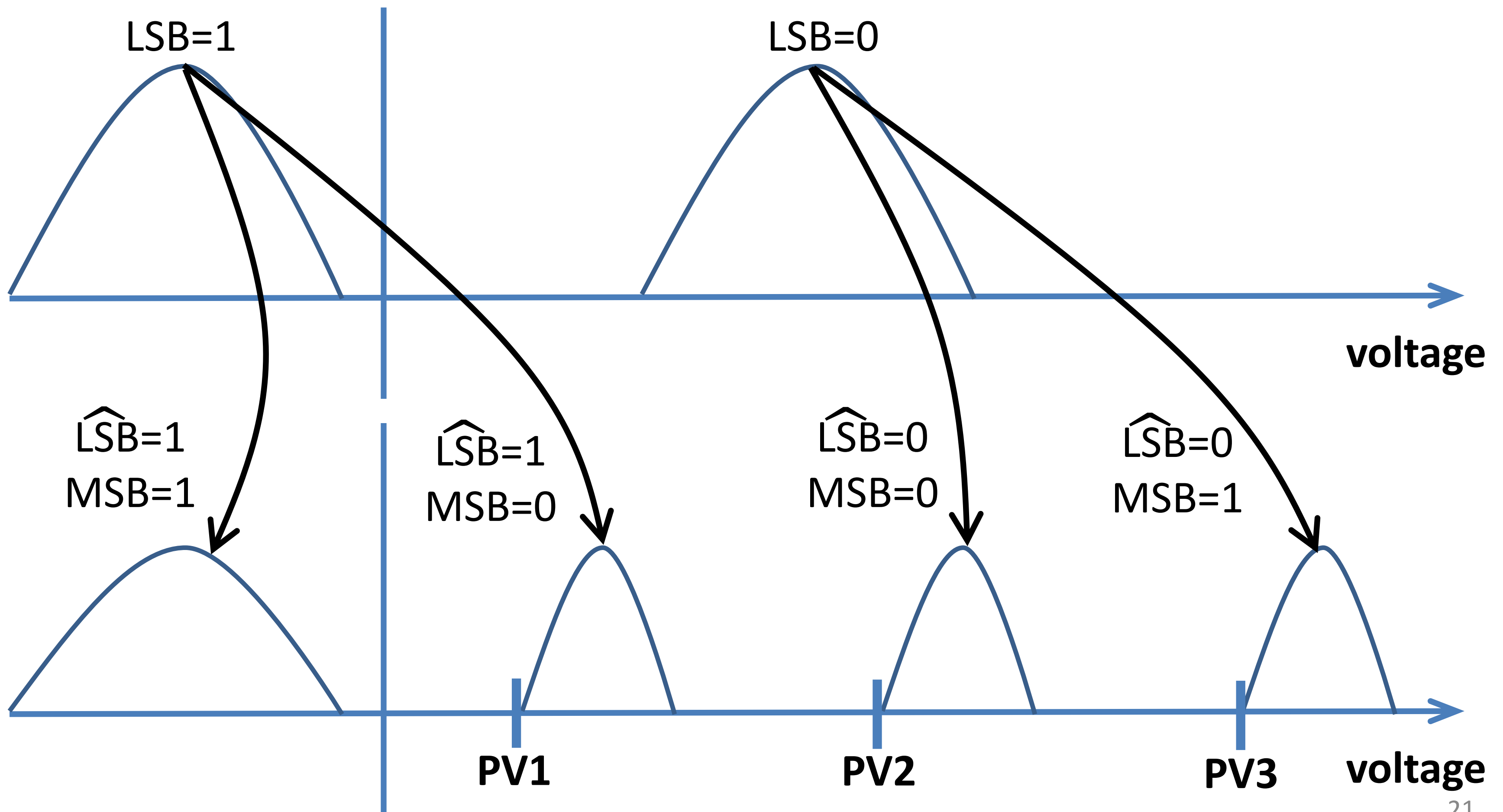
MLC (MSB) Write Process



MLC (MSB) Write Process

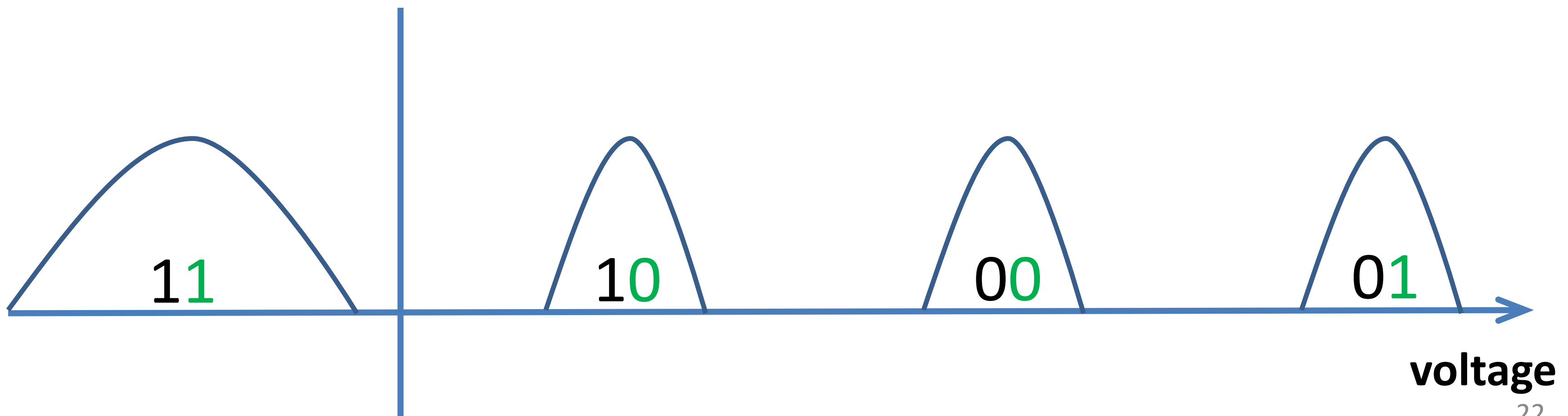


MLC (MSB) Write Process



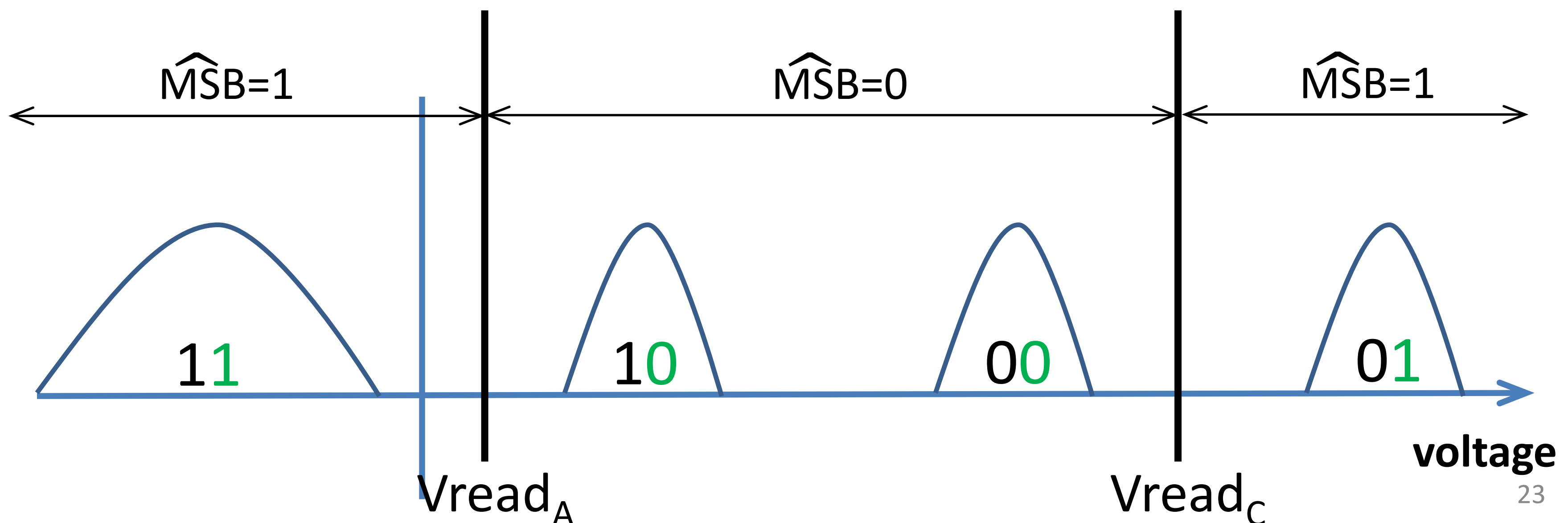
MLC (MSB) Read Process

- Two reference voltages ($Vread_A$ & $Vread_C$) specified.
- Two reads are conducted and the output is:
  - 0: $Vread_A \leq Vth \leq Vread_C$
  - 1: $Vth < Vread_A$ or $Vth \geq Vread_C$



MLC (MSB) Read Process

- Two reference voltages ($Vread_A$ & $Vread_C$) specified.
- Two reads are conducted and the output is:
  - 0: $Vread_A \leq Vth \leq Vread_C$
  - 1: $Vth < Vread_A$ or $Vth \geq Vread_C$



Noise Sources

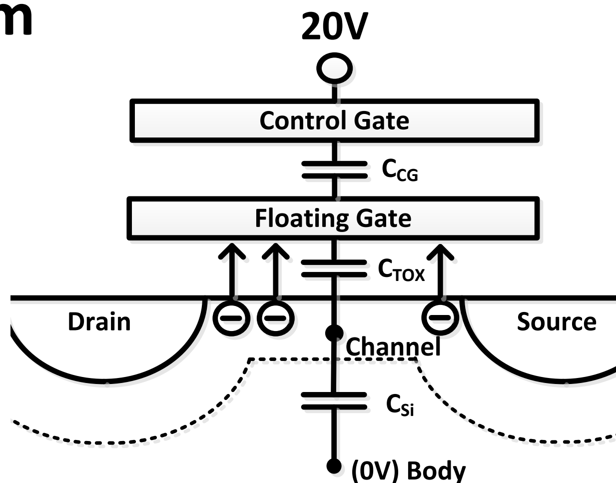
- **Endurance & Retention (single cell)**
- **NAND Array Based Noise**
  - **The Write Process**
    - Capacitive Coupling
    - Program Disturb
  - **The Read Process**
    - Read Disturb
    - Read Noise (RTN)
  - **Data Pattern**
    - Back Pattern Effect

Single Cell Program/Erase

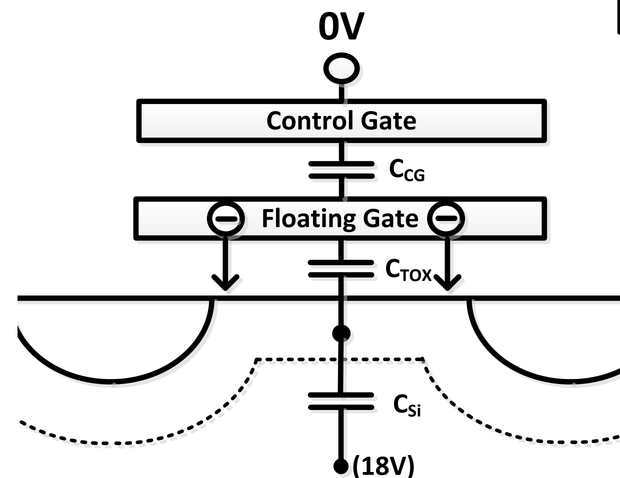
- Program/erase operations force charge on/off the floating-gates of NAND cells through Fowler-Nordheim (FN) tunneling.

$$J_{FN} = A_t \times E_{ox}^2 \times e^{-B_t/E_{ox}}$$

Program

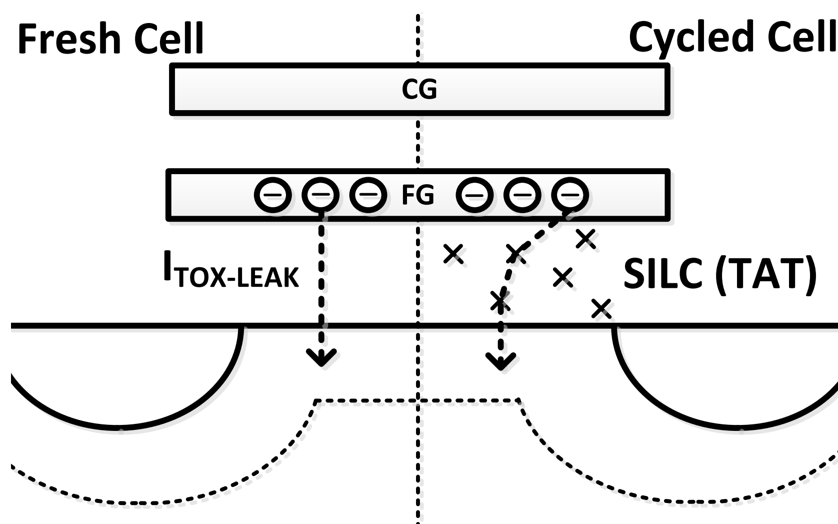


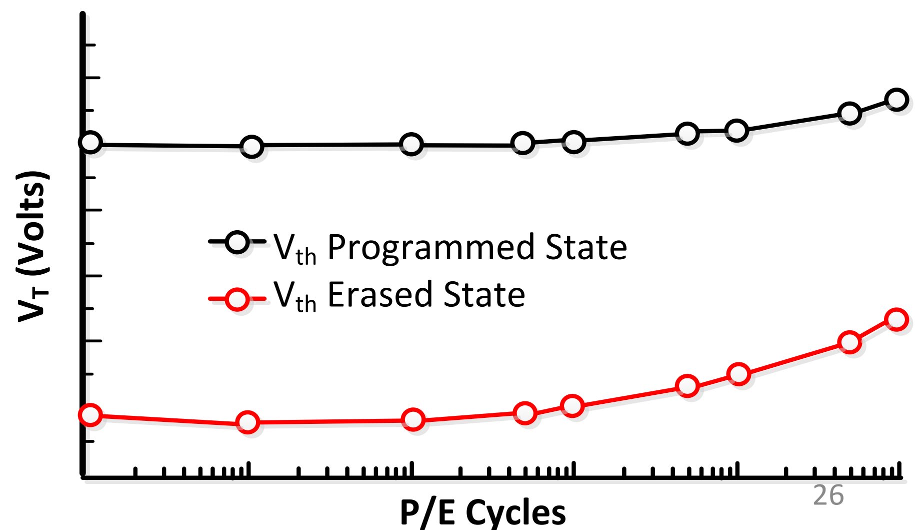
Erase



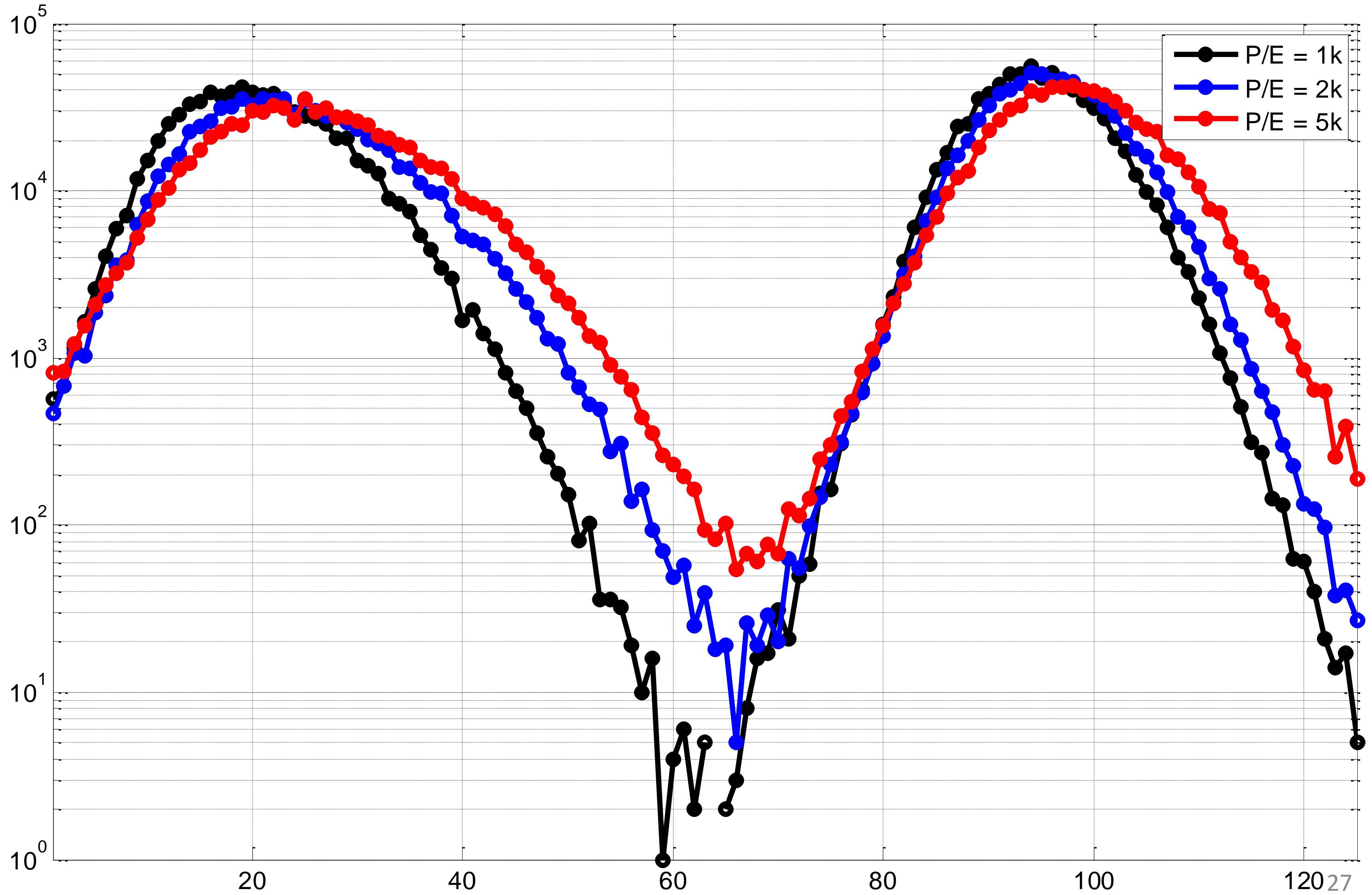
Cycling/Retention Effects

- As a cell is cycled the tunneling oxide forms traps
  - Broken atomic bonds in oxide matrix due to tunneling.
- Electrons can more easily leak from the FG to the channel by Trap Assisted Tunneling (TAT) .
- When filled with electrons, traps can increase the potential barrier, reducing the tunneling current and increase Vth.

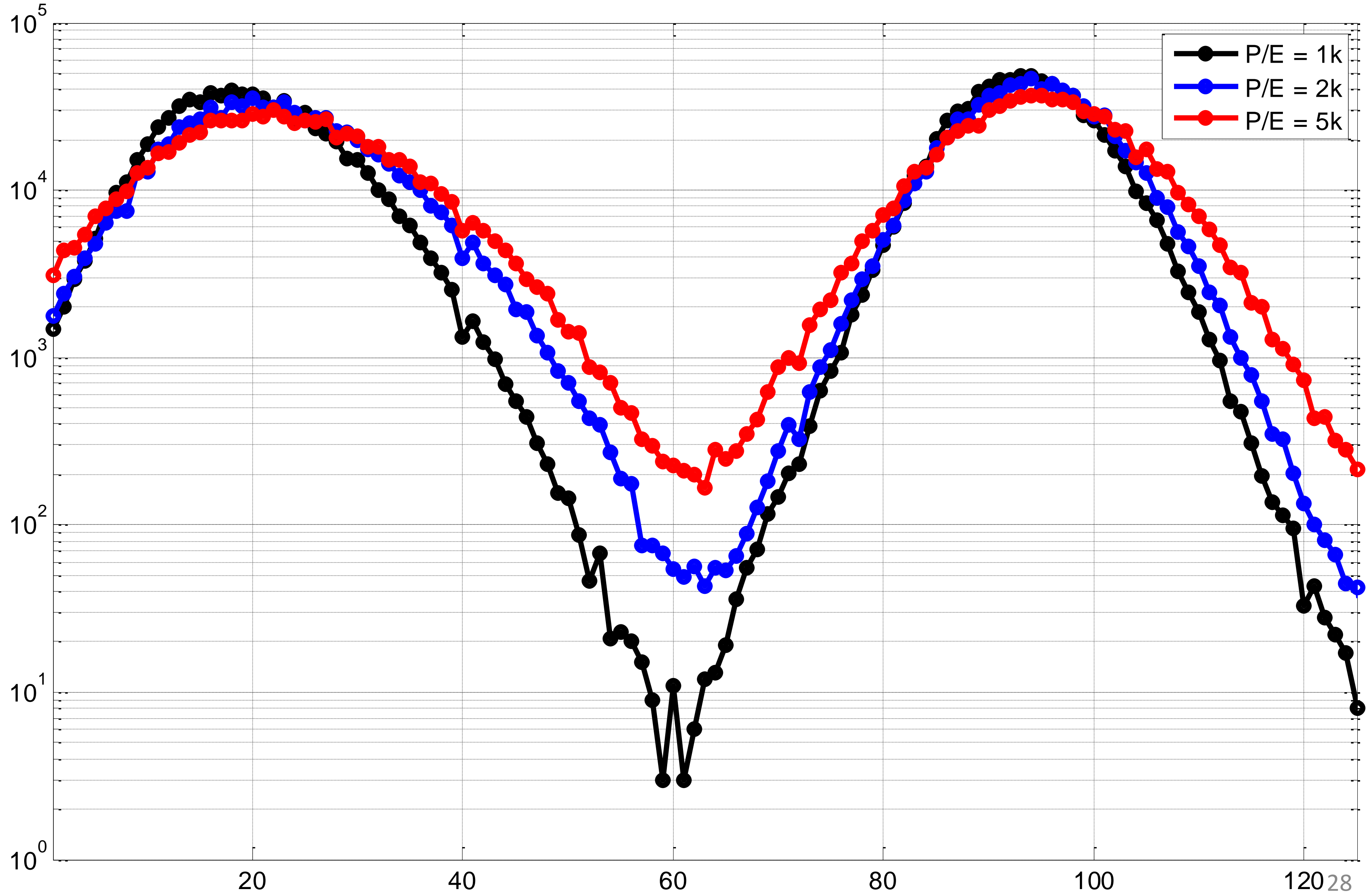




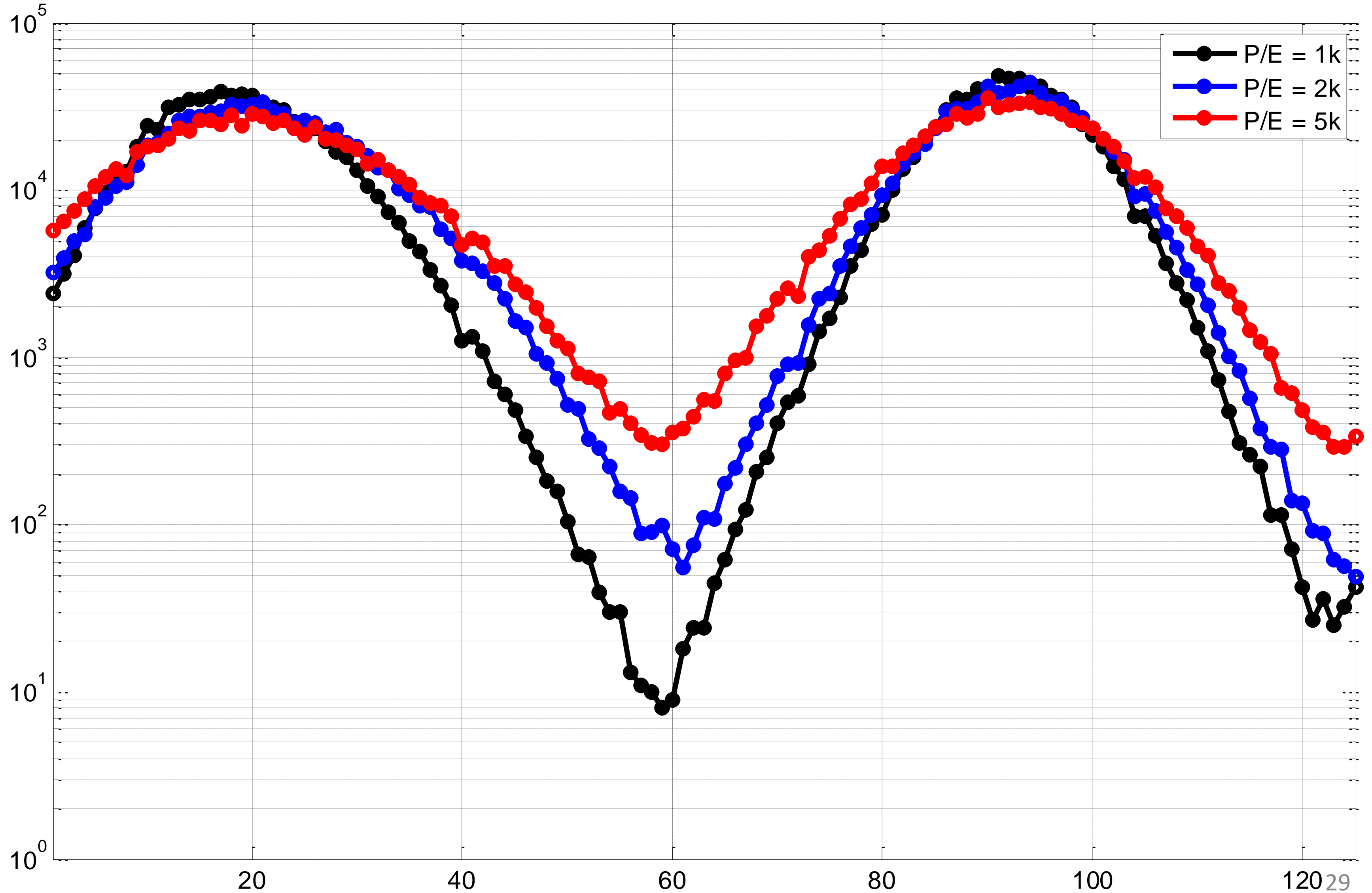
0m Retention



3m Retention



12m Retention

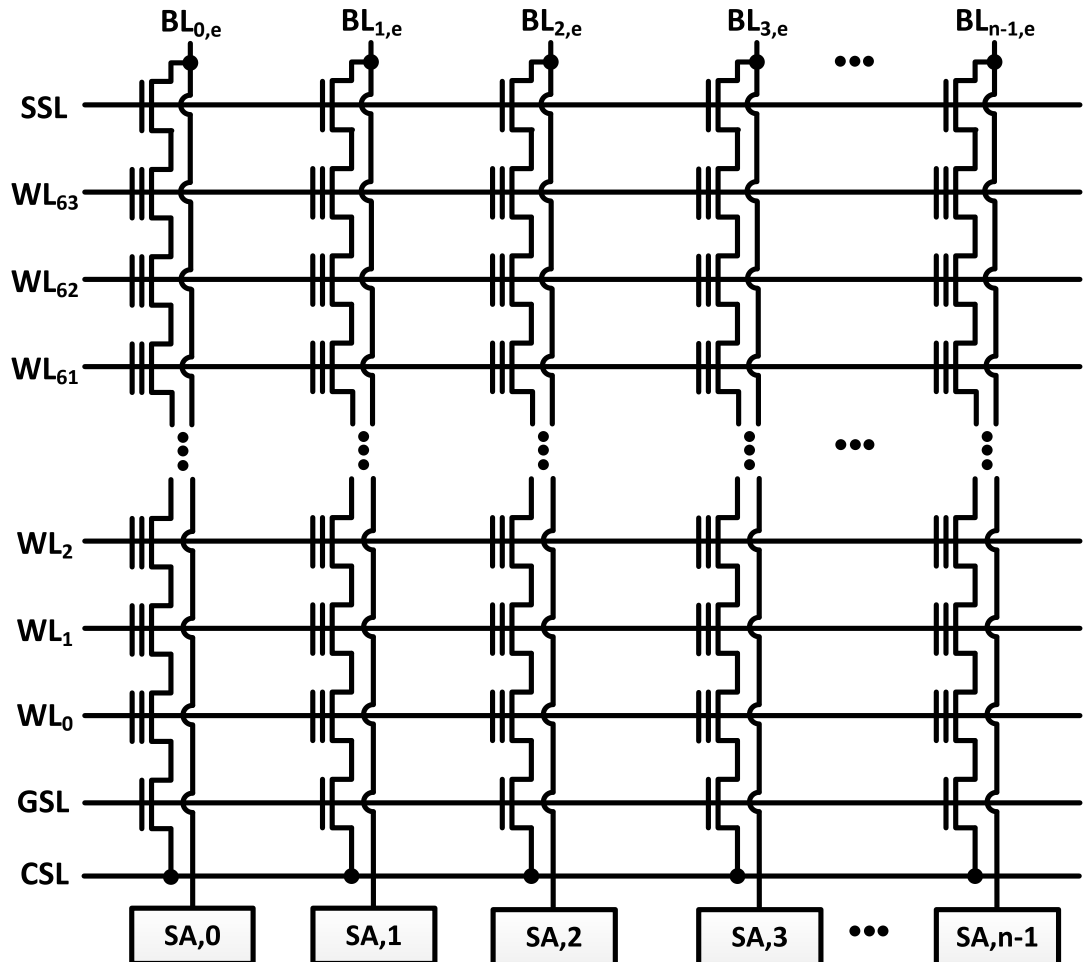


Mitigating Endurance/Retention

- **Endurance:**
  - Reduce the amount of data written to the NAND
    - Data Compression.
    - Reduce write amplification = NAND writes/host writes.
  - Wear-Leveling: Ensure all blocks are used equally
    - All blocks reach EOL at the same time.
- **Retention:**
  - Refreshing old blocks.
    - Background media scan.
- **Both:**
  - Stronger ECC.
  - Better Signal processing.

WRITING TO THE NAND ARRAY

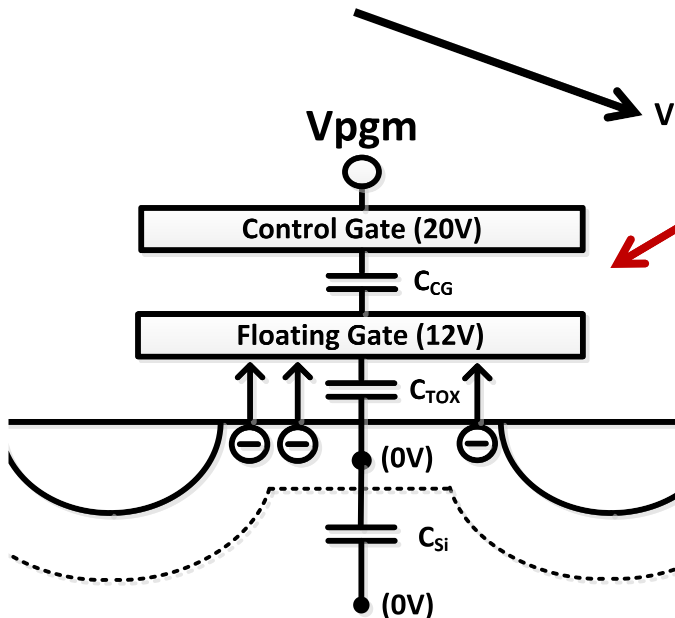
Writing to the NAND Array



Writing to the NAND Array

Vpass ~8-10V
Vpgm ~20-25V

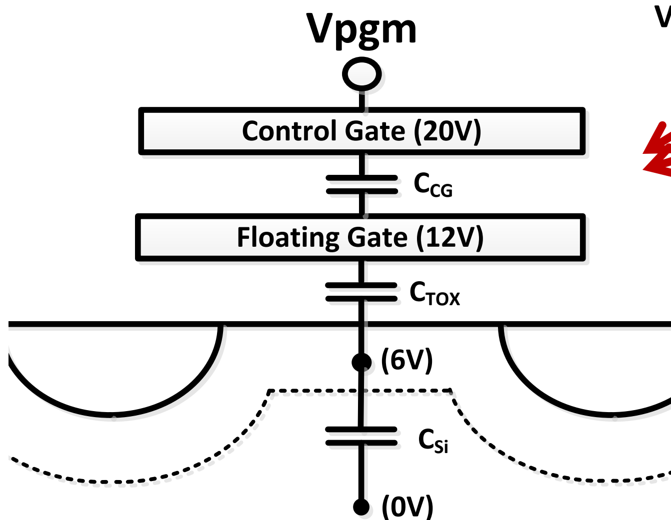
Programmed Wordline



GND GND GND GND ... GND
Vcc
Vpass
Vpass
Vpass
Vpgm
Vpass
Vpass
GND
GND
SA,0 SA,1 SA,2 SA,3 ... SA,n-1

Self-Boosting Program Inhibit

- Inhibited bitlines raised to Vcc, SSL → off.
- Wordline voltages cause the channel voltage to capacitively raise (*channel boosting*).
- Effective program voltage Vpgm-Vch.

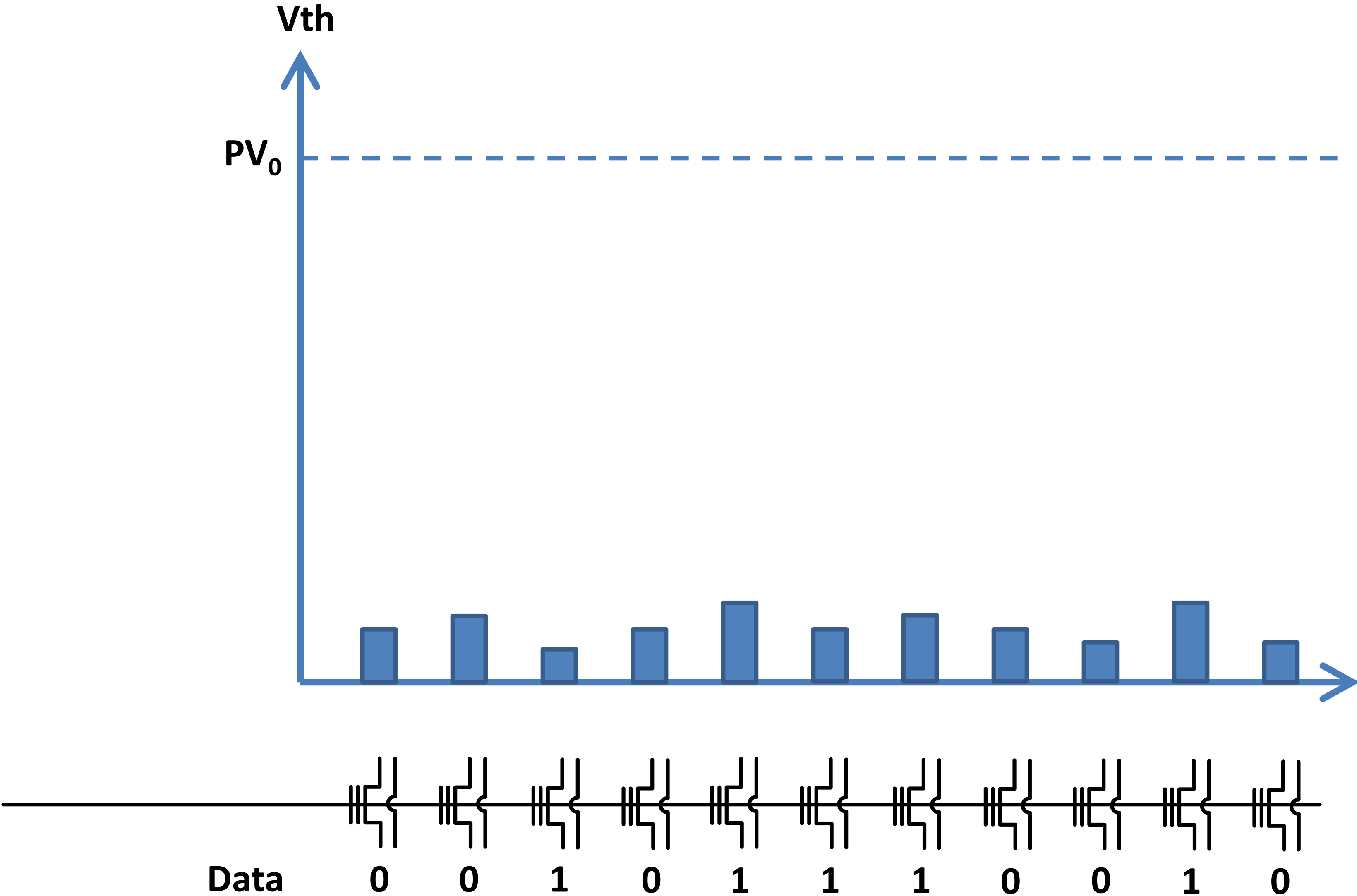


Vcc GND Vcc GND GND

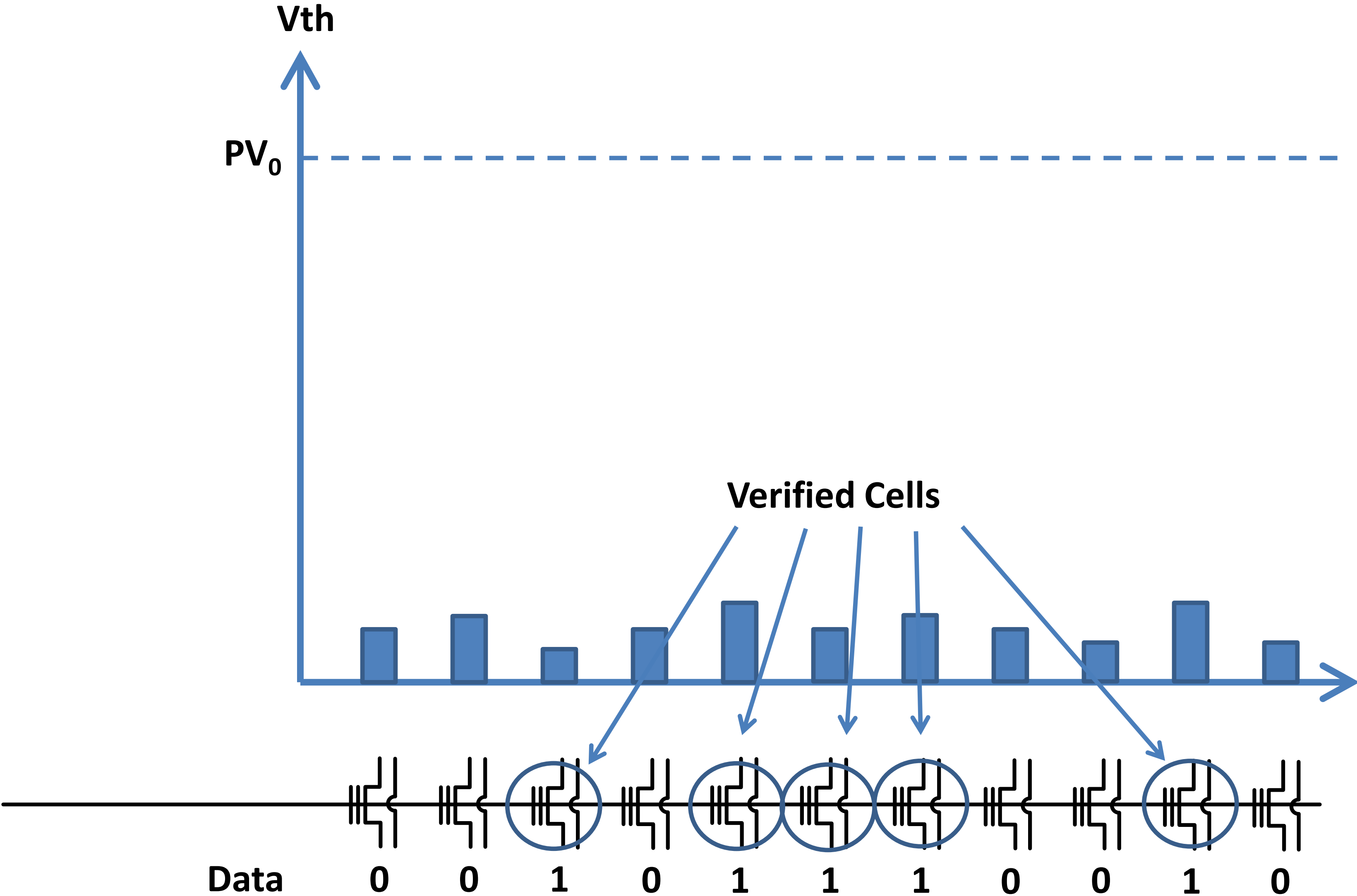
Vcc
Vpass
Vpass
Vpass
Vpgm
Vpass
Vpass
GND
GND

SA,0 SA,1 SA,2 SA,3 SA,n-1

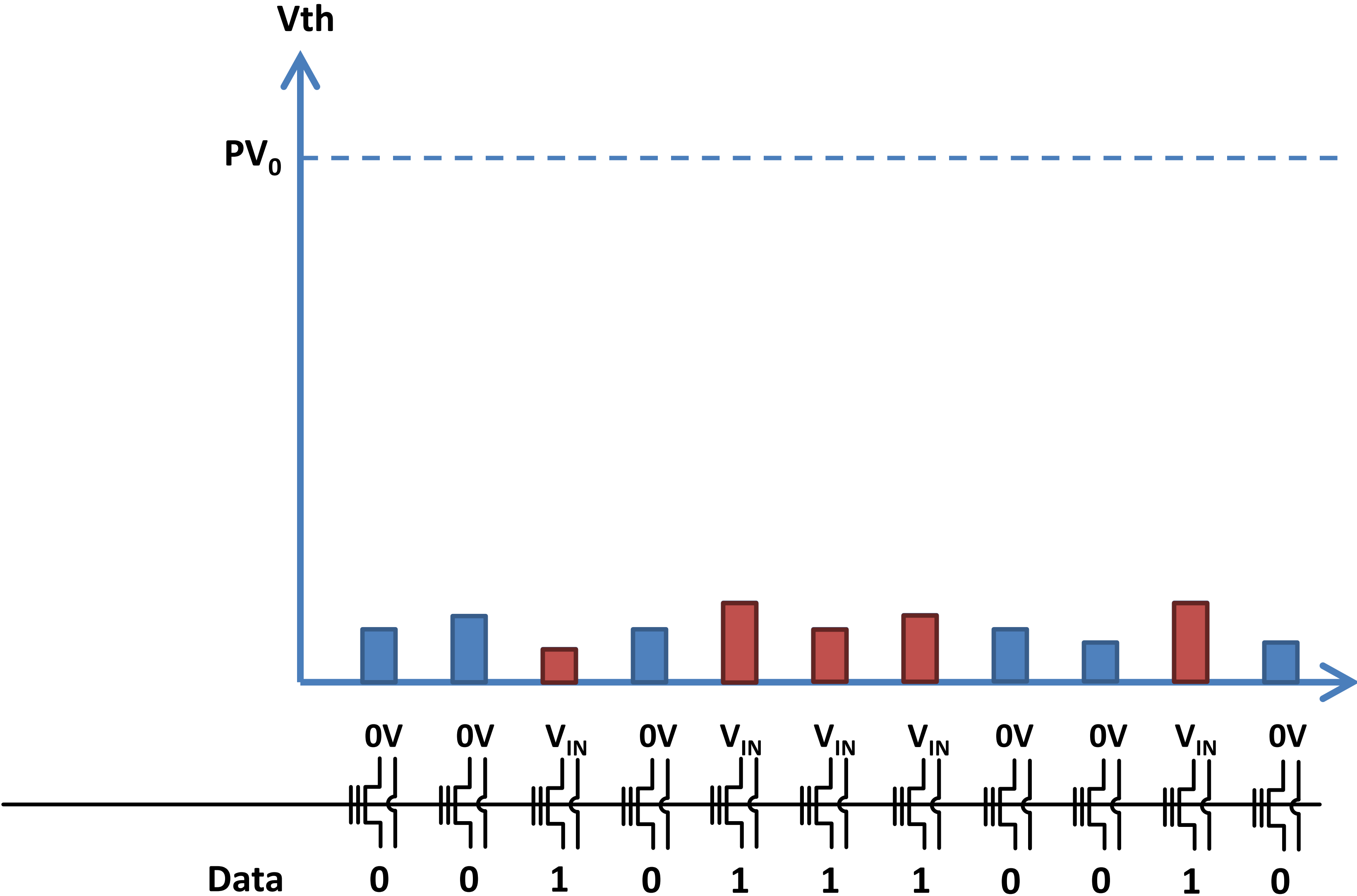
The Write Process ISPP



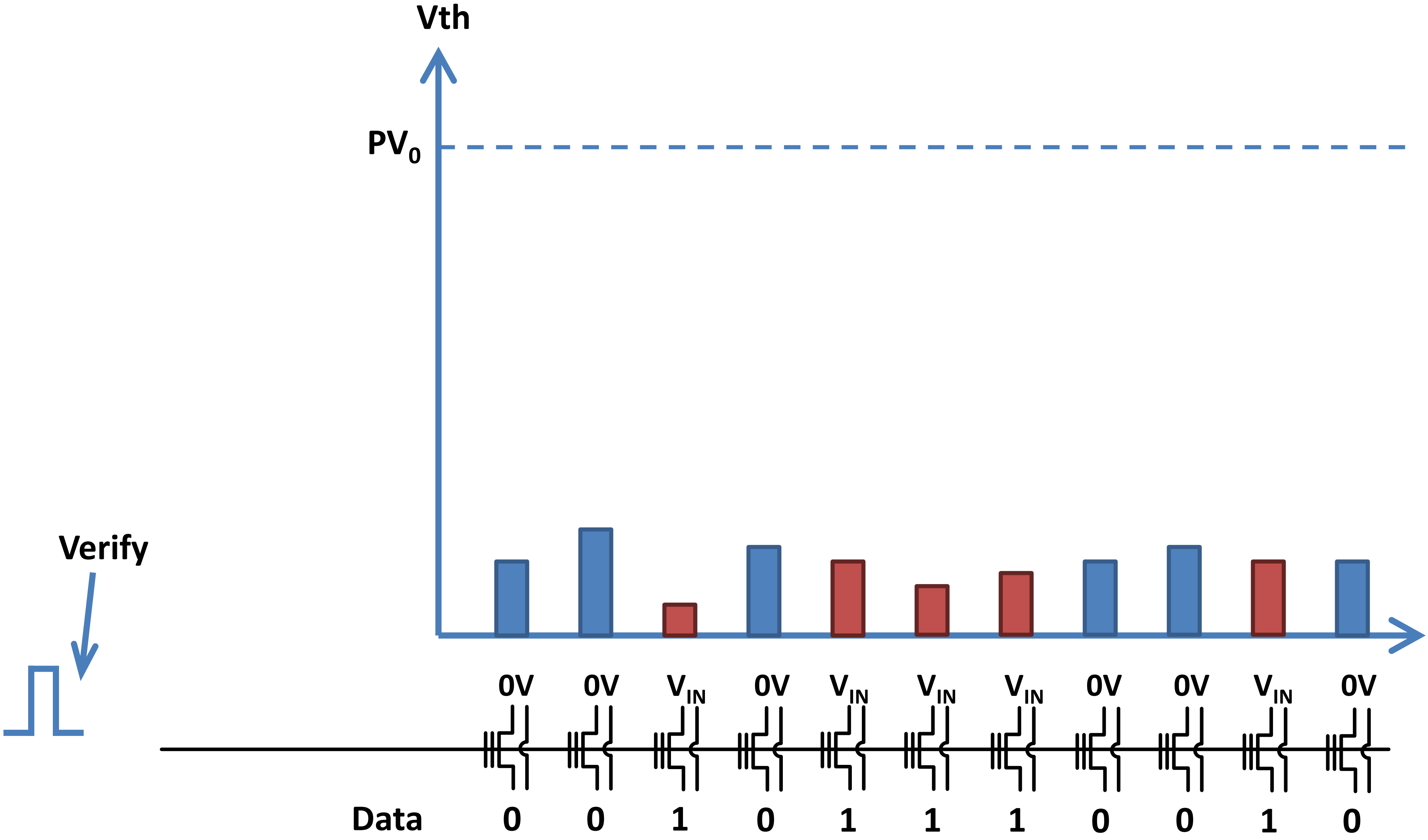
The Write Process ISPP



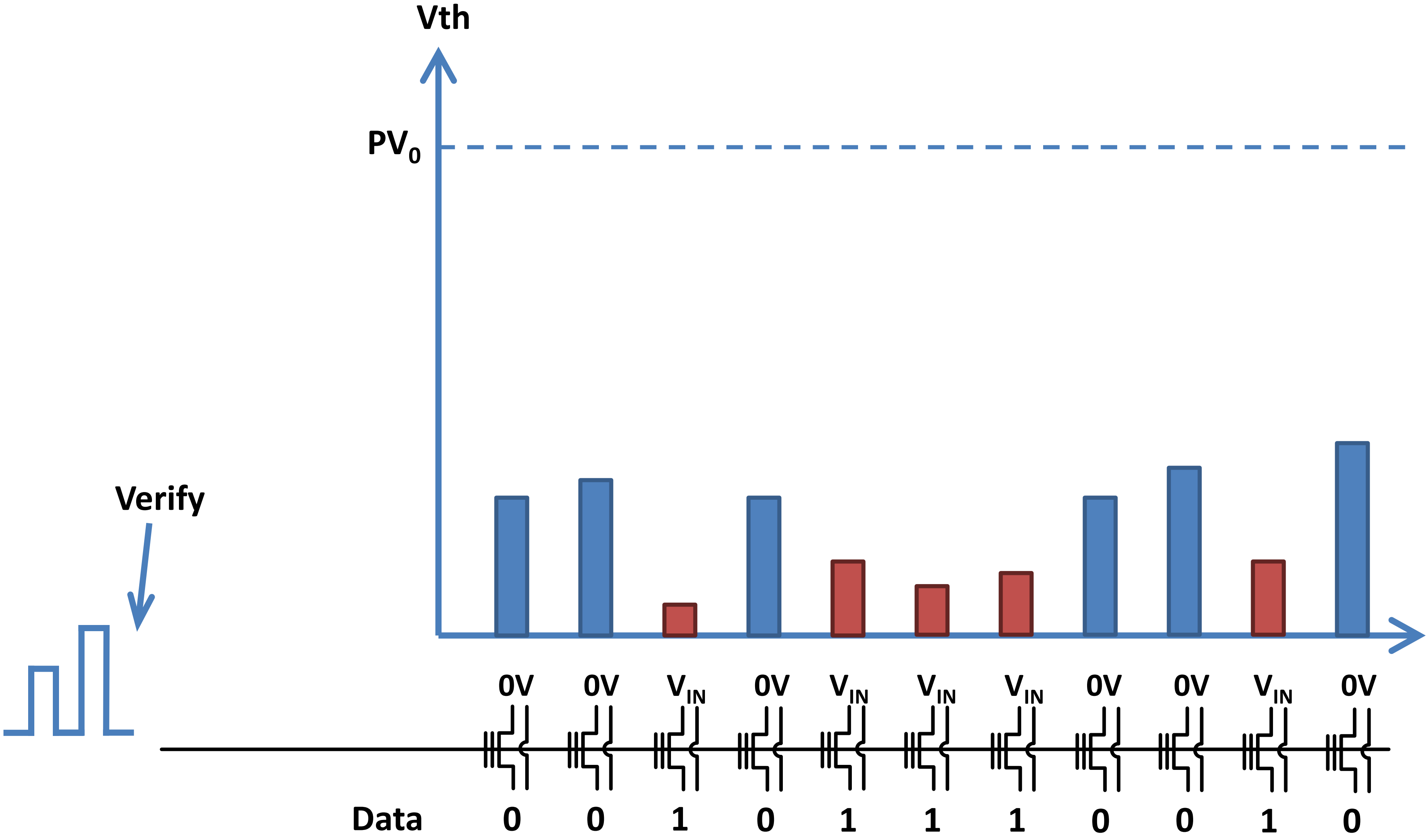
The Write Process ISPP



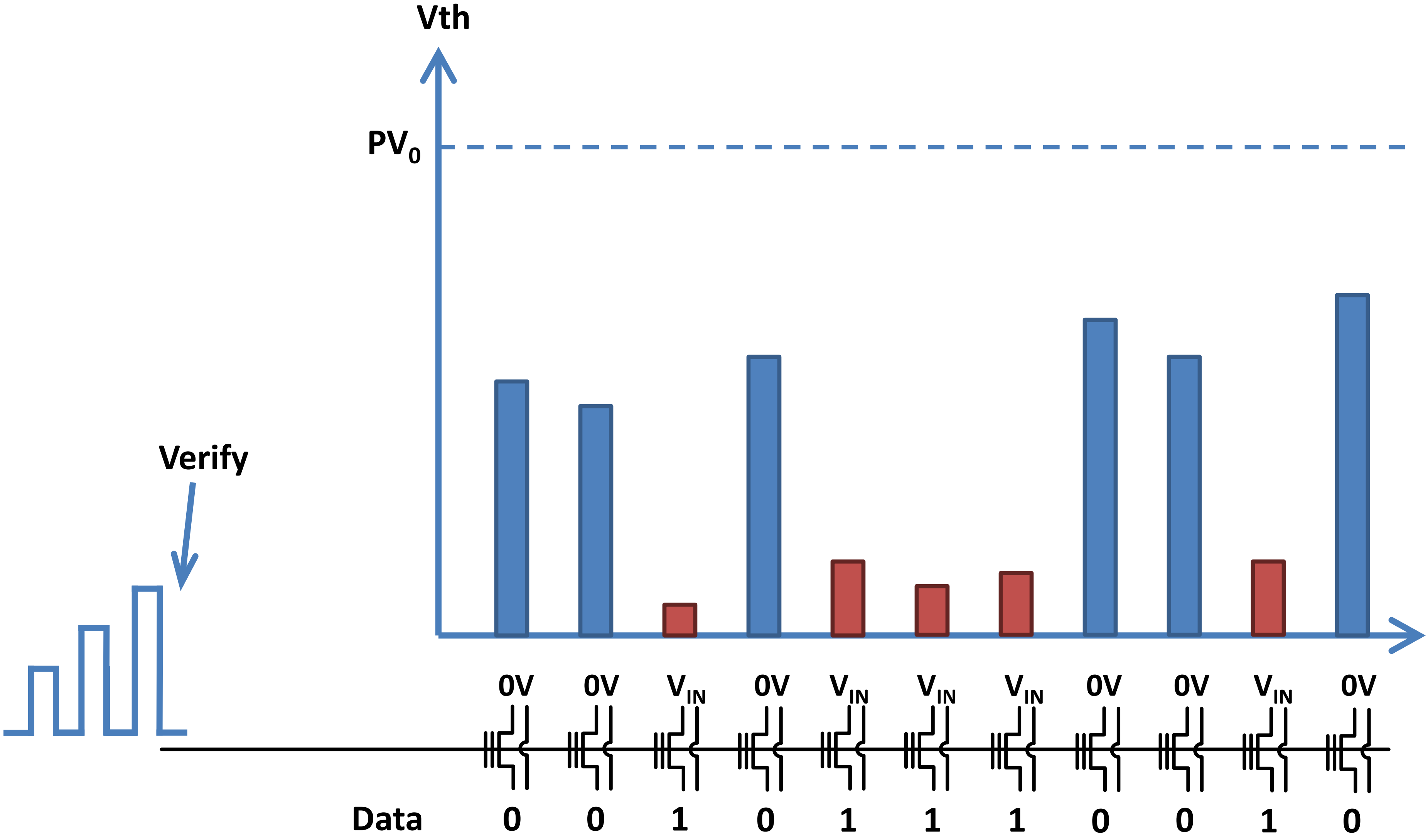
The Write Process ISPP



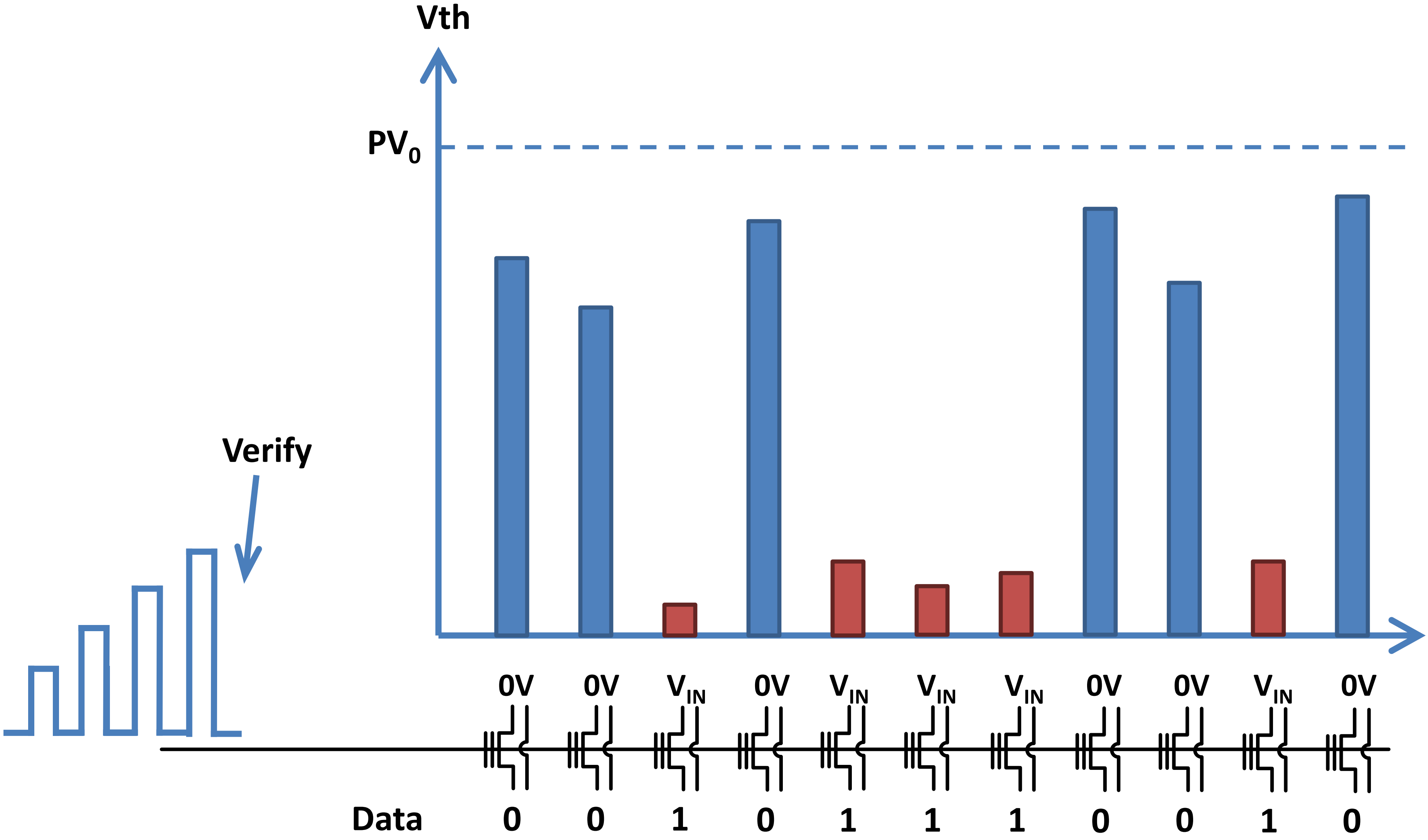
The Write Process ISPP



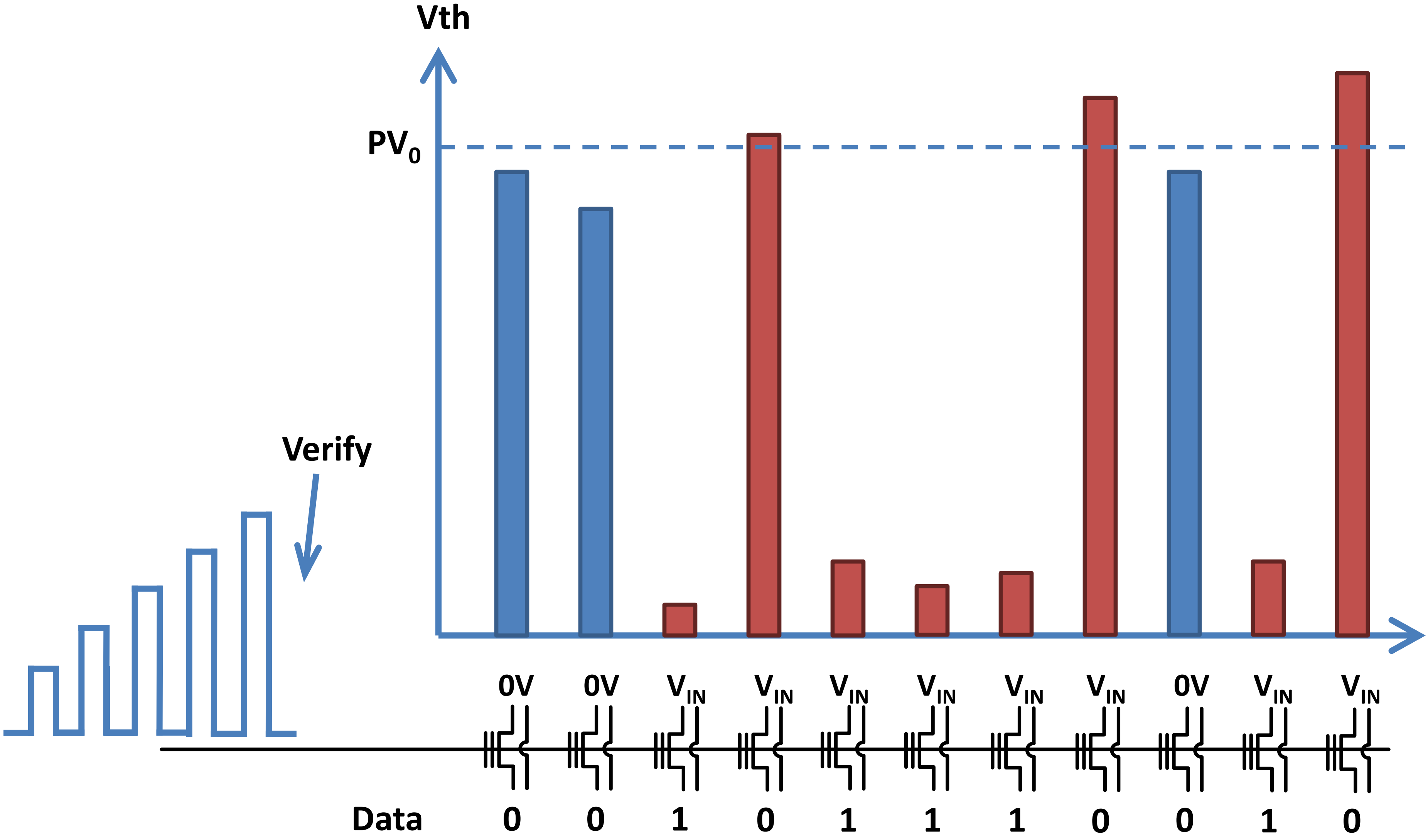
The Write Process ISPP



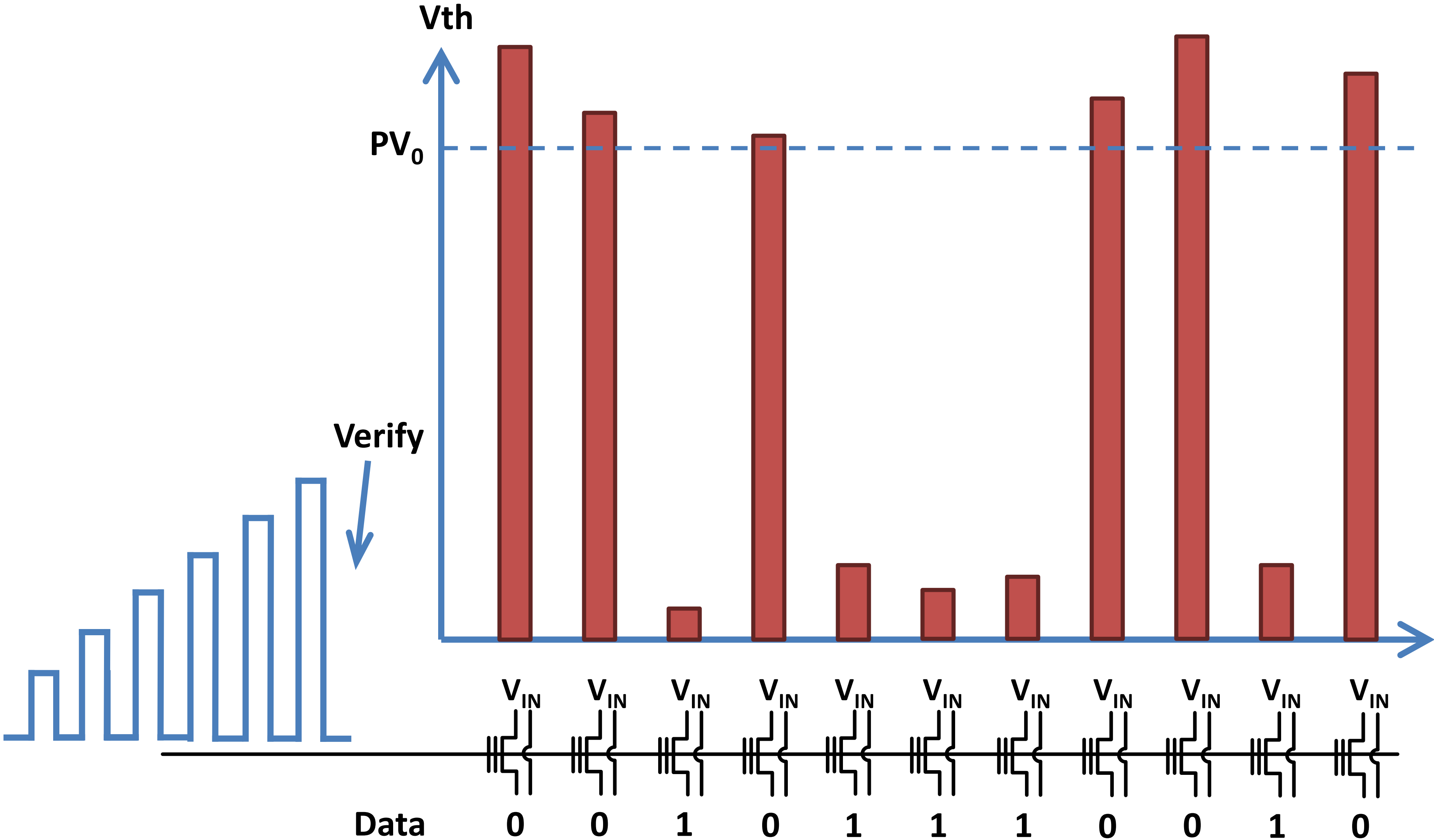
The Write Process ISPP



The Write Process ISPP



The Write Process ISPP



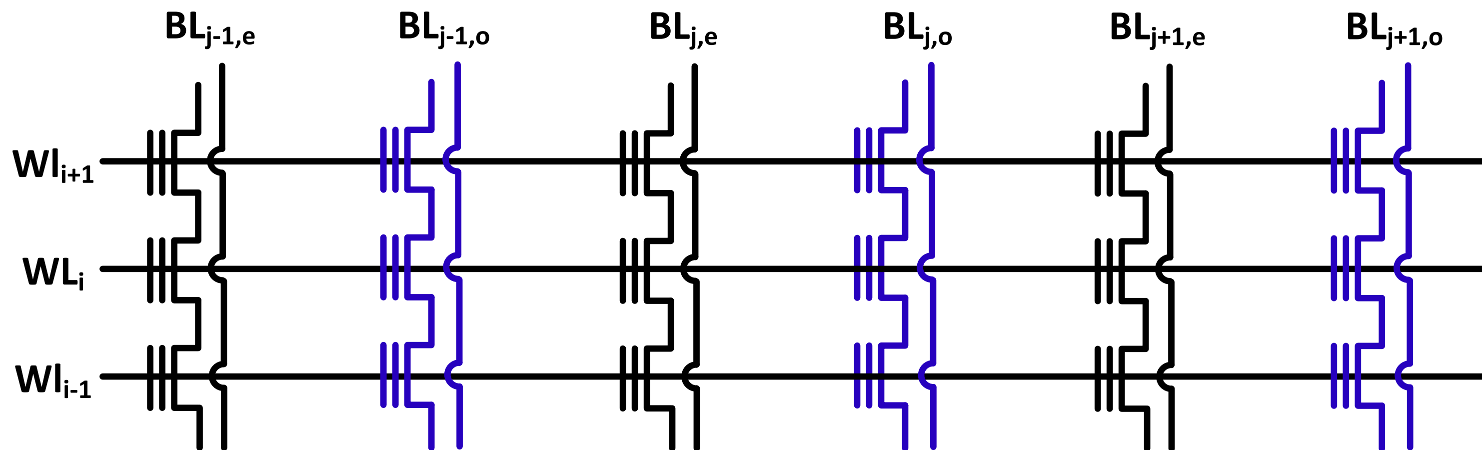
CAPACITIVE COUPLING

Capacitive Coupling

- Each floating gate is coupled to its neighbors.
  - Writing adds voltage to adjacent cells.

$V_{Add,Victim} \propto \alpha \times \left(V_{End,Aggressor} - V_{Start,Aggressor}\right)$

  - α depends on geometry (distance) and process.

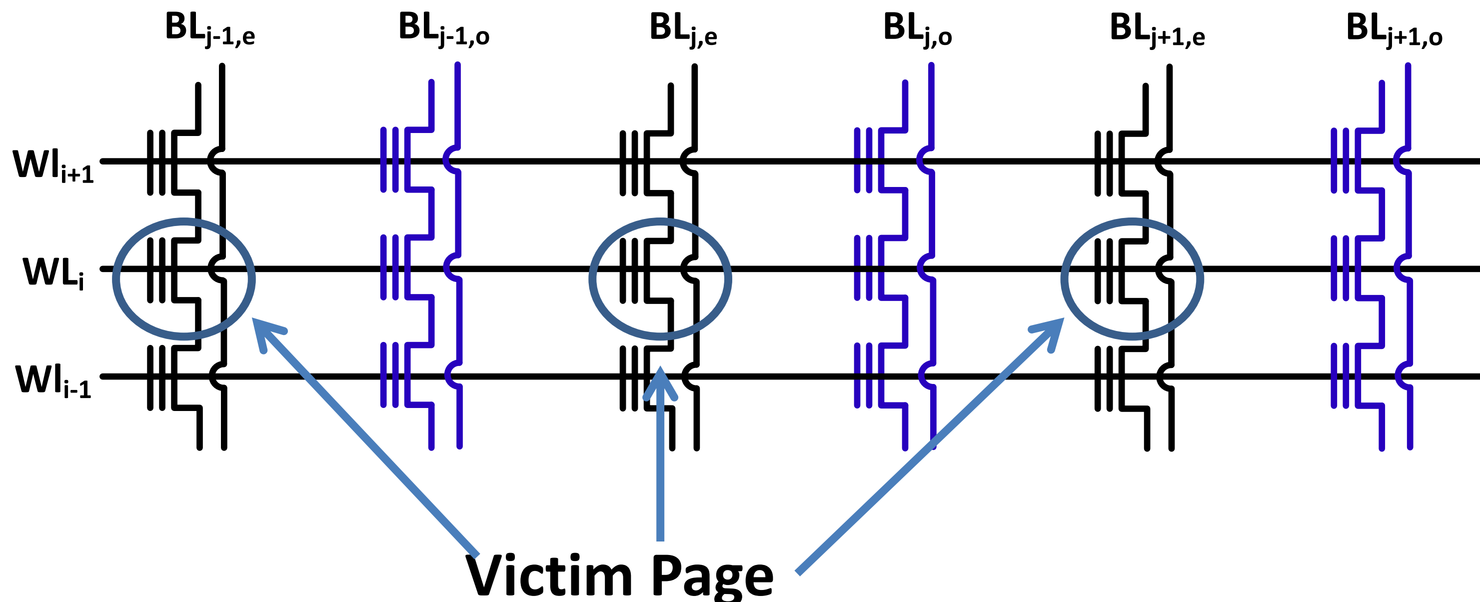


Capacitive Coupling

- Each floating gate is coupled to its neighbors.
  - Writing adds voltage to adjacent cells.

$$V_{Add,Victim} \propto \alpha \times (V_{End,Aggressor} - V_{Start,Aggressor})$$

  - α depends on geometry (distance) and process.

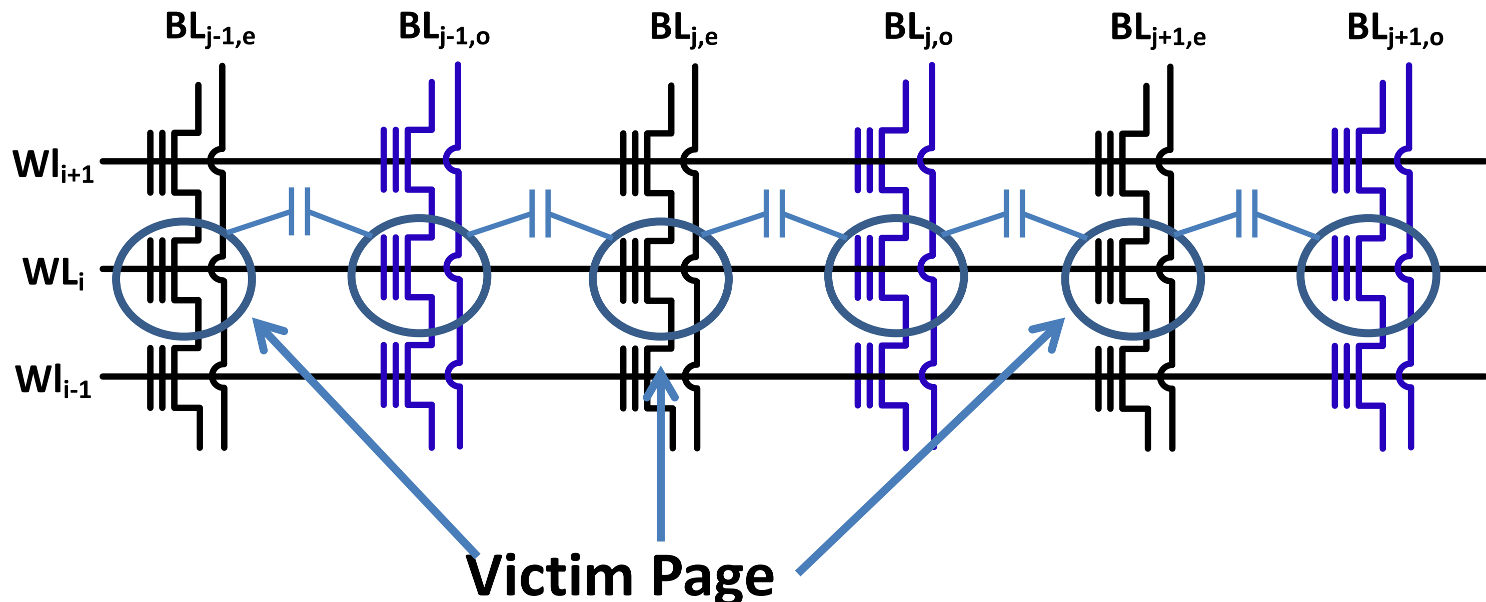


Capacitive Coupling

- Each floating gate is coupled to its neighbors.
  - Writing adds voltage to adjacent cells.

$$V_{Add,Victim} \propto \alpha \times (V_{End,Aggressor} - V_{Start,Aggressor})$$

  - α depends on geometry (distance) and process.

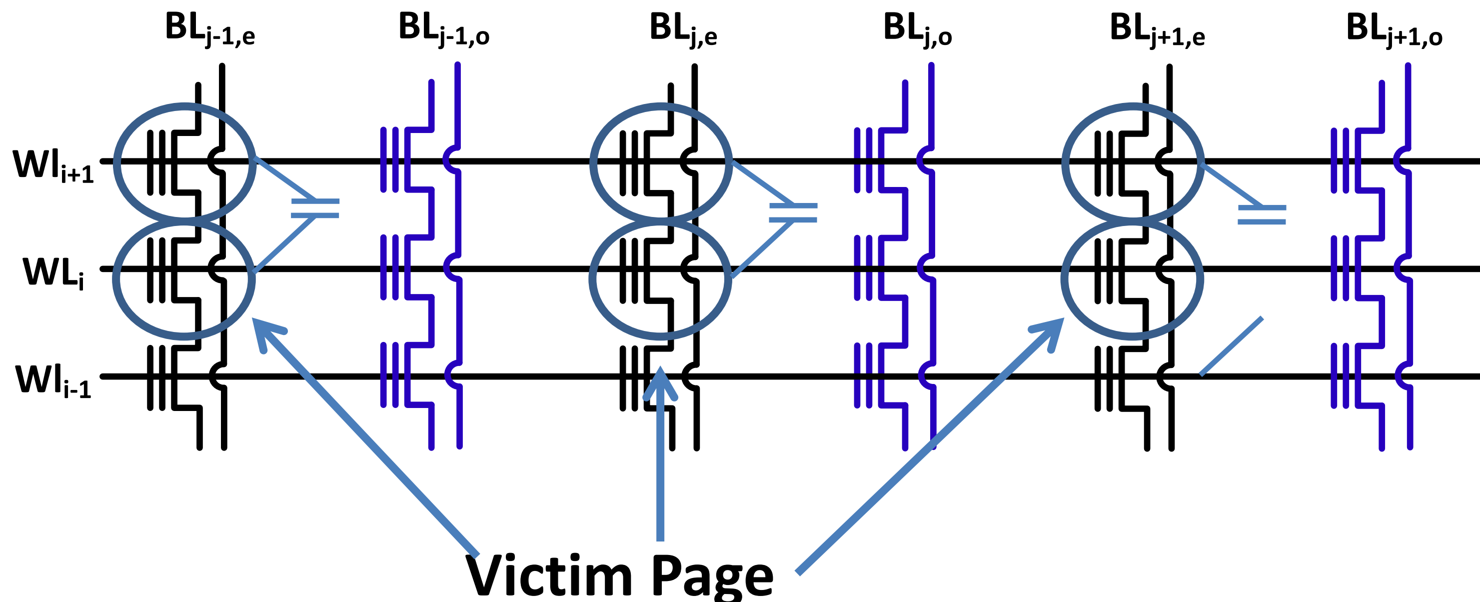


Capacitive Coupling

- Each floating gate is coupled to its neighbors.
  - Writing adds voltage to adjacent cells.

$$V_{Add,Victim} \propto \alpha \times \left(V_{End,Aggressor} - V_{Start,Aggressor}\right)$$

  - α depends on geometry (distance) and process.

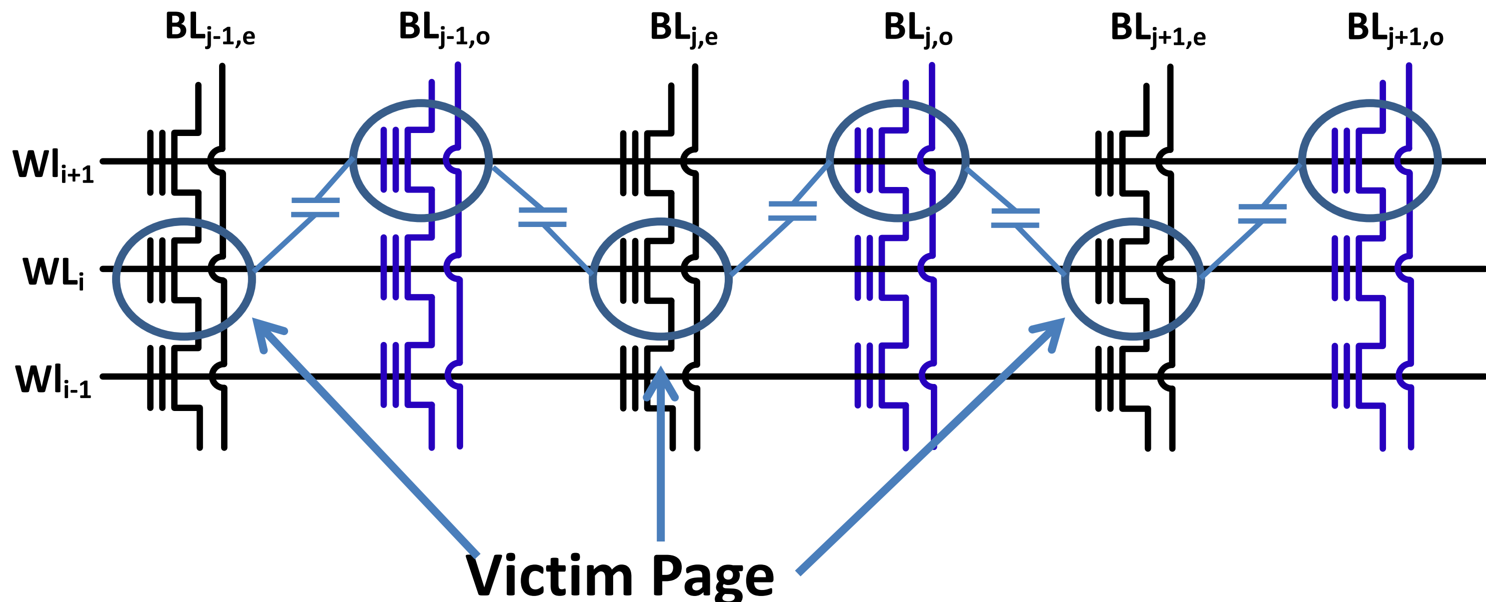


Capacitive Coupling

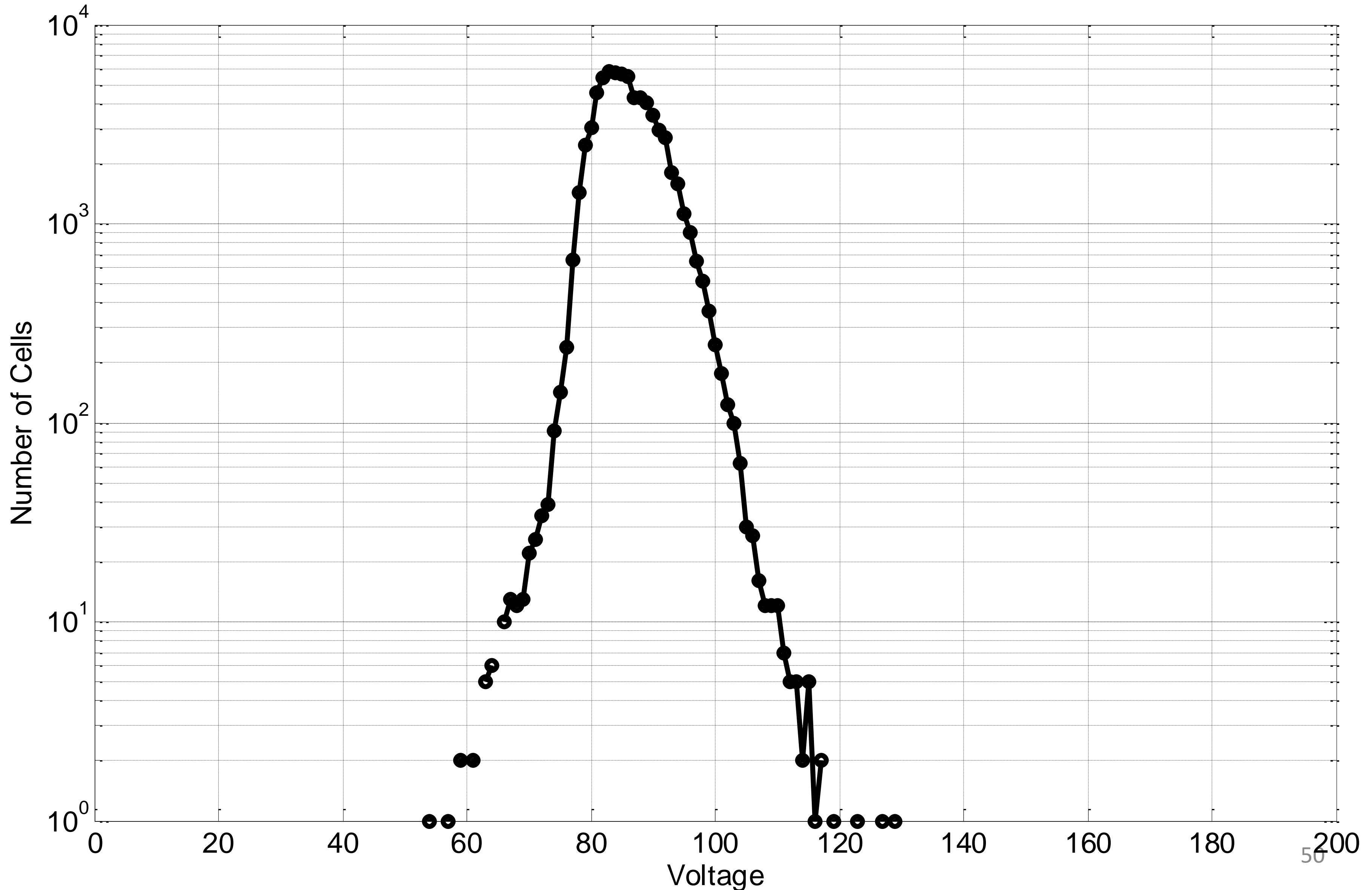
- Each floating gate is coupled to its neighbors.
  - Writing adds voltage to adjacent cells.

$$V_{Add,Victim} \propto \alpha \times \left(V_{End,Aggressor} - V_{Start,Aggressor}\right)$$

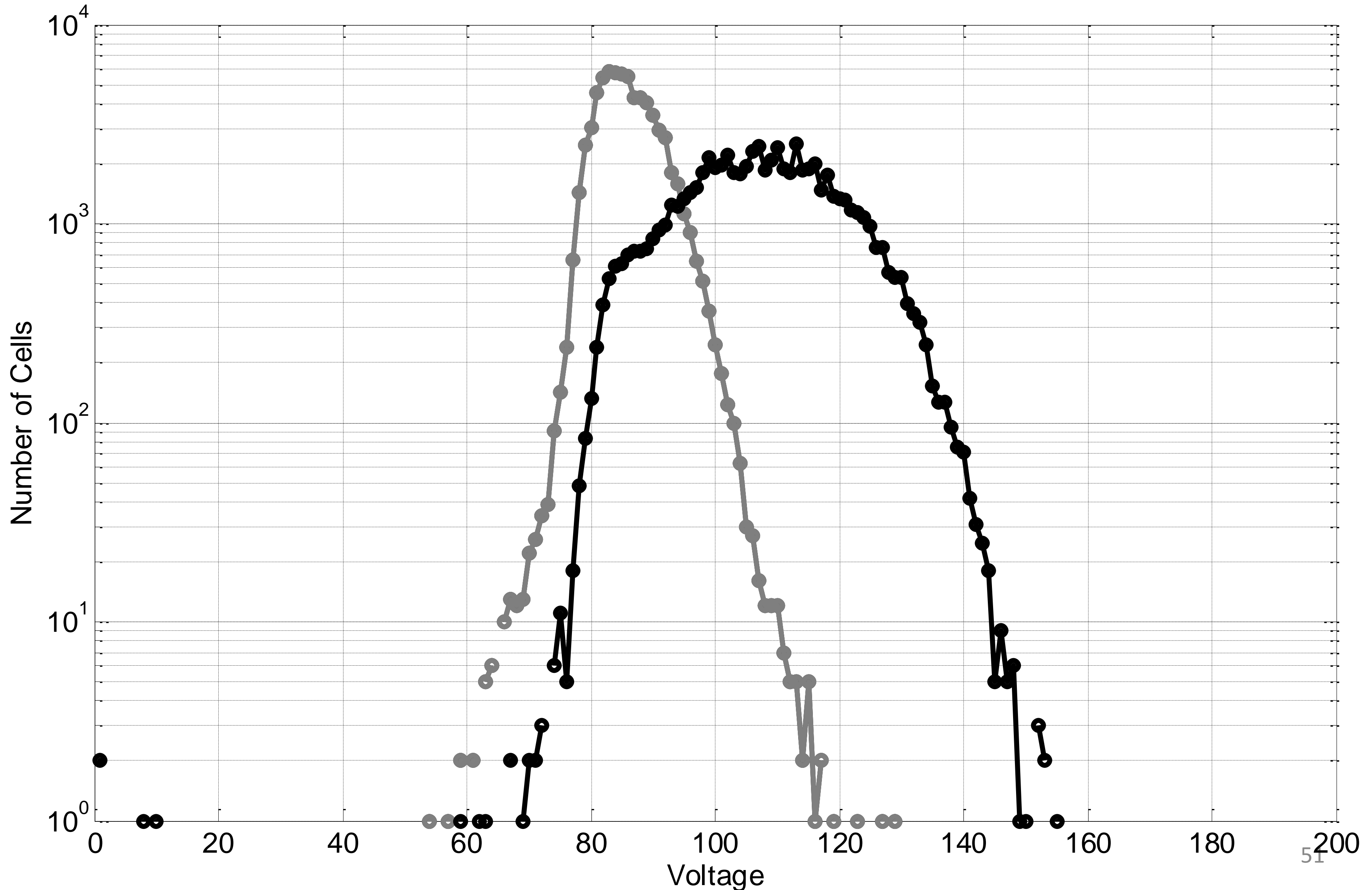
  - α depends on geometry (distance) and process.



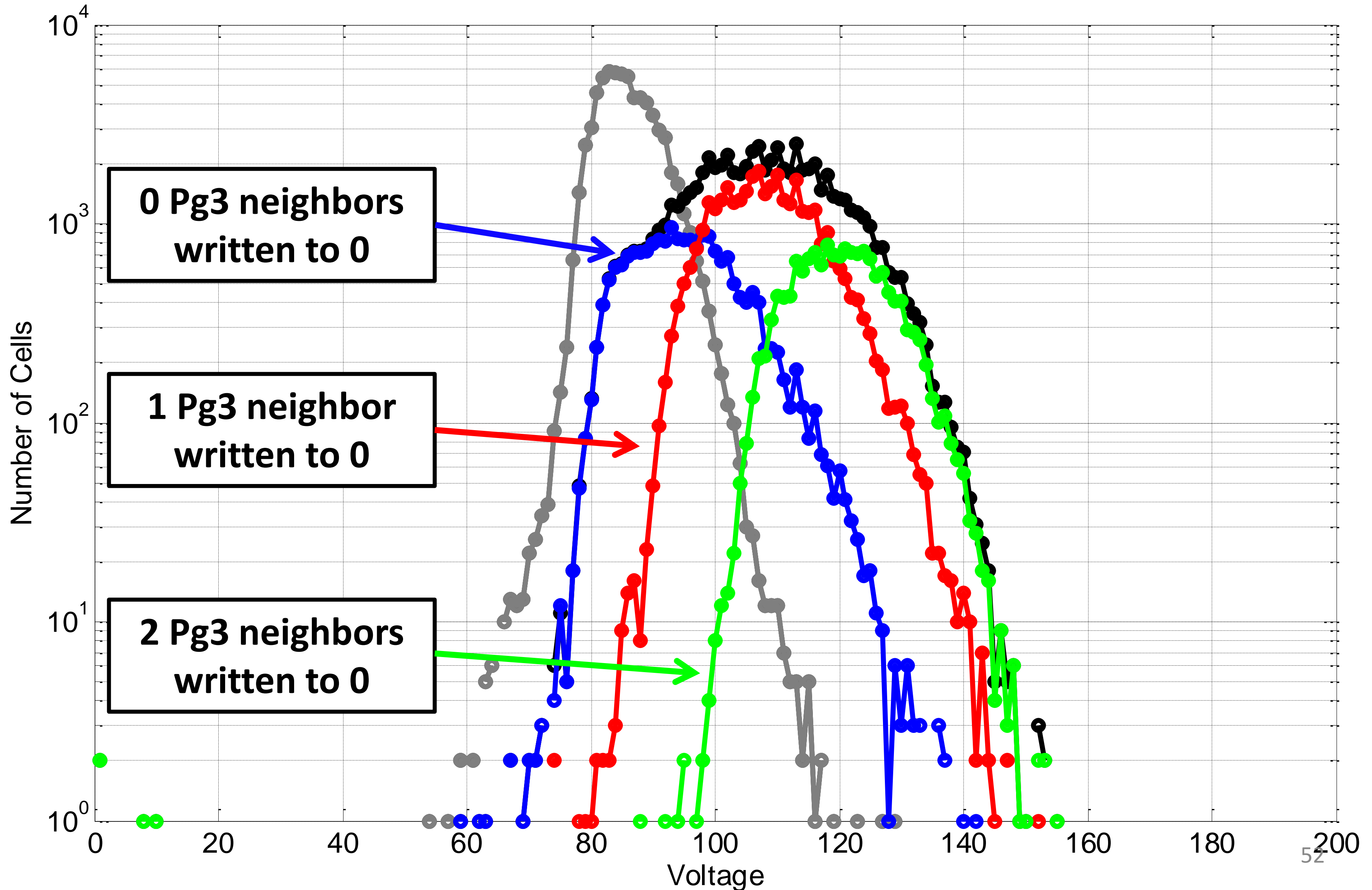
Actual LSB Write (Pg2 after Pg2 Write)



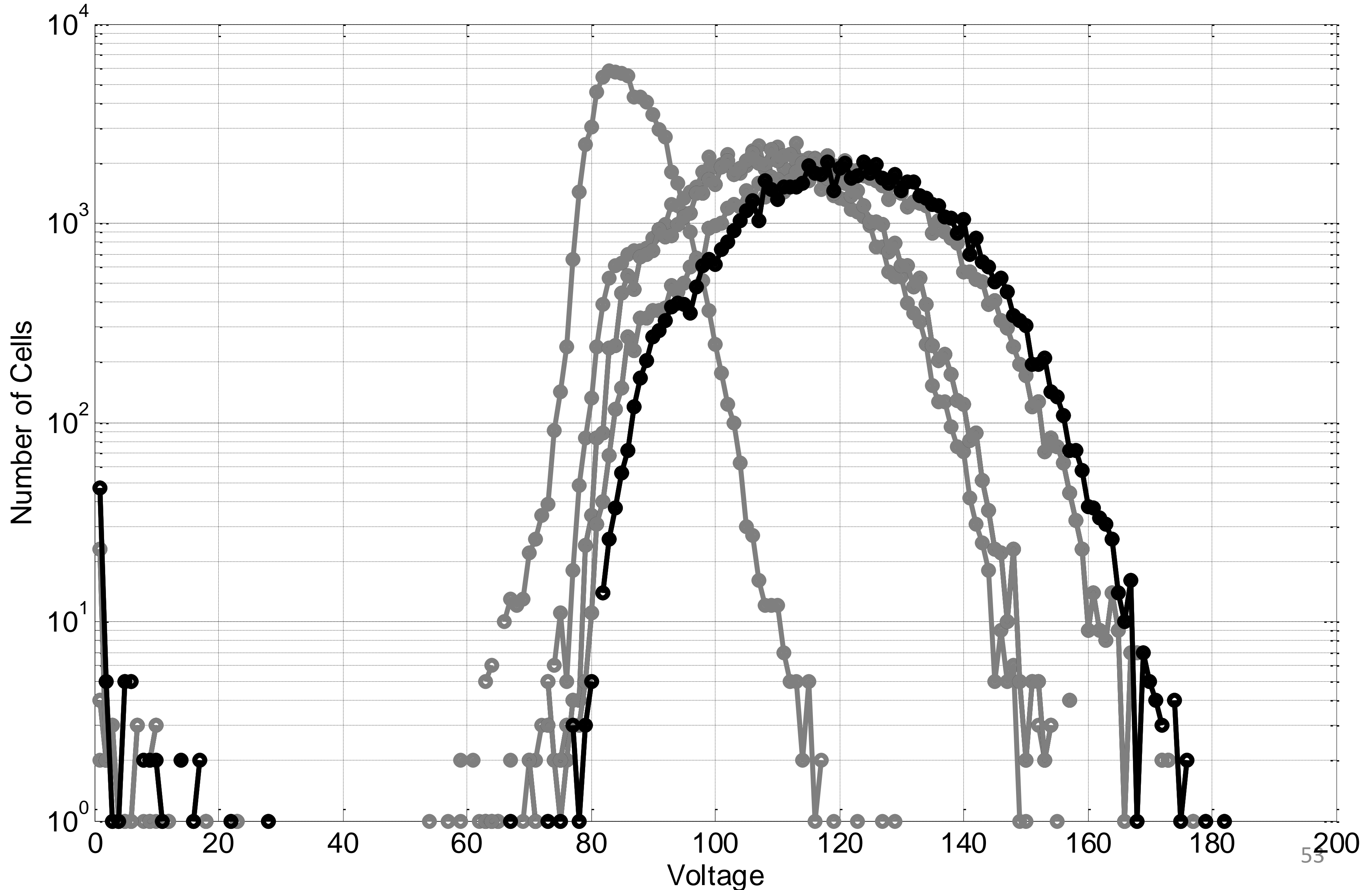
Actual LSB Write (Pg2 after Pg3 Write)



Actual LSB Write (Pg2 after Pg3 Write)

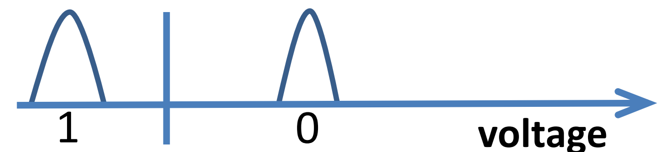


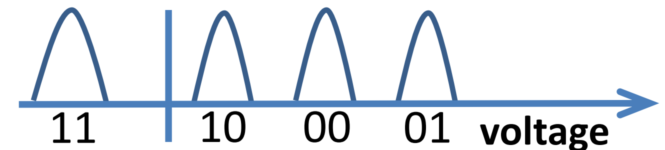
Actual LSB Write (Pg2 after Pg7 Write)



Write Sequence

- The sequence in which pages are written affects the induced capacitive coupling.
  - LSB applies ~0/2.5 volts.
  - MSB applies ~0/1.25 volts.





- The overall goal is to degrade the final distributions (after MSB) minimally.
- Write sequence acts to minimize this effect.

Even/Odd Bit Line (EOBL) Writing

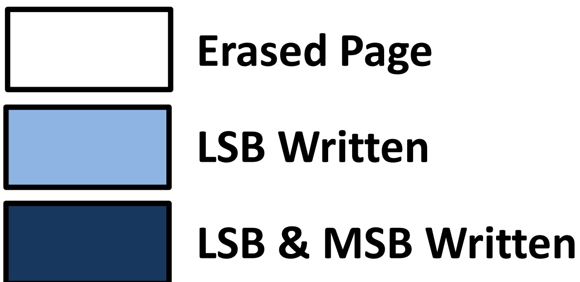
bitline

6 7 6 7 6 7 6 7 6

4 5 4 5 4 5 4 5 4

2 3 2 3 2 3 2 3 2

0 1 0 1 0 1 0 1 0 wordline



Even/Odd Bit Line (EOBL) Writing

bitline

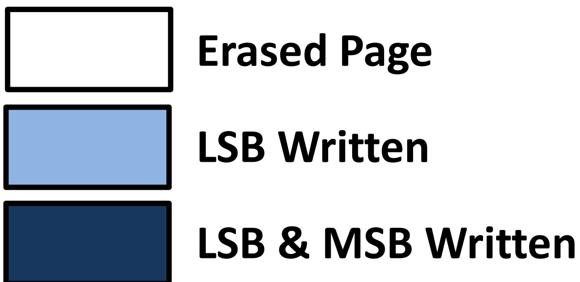
6 7 6 7 6 7 6 7 6

4 5 4 5 4 5 4 5 4

2 3 2 3 2 3 2 3 2

0 1 0 1 0 1 0 1 0

wordline



Even/Odd Bit Line (EOBL) Writing

bitline

6 7 6 7 6 7 6 7 6

4 5 4 5 4 5 4 5 4

2 3 2 3 2 3 2 3 2

0 1 0 1 0 1 0 1 0 **wordline**

Erased Page

LSB Written

LSB & MSB Written

Even/Odd Bit Line (EOBL) Writing

bitline

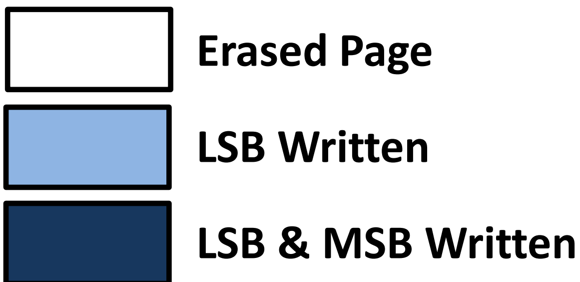
6 7 6 7 6 7 6 7 6

4 5 4 5 4 5 4 5 4

2 3 2 3 2 3 2 3 2

0 1 0 1 0 1 0 1 0

wordline



Even/Odd Bit Line (EOBL) Writing

bitline

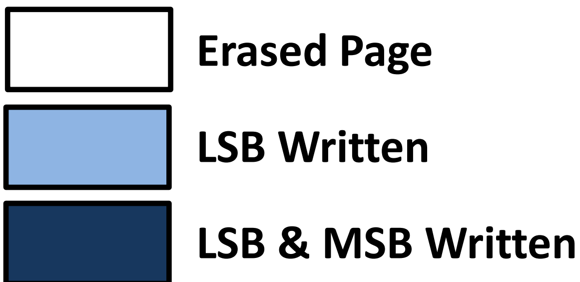
6 7 6 7 6 7 6 7 6

4 5 4 5 4 5 4 5 4

2 3 2 3 2 3 2 3 2

0 1 0 1 0 1 0 1 0

wordline

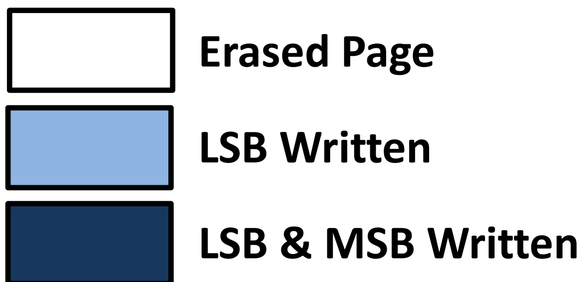


Even/Odd Bit Line (EOBL) Writing

bitline

6	7	6	7	6	7	6	7	6
4	5	4	5	4	5	4	5	4
2	3	2	3	2	3	2	3	2
0	1	0	1	0	1	0	1	0

wordline



Even/Odd Bit Line (EOBL) Writing

bitline

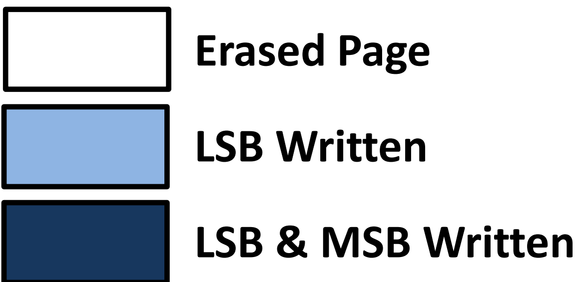
6 7 6 7 6 7 6 7 6

4 5 4 5 4 5 4 5 4

2 3 2 3 2 3 2 3 2

0 1 0 1 0 1 0 1 0

wordline



Even/Odd Bit Line (EOBL) Writing

bitline

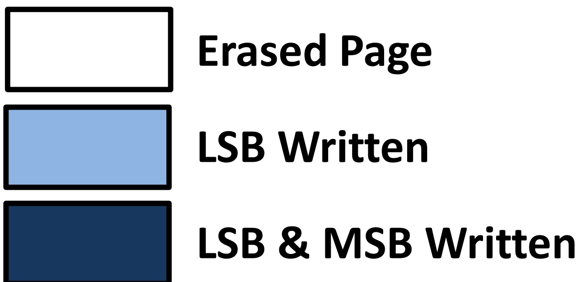
6 7 6 7 6 7 6 7 6

4 5 4 5 4 5 4 5 4

2 3 2 3 2 3 2 3 2

0 1 0 1 0 1 0 1 0

wordline



Even/Odd Bit Line (EOBL) Writing

bitline

6	7	6	7	6	7	6	7	6
4	5	4	5	4	5	4	5	4
2	3	2	3	2	3	2	3	2
0	1	0	1	0	1	0	1	0

wordline

Erased Page

LSB Written

LSB & MSB Written

Even/Odd Bit Line (EOBL) Writing

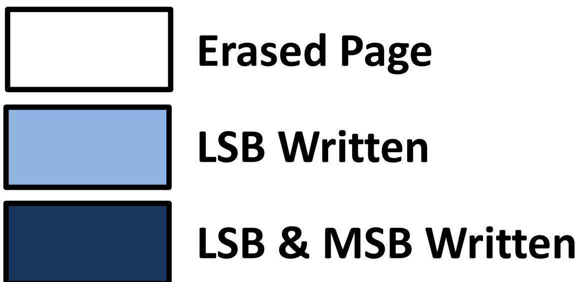
bitline

6 7 6 7 6 7 6 7 6

4 5 4 5 4 5 4 5 4

2 3 2 3 2 3 2 3 2

0 1 0 1 0 1 0 1 0 wordline

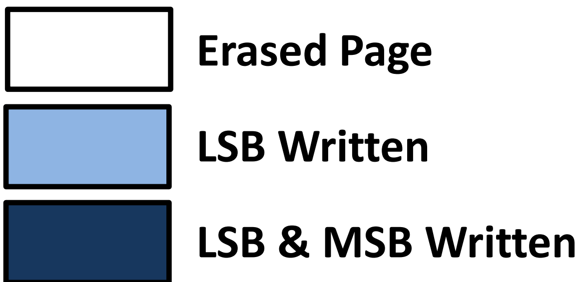


Even/Odd Bit Line (EOBL) Writing

bitline

6	7	6	7	6	7	6	7	6
4	5	4	5	4	5	4	5	4
2	3	2	3	2	3	2	3	2
0	1	0	1	0	1	0	1	0

wordline



Even/Odd Bit Line (EOBL) Writing

bitline

6 7 6 7 6 7 6 7 6

4 5 4 5 4 5 4 5 4

2 3 2 3 2 3 2 3 2

0 1 0 1 0 1 0 1 0

wordline

Erased Page

LSB Written

LSB & MSB Written

Even/Odd Bit Line (EOBL) Writing

bitline

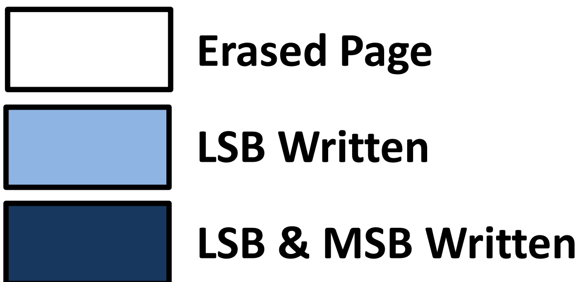
6 7 6 7 6 7 6 7 6

4 5 4 5 4 5 4 5 4

2 3 2 3 2 3 2 3 2

0 1 0 1 0 1 0 1 0

wordline



Even/Odd Bit Line (EOBL) Writing

bitline

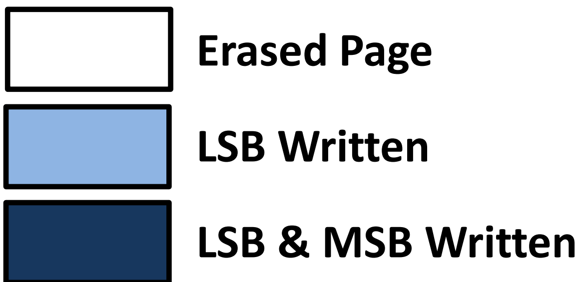
6 7 6 7 6 7 6 7 6

4 5 4 5 4 5 4 5 4

2 3 2 3 2 3 2 3 2

0 1 0 1 0 1 0 1 0

wordline



Even/Odd Bit Line (EOBL) Writing

bitline

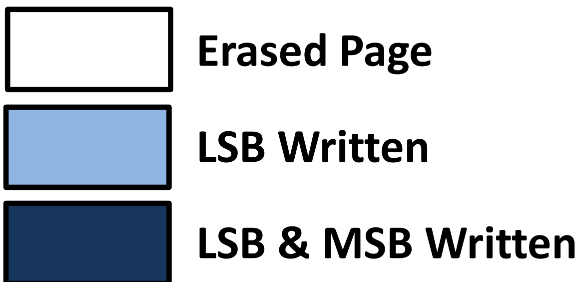
6 7 6 7 6 7 6 7 6

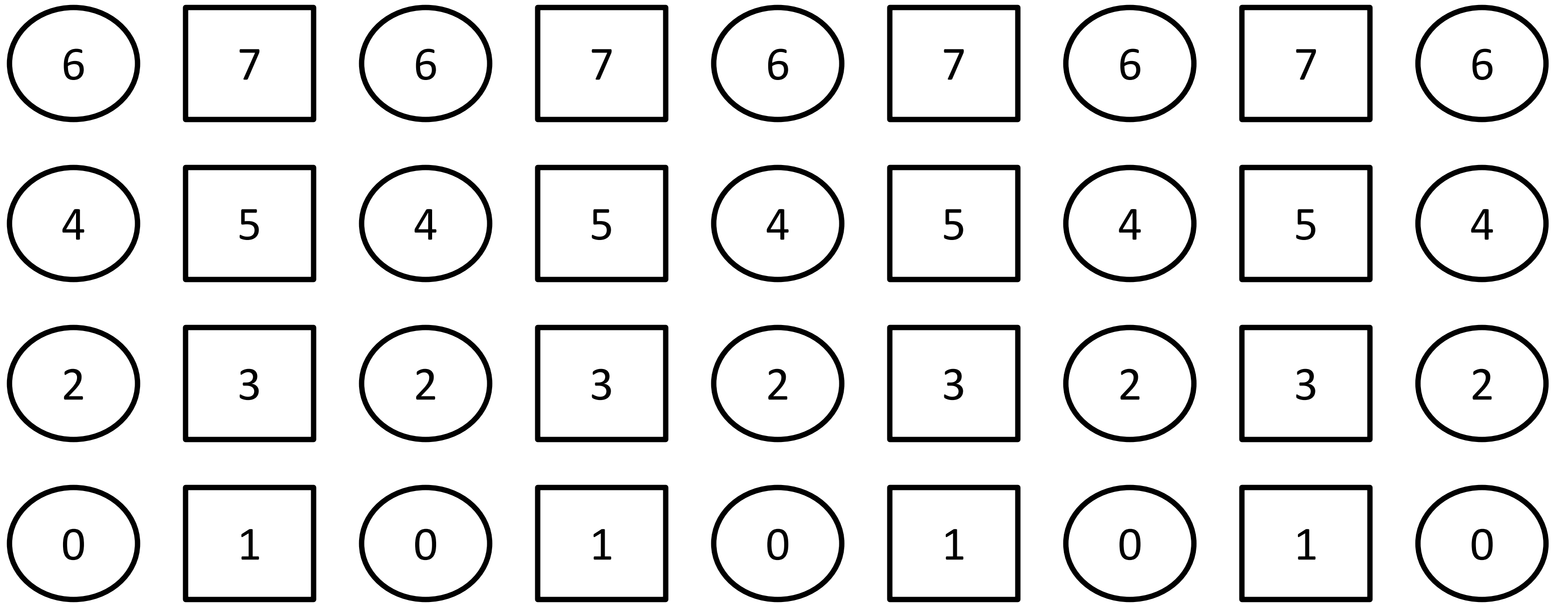
4 5 4 5 4 5 4 5 4

2 3 2 3 2 3 2 3 2

0 1 0 1 0 1 0 1 0

wordline





6
7
6
7
6
7
6
7
6
4
5
4
5
4
5
4
5
4
2
3
2
3
2
3
2
3
2
0
1
0
1
0
1
0
1
0

6 7 6 7 6 7 6 7 6

4 5 4 5 4 5 4 5 4

2 3 2 3 2 3 2 3 2

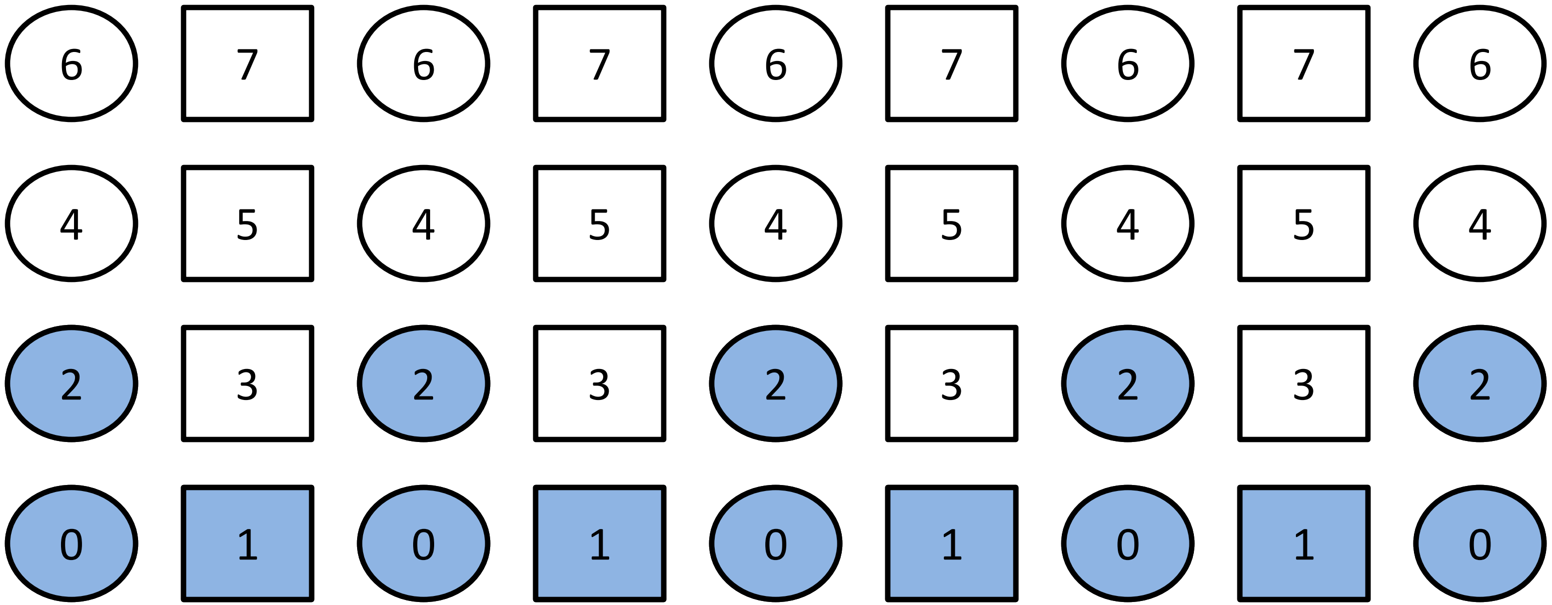
0 1 0 1 0 1 0 1 0

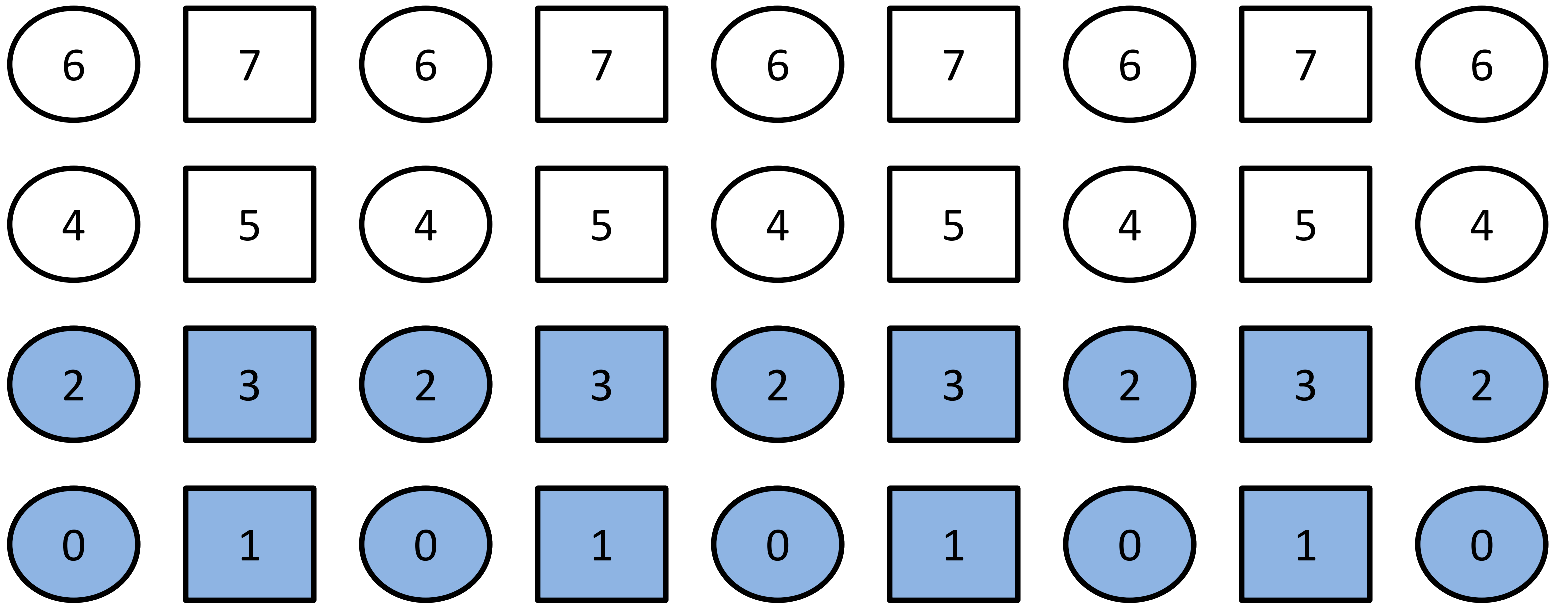
6 7 6 7 6 7 6 7 6

4 5 4 5 4 5 4 5 4

2 3 2 3 2 3 2 3 2

0 1 0 1 0 1 0 1 0


6 7 6 7 6 7 6 7 6
4 5 4 5 4 5 4 5 4
2 3 2 3 2 3 2 3 2
0 1 0 1 0 1 0 1 0
Page 2 Cells
Voltage
10^0 10^1 10^2 10^3 10^4
0 50 100 150 200 250

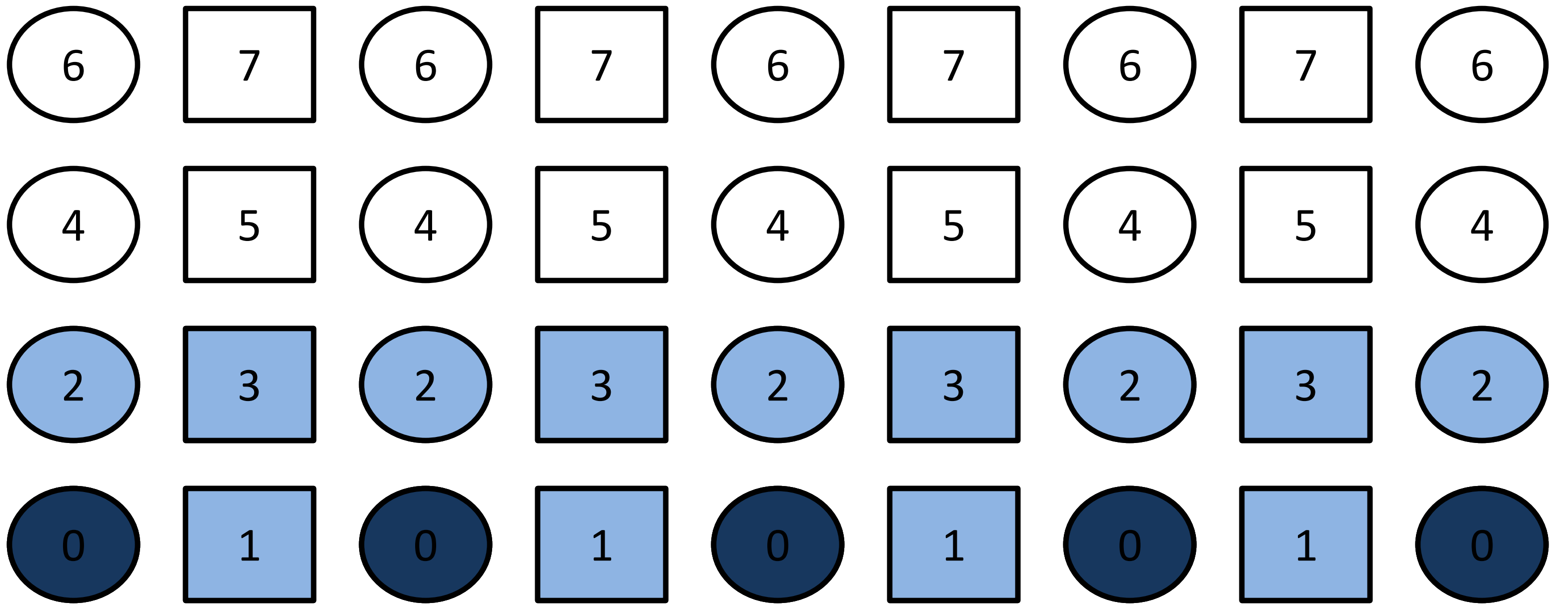


10^4
10^3
10^2
10^1
10^0

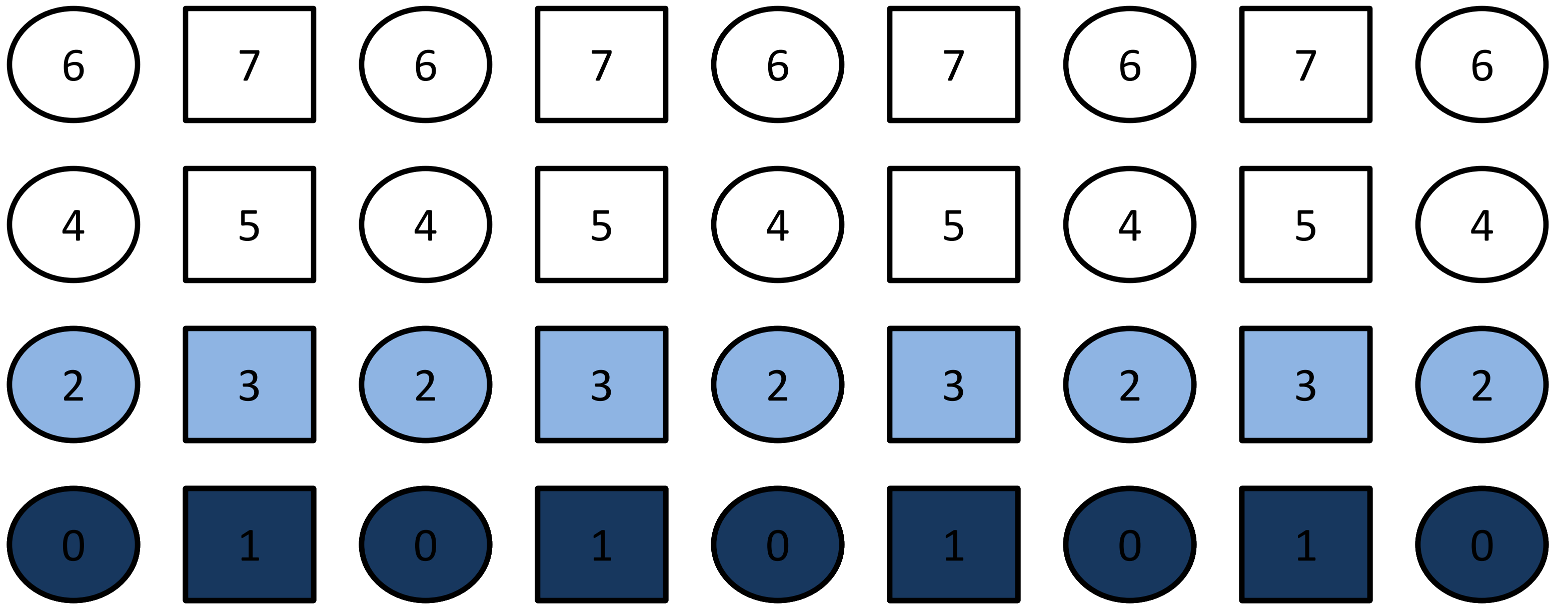
Page 2 Cells

0 50 100 150 200 250

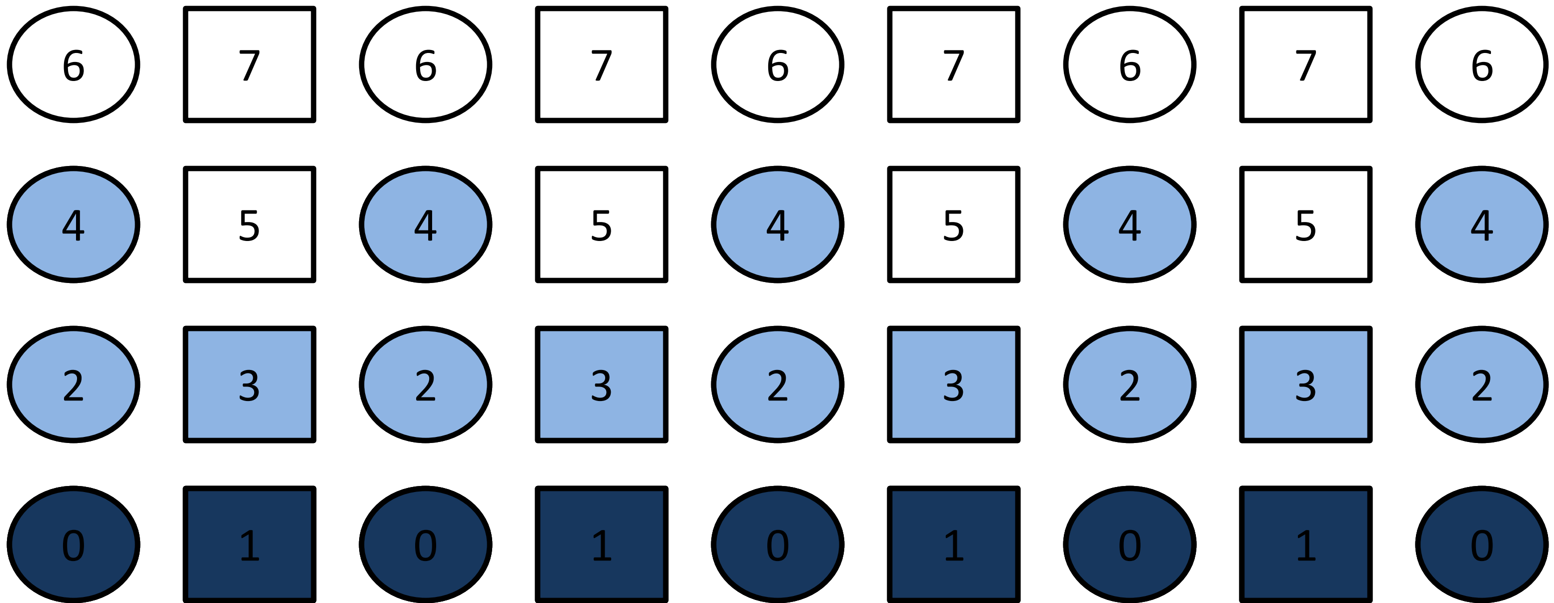
Voltage



6 7 6 7 6 7 6 7 6
4 5 4 5 4 5 4 5 4
2 3 2 3 2 3 2 3 2
0 1 0 1 0 1 0 1 0
Page 2 Cells
10^4
10^3
10^2
10^1
10^0
0 50 100 150 200 250
Voltage

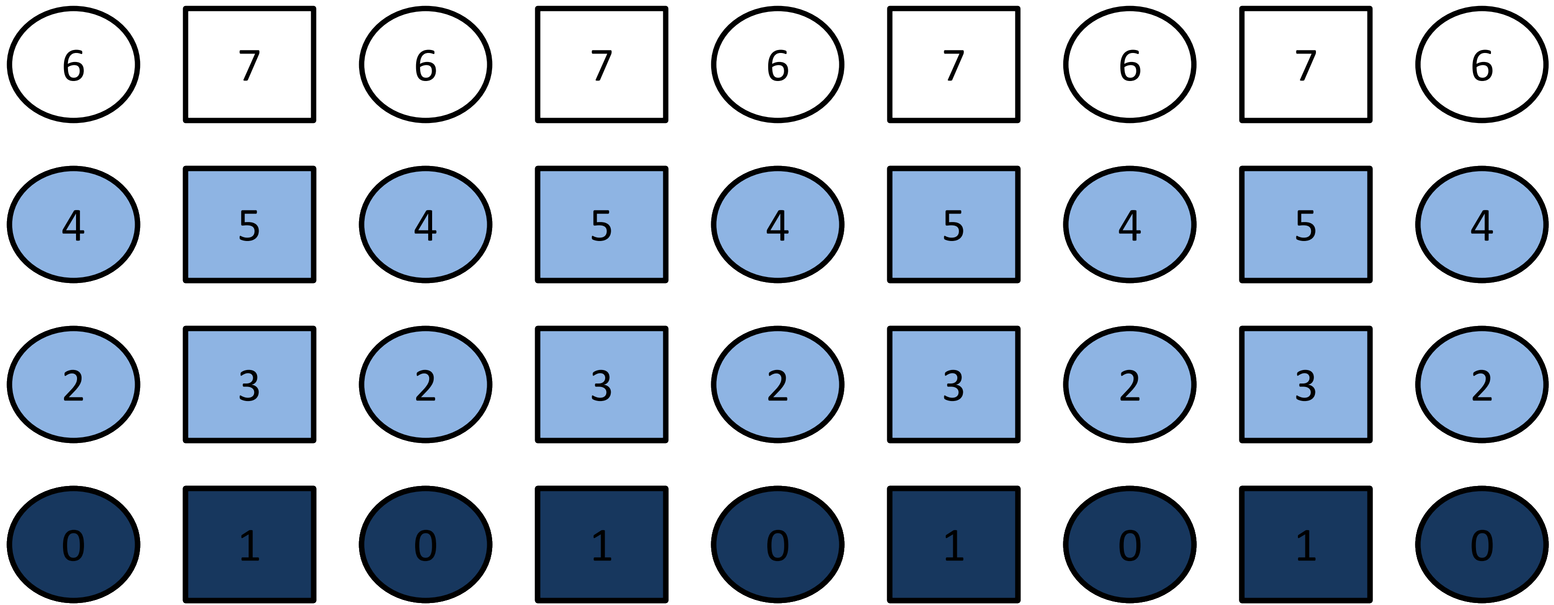

6 7 6 7 6 7 6 7 6
4 5 4 5 4 5 4 5 4
2 3 2 3 2 3 2 3 2
0 1 0 1 0 1 0 1 0

Page 2 Cells
10^4
10^3
10^2
10^1
10^0
0 50 100 150 200 250
Voltage

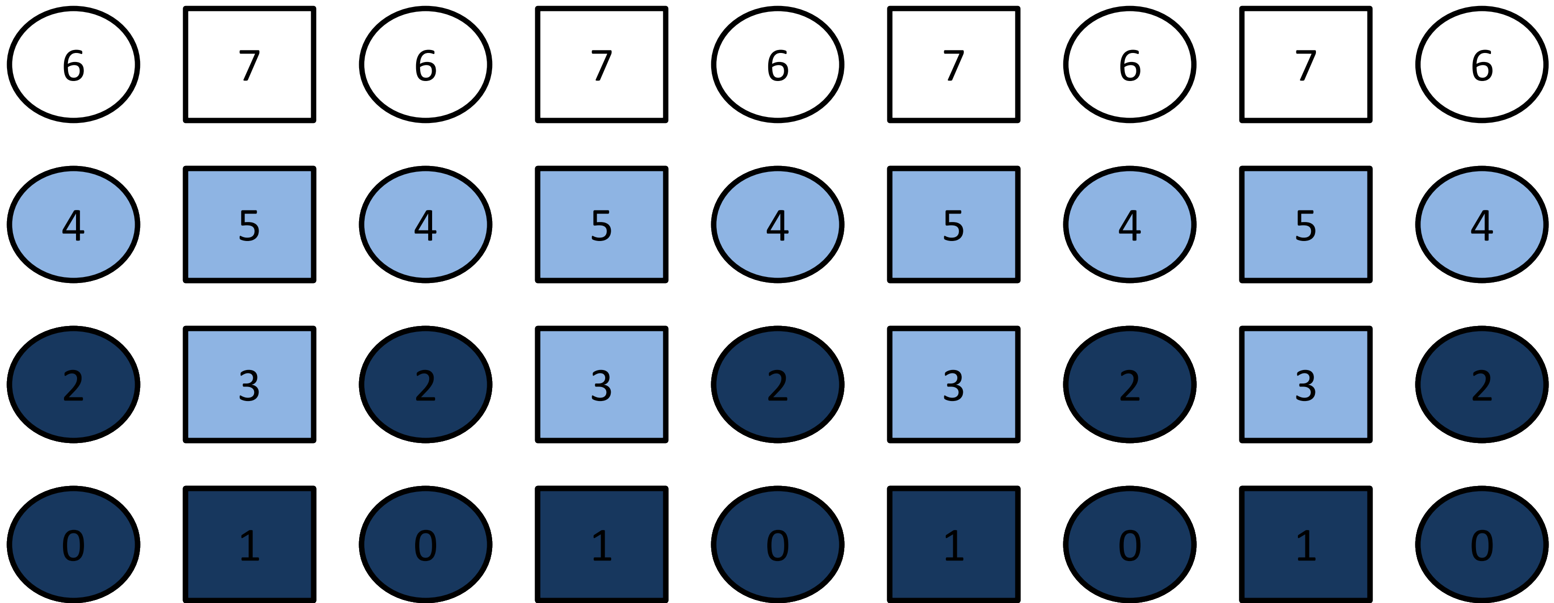


6 7 6 7 6 7 6 7 6
4 5 4 5 4 5 4 5 4
2 3 2 3 2 3 2 3 2
0 1 0 1 0 1 0 1 0

10^4
10^3
10^2
10^1
10^0
Page 2 Cells
0
50
100
150
200
250
Voltage

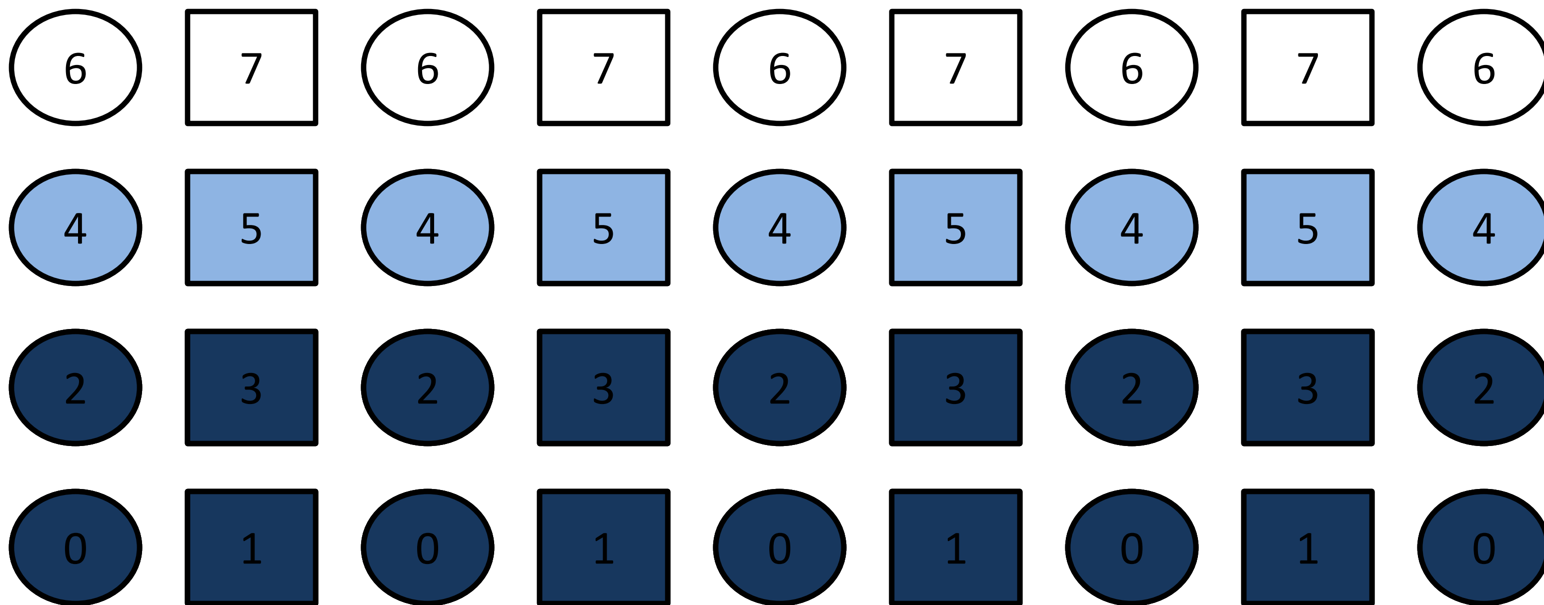

6 7 6 7 6 7 6 7 6
4 5 4 5 4 5 4 5 4
2 3 2 3 2 3 2 3 2
0 1 0 1 0 1 0 1 0

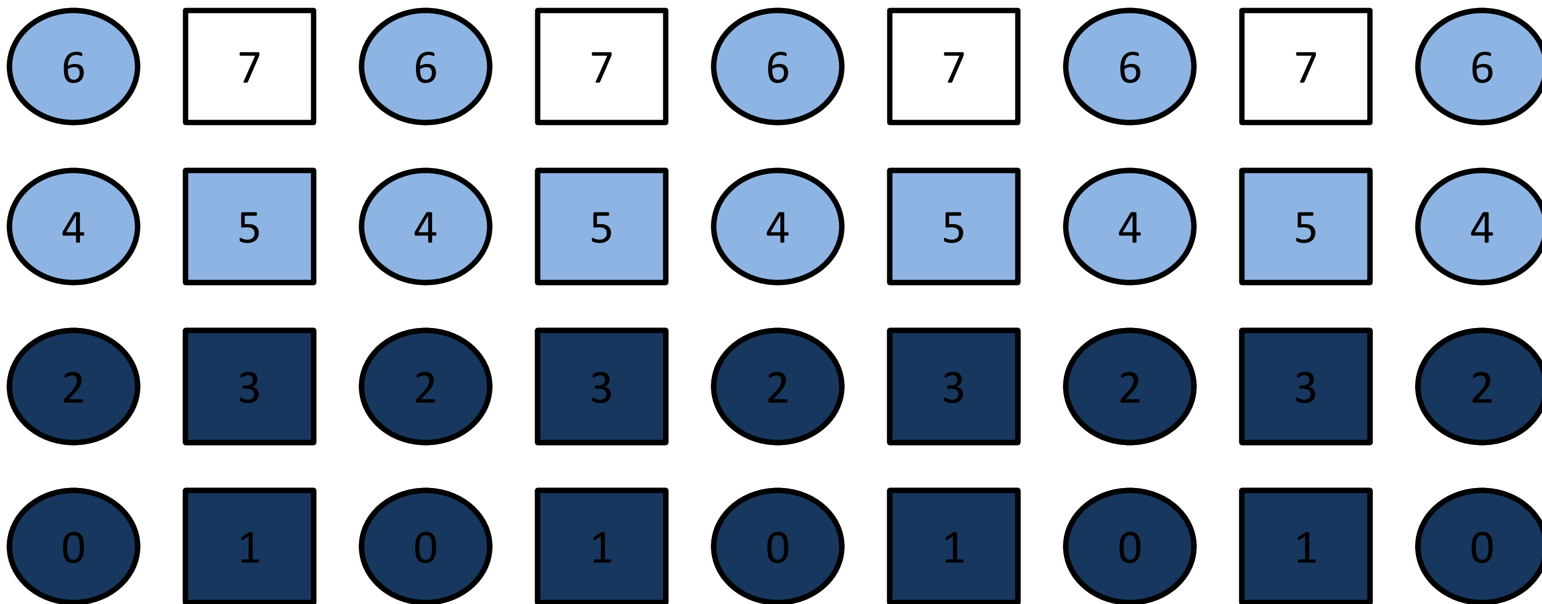
10^4
10^3
10^2
10^1
10^0
Page 2 Cells
0
50
100
150
200
250
Voltage

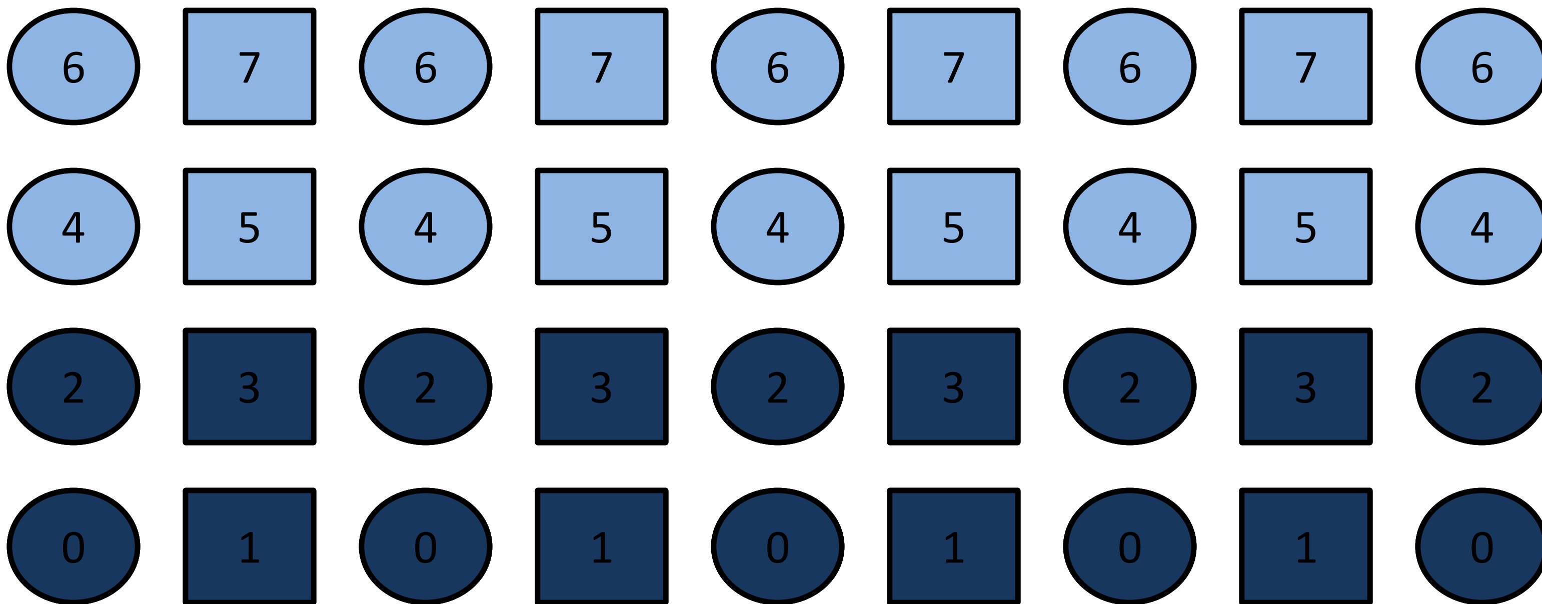


6 7 6 7 6 7 6 7 6
4 5 4 5 4 5 4 5 4
2 3 2 3 2 3 2 3 2
0 1 0 1 0 1 0 1 0

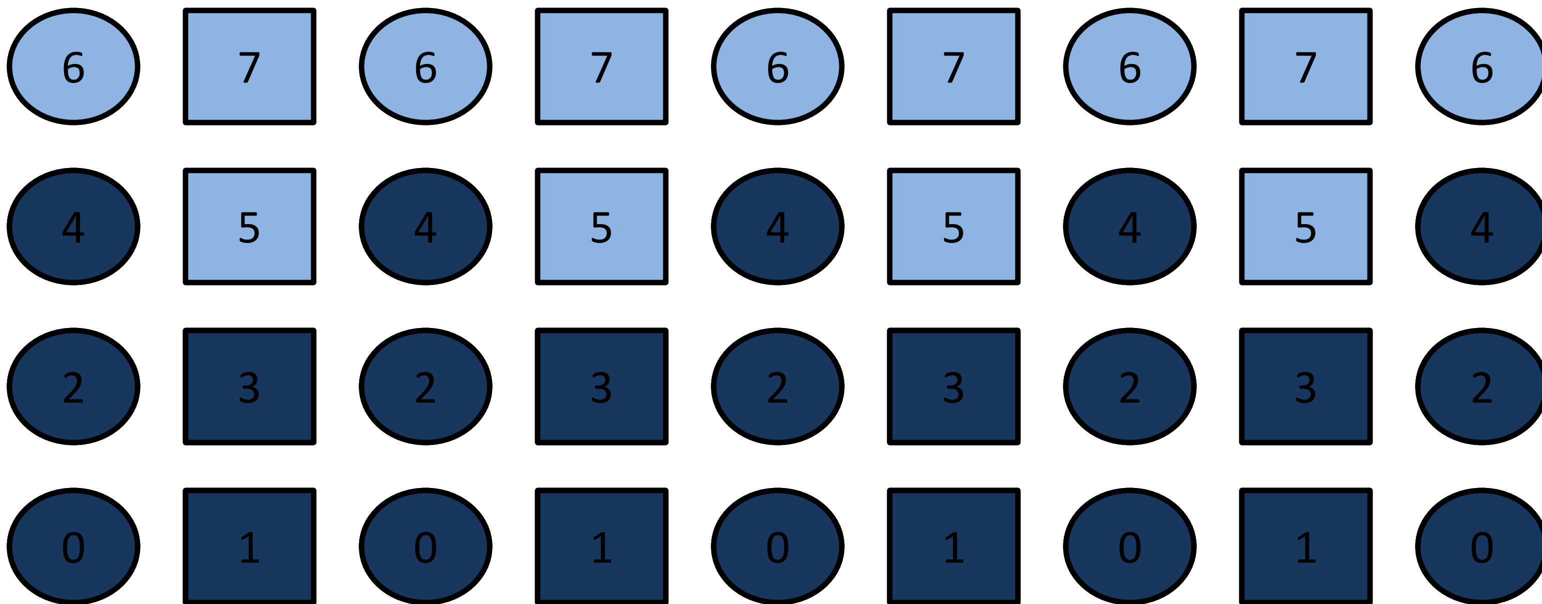
10^4
10^3
10^2
10^1
10^0
Page 2 Cells
0
50
100
150
200
250
Voltage


6 7 6 7 6 7 6 7 6
4 5 4 5 4 5 4 5 4
2 3 2 3 2 3 2 3 2
0 1 0 1 0 1 0 1 0
Page 2 Cells
Voltage
10^0
10^1
10^2
10^3
10^4
0
50
100
150
200
250


6 7 6 7 6 7 6 7 6
4 5 4 5 4 5 4 5 4
2 3 2 3 2 3 2 3 2
0 1 0 1 0 1 0 1 0
Page 2 Cells
10^4
10^3
10^2
10^1
10^0
0 50 100 150 200 250
Voltage


6 7 6 7 6 7 6 7 6
4 5 4 5 4 5 4 5 4
2 3 2 3 2 3 2 3 2
0 1 0 1 0 1 0 1 0

Page 2 Cells
10^4
10^3
10^2
10^1
10^0
0
50
100
150
200
250
Voltage

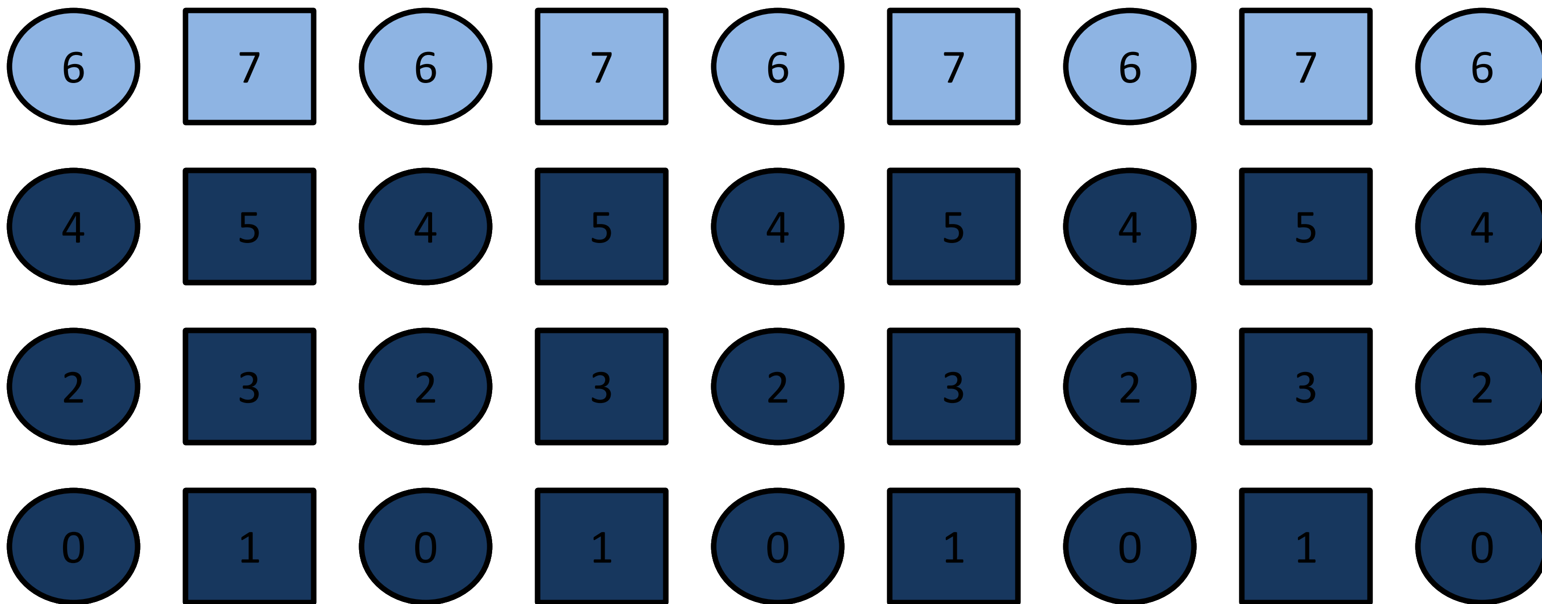


Page 2 Cells

10^4
10^3
10^2
10^1
10^0

0 50 100 150 200 250

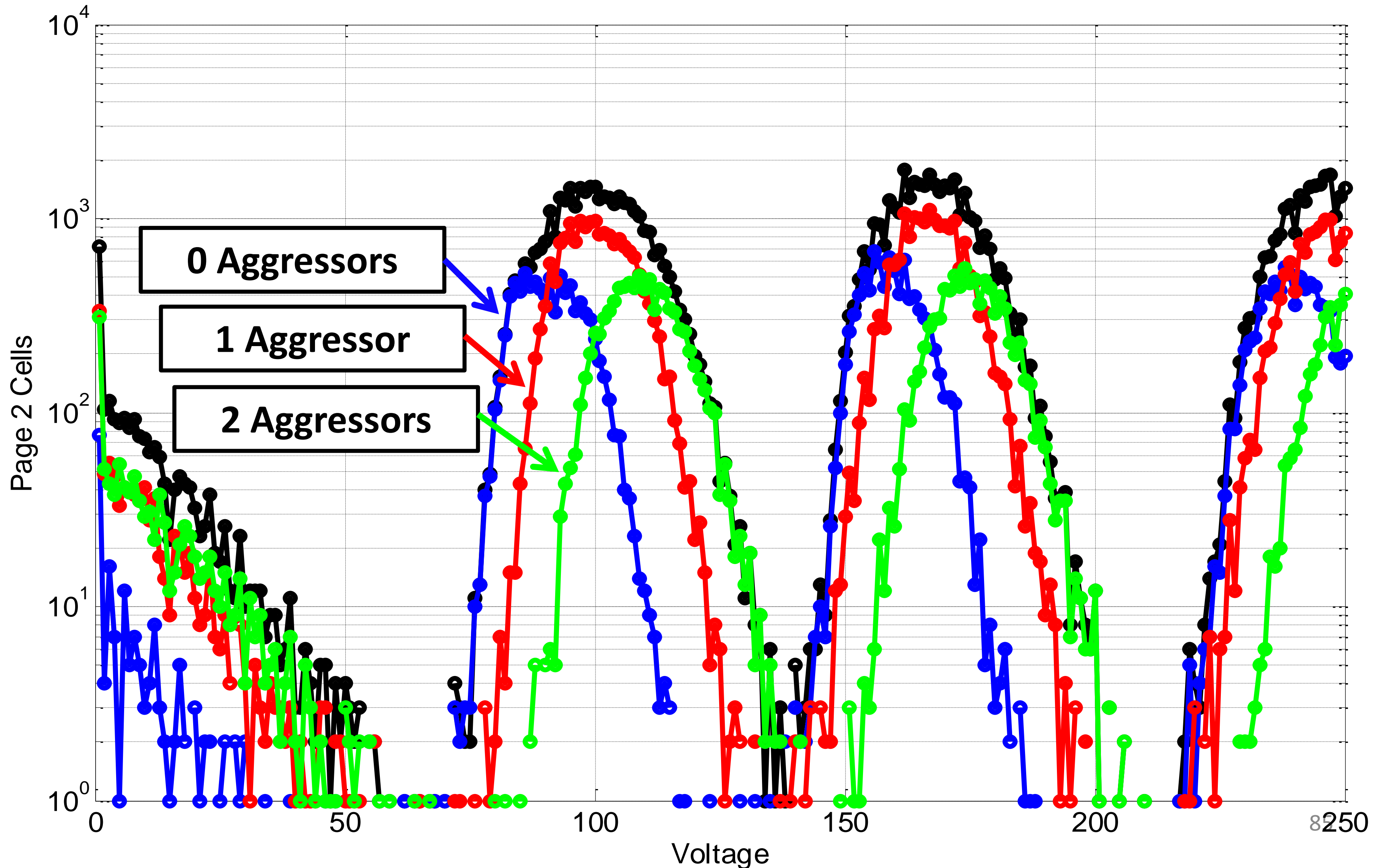
Voltage



6 7 6 7 6 7 6 7 6
4 5 4 5 4 5 4 5 4
2 3 2 3 2 3 2 3 2
0 1 0 1 0 1 0 1 0

Page 2 Cells
10^4
10^3
10^2
10^1
10^0
0
50
100
150
200
250
Voltage

Final Distributions (Cond. on Pg 3 &4)



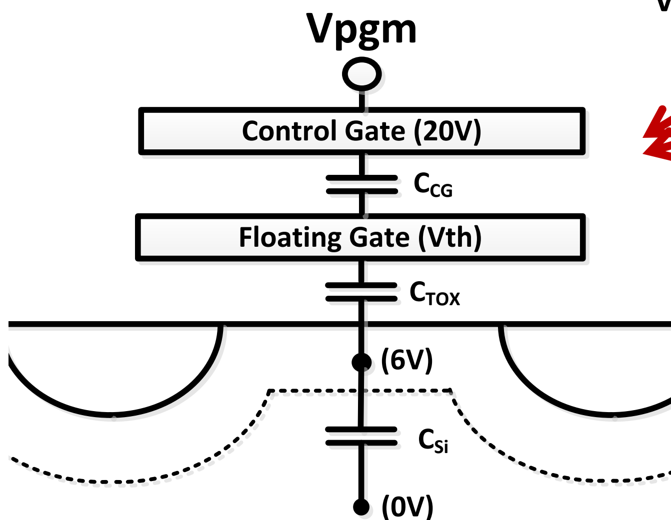
Mitigating Capacitive Coupling

- Many manufacturers are adopting all bitline (ABL) structure to minimize the number of aggressors.
- Capacitive coupling is inter-symbol interference (ISI), i.e. largely deterministic.
- Traditional methods for handling ISI
  - Write-precompensation?
  - Signal processing methods (ISI cancellation)?

PROGRAM DISTURB

Program Disturb

- After each cell reaches its PV level, it is inhibited (as shown).
- An inhibited cell has its channel voltage raised, thus reducing the voltage difference to its control gate.

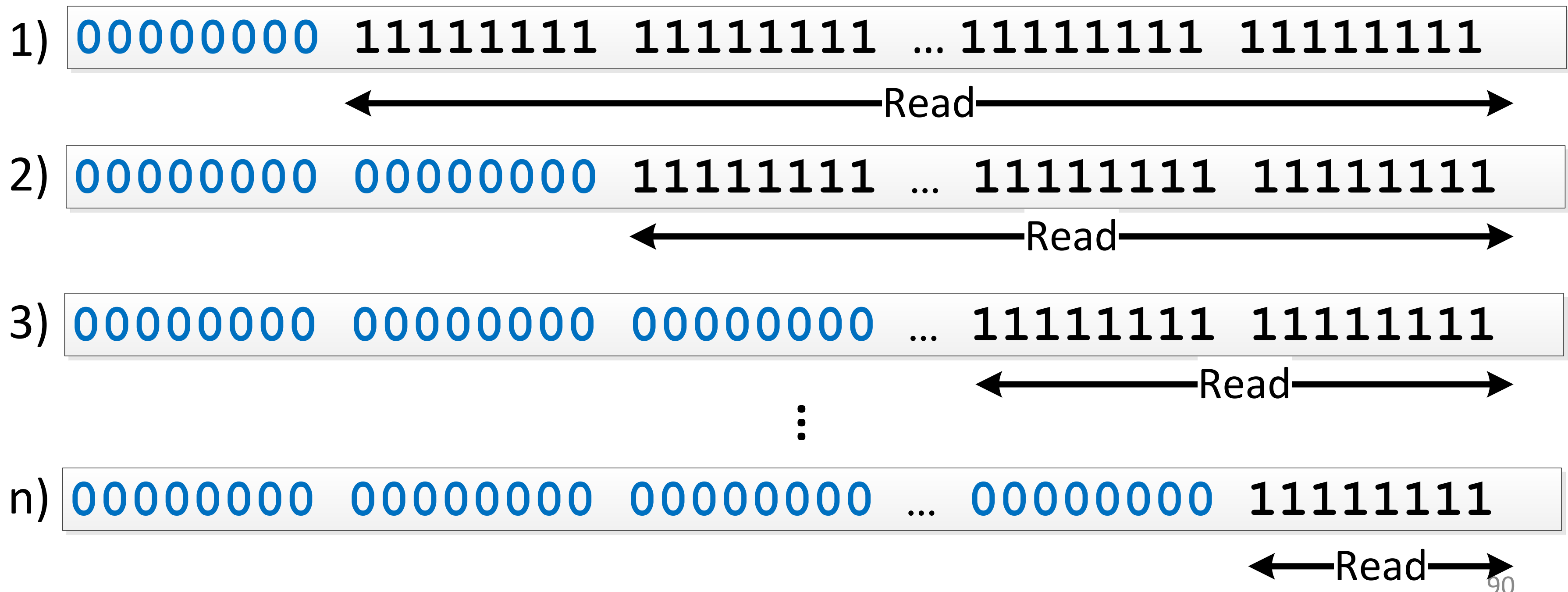


Program Disturb

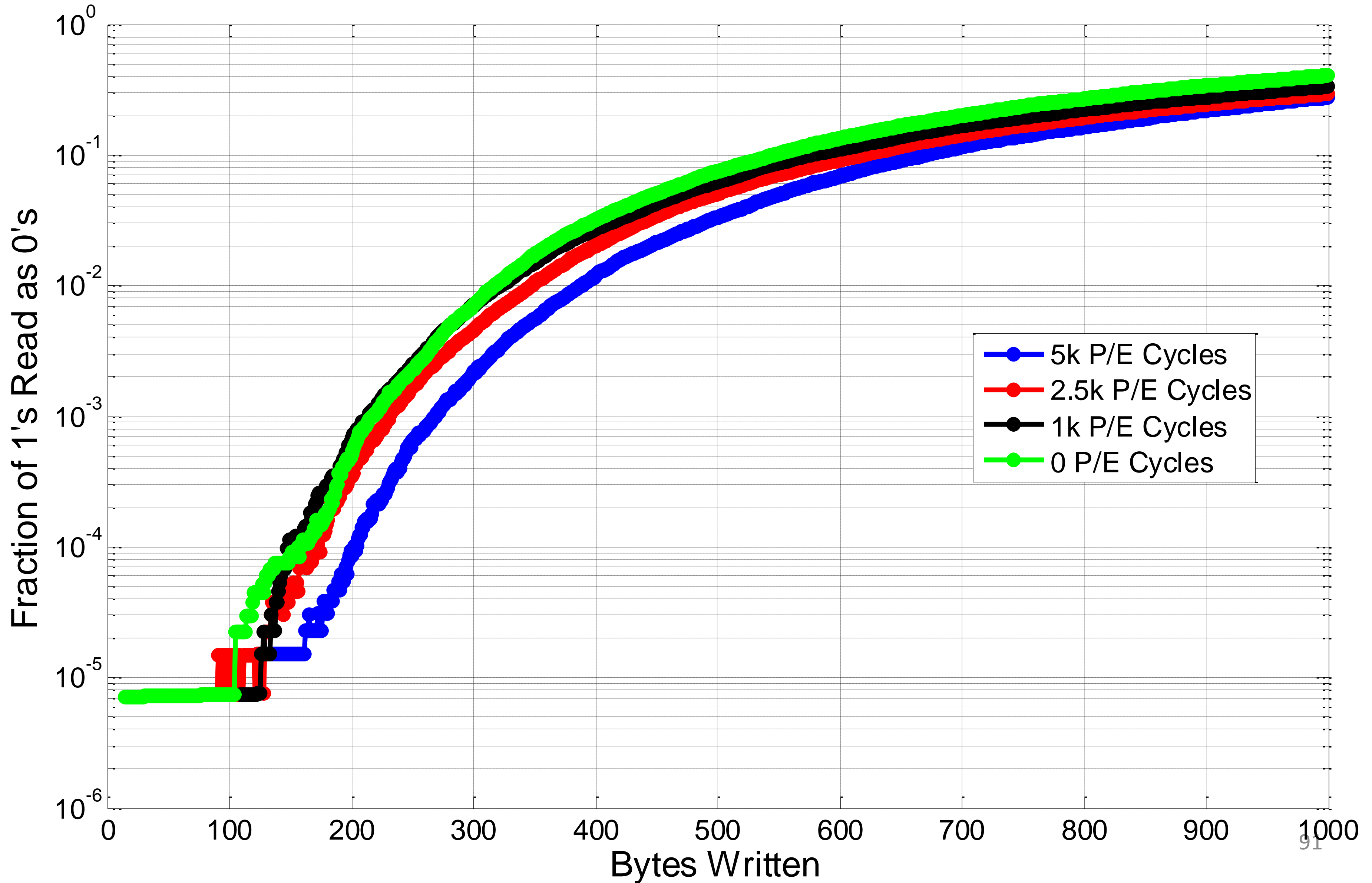
- Although the inhibit process acts to reduce the electric field in the tunneling oxide, it does not eliminate it.
- Some excess charge will be transferred to the floating-gate of inhibited cells.
- This is most severe for cells in the erased level since they are inhibited throughout the write process.
  - Receive the most write-pulses after being inhibited.

Program Disturb (Experiment)

- Continually re-write a single LSB page bringing successive bytes to the “0” level, i.e.
- Remaining bytes read with Vread = 0.



Experimental Results



Program Disturb (Experiment)

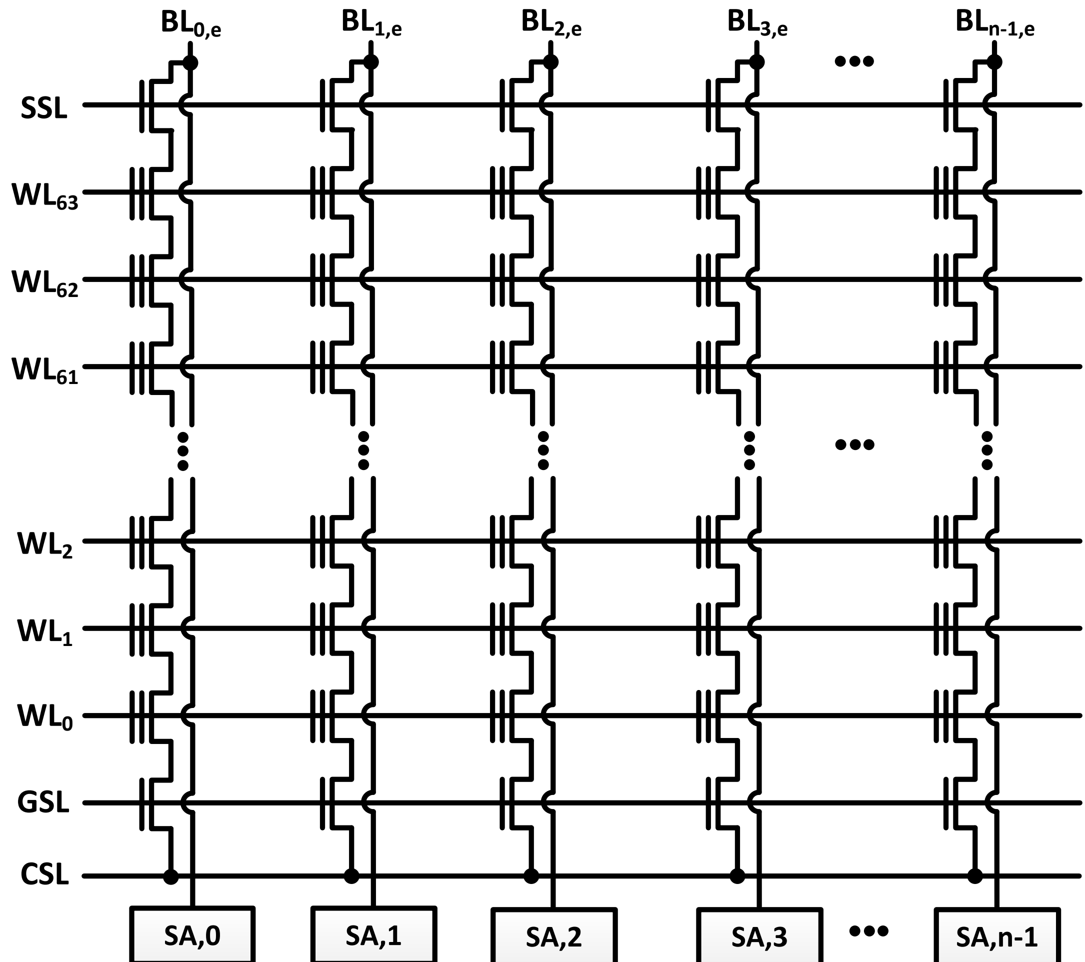
- All P/E cycles are affected by program-disturb at similar rates (lower P/E is affected slightly more).
  a) Higher P/E cycled pages require less pulses to program (minimizing program disturb).
  b) Higher P/E cycled pages will more readily take on excess charge (maximizing program disturb).
- The effect of a) outweighs the effect of b).

Mitigating Program Disturb

- Changing the write strategy by increasing Vpass will boost the channel voltage, lessening its effects.
  - Increasing Vpass increases pass disturb.
- The effects of program disturb are primarily on the lowest level.
  - Can be taken into account when data is processed.
- Pages are only programmed a single time.

READING FROM THE NAND ARRAY

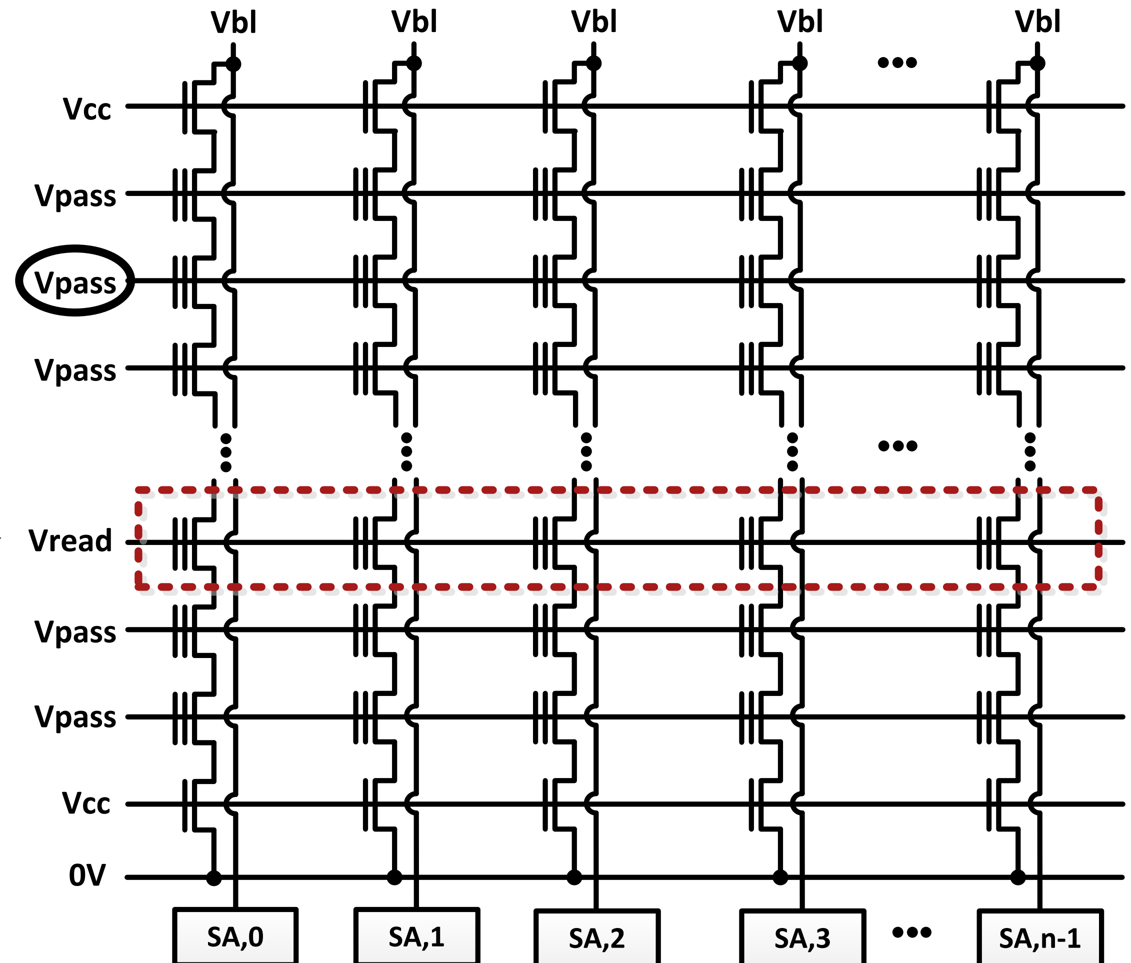
Reading from the NAND Array



Read Process (WL_2)

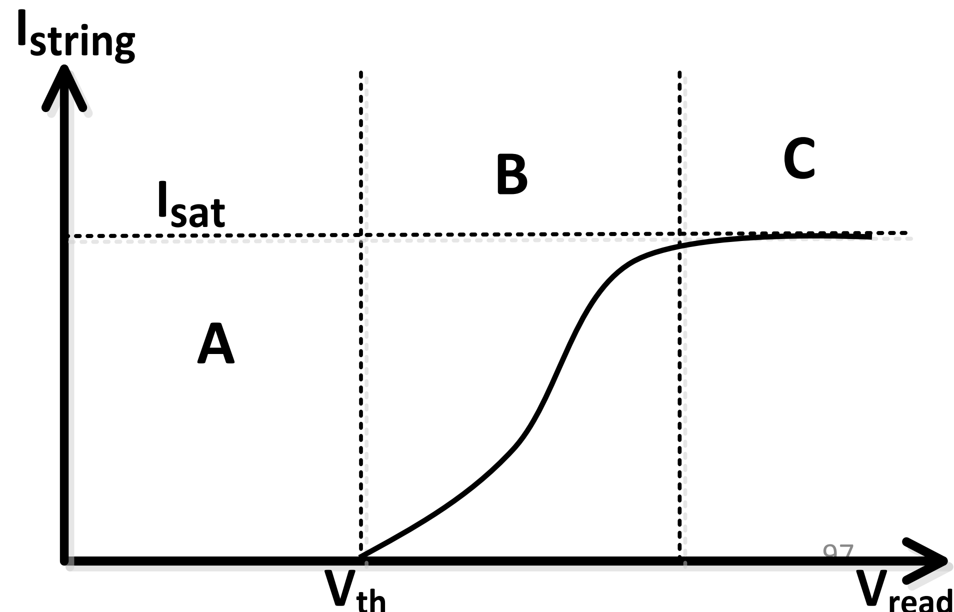
Vpass ~4-5V > Vth_{max}

Isolated Wordline



Read Process

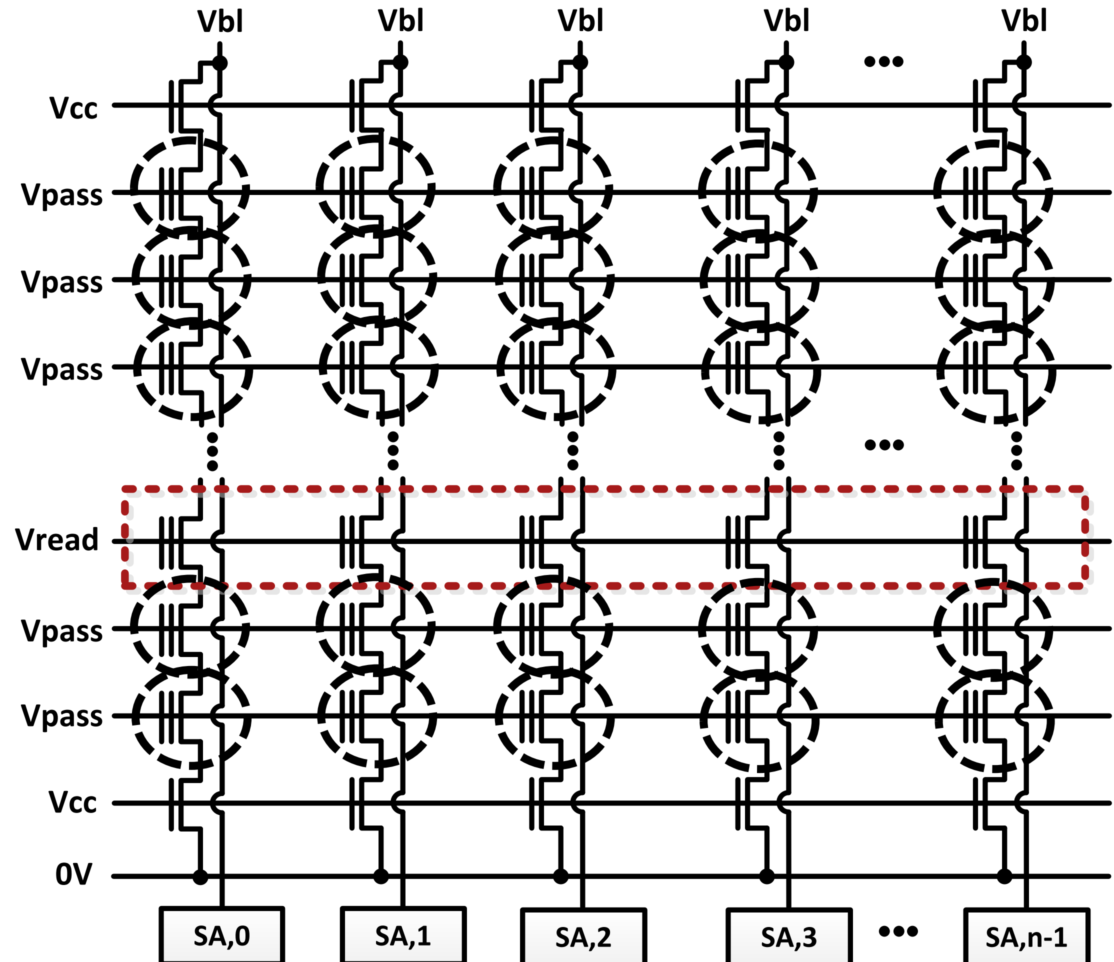
- Bitline (I_{string}) current operates in 1 of 3 regions:
  - A) Addressed cell not conductive (Vth < Vread)
  - B) Vread makes addressed cell conductive (Vth > Vread)
  - C) Cell is completely on, series resistance of pass transistors saturates current (Vth >> Vread).
- In practice, NAND currents of ~10nA must be read.
- Capacitors used to integrate current to make sensing possible.



READ DISTURB

Read Disturb

- Unselected wordlines have Vpass applied to CG.
- Selected wordline has Vread applied to CG.
- Applied voltages cause unintended tunneling of charge.
- Since Vread < Vpass, unselected wordlines are most severely affected.



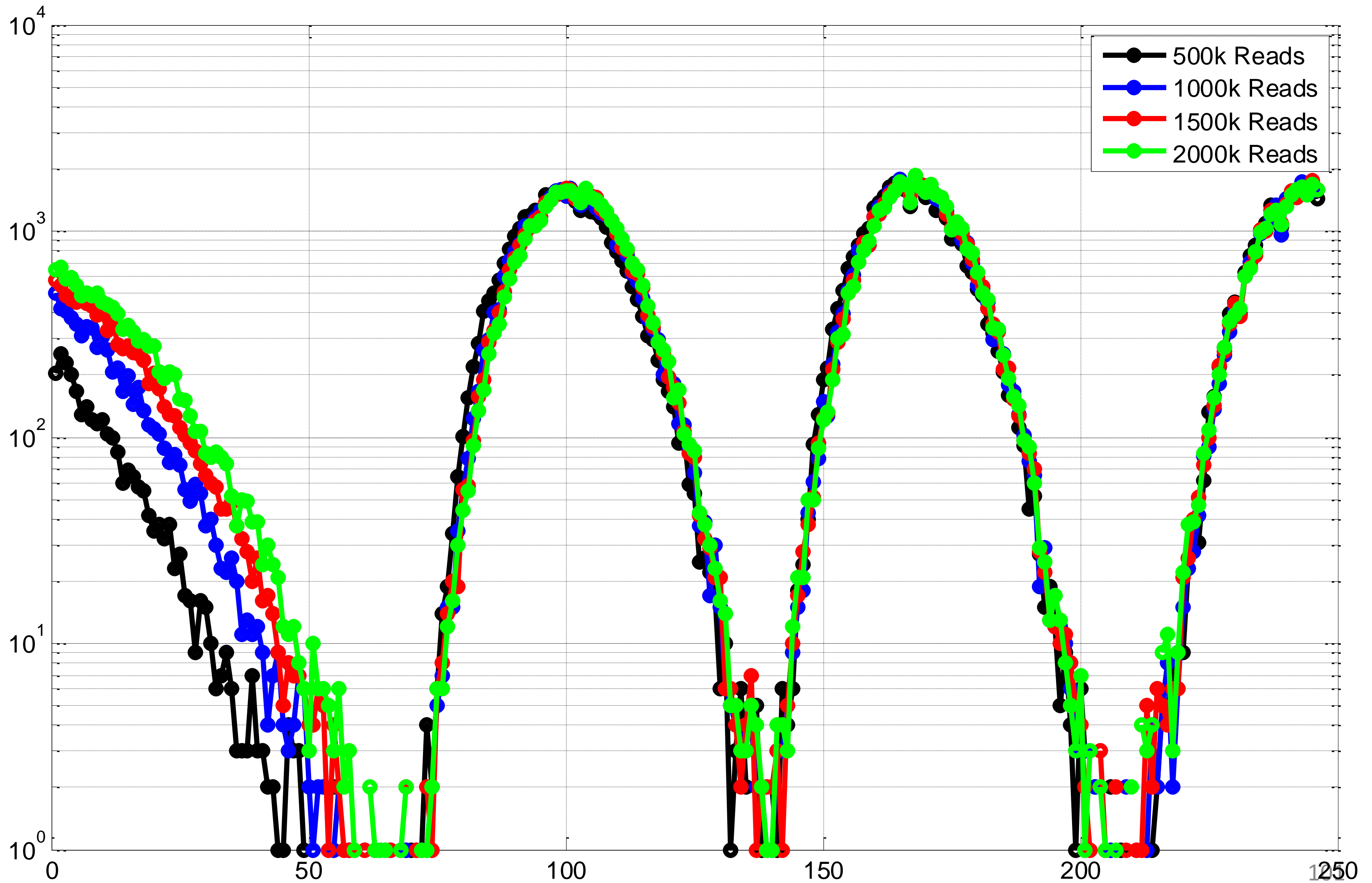
Read Disturb

- The total stress time for wordline j depends on the total reads to ***other*** wordlines.

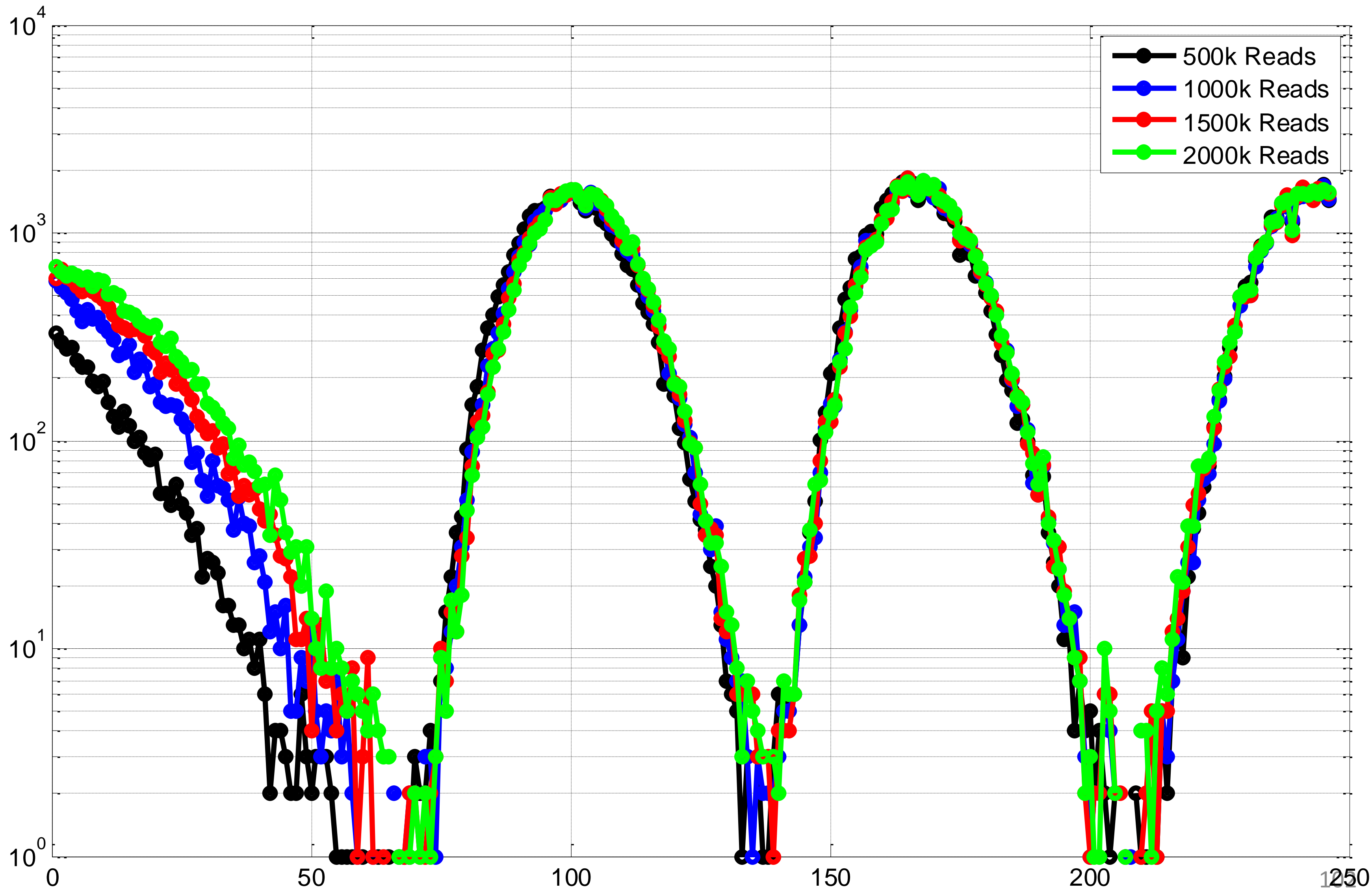
$$MaxST(j) = T_{read} \times \sum_{i=0, i \neq j}^{63} N_{read}(i)$$

- Wordline read the fewest (most) times in a block incurs the most (least) read-disturb.
- Incidental tunneling is endurance dependent.
- Read disturb most severely affects lowest levels.

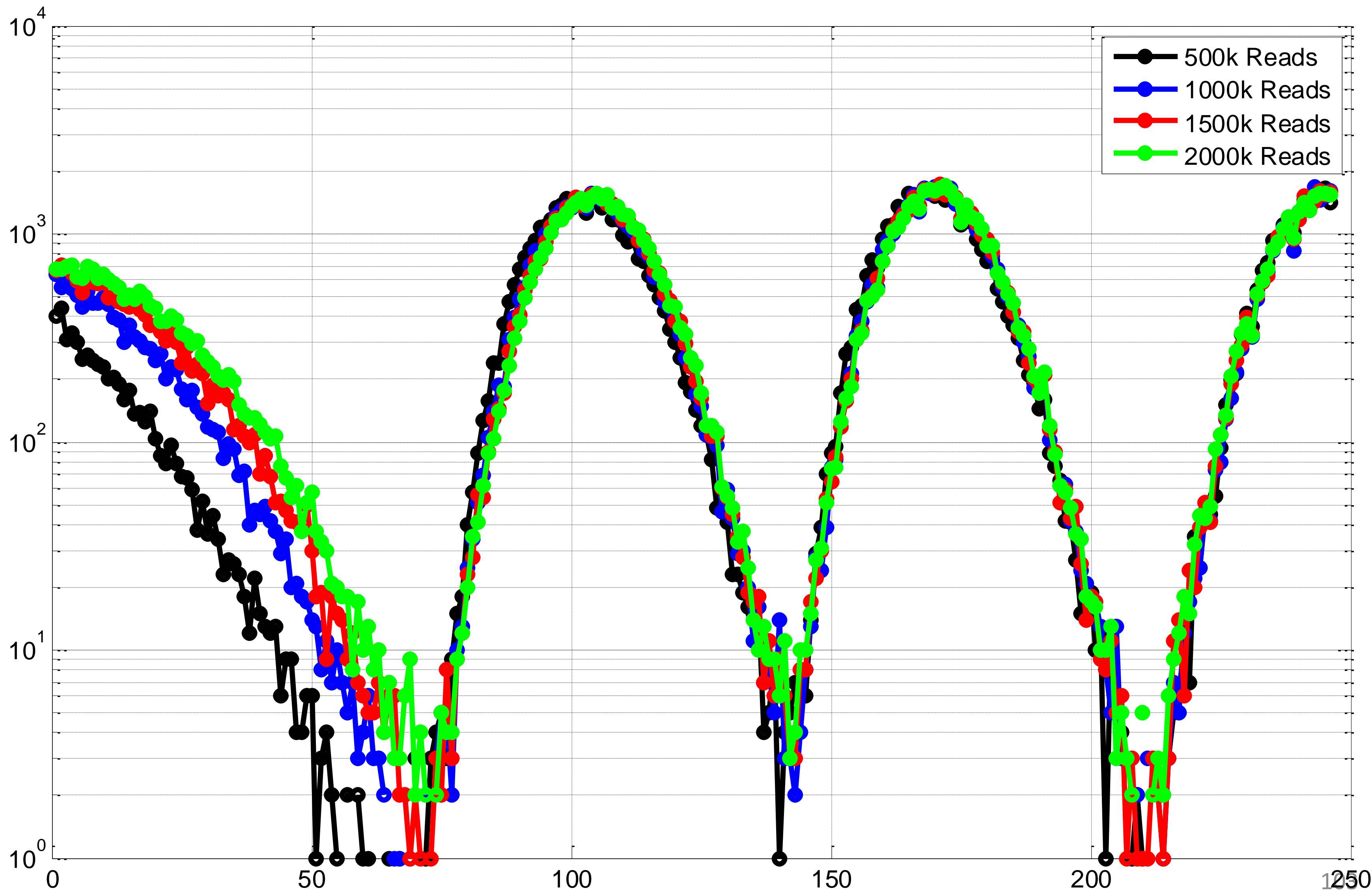
0 P/E



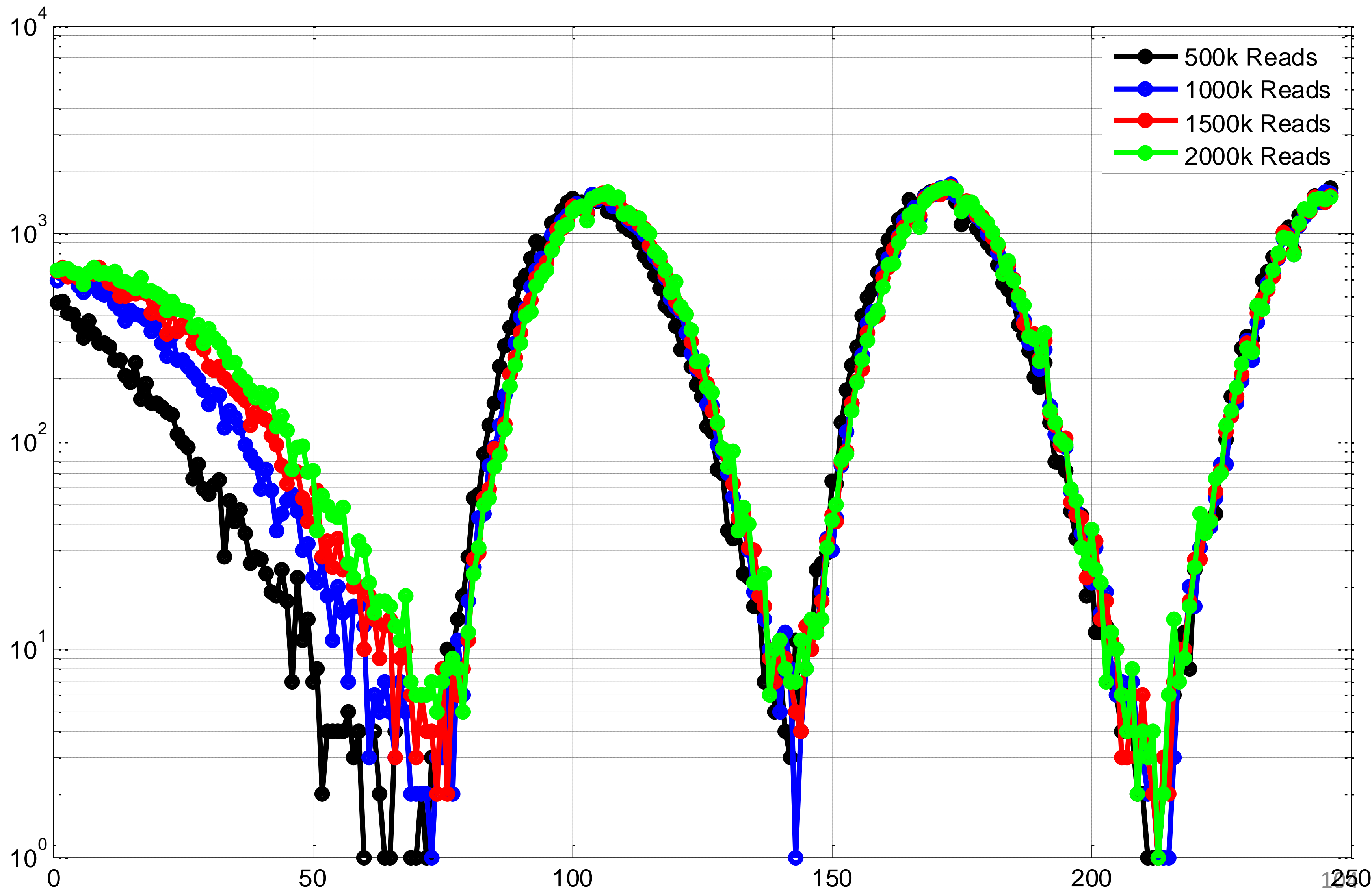
1k P/E



2.5k P/E



5k P/E



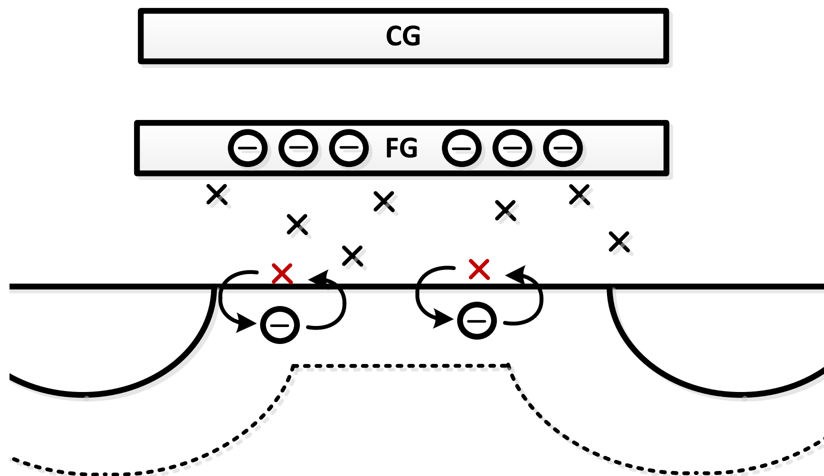
Mitigating Read Disturb

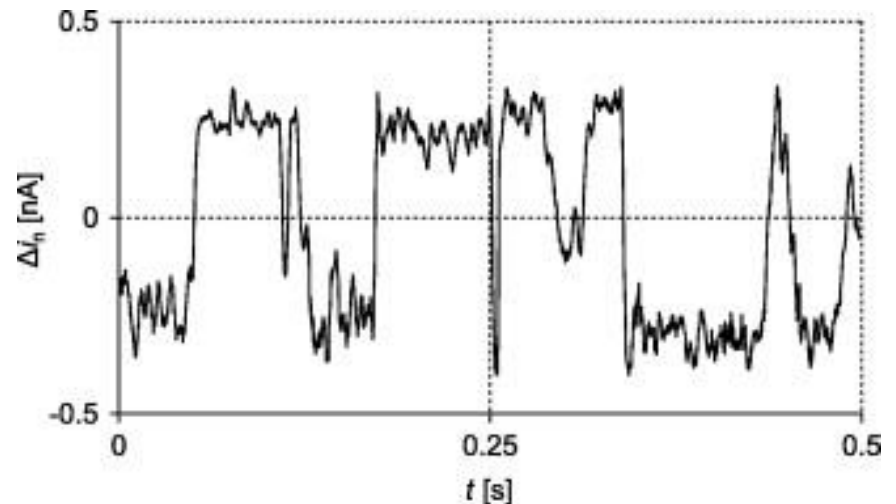
- In general, read-disturb is difficult to detect during normal read operations.
  - Page being read is minimally disturbed.
  - Since only lowest level affected, only MSB will be affected.
- Continually reading the same page (LBA) will not show signs of read-disturb.
- Read counters can be cumbersome to implement (firmware overhead) and expensive to store.

READ NOISE

Read Noise (RTN)

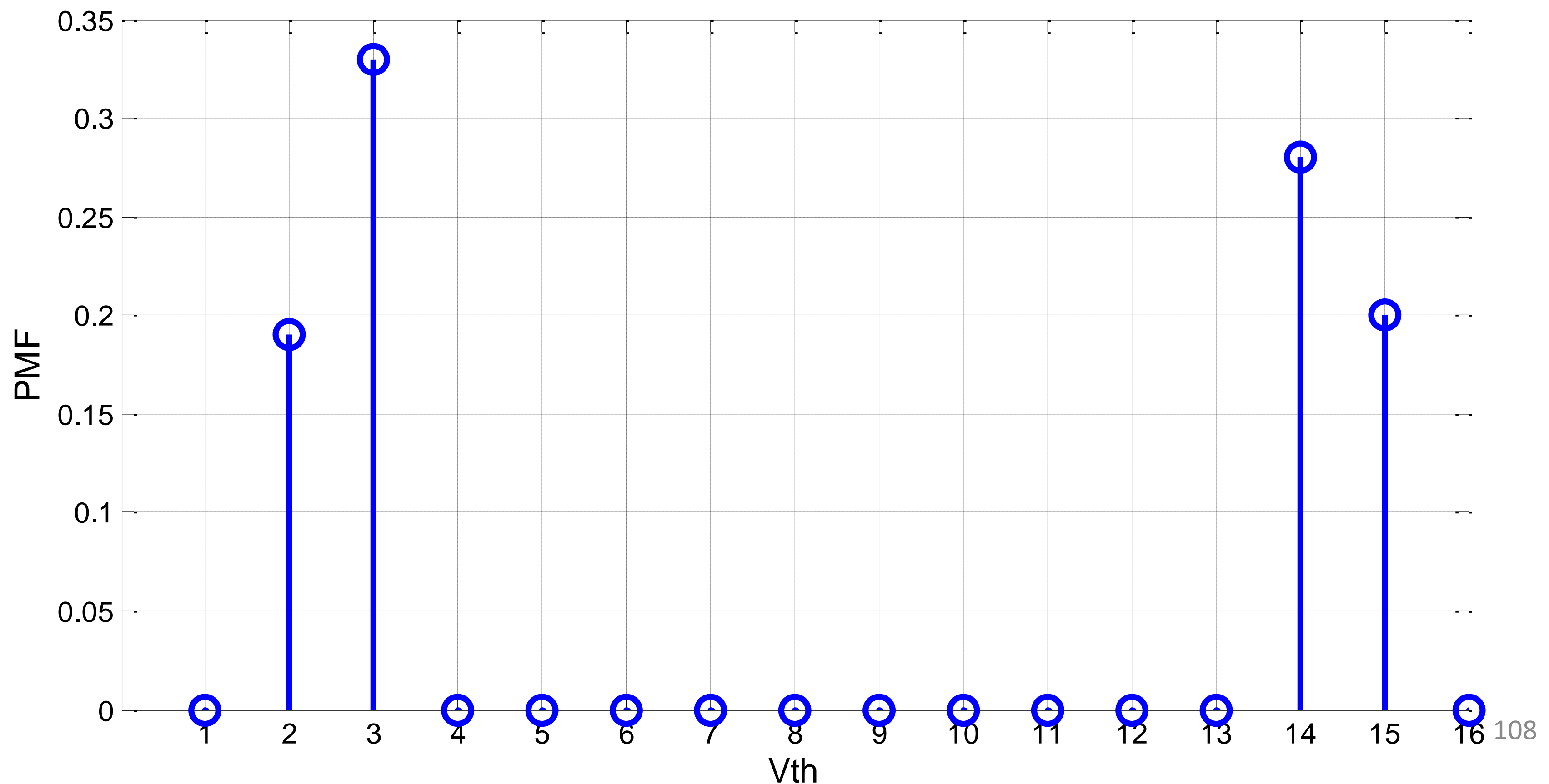
- Traps that reside in the tunneling oxide near the channel can easily gain/lose electrons.
- Cell voltage fluctuates in discrete states as this happens.





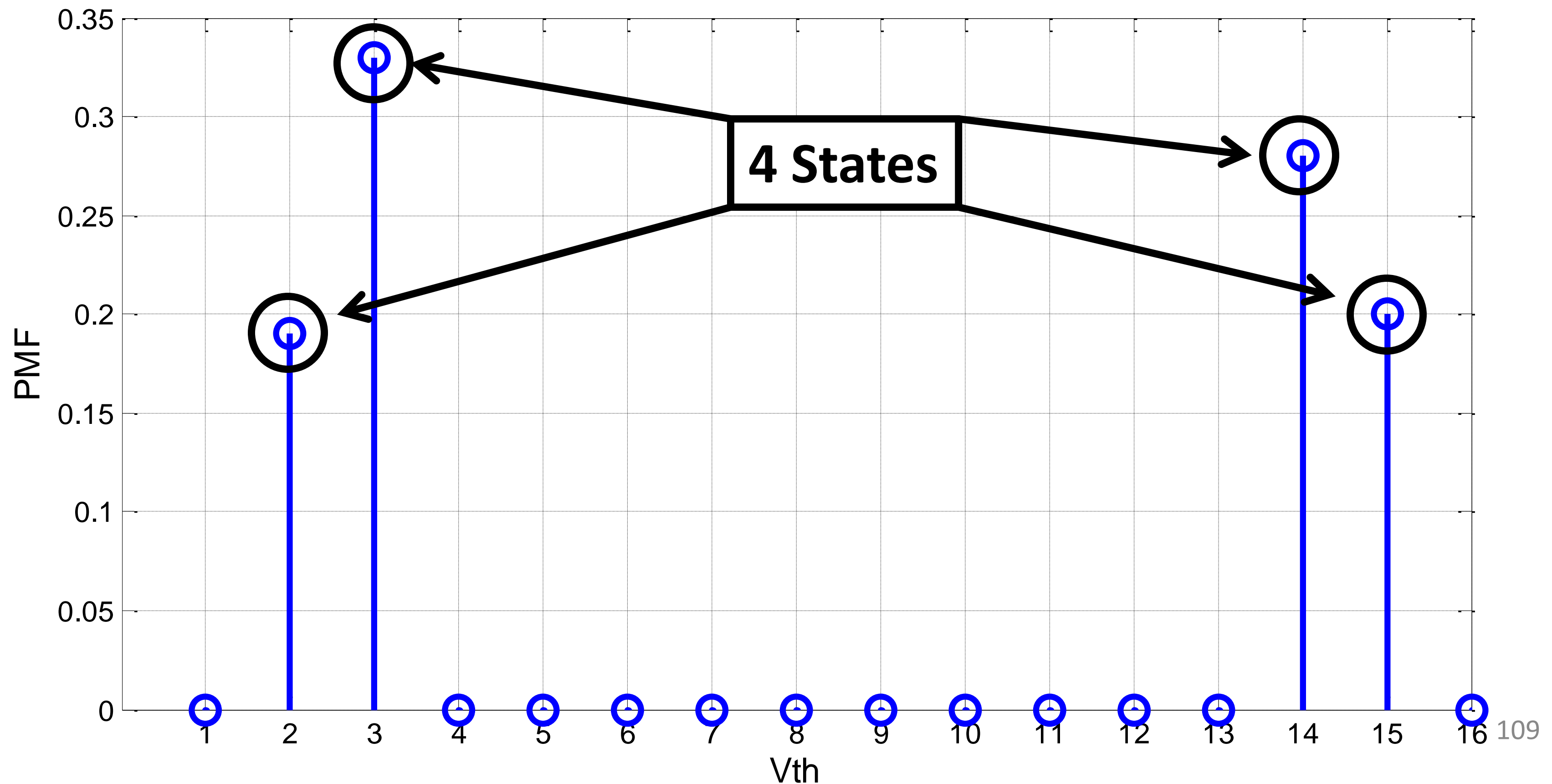
Read Noise

- To analyze the read-noise in 2ynm NAND, a page was read 100x at each read-threshold.



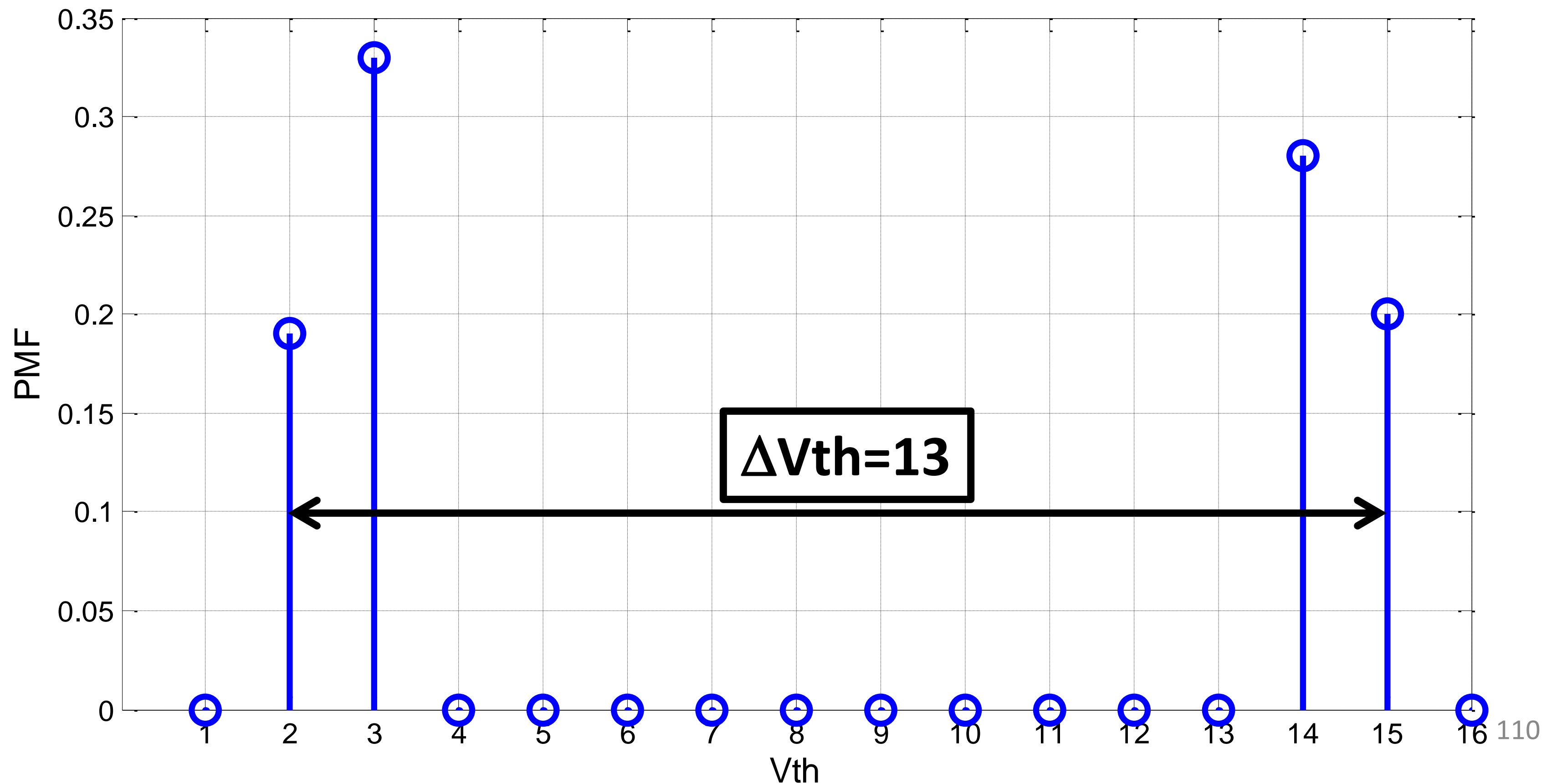
Read Noise

- To analyze the read-noise in 2ynm NAND, a page was read 100x at each read-threshold.



Read Noise

- To analyze the read-noise in 2ynm NAND, a page was read 100x at each read-threshold.



Read Noise

Mitigating Read Noise

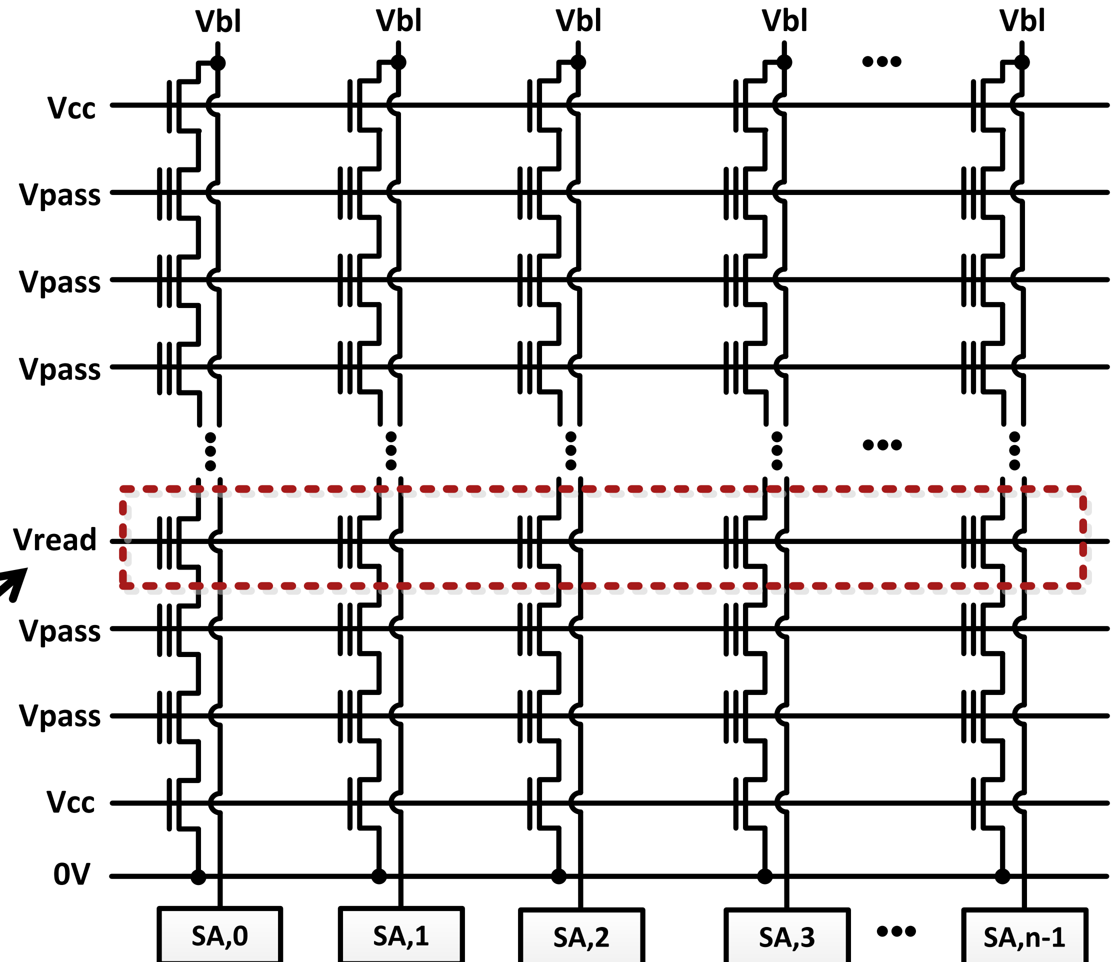
- Signal processing techniques such as read-averaging can be used to mitigate its effects.
  - Particularly since multiple reads (i.e. read-shifts) are often used to recover data.
- Useful for randomizing error locations.
  - This randomization helps to reduce error-floors for some coding methodologies (i.e. LDPC).

BACK PATTERN EFFECT

Back Pattern Effect

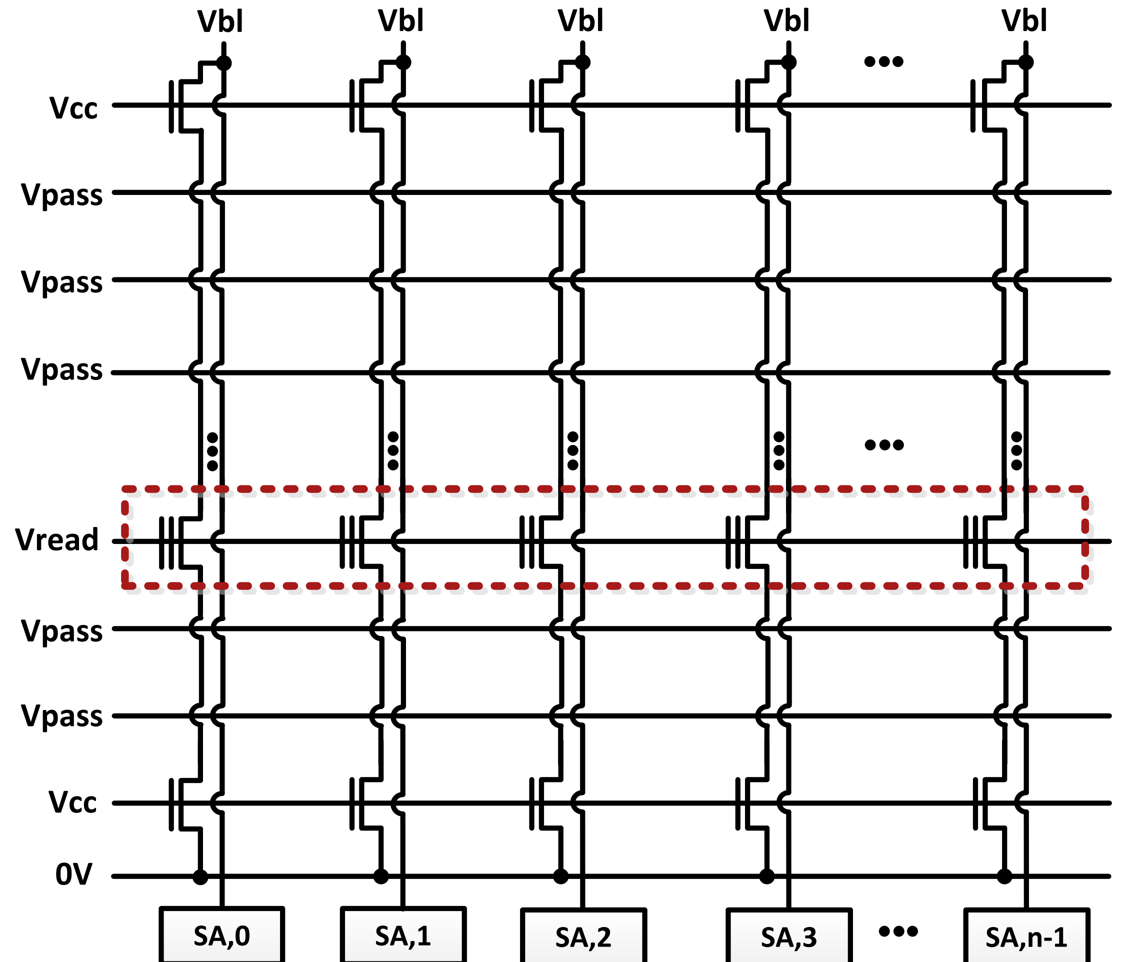
- During the read-process, Vpass is applied to all non-selected wordlines.
- Cells along these wordlines are (ideally) set to pass.

Selected Wordline (k)



Idealized Selection

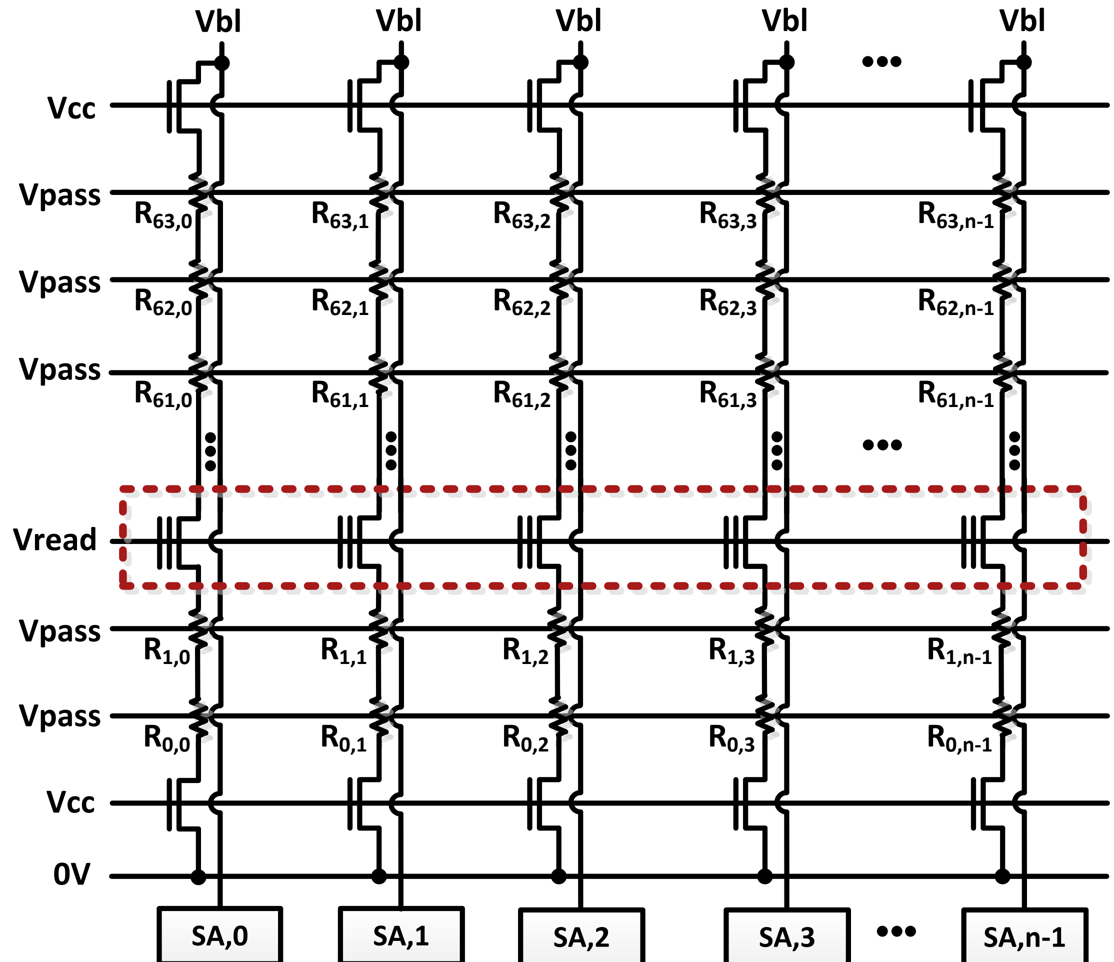
- In the ideal case, pass cells behave as short circuits.
- The string-current (I_{string}), thus, is only a function of the selected cell.
- Selected cells begins to conduct when Vread exceeds Vth.



Realistic Selection

- Each transistor has a resistance which is a function of its threshold voltage and control gate voltage, i.e.,

$$R_{i,j} \propto \frac{1}{\left(V_{TH}^{i,j} - V_{GS}^{i,j}\right)}$$



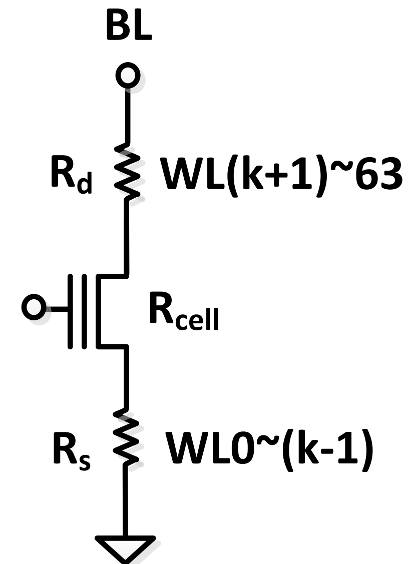
Back Pattern Effect

- Total resistance for bit-line j,

$$R^{B_j} \propto R\left(\frac{1}{\left(V_{TH}^{k,j} - V_{read}\right)}\right) + \sum_{i=0, i\neq k}^{64} R\left(\frac{1}{\left(V_{TH}^{i,j} - V_{pass}\right)}\right)$$

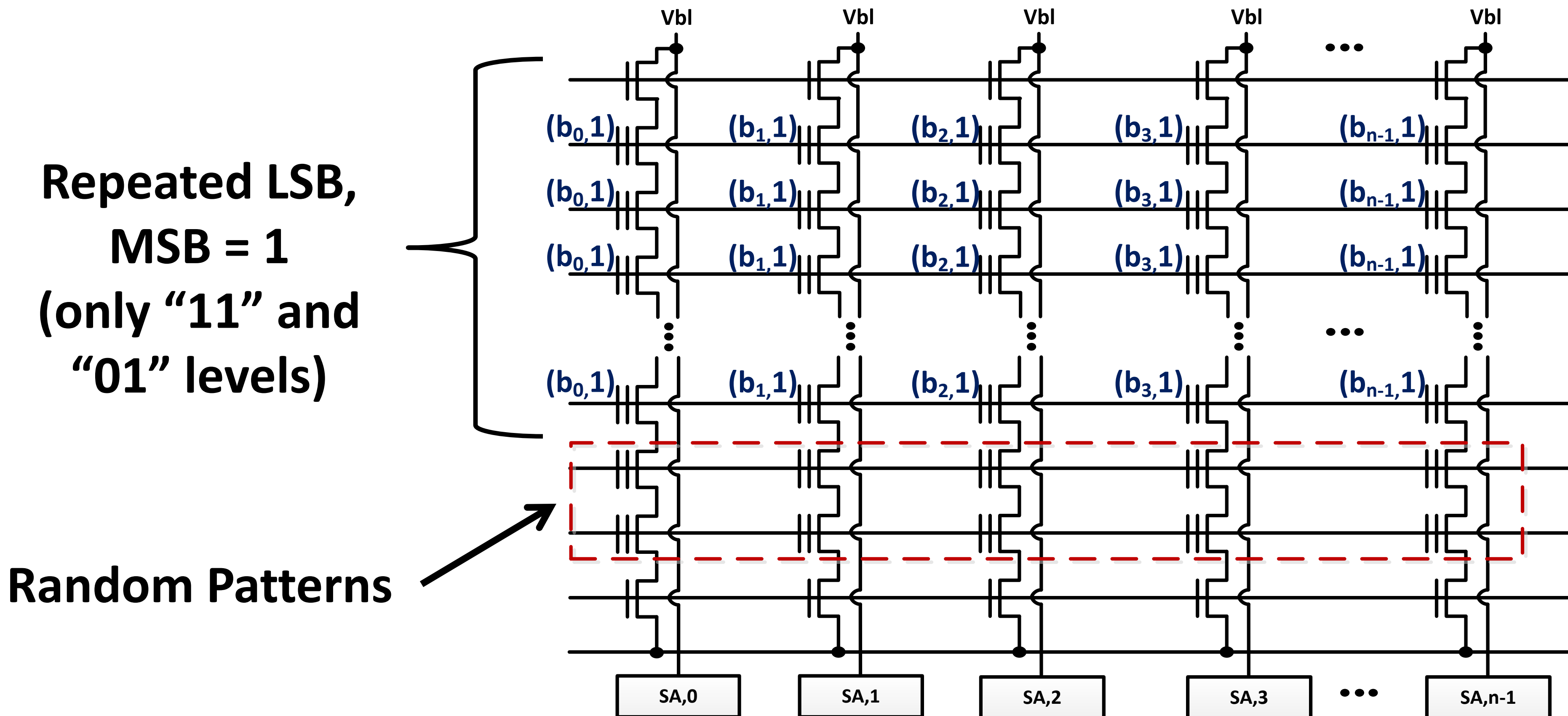
- Bit-line current depends on threshold voltages (i.e. data) of *every* cell the bit-string,

$$I_{B_j} = \frac{V_{bl}}{R^{B_j}}$$

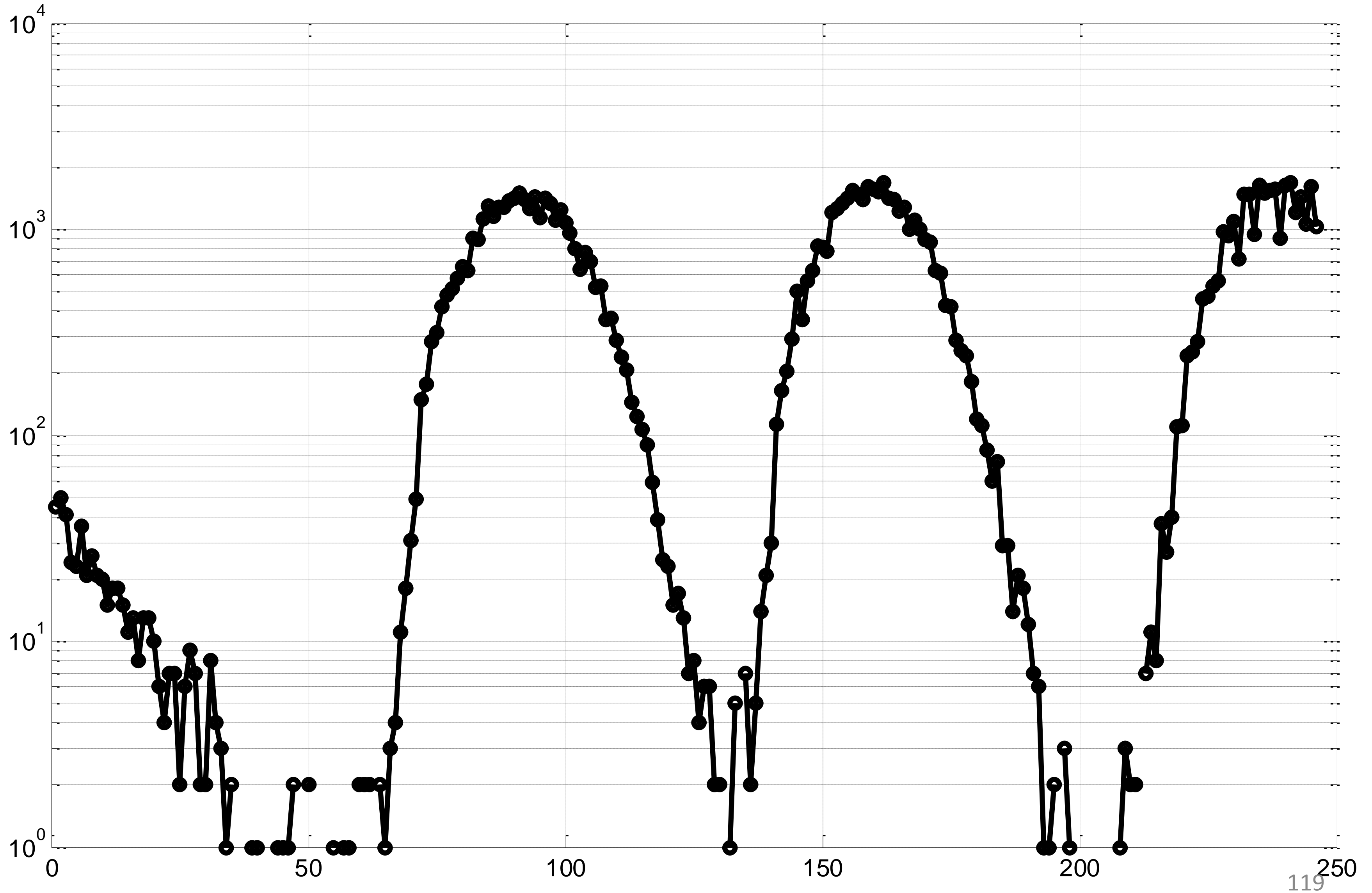


Back Pattern Effect

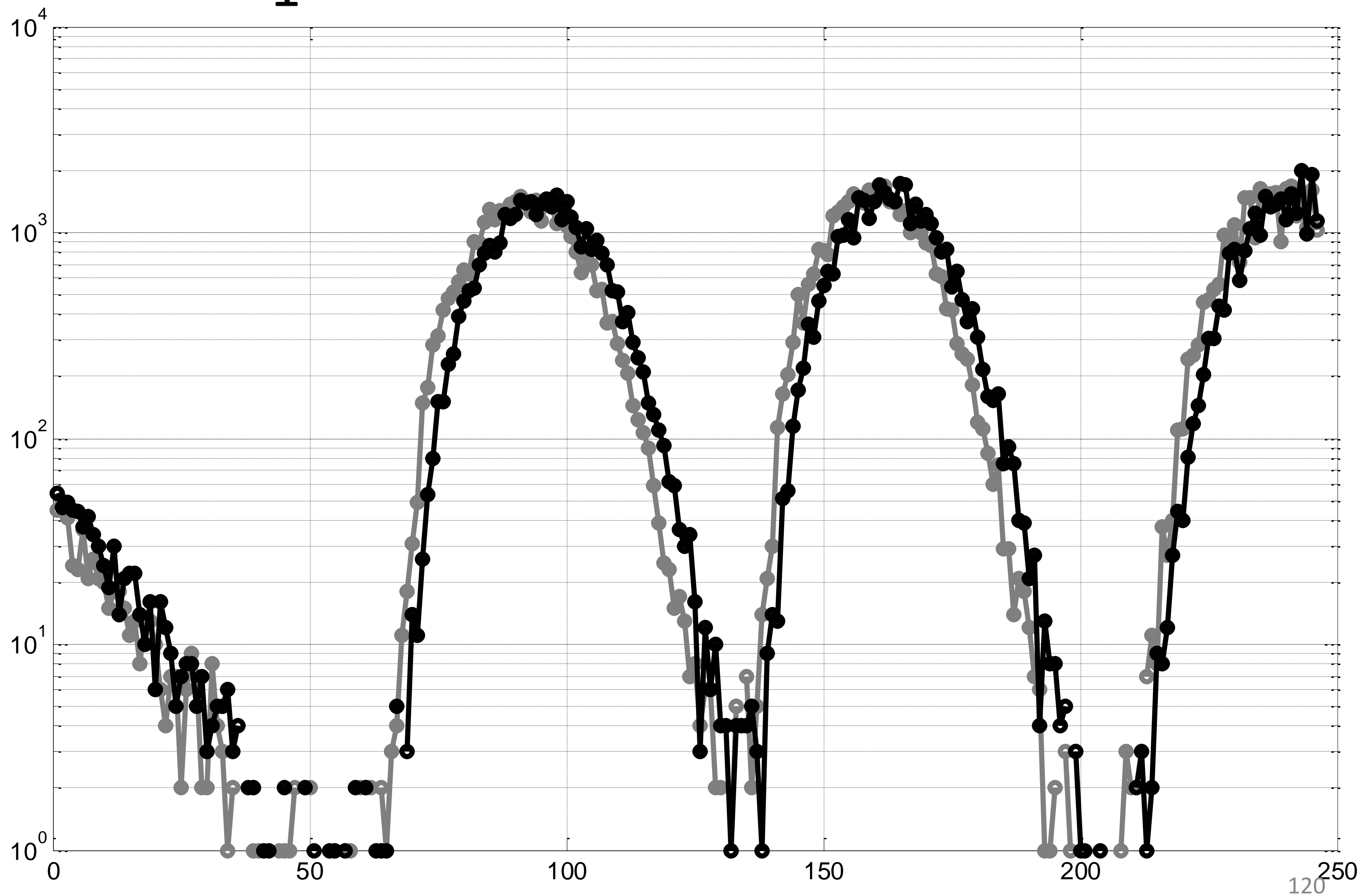
- To demonstrate, a block was written as follows



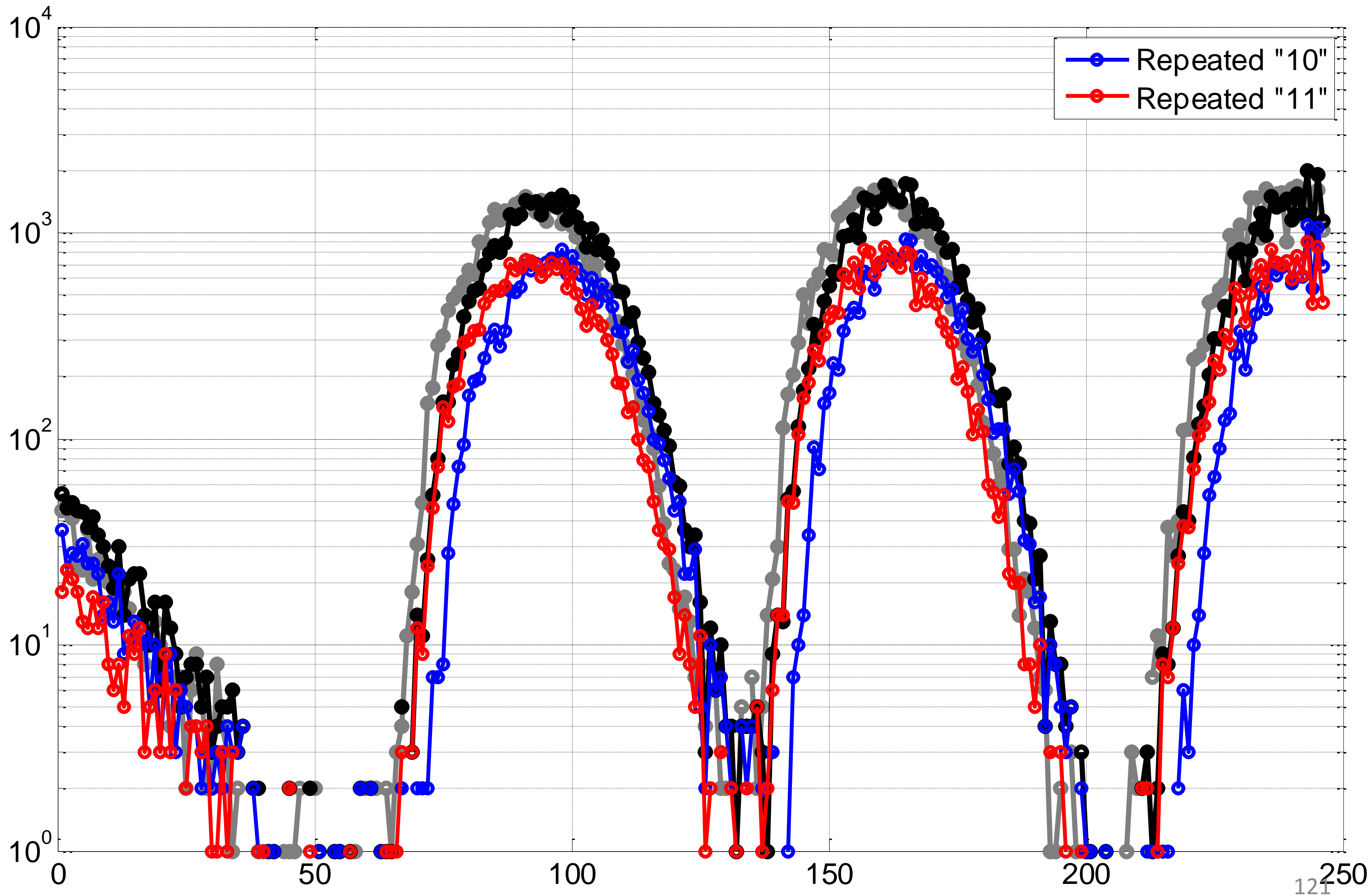
WL_1 Histogram (After 1st 2 wordlines)



WL_1 After Block Completion



WL_1 After Block Completion (Conditional)

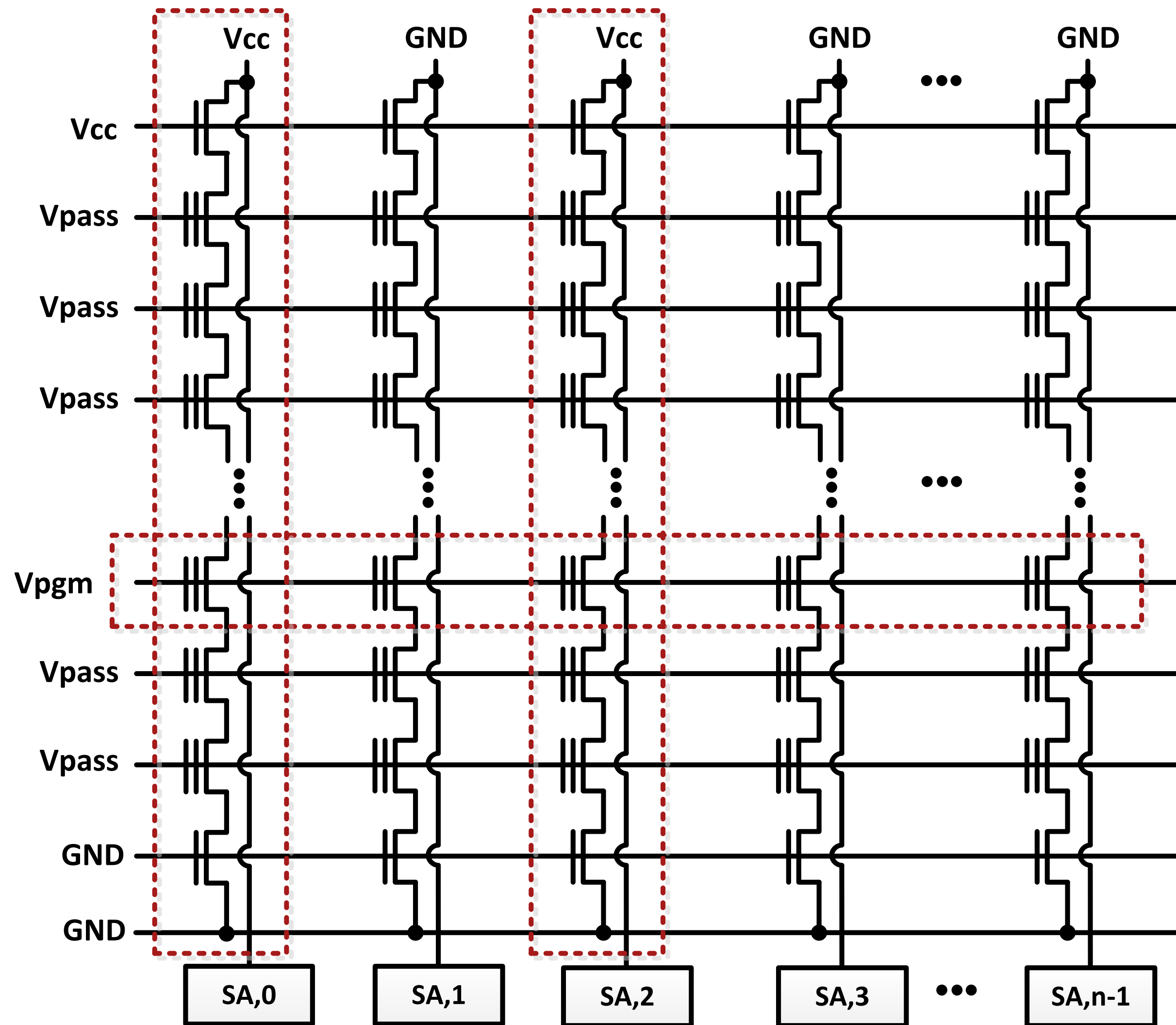


Mitigating Back Pattern Effect

- To avoid the read-back dependence of a page on the data written to the remainder of the block, the data must be properly randomized.
- Since user-data pattern may be repeated within a block, data scrambling must be used.
- Scrambler pattern must ensure sufficient randomization is achieved in all cases.

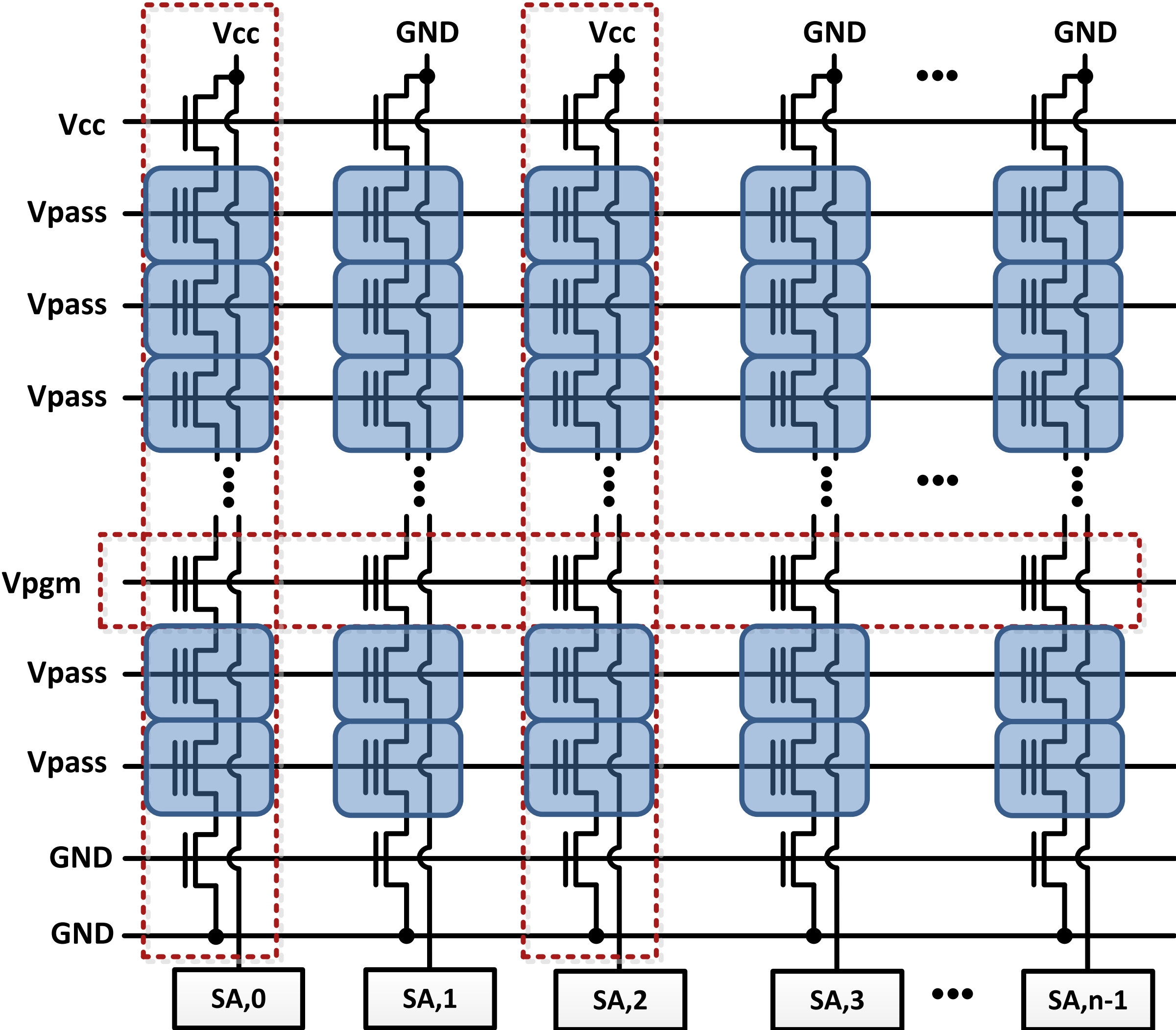
SUMMARY

Write Noise



Write Noise

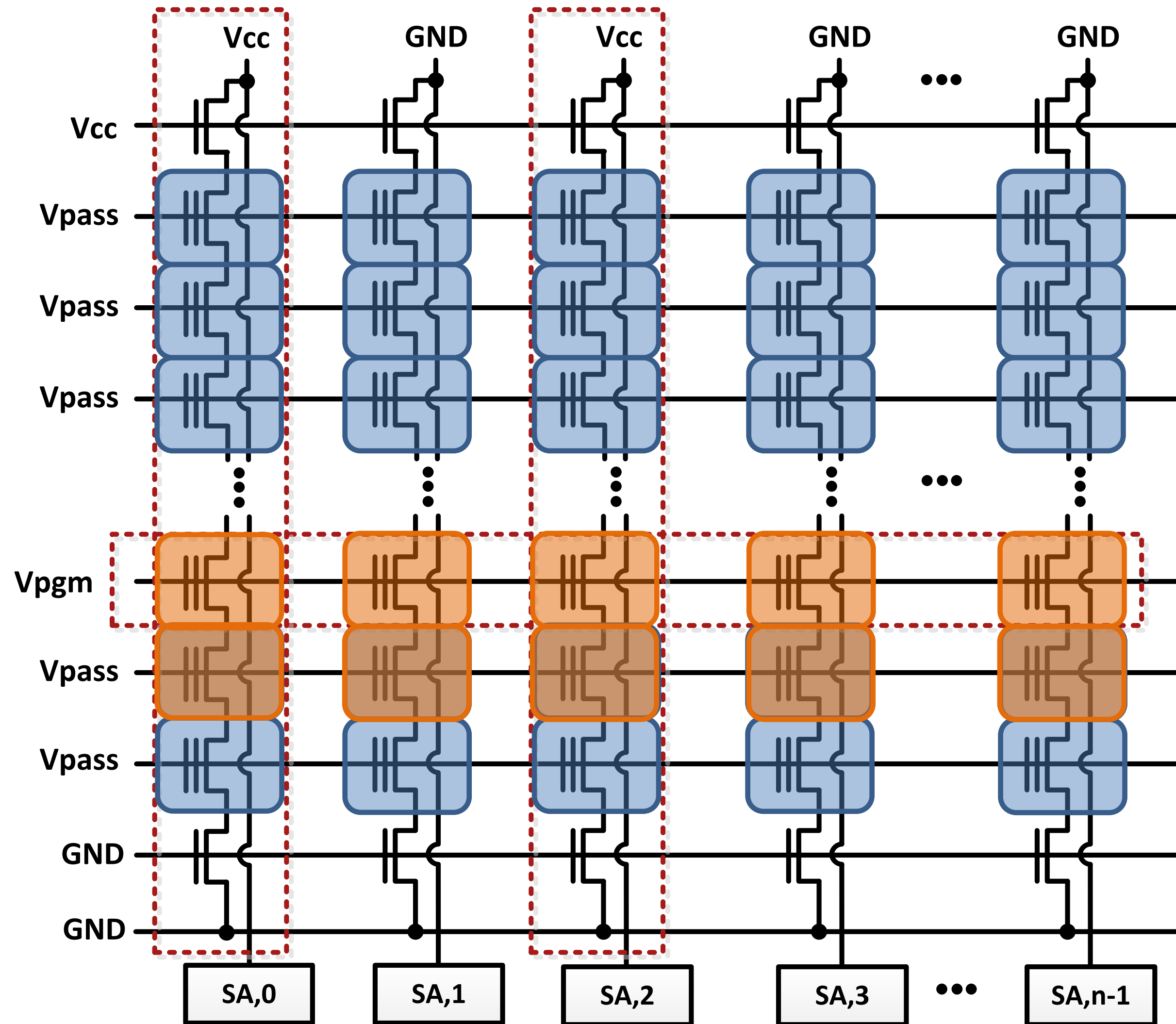
Pass Disturb



Write Noise

Pass Disturb

Capacitive Coupling

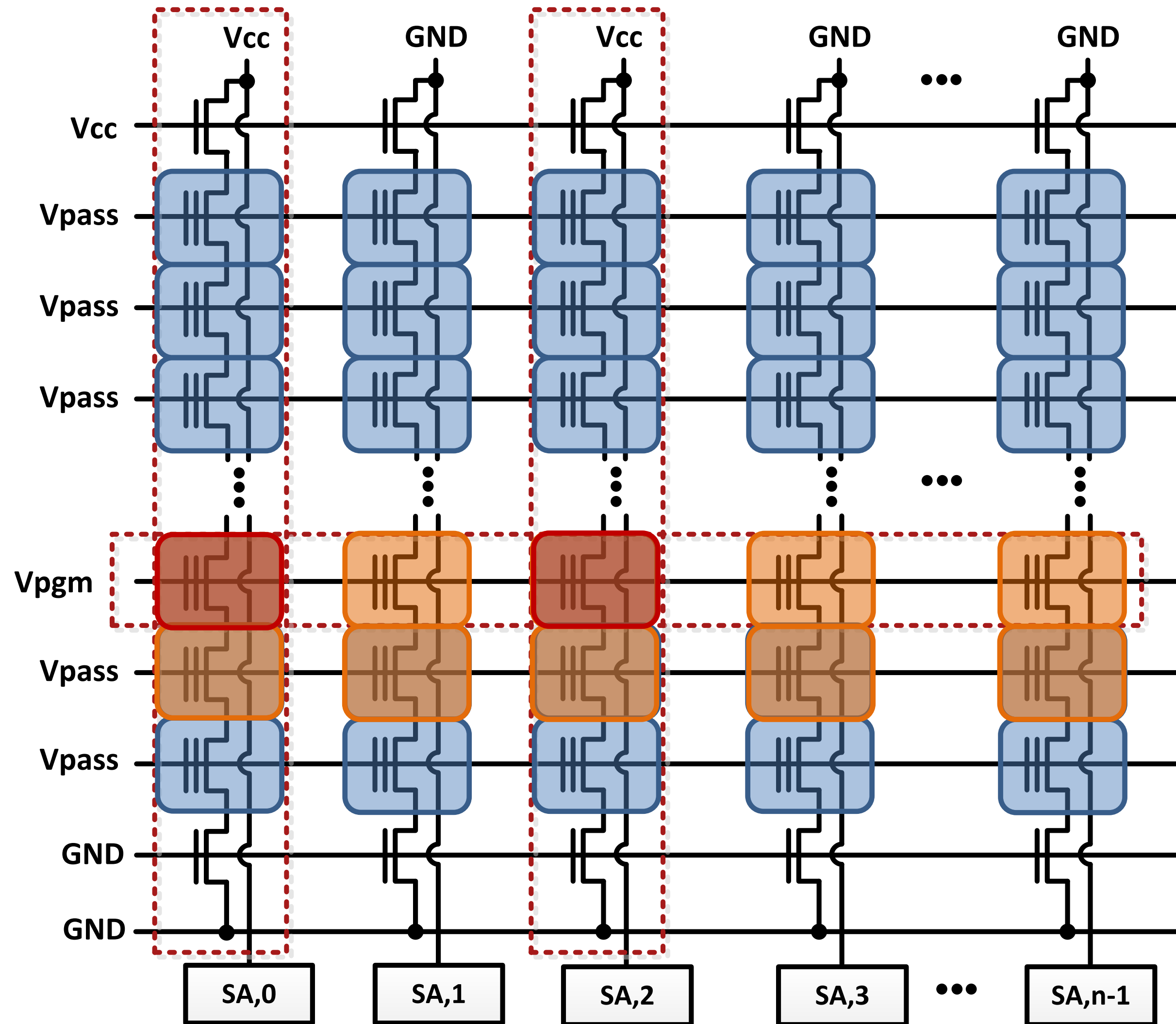


Write Noise

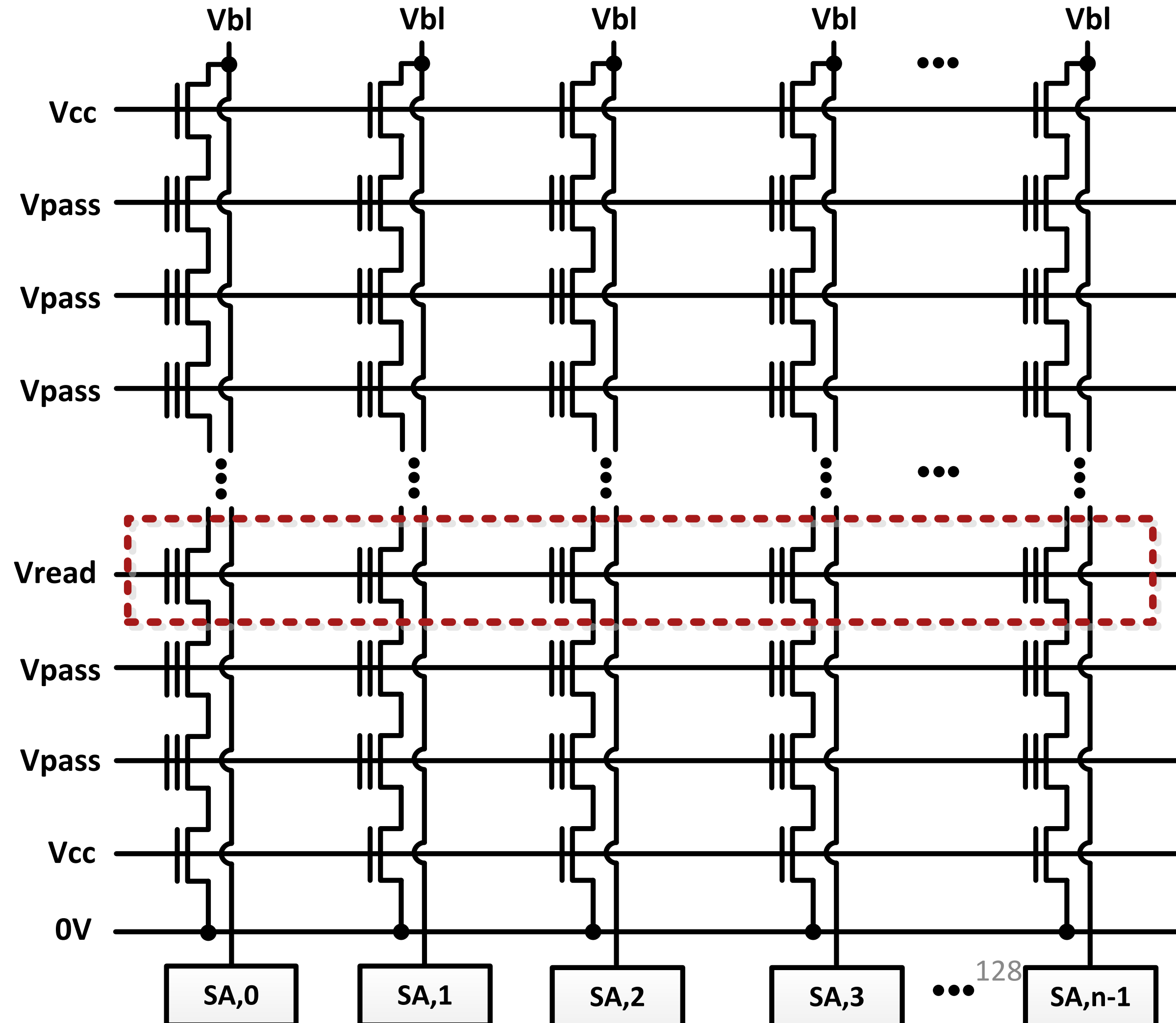
Pass Disturb

Capacitive Coupling

Program Disturb

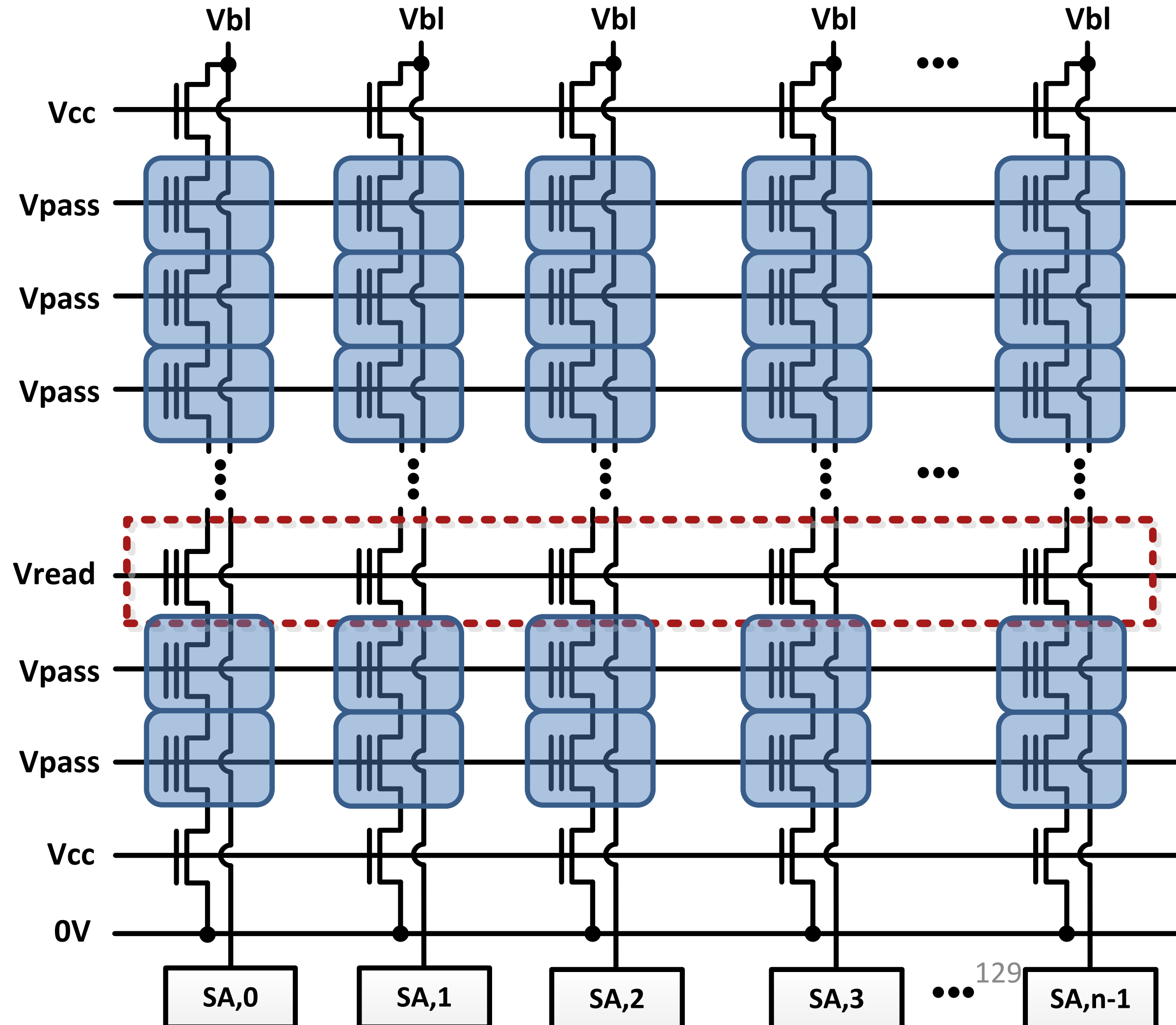


Read Noise



Read Noise

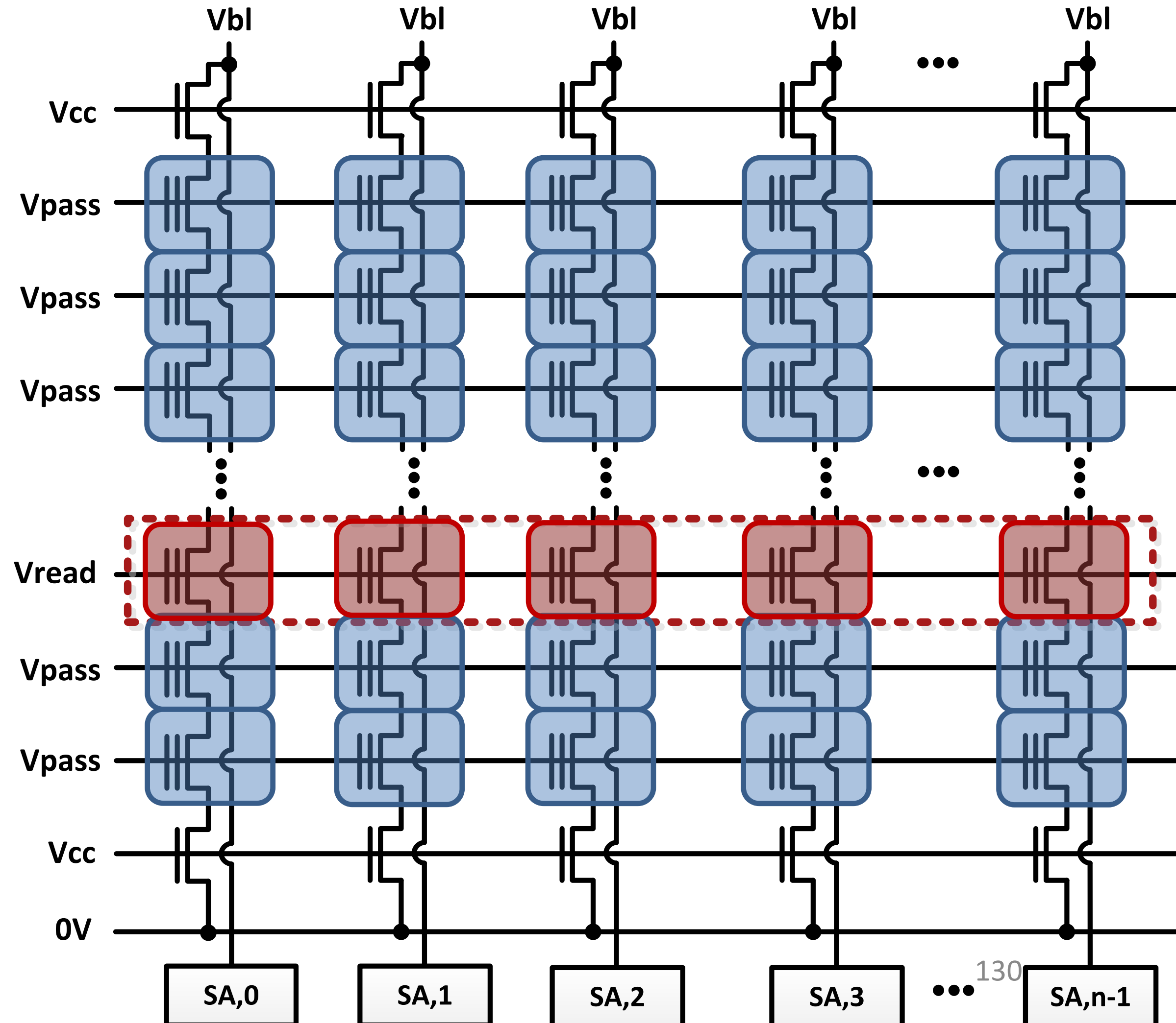
Read Disturb



Read Noise

Read Disturb

Read Noise (RTN)

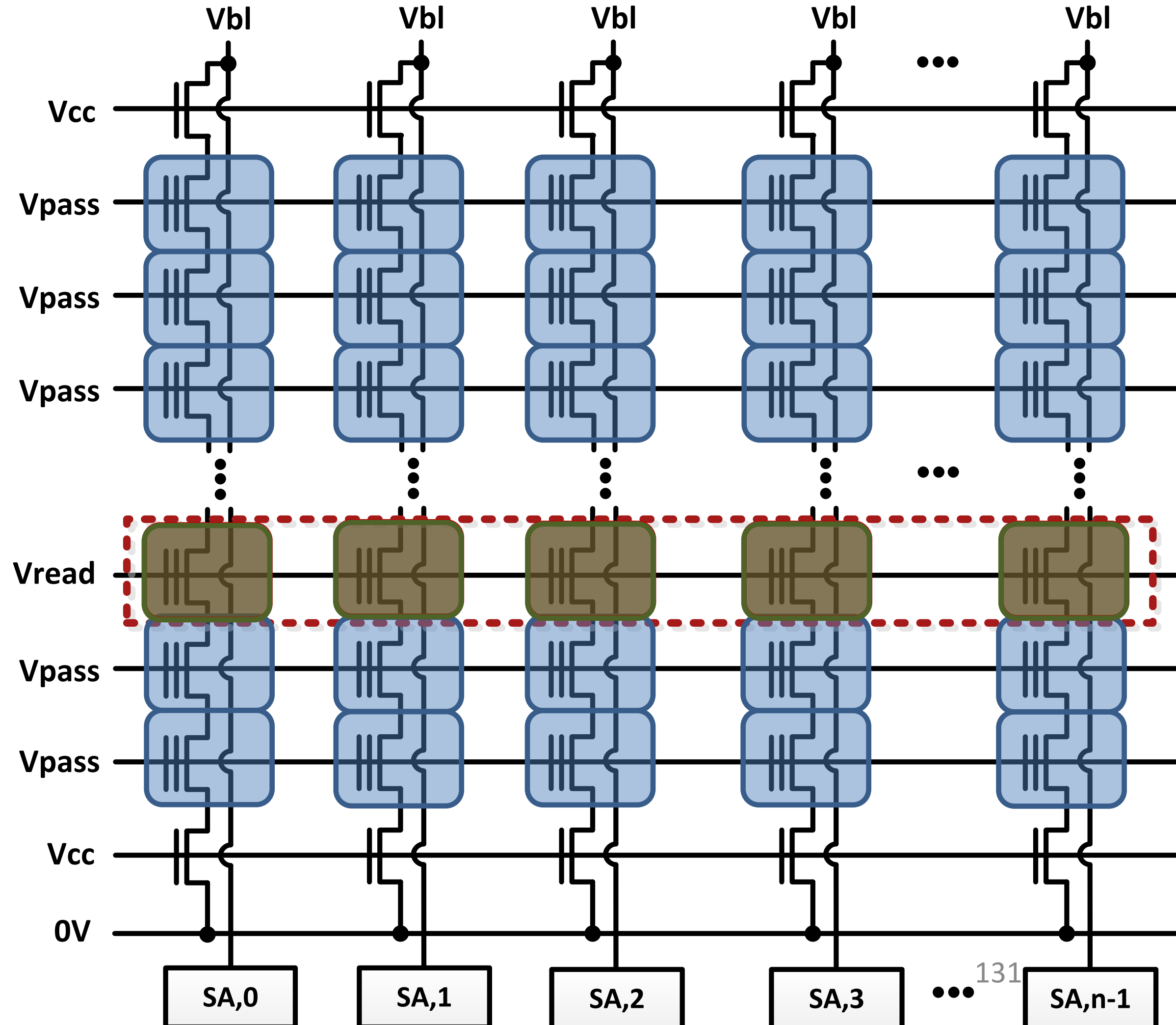


Read Noise

Read Disturb

Read Noise (RTN)

Back Pattern Effect



Conclusion

- There are many noise sources present in NAND flash memory.
  - Some are properties of the NAND flash cell.
  - Some are inherent to the array structure.
- Many of these noise sources are exacerbated by the reduction in process.
- By understanding these noise sources, algorithms can be utilized to maintain reliability through this process scaling.

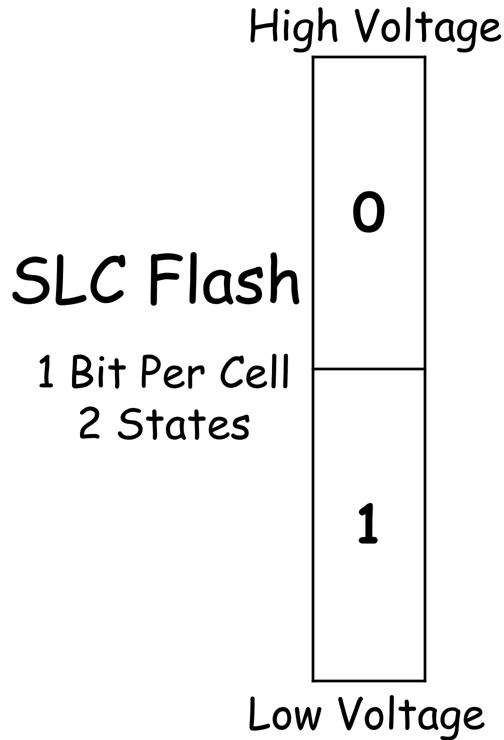
Part II: Error-Correction and Rewriting Codes for Non-Volatile Memories

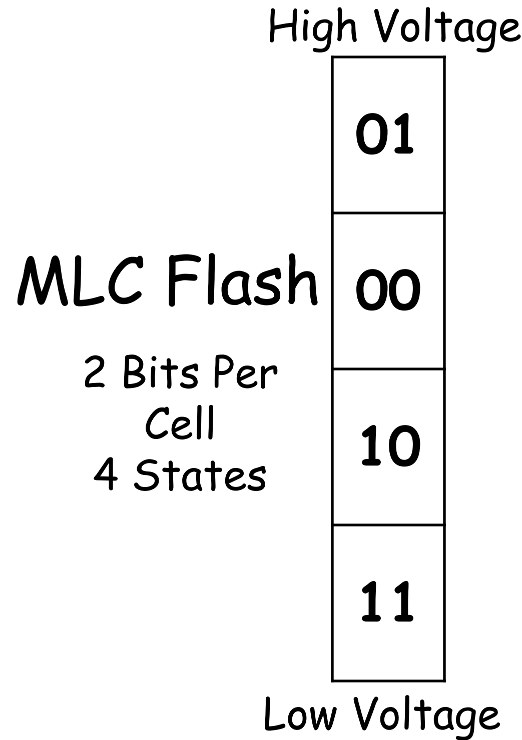
Eitan Yaakobi, Ph.D.
California Institute of Technology

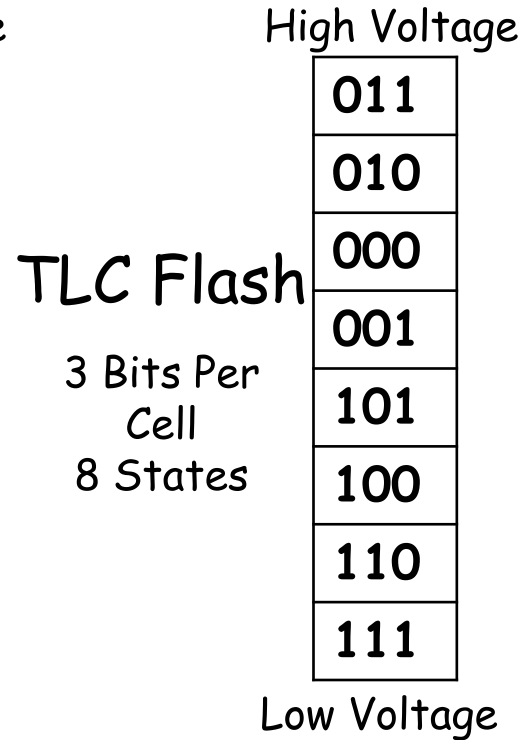
Outline

- Error Correction Codes
- Constrained Codes
- Rewriting Codes

SLC, MLC and TLC Flash







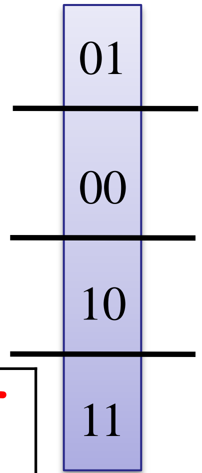
Flash Memory Structure

- A group of cells constitute a page
- A group of pages constitute a block
  - In SLC flash, a typical block layout is as follows

page 0	page 1
page 2	page 3
page 4	page 5
. . .	. . .
page 62	page 63

Flash Memory Structure

- In MLC flash the two bits within a cell **DO NOT** belong to the same page – **MSB page** and **LSB page**
- Given a group of cells, all the MSB's constitute one page and all the LSB's constitute another page

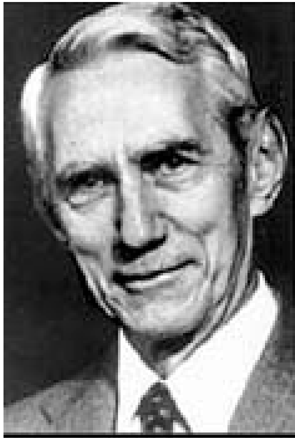


Row index	MSB of first 2^{14} cells	LSB of first 2^{14} cells	MSB of last 2^{14} cells	LSB of last 2^{14} cells
0	page 0	page 4	page 1	page 5
1	page 2	page 8	page 3	page 9
2	page 6	page 12	page 7	page 13
3	page 10	page 16	page 11	page 17
⋮	⋮	⋮	⋮	⋮
30	page 118	page 124	page 119	page 125
31	page 122	page 126	page 123	page 127

TLC Structure

Row index	MSB Page **MSB** of **first** 2^{16} cells	CSB Page **CSB** of **first** 2^{16} cells	LSB Page **LSB** of **first** 2^{16} cells	MSB Page **MSB** of **last** 2^{16} cells	CSB Page **CSB** of **last** 2^{16} cells	LSB Page **LSB** of **last** 2^{16} cells
0	page 0			page 1		
1	page 2	page 6	page 12	page 3	page 7	page 13
2	page 4	page 10	page 18	page 5	page 11	page 19
3	page 8	page 16	page 24	page 9	page 17	page 25
4	page 14	page 22	page 30	page 15	page 23	page 31
⋮	⋮		⋮	⋮		⋮
62	page 362	page 370	page 378	page 363	page 371	page 379
63	page 368	page 376		page 369	page 377	
64	page 374	page 382		page 375	page 383	
65	page 380			page 381		

Shannon Capacity

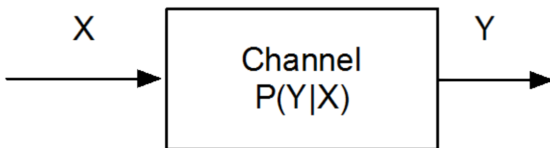

Claude Elwood Shannon
1916 - 2001

Every communication channel is characterized by a single number **C**, called the **channel capacity**.
It is possible to transmit information over this channel reliably (with probability of error $\rightarrow$ 0) if and only if:

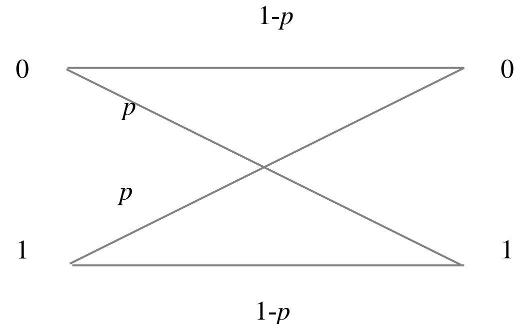
$$R \overset{\text{def}}{=} \frac{\#\,\textit{information bits}}{\textit{channel use}} < C$$

Shannon Capacity

General Channel



Discrete Channel



Capacity (maximized by uniform P(X) for binary input symmetric channel)

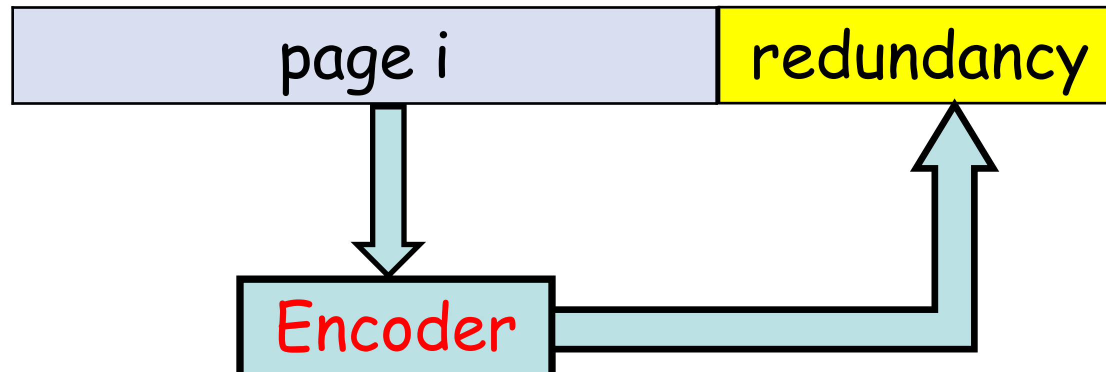
$$I(X;Y) = H(Y) - H(Y|X)$$

$$H(Y) = -\sum_{i=1}^{n} P(y_i)\log_2 P(y_i)$$

$$H(Y|X) = -\sum_{j=1}^{m} P(x_i)H(Y|X = x_i)$$

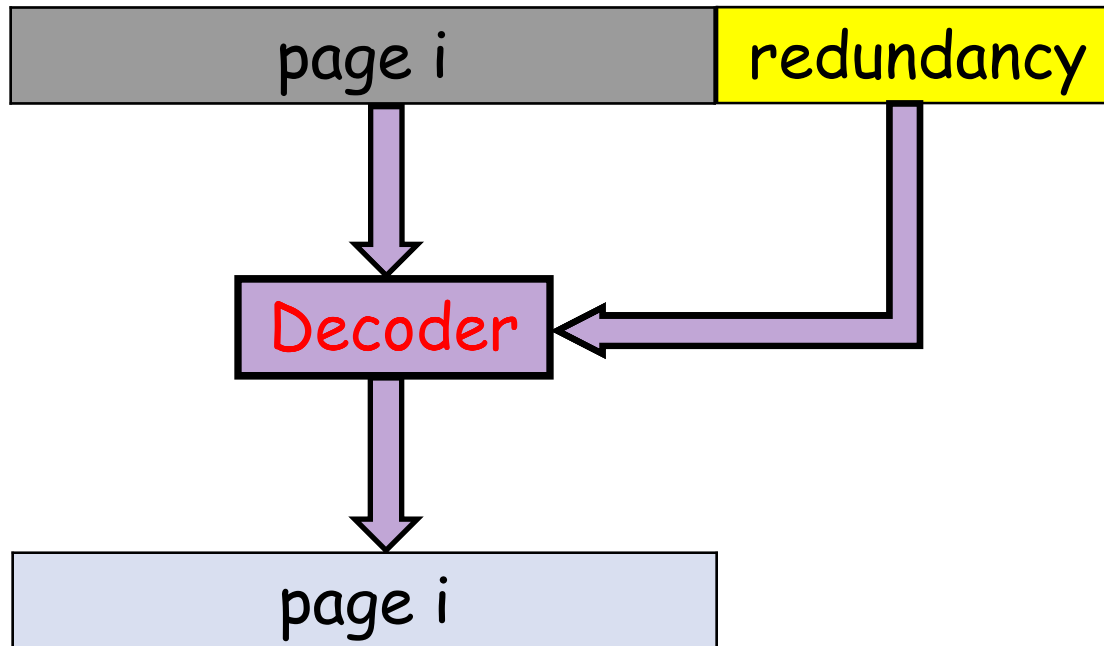
Error Correction Codes

- How does an Error Correction Code (ECC) work?



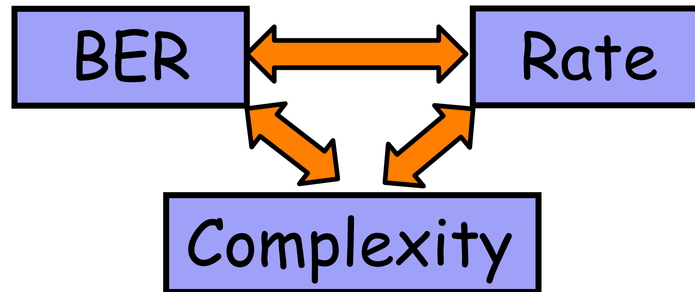
Error Correction Codes

- How does an Error Correction Code (ECC) work?

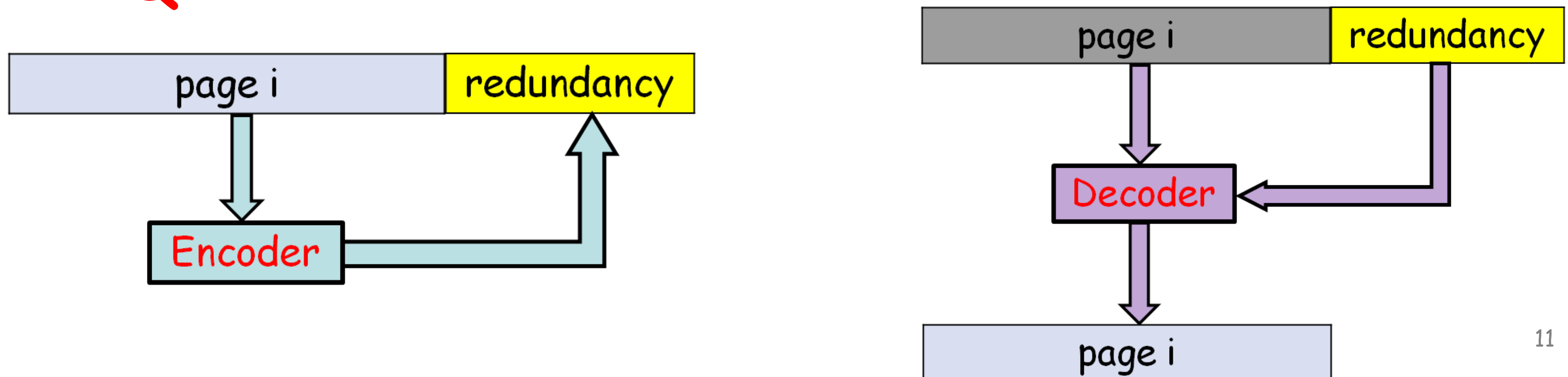


Error Correction Codes

- **Code Rate** = $\frac{\#\text{info. bits}}{\#\text{info. Bits} + \#\text{ redun. bits}}$
- **Tradeoff**:



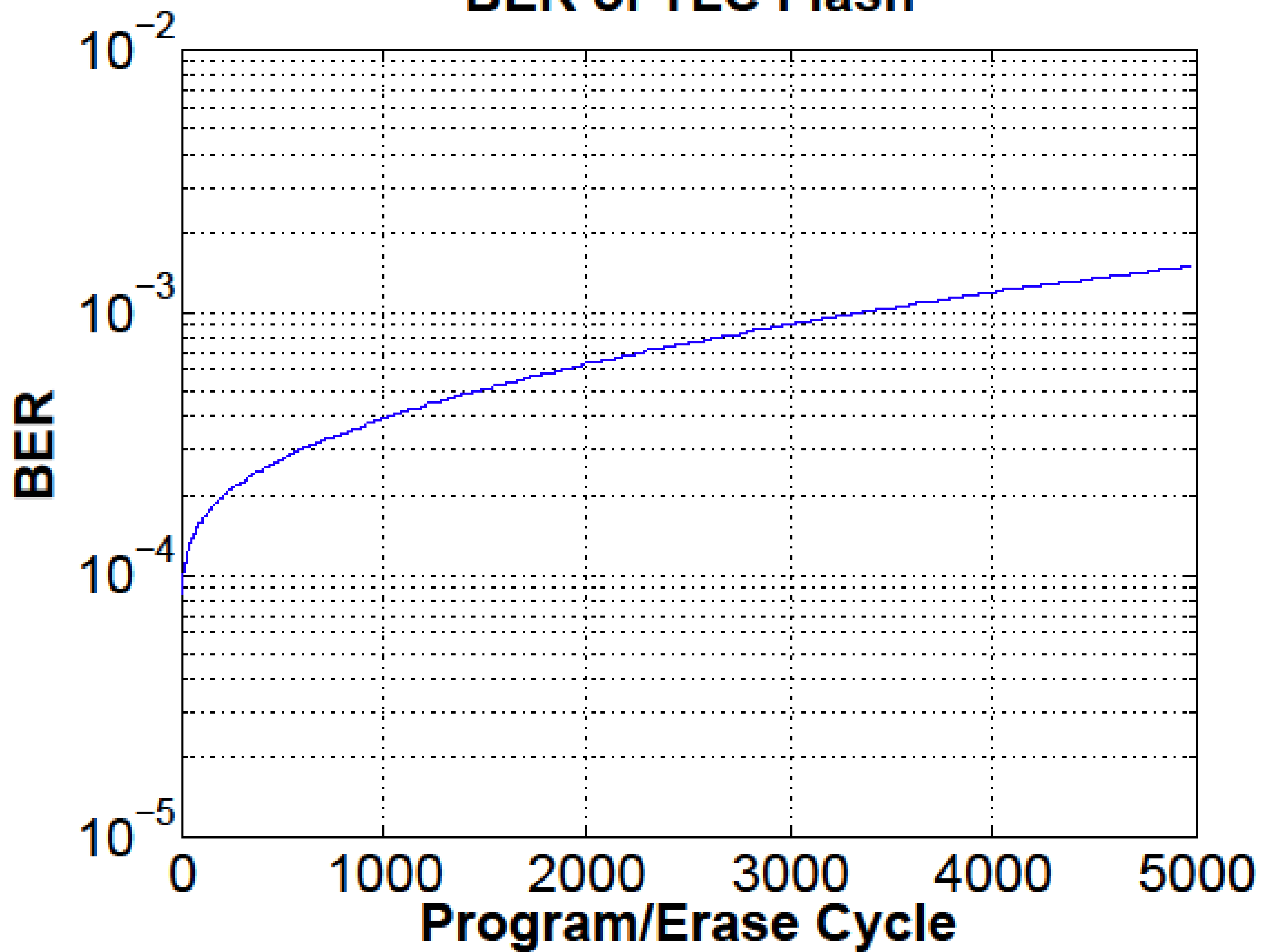
- **Many ECCs**: BCH, RS, Turbo, LDPC, Polar codes…
- **Question**: What ECC to use…?



Error Characterization

- We tested several blocks of SLC/MLC/TLC chips
- For each block the following steps were repeated:
  - The block is **erased**.
  - Pseudo-random data are **programmed** to the block.
  - The data are **read** and **errors** are identified.

- **Disclaimers**:
  - We measured many more P/E cycles than the manufacturer's guaranteed lifetime of the device
  - The experiments were done in laboratory conditions and related factors such as temperature change, intervals between erasures, or multiple readings before erasures were not considered.


BER of TLC Flash
10^{-2}
10^{-3}
10^{-4}
10^{-5}
BER
0
1000
2000
3000
4000
5000
Program/Erase Cycle

ECC Comparison for TLC flash

- **BCH Codes**
- **LDPC Codes**
  - Gallager codes (3,k)-regular, R=0.8, 0.9, 0.925, length 2^{16}
  - AR4JA protograph-based codes, R=0.8, lengths 1280, 5120, 20480
  - MacKay codes variable-regular degree (3 or 4) ; no 4-cycles, R=0.82, 0.87, 0.93; lengths 4095, 16383, 32000
  - IEEE 802.3an* (10Gb/s Ethernet), R ≈0.84, length 2048
- **BCH decoder**: corrects error patterns with up to **t** errors; detects and leaves unchanged more than **t** errors
- **LDPC decoders**: assume binary symmetric channel model **BSC(p)**, with empirical error probability **p**

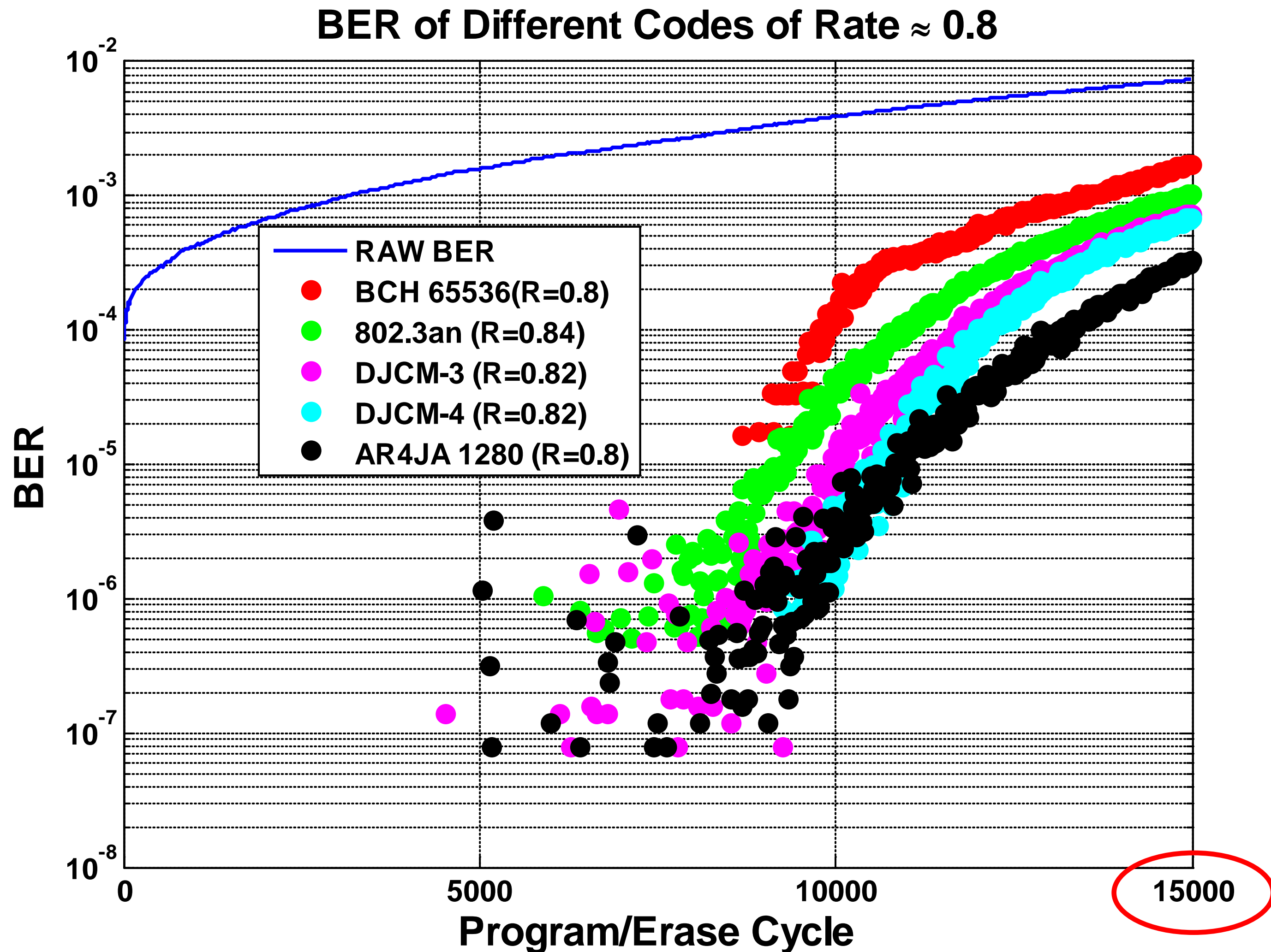
* Djurdjevic et al., *IEEE Commun. Letters*, July 2003

LDPC Decoders

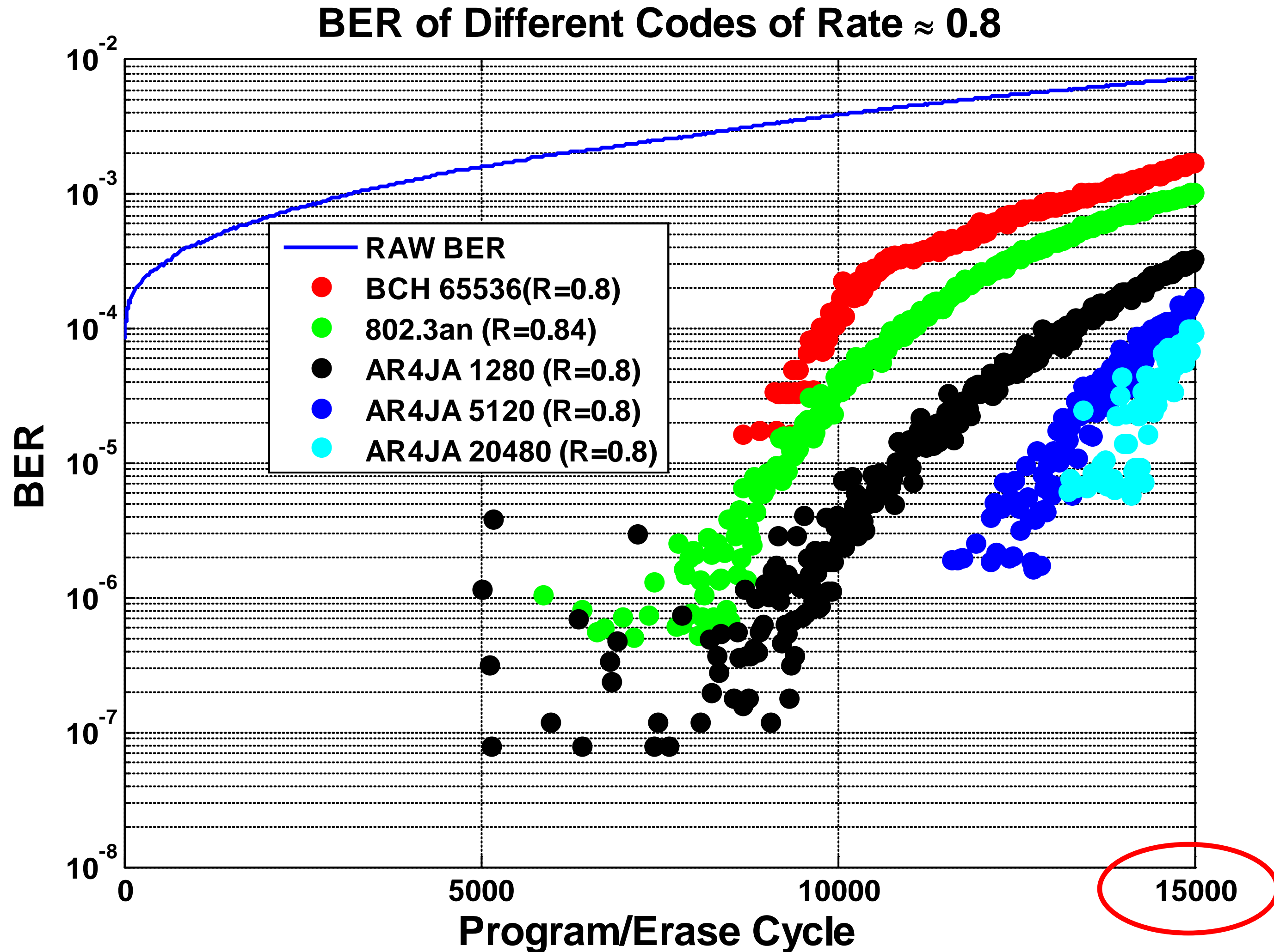
- **Sum-product algorithm (SPA)**
  - Floating-point, max iterations 200
  - (5+1)-bit quasi-uniform quantization
- **Min-sum algorithm (MSA)**
  - No LLR limits, max iterations 200
- **Linear programming (LP) decoding**
  - Alternating Direction Method of Multipliers (ADMM)* with new fast "projection step"

* Barman, et al., *Proc. 46th Allerton Conference,* Sept. 2011.

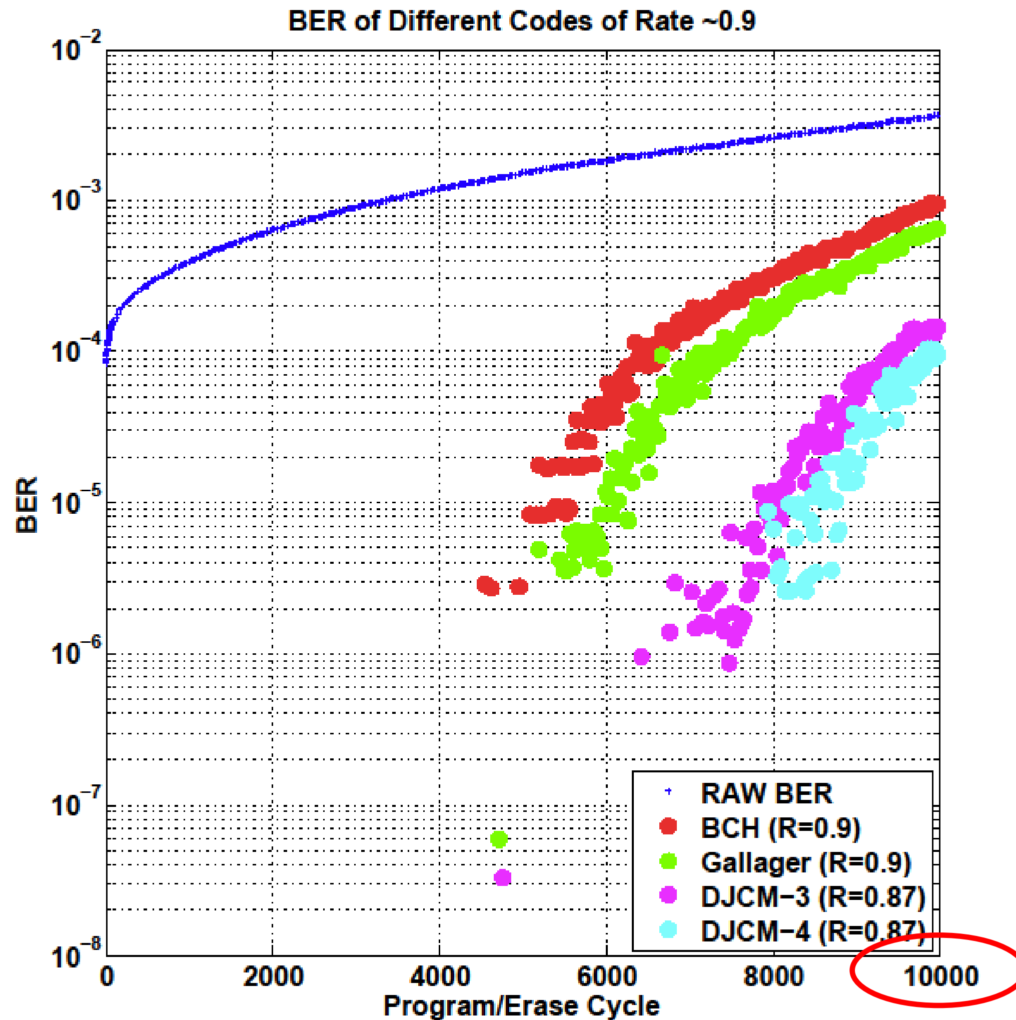
R≈0.8, LDPC with SPA Decoding



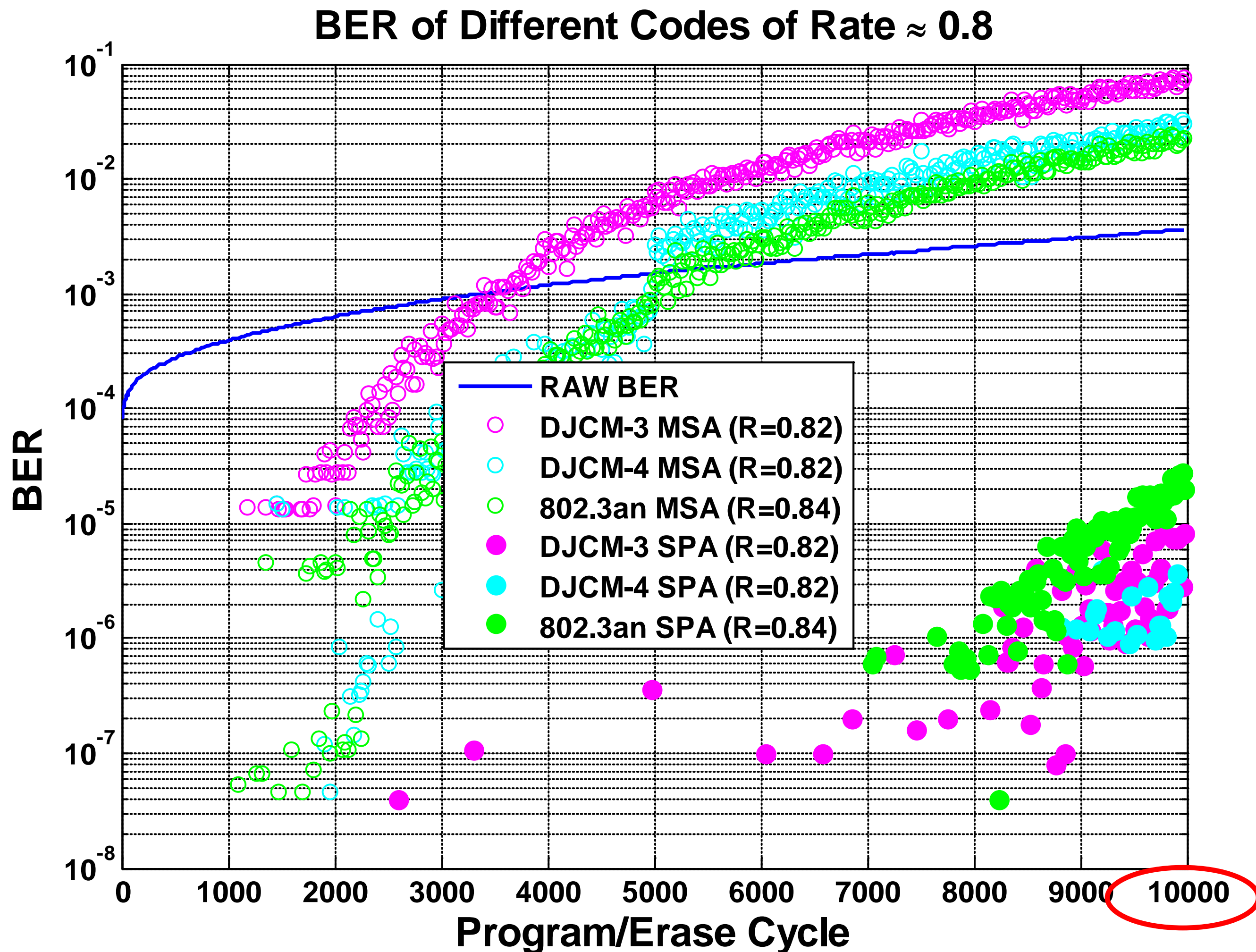
R≈0.82, LDPC with SPA Decoding



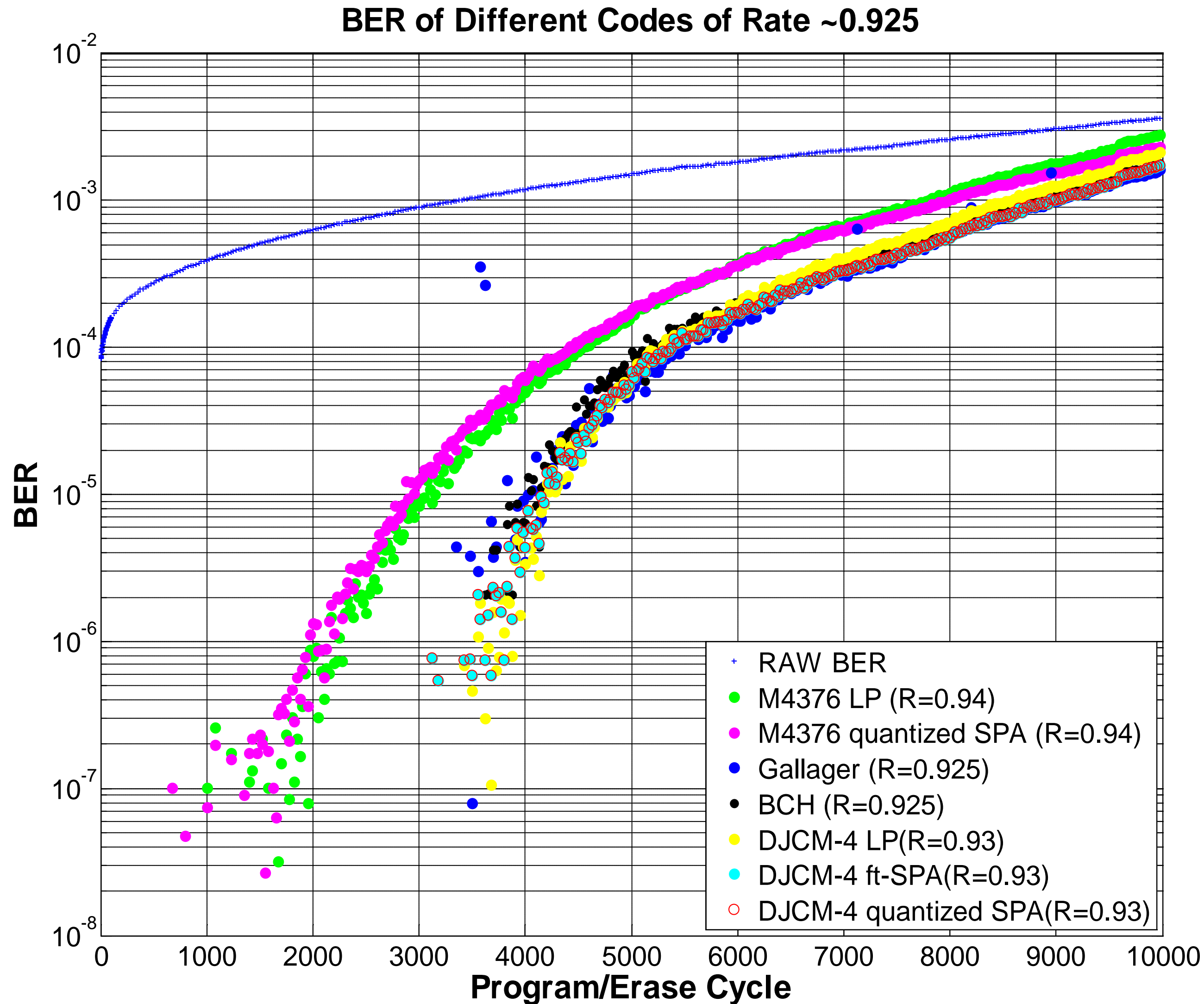
R≈0.9, LDPC with SPA Decoding



R≈0.8, MSA vs. SPA Decoding



R≈0.925, LP vs. SPA Decoding



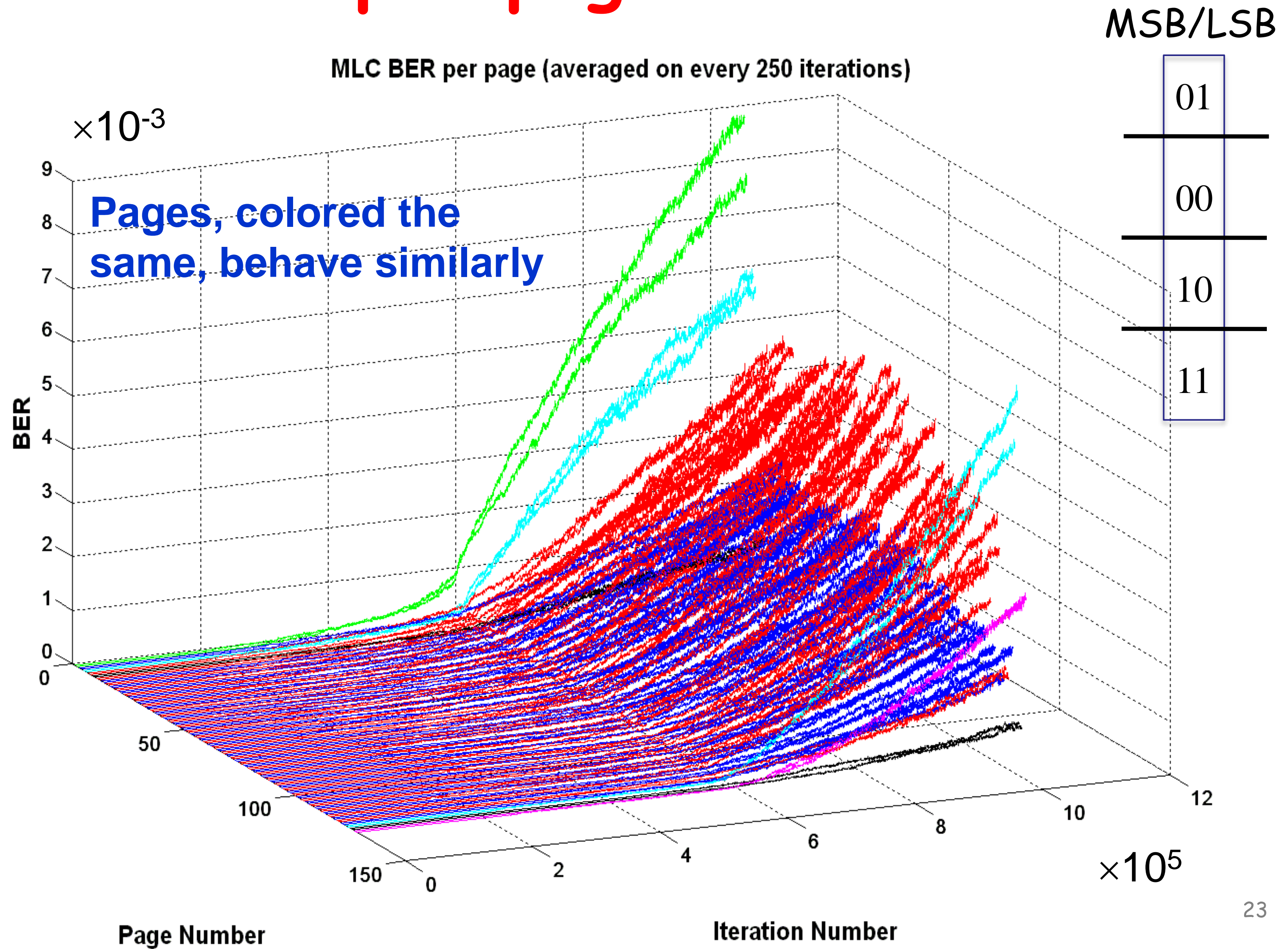
General Observations

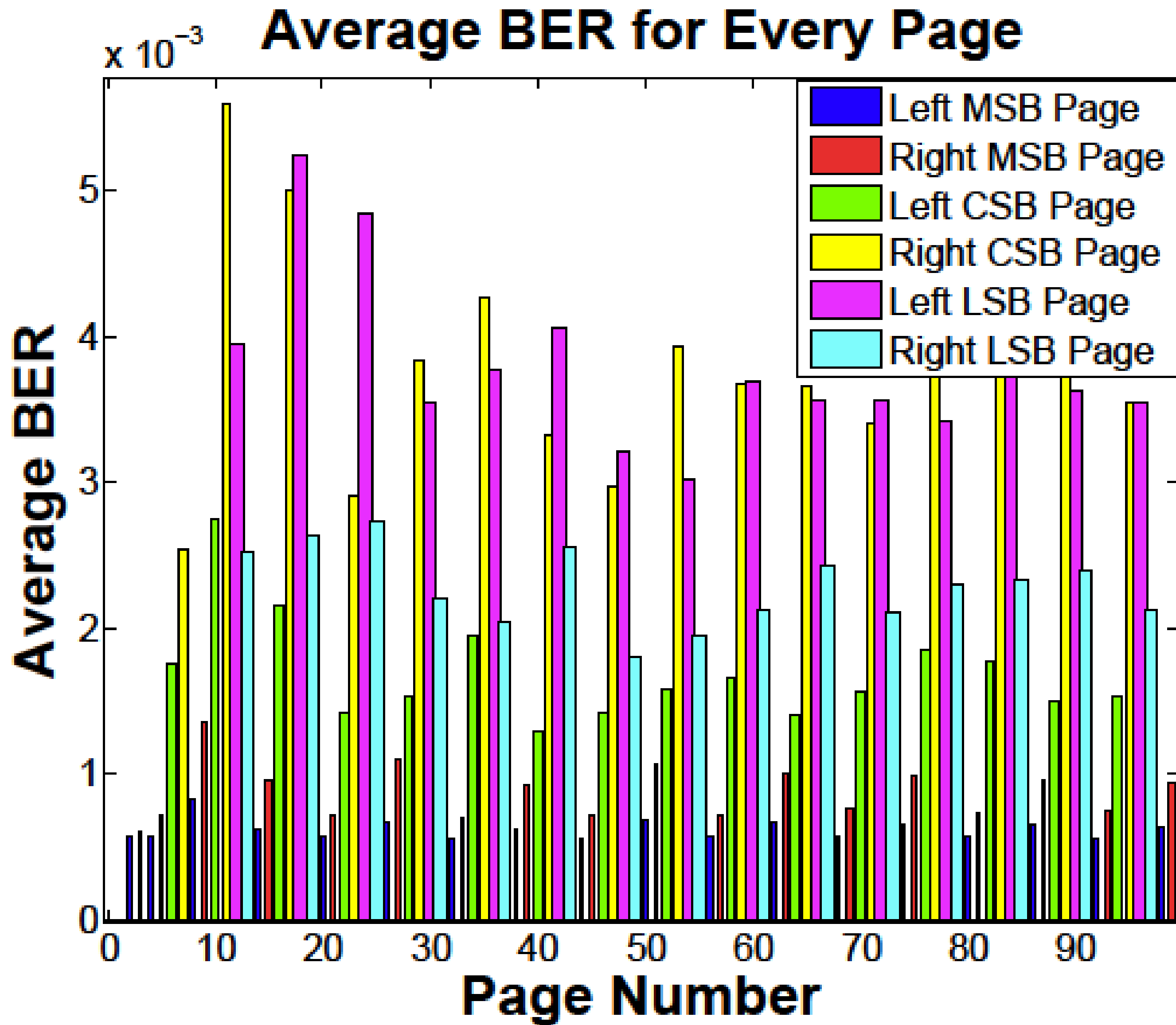
- Best LDPC performance surpasses BCH at all code rates R≈ 0.8, 0.9, 0.925
- MSA was inferior to SPA decoding at R≈0.8
- LP-ADMM was comparable to SPA decoding at R≈0.925, with slightly steeper slope
- (5+1)-bit quasi-uniform quantized SPA (not optimized) matches floating-point SPA
- Soft Vs. Hard input

Error Correction Codes

- **Question**: Is it possible to construct better ECCs?
- **Answer**: Yes! If there is better knowledge on the error model

BER per page - MLC



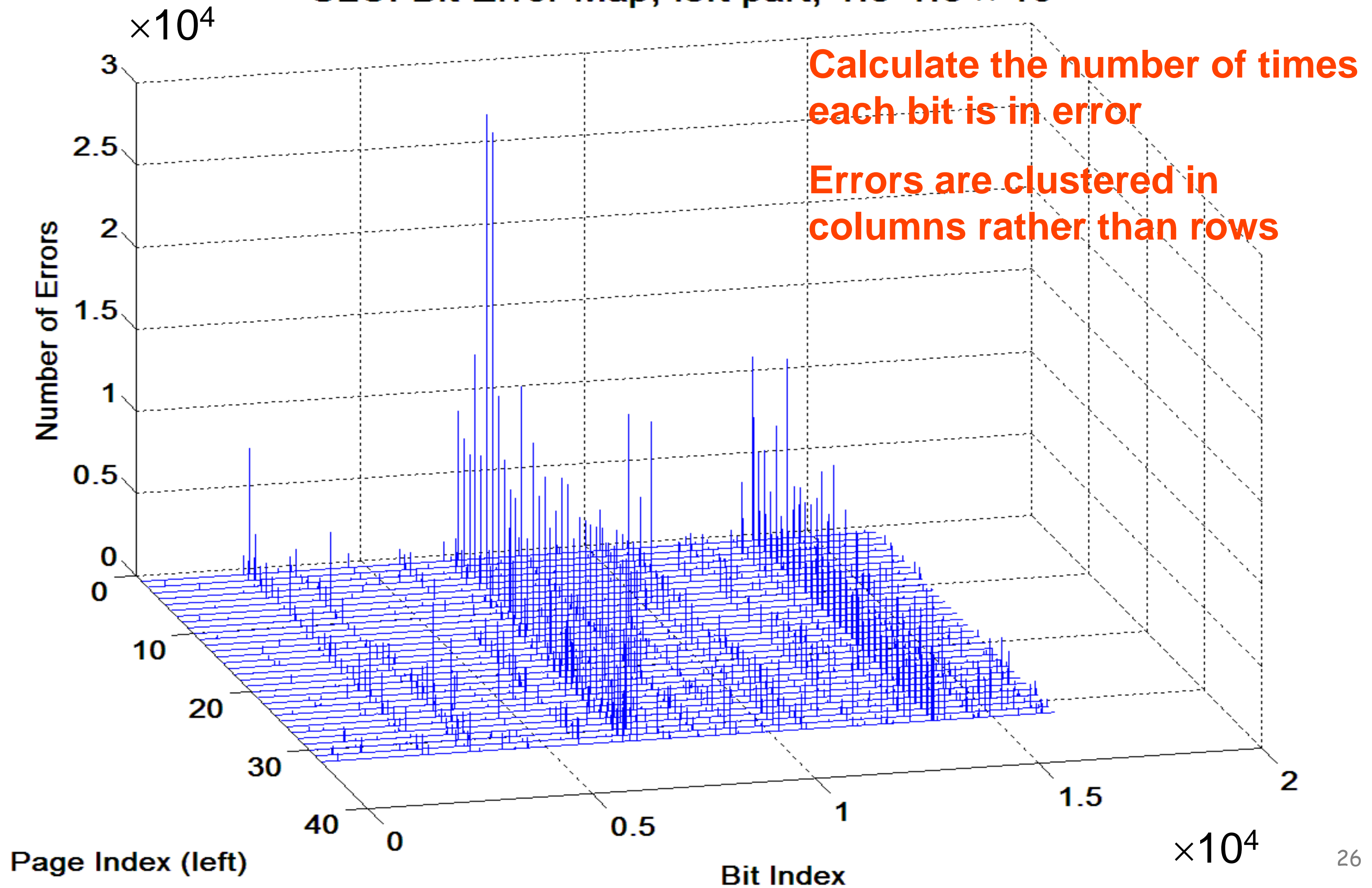


M/C/L
011
010
000
001
101
100
110
111

Bit Error Map in SLC

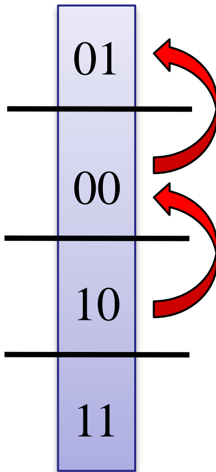
- We checked how the errors behave per bit
- For a small window of iterations, $1.5\text{-}1.6 \times 10^6$ iterations (BER is roughly fixed), we measured the number of times each bit was in error

Bit Error Map for Odd Pages in SLC



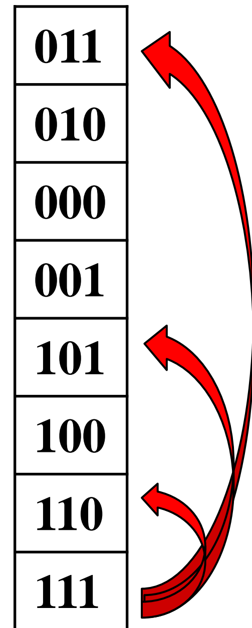
Cell-based ECC

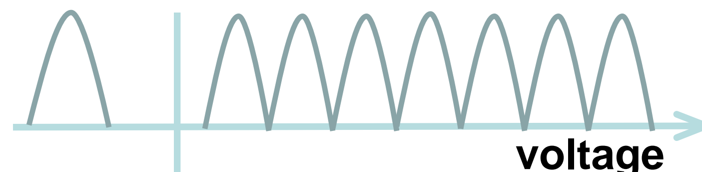
- Experiments have shown that certain specific cell-error types are dominant in MLC and TLC flash memories
- The dominant cell errors in MLC involved a change in cell voltage by only one level: 10 to 00 or 00 to 01
- An algebraic code that targets such errors by sharing redundancy between MSB and LSB pages showed improved BER vs. P/E



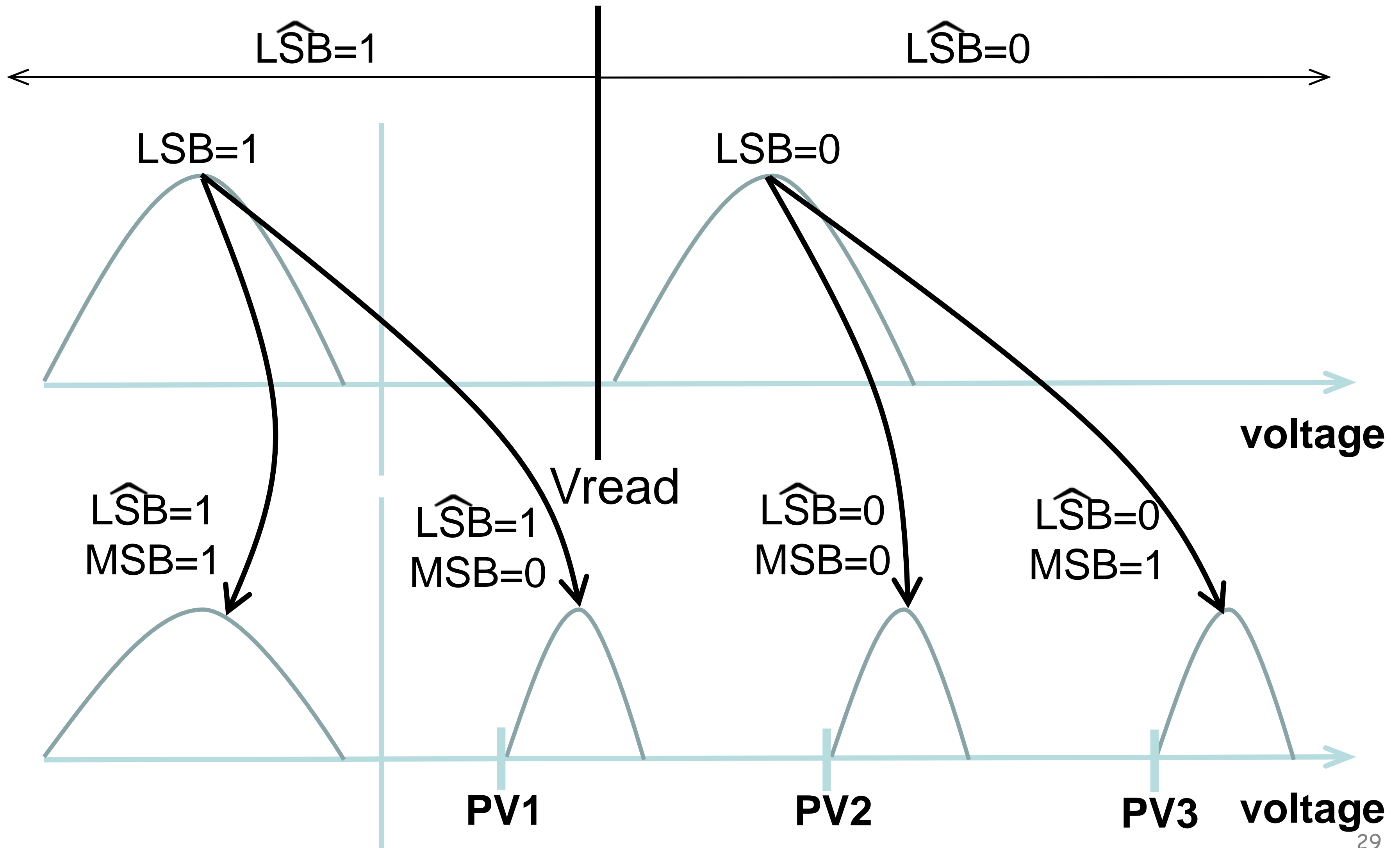
ECC Scheme for TLC Flash

- If a TLC cell is in error, then with high probability only one of the three bits in the cell is in error
- The probability of a bit being in error does not depend on the target cell level
- Algebraic coding schemes that target such errors offer potential BER improvements



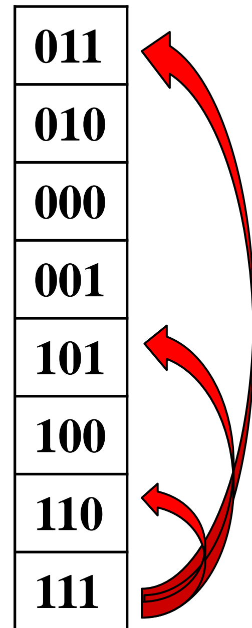


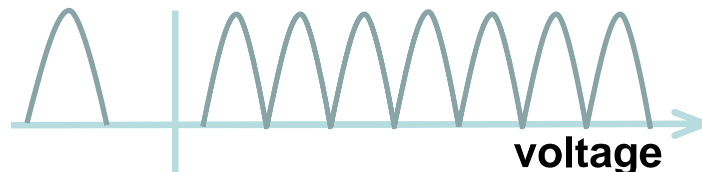
MLC (MSB) Write Process



ECC Scheme for TLC Flash

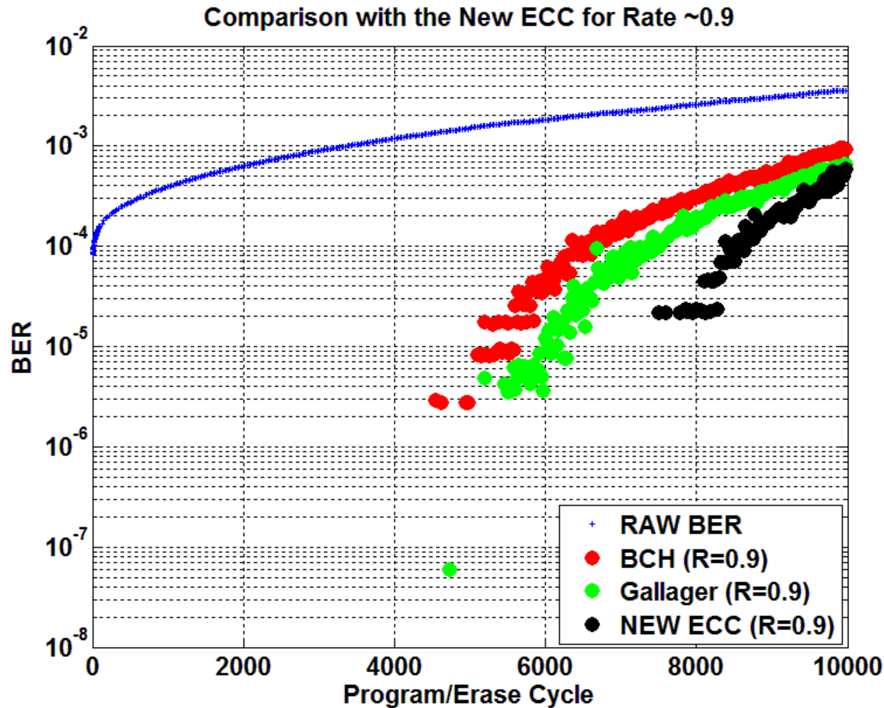
- If a TLC cell is in error, then with high probability only one of the three bits in the cell is in error
- The probability of a bit being in error does not depend on the target cell level
- Algebraic coding schemes that target such errors offer potential BER improvements



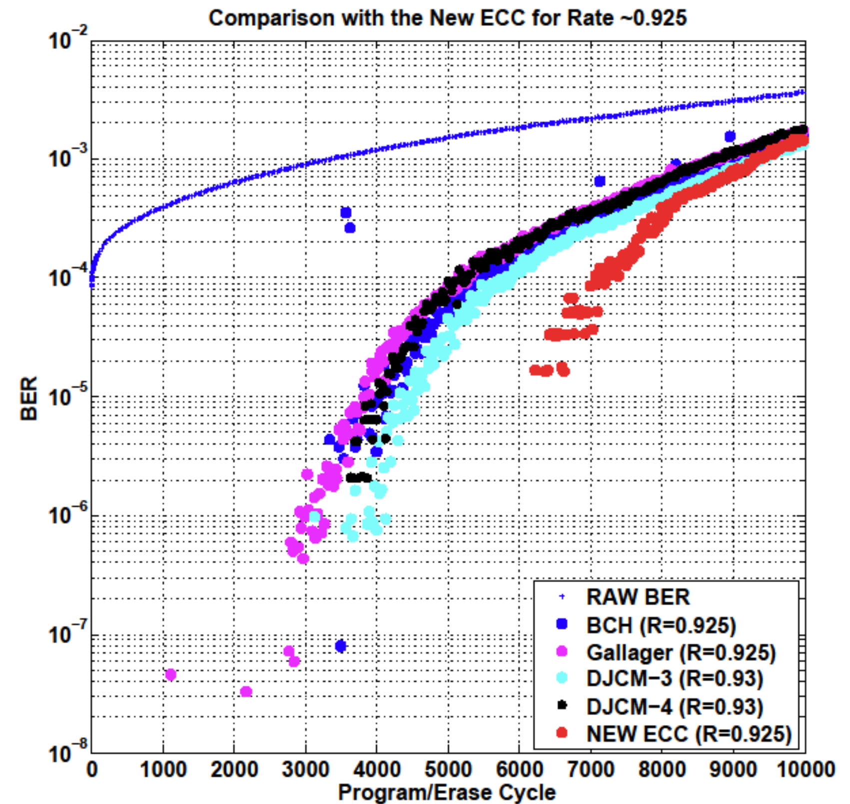


BER for Cell-based Code for TLC Flash

R≈0.9

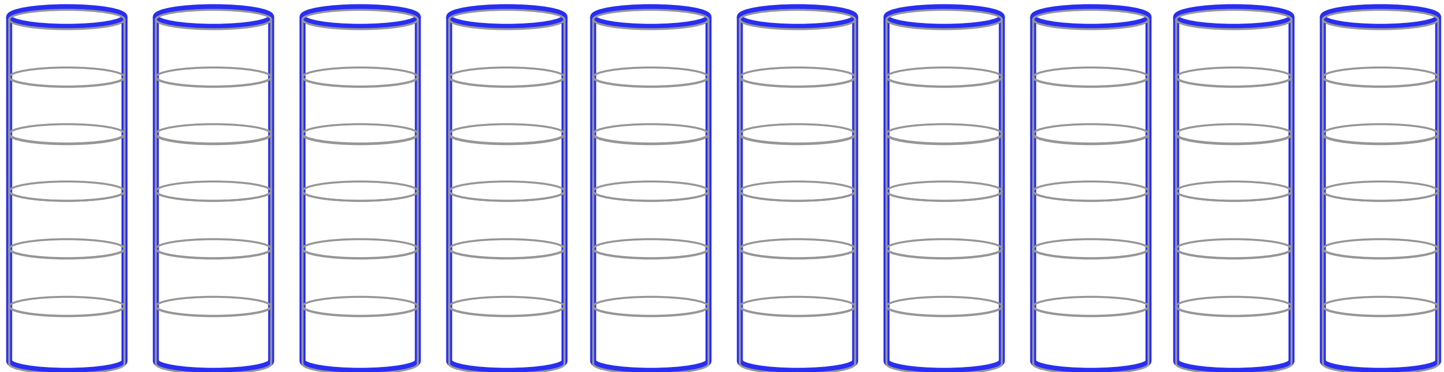


R≈0.925



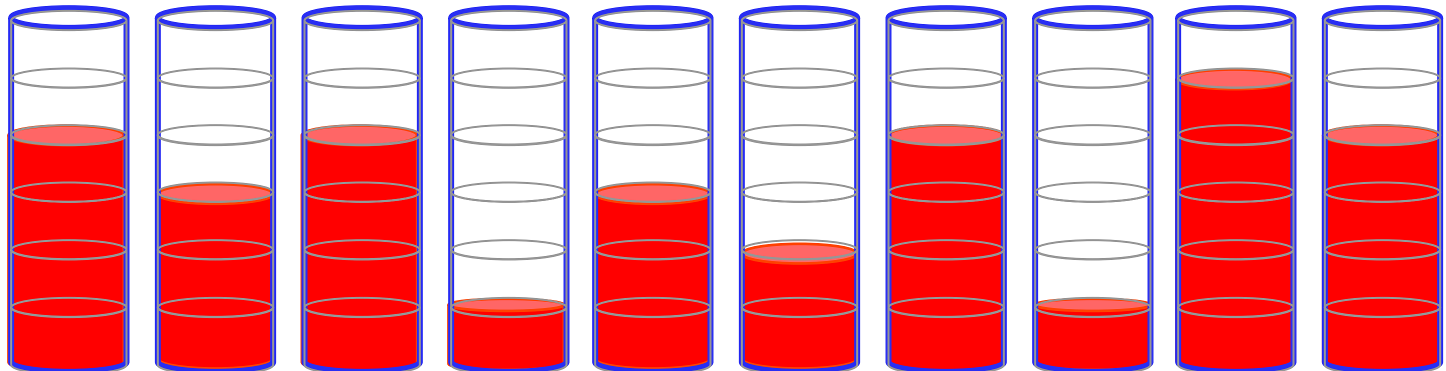
Limited Magnitude Error-Correcting Codes

- Many storage applications, e.g. **flash memories**, **phase-change memories** and more, share the following common properties:
  - Cells have multiple levels: **0,1,…,q-1**
  - Errors have an **asymmetric** behavior



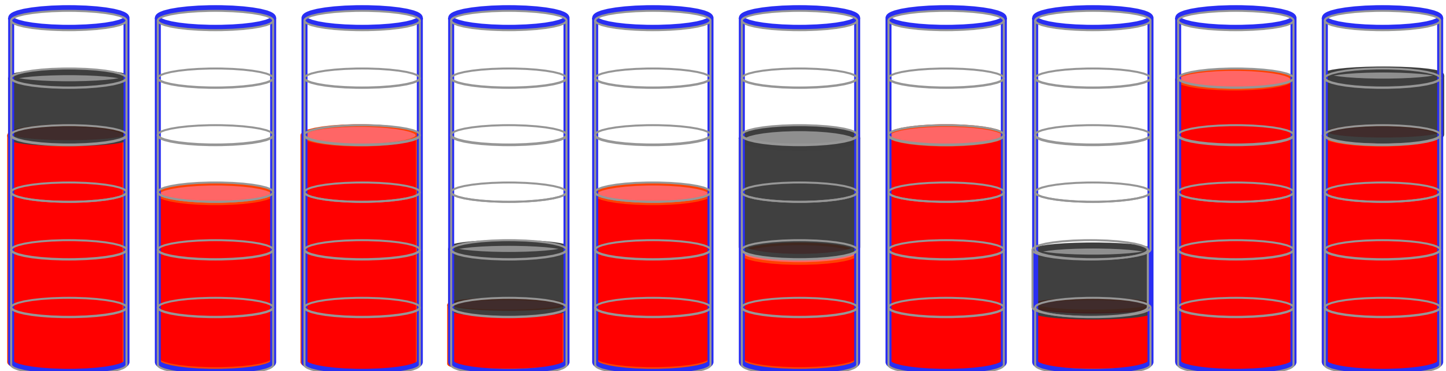
Limited Magnitude Error-Correcting Codes

- Many storage applications, e.g. **flash memories**, **phase-change memories** and more, share the following common properties:
  - Cells have multiple levels: **0,1,…,q-1**
  - Errors have an **asymmetric** behavior
  - If a cell error occurs, then the cell level increases (or decreases) by at most l levels



Limited Magnitude Error-Correcting Codes

- Many storage applications, e.g. **flash memories**, **phase-change memories** and more, share the following common properties:
  - Cells have multiple levels: **0,1,…,q-1**
  - Errors have an **asymmetric** behavior
  - If a cell error occurs, then the cell level increases (or decreases) by at most l levels



Limited Magnitude Error-Correcting Codes

- **Flash memories**
  - Cells **increase** their level during the programming process due to **over-shooting**
  - Cells **decrease** their level due to **data retention**
  - Errors become more prominent as the device is **cycled**
- **Phase change memories**
  - The **drift** in these memories changes the cells' levels in one direction

MLC Data Retention

Count_B-MLC8_400

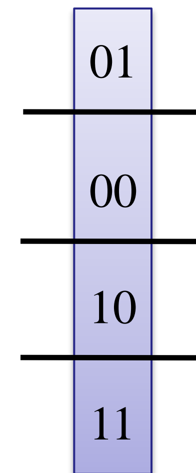
Count of Bits: 10000, 1000, 100, 10, 1, 0.1

Time (yrs): 0, 2, 4, 6, 8, 10, 12

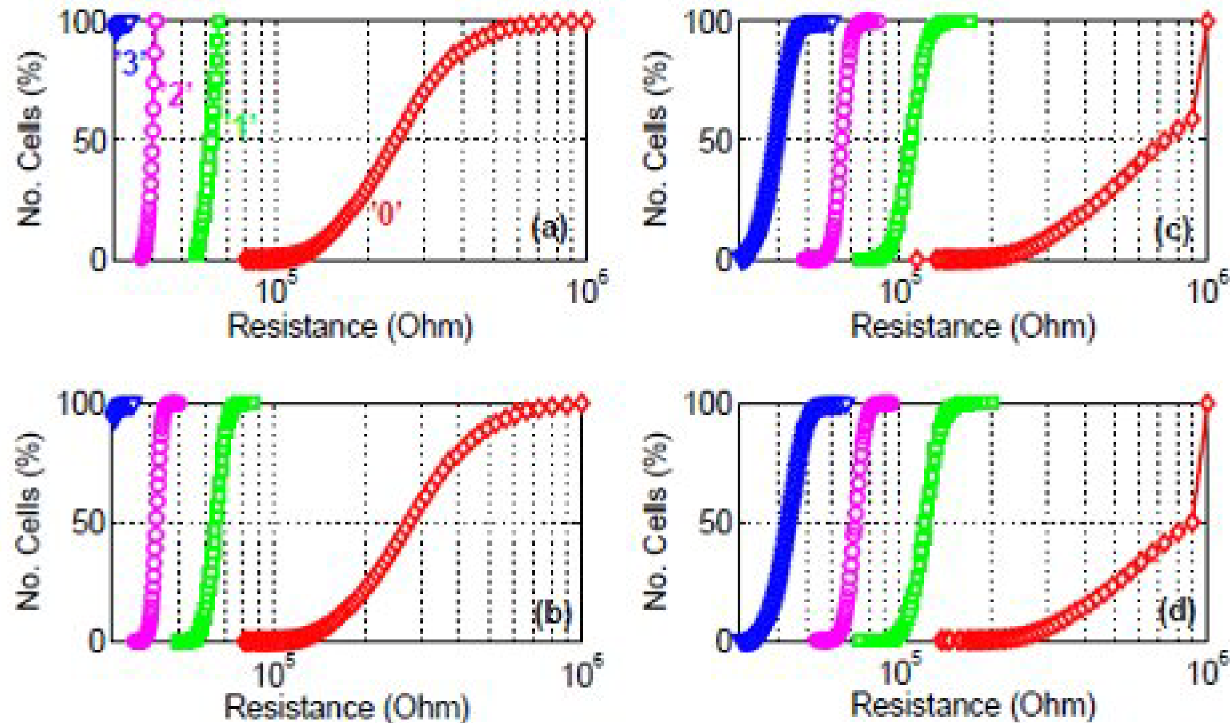
slow in err
00->01
01->00
10->11
11->10
fast in err
00->10
01->11
10->00
11->01
both in err
00->11
01->10
10->01
11->00

Cycle chip to 400% of lifetime

Bake at 125°C for 9hrs20mins per year of aging



Cell Drift in PCM



Time evolution of programmed resistance distributions of 200 kcells due to drift: (a) as programmed, and (b) 40μs, (c) 1000s, (d) 46,000s after programming.

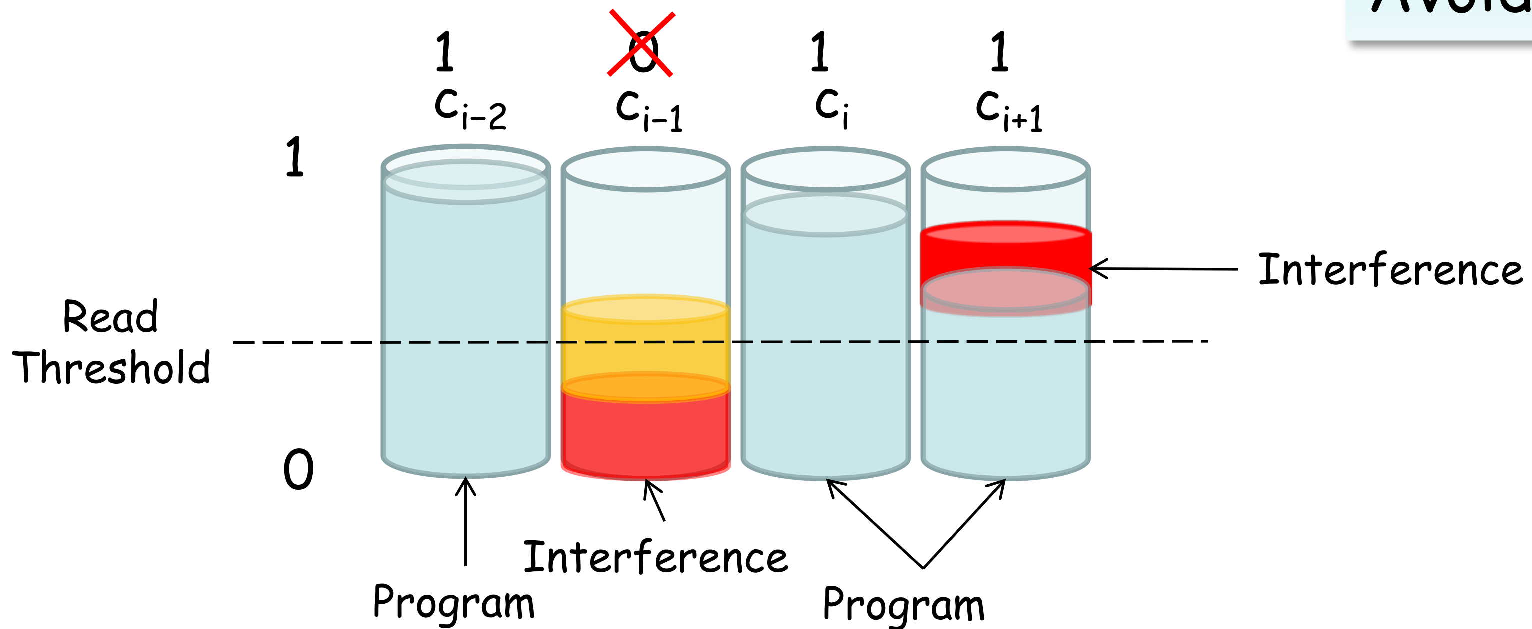
Figure from: N. Papandreou, H. Pozidis, T. Mittelholzer, G. F. Close, M. Breitwisch, C. Lam, and E. Eleftheriou, "Drift-Tolerant Multilevel Phase-Change Memory", 3rd IEEE Memory Workshop, May 2011

Constrained Codes

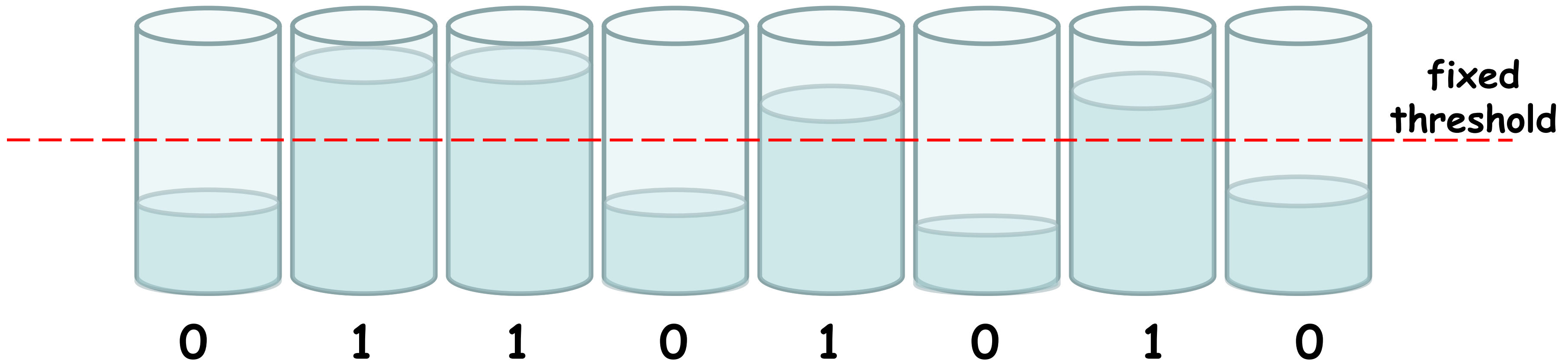
- Codes designed to prevent specific data patterns
  - Ex. **Run Length Limited** codes **RLL** **(d,k)**
  - Number of **0s** b/w consecutive **1s** is at least **d** and at most **k**
  - Used in telecommunications and storage systems for **synchronization purposes**
- What are the typical constraints in flash?

Inter-Cell Interference (ICI)

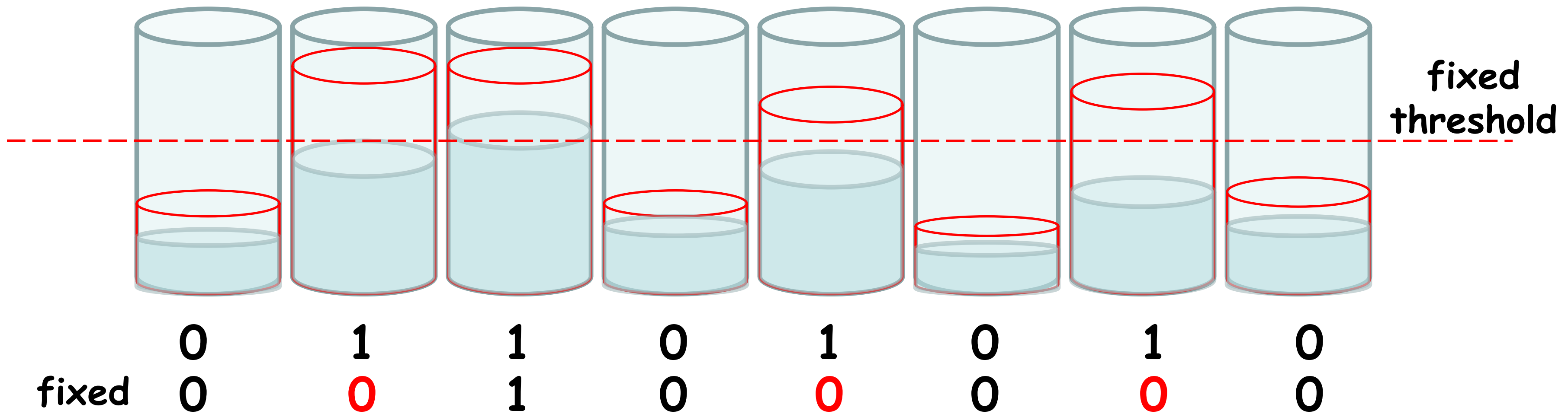
- Mitigate inter-cell interference → 101 is forbidden



Balanced Codes

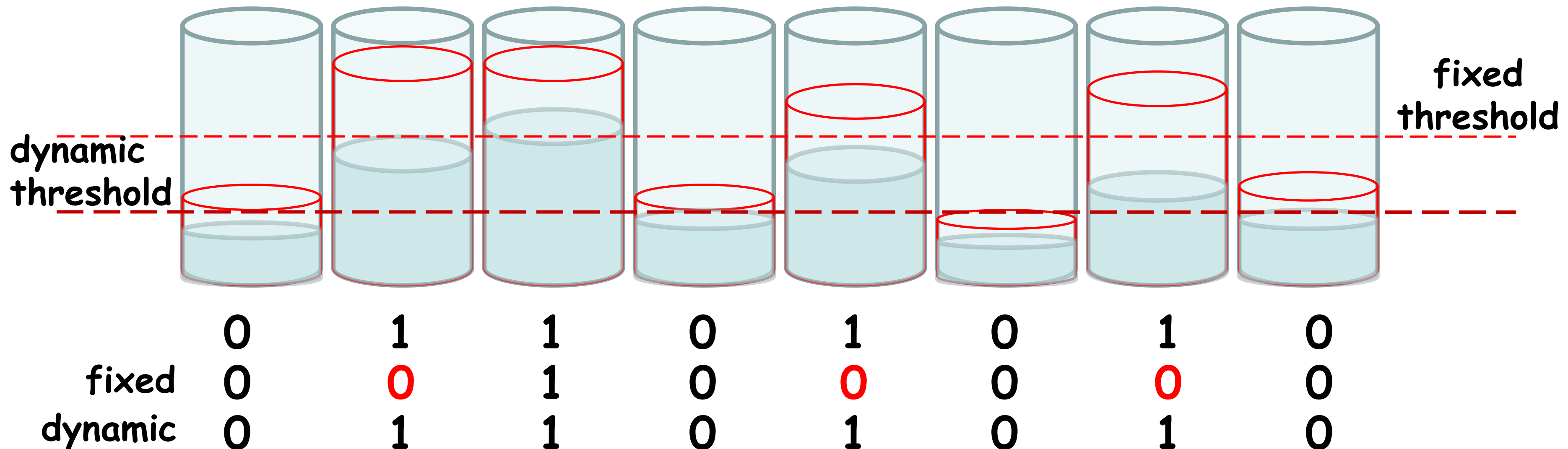


Balanced Codes



Balanced Codes

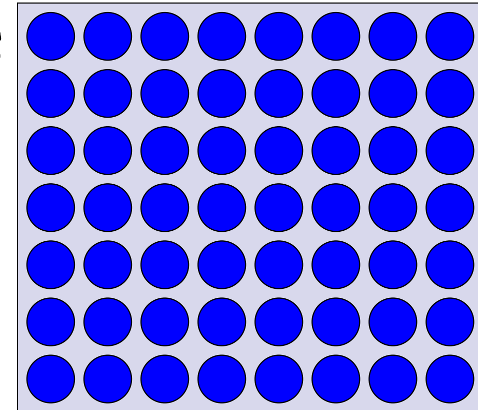
- Write only **balanced** words: **#0s = #1s**
- In reading: the **n/2** low cells are read as **0**
  the **n/2** high cells are read as **1**
- **Relative ranking** is most likely preserved

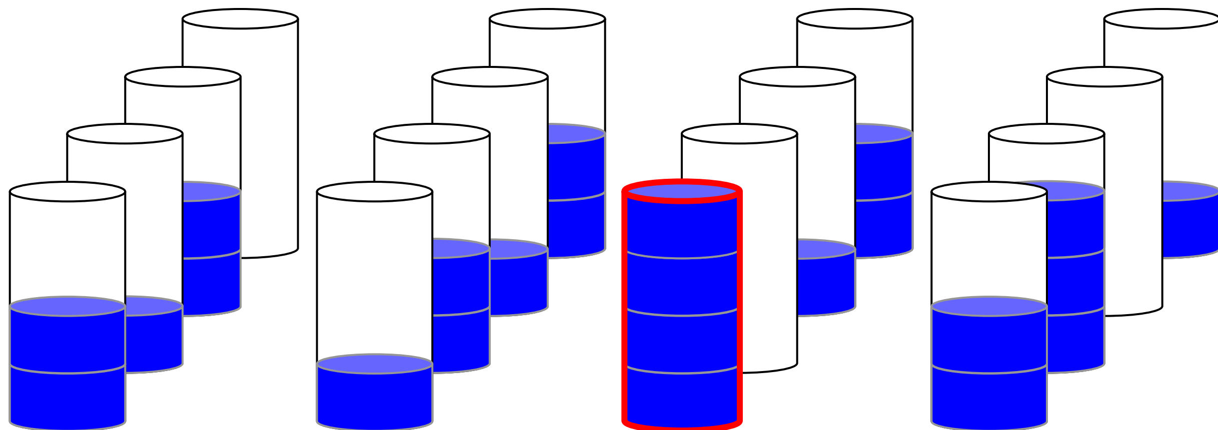


Rewriting Codes

- Array of cells, made of floating gate transistors
  - Each cell can store **q** different levels
  - Today, **q** typically ranges between **2** and **16**
  - The levels are represented by the number of electrons
  - The cell's level is increased by pulsing electrons
  - To reduce a cell level, all cells in its containing block must first be reset to level 0

A VERY EXPENSIVE OPERATION





Rewriting Codes

- **Problem**: Cannot rewrite the memory without an erasure
- **However**… It is still possible to rewrite if only cells in low level are programmed

From Wikipedia:

One limitation of flash memory is that, although it can be read or programmed a byte or a word at a time in a random access fashion, it can only be erased a "block" at a time. This generally sets all bits in the block to 1. Starting with a freshly erased block, any location within that block can be programmed. ***However, once a bit has been set to 0, only by erasing the entire block can it be changed back to 1***. In other words, flash memory (specifically NOR flash) offers random-access read and programming operations, but does not offer arbitrary random-access rewrite or erase operations. ***A location can, however, be rewritten as long as the new value's 0 bits are a superset of the over-written values***. For example, a nibble value may be erased to 1111, then written e.g. as 1110. Successive writes to that nibble can change it to 1010, then 0010, and finally 0000. Essentially, erasure sets all bits to 1, and programming can only clear bits to 0. File systems designed for flash devices can make use of this capability, for example to represent sector metadata.

Rewriting Codes

- **Problem**: Cannot rewrite the memory without an erasure
- **However**… It is still possible to rewrite if only cells in low level are programmed
- **Naive Example**:
  - First write: program only the even pages
  - Second write: program only the odd pages

page 0	page 1
page 2	page 3
page 4	page 5
. . .	. . .
page 62	page 63

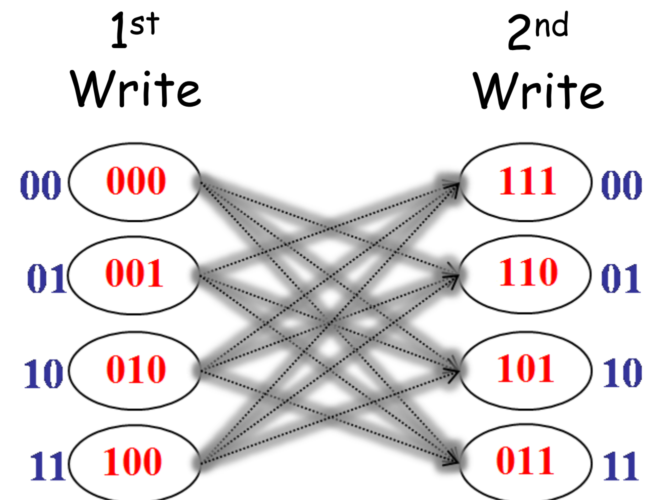
Rewriting Codes

- One of the most efficient schemes to decrease the number of block erasures
- Floating Codes
- Buffer Codes
- Trajectory Codes
- Rank Modulation Codes
- WOM Codes

Write-Once Memories (WOM)

- Introduced by **Rivest and Shamir**, "*How to reuse a write-once memory*", 1982
- The memory elements represent bits (2 levels) and are irreversibly programmed from **'0'** to **'1'**

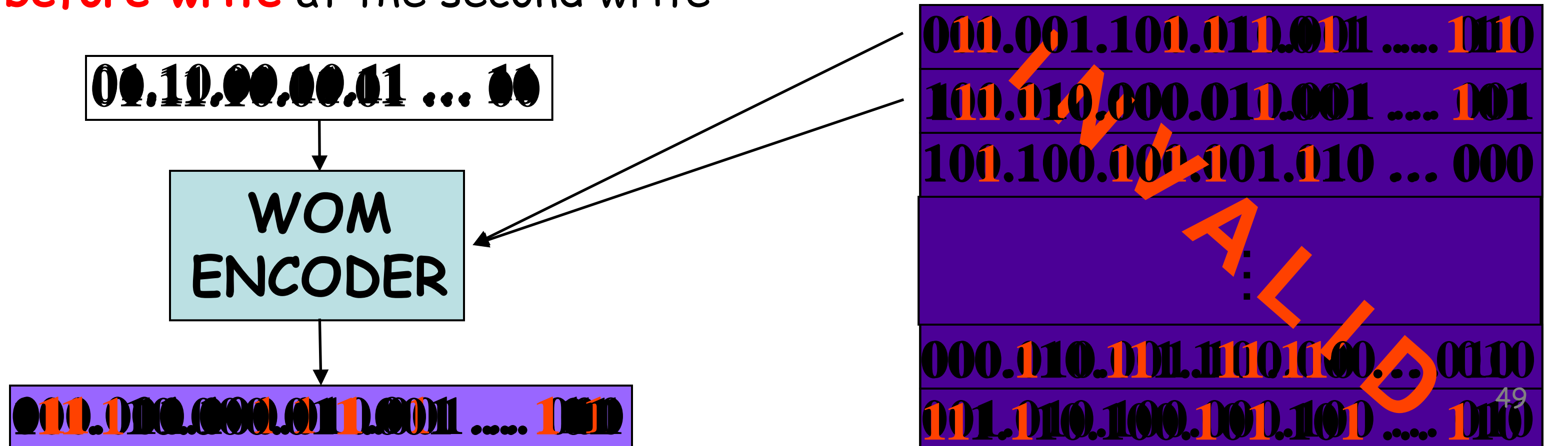
Bits Value	1st Write	2nd Write
00	000	111
01	001	110
10	010	101
11	100	011



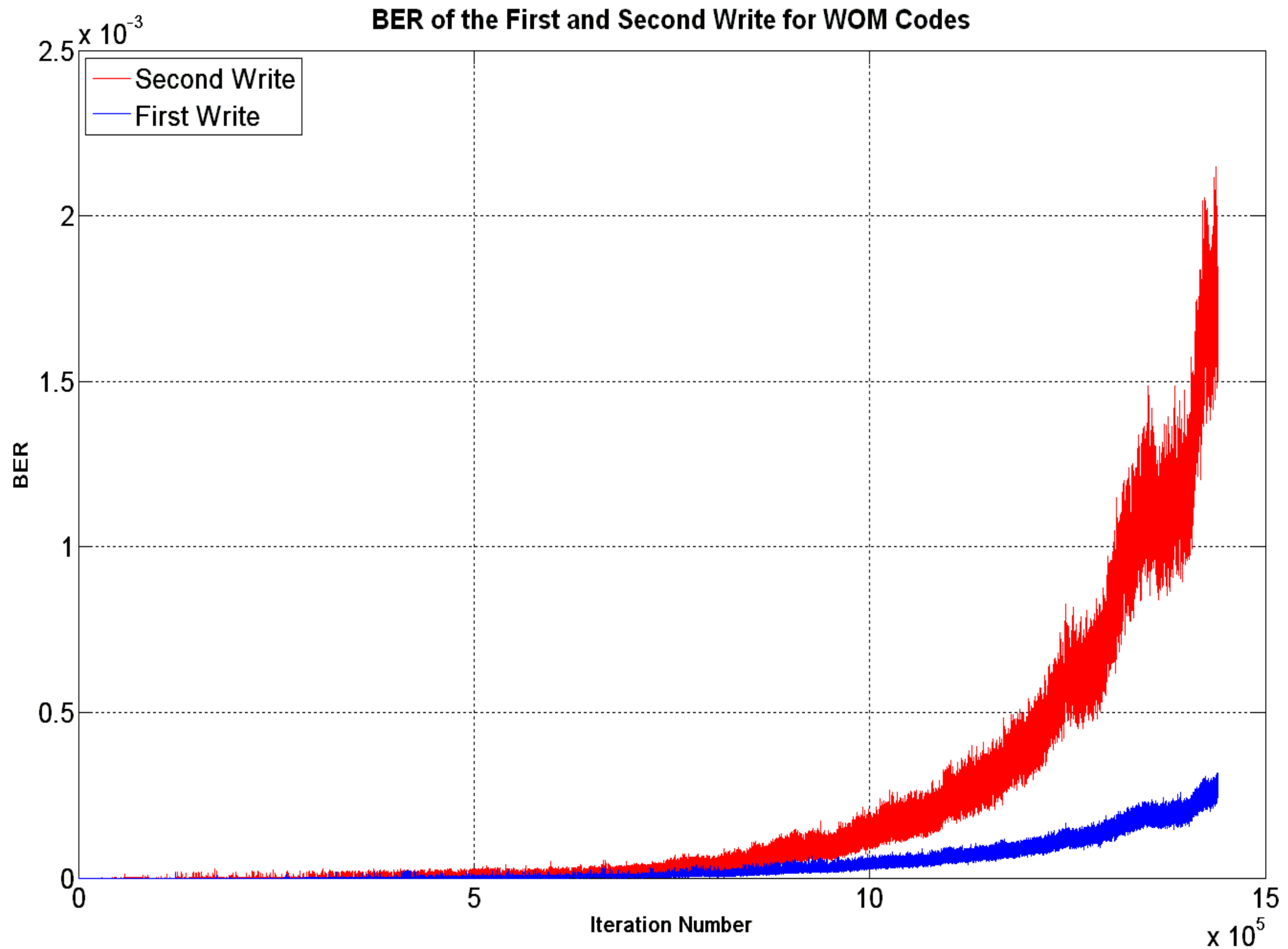
WOM Implementation in SLC Flash

- A scheme for storing two bits twice using only three cells **before erasing the cells**
- The cells only **increase** their level
- How to implement? (in **SLC** block)
  - Each page stores **2KB/1.5 = 4/3KB** per write
  - A page can be written twice before erasing
  - Pages are **encoded** using the **WOM code**
  - When the block has to be rewritten, mark its pages as **invalid**
  - Again write pages using the WOM code **without erasing**
  - **Read before write** at the second write

data	1st write	2nd write
00	000	111
01	100	011
10	010	101
11	001	110



BER for the First and Second Write



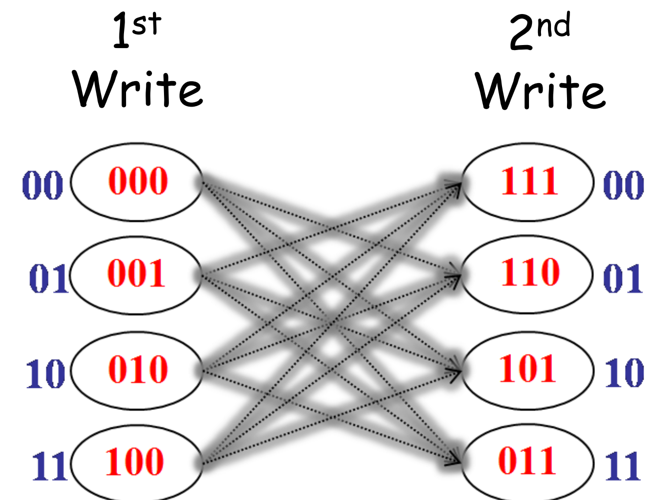
Write-Once Memories (WOM)

- Introduced by **Rivest and Shamir**, "*How to reuse a write-once memory*", 1982
- The memory elements represent bits (2 levels) and are irreversibly programmed from '0' to '1'

Bits Value	1st Write	2nd Write
00	000	111
01	001	110
10	010	101
11	100	011

The problem:

What is the total number of bits that is possible to write in **n** cells in **t** writes?



Binary WOM-Codes

- $k_1, \ldots, k_t$:the number of bits on each write
  - n cells and t writes
- The **sum-rate** of the WOM-code is

  $$R = (\Sigma_1^t k_i)/n$$

  - Rivest Shamir: R = (2+2)/3 = 1.333
- Fixed-rate and Unrestricted-rate WOM-codes

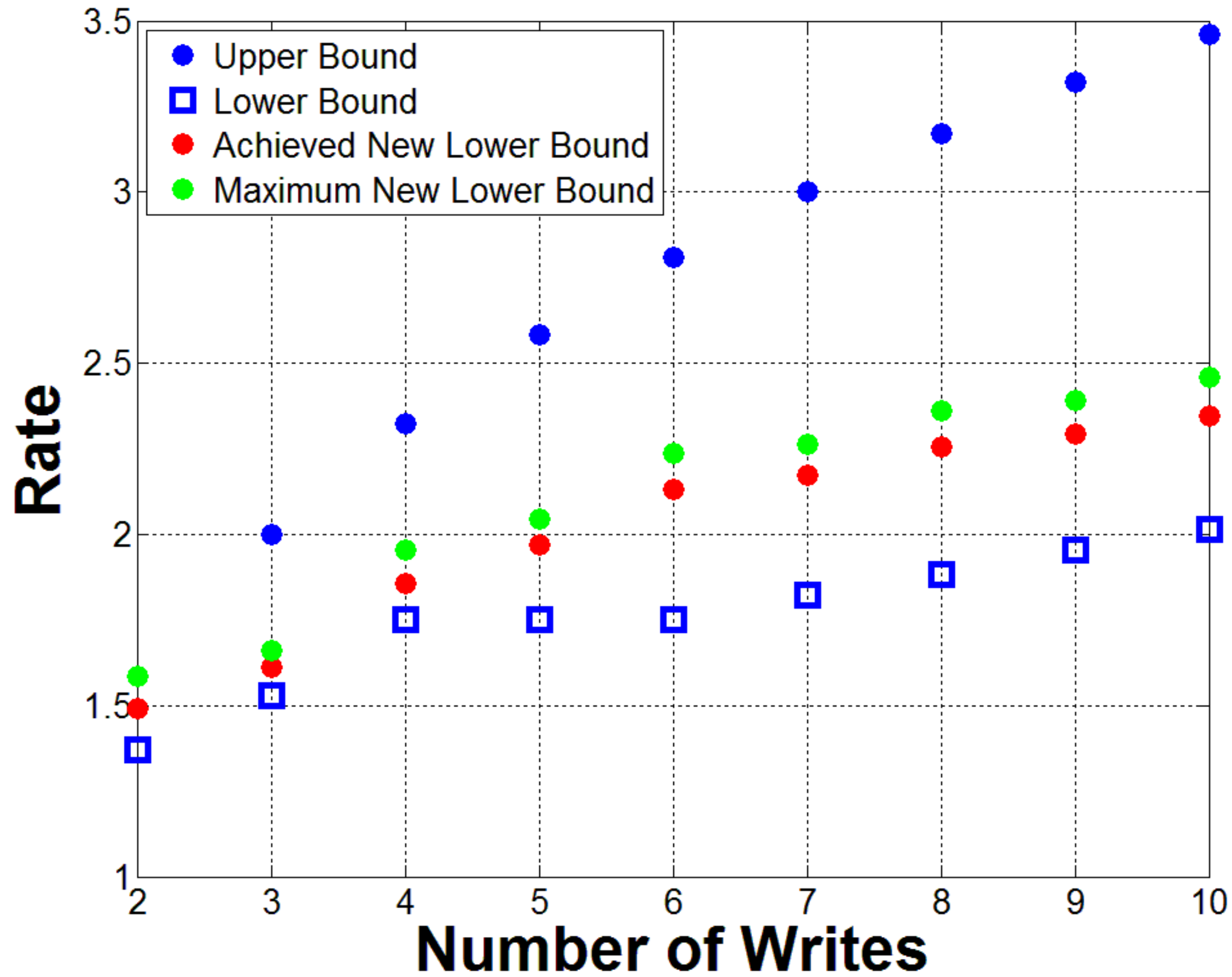
Capacity and Constructions

- **Capacity region** (Heegard '86, Fu and Han Vinck '99)

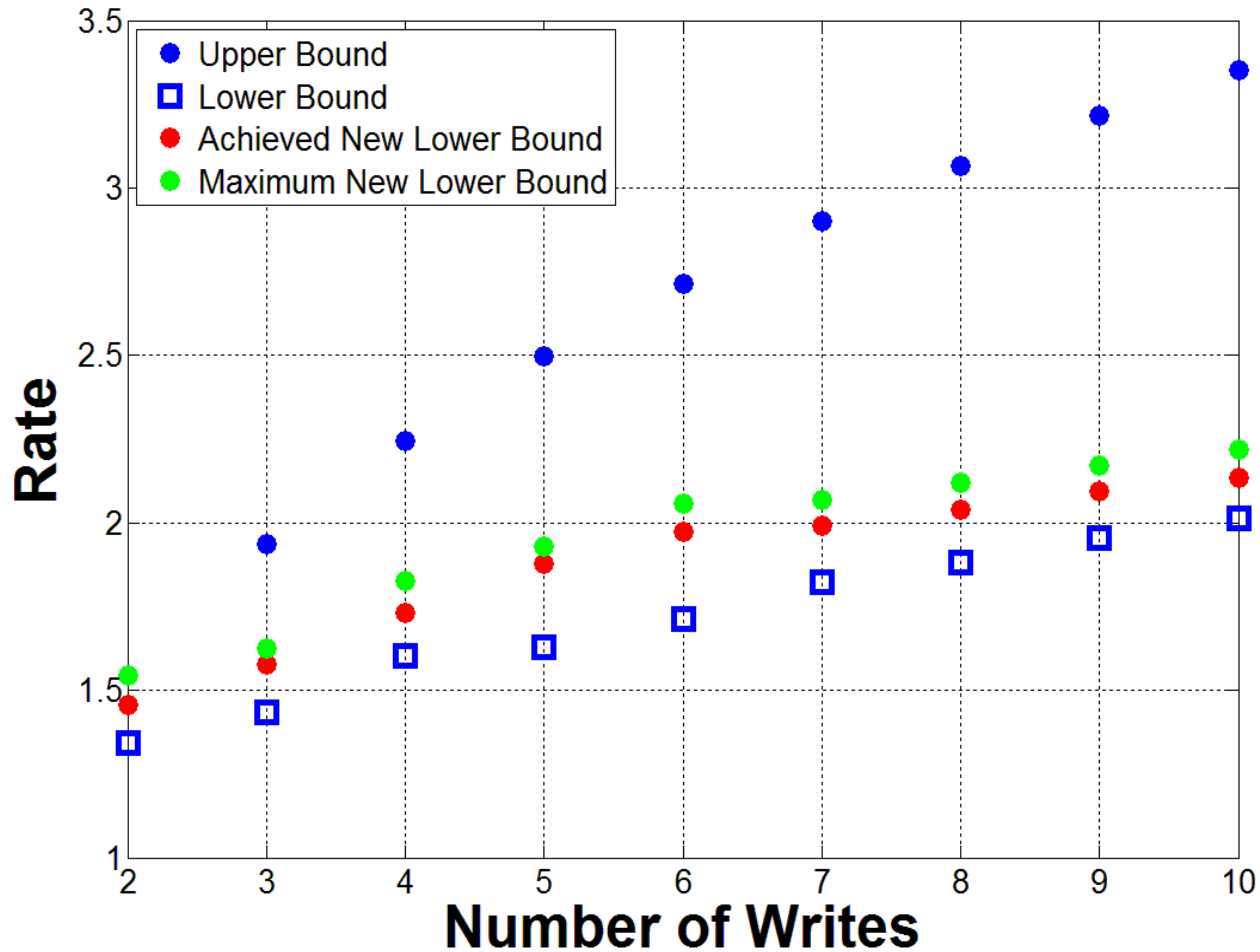
$$C_{t\text{-}WOM}=\{(R_1,\dots,R_t) \mid \begin{array}{l} R_1 \leq h(p_1), \\ R_2 \leq (1-p_1)h(p_2),\dots, \\ R_{t-1} \leq (1-p_1)\cdots(1-p_{t-2})h(p_{t-1}) \\ R_t \leq (1-p_1)\cdots(1-p_{t-2})(1-p_{t-1}) \end{array}\}$$

- Maximum achievable sum-rate is $\log(t+1)$
- **Constructions**:
  - Rivest, Shamir '82
  - Wolf, Wyner, Ziv, Korner '84
  - Merkx '84
  - Cohen, Godlewski, and Merkx '86
  - Wu and Jiang '09
  - Wu '10
  - Yaakobi, Kayser, Siegel, Vardy, Wolf '10
  - Kayser, Yaakobi, Siegel, Vardy, Wolf '10

Results: Unrestricted-rate



Results: Fixed-rate



Recent Results

- **Shpilka**, "*New constructions of WOM codes using the Wozencraft ensemble*", '12
  - Capacity achieving construction
  - 3-write WOM codes of sum-rate 1.81
- **Burshtein, Strugatski**, "*Polar write once memory codes*", '12
- **Yaakobi**, **Shpilka**, "*High sum-rate three-write and non-binary WOM codes*", '12
  - 3-write WOM codes of sum-rate 1.88
- **Shpilka**, "*Capacity Achieving Multiwrite WOM Codes*", '12
- **The Challenge**: Constructing WOM codes with high sum-rate and low encoding/decoding complexities

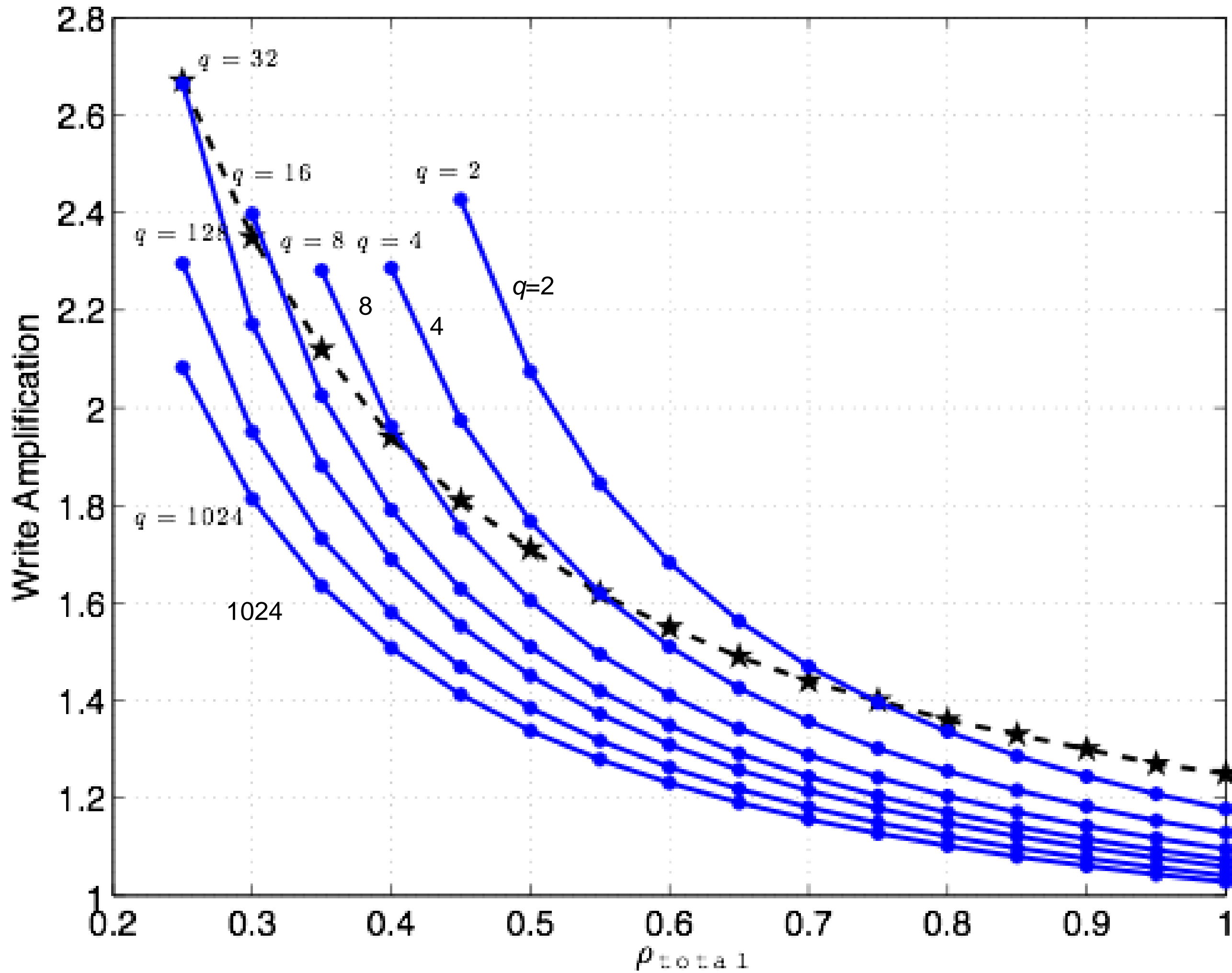
Why/When to Use WOM Codes?

- **Disadvantage**: sacrifice a large amount of the capacity
  - **Ex**: Two write WOM codes
    - The best sum-rate is **log3≈1.58**
    - Can write (at most) only **0.79n** bits so there is a lost of (at least) **21%** of the capacity
- **Advantage**: Can increase the lifetime of the memory and reduce the write amplification

Why/When to Use WOM Codes?

- **Advantage**: Can increase the lifetime of the memory and reduce the write amplification
- **Example:**
  - User has **3GB** of flash with lifetime **100 P/E**
  - Each day the user writes **2GB** of new data (no need to store the old data)
  - Without WOM, the memory lasts **3/2*100=150** days
  - With WOM (the Rivest Shamir scheme)
    every two days the memory is erased once
    the memory lasts **2*100=200** days
  - Can improve if there is dependency between the data written on every day

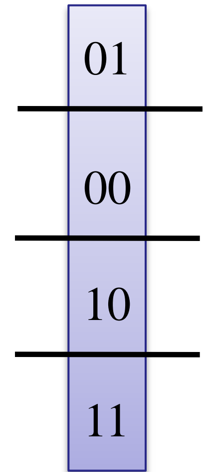
Write Amplification for t=2 WOM Codes



Write amplification decreases for increasing q

Non-Binary WOM

- Many constructions
  - Huang, Lin, and Abdel-Ghaffar '10
  - Gabrys and Dolecek '11
  - Jiang, Zhou, Bruck '11
  - Gabrys, Yaakobi, Dolecek, Siegel, Vardy, and Wolf '11
  - Kurkoski '11, Kurkosi '12
  - Haymaker, Kelley '12
  - Burshtein, Strugatski '12
  - Cassuto, Yaakobi '12
  - Yaakobi, Shpilka '12
  - Bhatia, Iyengar, Siegel '12
- Might be harder to implement in real flash devices



Thanks

Aman Bhatia	Andrew Jiang	Paul Siegel
Brian Butler	Scott Kayser	Steven Swanson
Yuval Cassuto	Young-Han Kim	Alexander Vardy
Lara Dolecek	Brian Kurkoski	Lele Wang
Ryan Gabrys	Jing Ma	Jack Wolf
Laura Grupp	Minghai Qin	Luojie Xiang
Aravind Iyengar	Amir Shpilka	Xiaojie Zhang

Signal Processing and Coding for Non-Volatile Memories

Part III: Emerging Coding Methods

Anxiao (Andrew) Jiang

Department of Computer Science and Engineering
Texas A&M University

Tutorial at Non-Volatile Memories Workshop (NVMW), March 3, 2013
Joint Presentation with Eitan Yaakobi and Jason Bellorado

Outline of this talk

We will learn about

- Joint rewriting and error correction scheme,

Outline of this talk

We will learn about

- Joint rewriting and error correction scheme,
- Rank modulation scheme,

Outline of this talk

We will learn about

- Joint rewriting and error correction scheme,
- Rank modulation scheme,
- Variable-level cell scheme,

Outline of this talk

We will learn about

- Joint rewriting and error correction scheme,
- Rank modulation scheme,
- Variable-level cell scheme,
- Summary and future directions.

Joint rewriting and error correction scheme

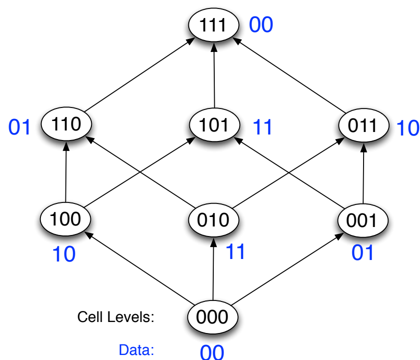
Review: Basic Problem for Write-Once Memory

Let us recall the basic question for Write-Once Memory (WOM):

- Suppose you have n binary cells. Every cell can change its value only from 0 to 1, not from 1 to 0.
  How can you write data, and then rewrite, rewrite, rewrite $\cdots$ the data?

Review: Write Once Memory (WOM) [1]

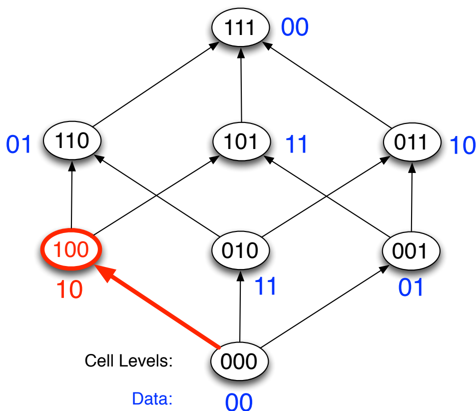
Example: Store 2 bits in 3 SLCs. Write the 2-bit data twice.



[1] R. L. Rivest and A. Shamir, "How to reuse a 'write-once' memory," in Information and Control, vol. 55, pp. 1-19, 1982.

Review: Write Once Memory (WOM)

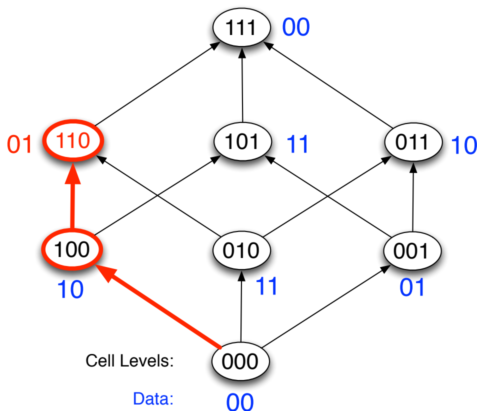
Example: Store 2 bits in 3 SLCs. Write the 2-bit data twice.



1st write: 10

Review: Write Once Memory (WOM)

Example: Store 2 bits in 3 SLCs. Write the 2-bit data twice.

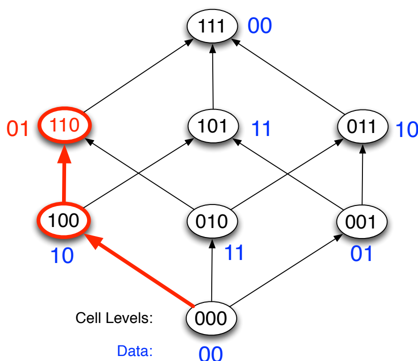


1st write: 10

2nd write: 01

Review: Write Once Memory (WOM)

Example: Store 2 bits in 3 SLCs. Write the 2-bit data twice.



1st write: 10

2nd write: 01

Sum rate: $\frac{2}{3} + \frac{2}{3} = 1.33$

Review: Write-Once Memory Code

This kind of code is called Write-Once Memory (WOM) code.

It is potentially a powerful technology for Flash Memories.

Review: Capacity of WOM [1][2]

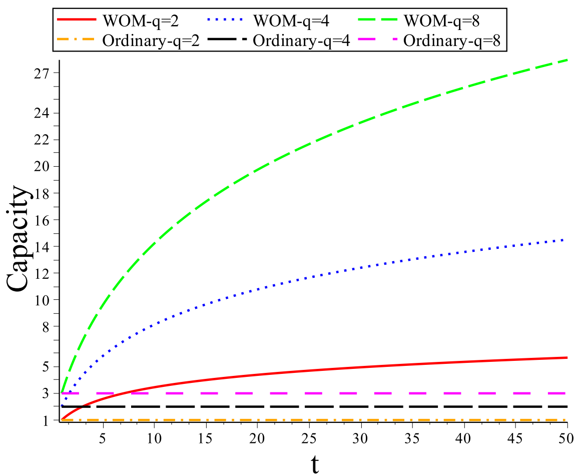
For WOM of q-level cells and t rewrites, the capacity (maximum achievable sum rate) is

$$\log_2 \binom{t+q-1}{q-1}.$$

bits per cell.

[1] C. Heegard, On the capacity of permanent memory, in *IEEE Trans. Information Theory*, vol. IT-31, pp. 34-42, 1985.
[2] F. Fu and A. J. Han Vinck, On the capacity of generalized write-once memory with state transitions described by an arbitrary directed acyclic graph, in *IEEE Trans. Information Theory*, vol. 45, no. 1, pp. 308-313, 1999.

Review: Capacity of WOM



Recent Developments

How to design good WOM codes?

Two capacity-achieving codes were published in 2012 – the same year!:

- A. Shpilka, Capacity achieving multiwrite WOM codes, 2012.
- D. Burshtein and A. Strugatski, **Polar write once memory codes**, 2012.

Two Parameters: α and ϵ

For a t-write WOM code, consider one of its t writes.

There are two important parameters for this write:

- α: The fraction of cells that are 0 before this write.
- ϵ: For the cells of level 0 before this write, ϵ is the fraction of them that are changed to 1 in this write.

For t-write WOM codes, the optimal values of α and ϵ are known for each of the t writes.

Polar WOM Code [1]

Idea of Burshtein and Strugatski: See a write as the decoding of a polar code:

- See the cells' state **BEFORE** the write as a noisy Polar codeword.
- See the cells' state **AFTER** the write as the correct (i.e., error-free) Polar codeword.

More precisely, they see the write as lossy data compression, using the method presented by Korada and Urbanke [2].

[1] D. Burshtein and A. Strugatski, Polar Write Once Memory Codes, in *Proc. ISIT*, 2012.

[2] S. Korada and R. Urbanke, Polar Codes Are Optimal For Lossy Source Coding, in *IEEE Transactions on Information Theory*, vol. 56, no. 4, pp. 1751–1768, 2010.

Polar WOM Code

Smart Idea by Burshtein and Strugatski:

1. Add dither to cell:
   - Let $s \in \{0, 1\}$ be the level of a cell.
   - Let $g \in \{0, 1\}$ be a pseudo-random number known to the encoder and decoder.
   - Let $v = s \oplus g$ be called the **value** of the cell.
2. Build a test channel for the write, which we shall call the WOM channel:

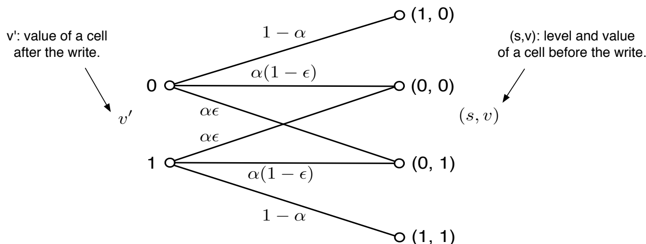
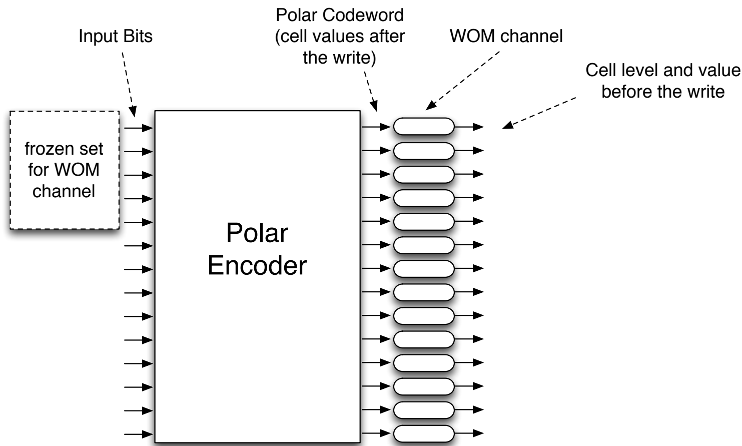
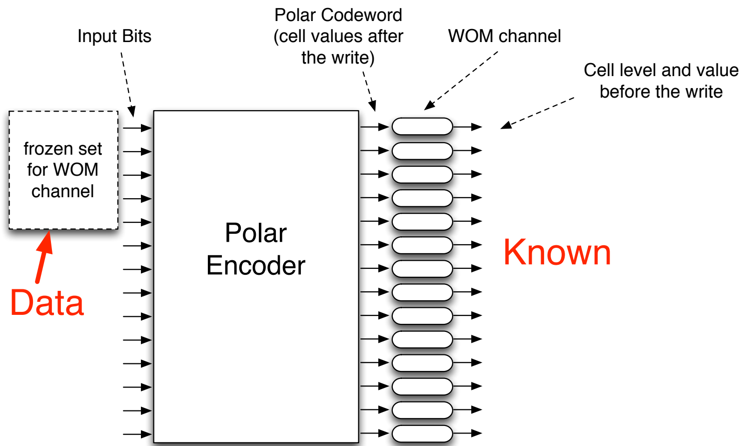


Fig. 1. The WOM channel $WOM(\alpha, \epsilon)$.

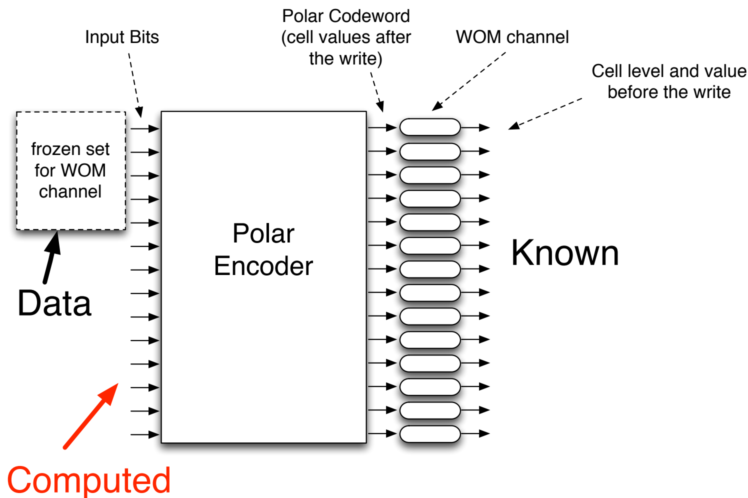
Polar WOM Code: Process of A Write: Encode



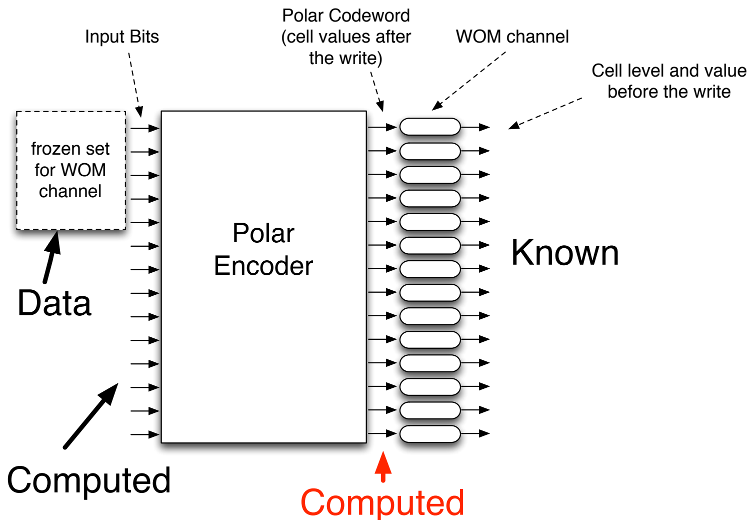
Polar WOM Code: Process of A Write: Encode



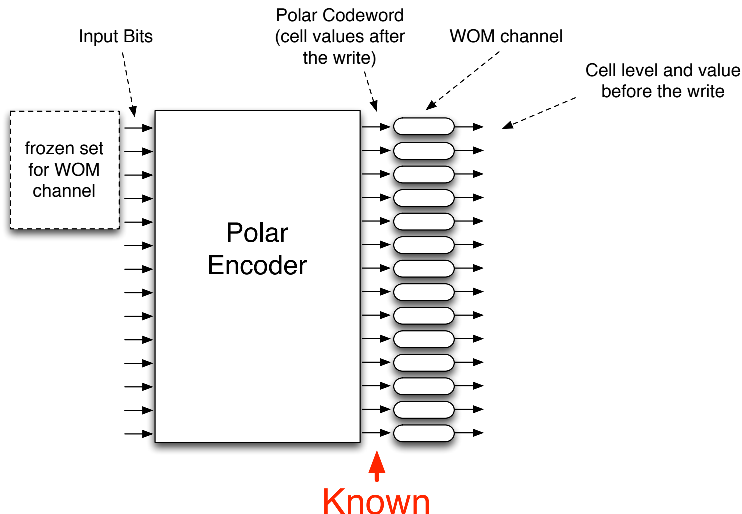
Polar WOM Code: Process of A Write: Encode



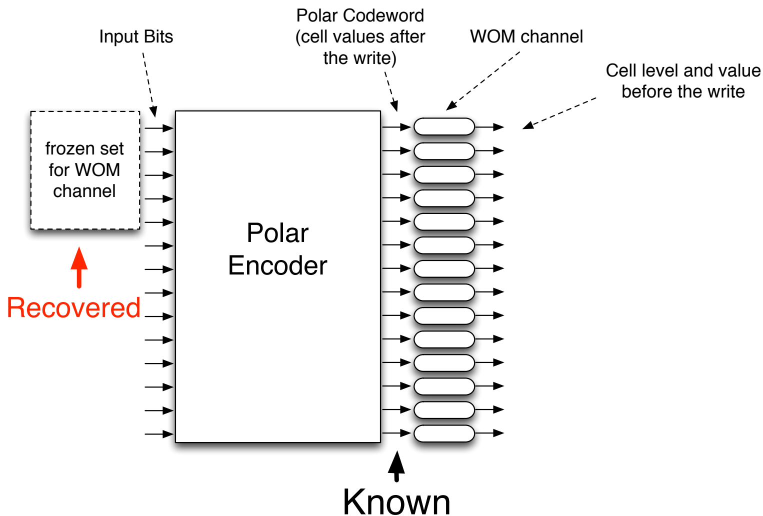
Polar WOM Code: Process of A Write: Encode



Polar WOM Code: Process of A Write: Decode



Polar WOM Code: Process of A Write: Decode



For **Rewriting** to be used in flash memories, it is **CRITICAL** to combine it with **Error-Correcting Codes**.

Some Codes for Joint Rewriting and Error Correction

Previous results are for correcting a few (up to 3) errors:

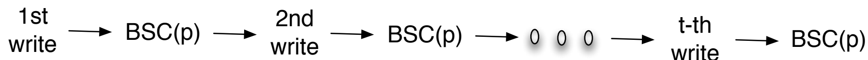
- G. Zemor and G. D. Cohen, Error-Correcting WOM-Codes, in *IEEE Transactions on Information Theory*, vol. 37, no. 3, pp. 730–734, 1991.
- E. Yaakobi, P. Siegel, A. Vardy, and J. Wolf, Multiple Error-Correcting WOM-Codes, in *IEEE Transactions on Information Theory*, vol. 58, no. 4, pp. 2220–2230, 2012.

New Code for Joint Rewriting and Error Correction

We now present a joint coding scheme for rewriting and error correction, which can correct a substantial number of errors and supports any number of rewrites.

- A. Jiang, Y. Li, E. En Gad, M. Langberg, and J. Bruck, Joint Rewriting and Error Correction in Write-Once Memories, 2013.

Model of Rewriting and Noise



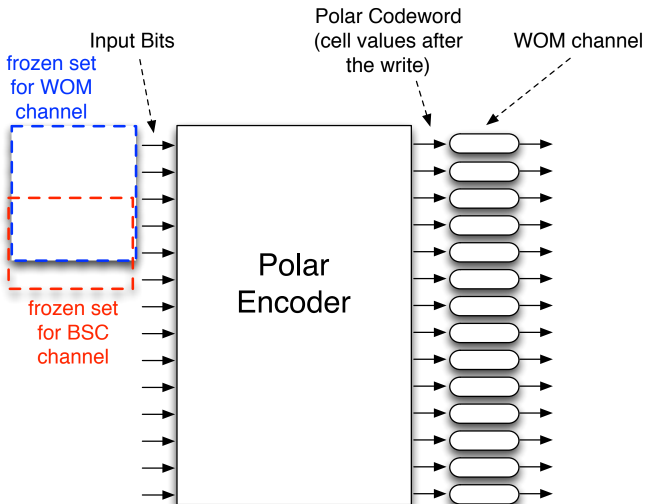
Two Channels

Consider one write.

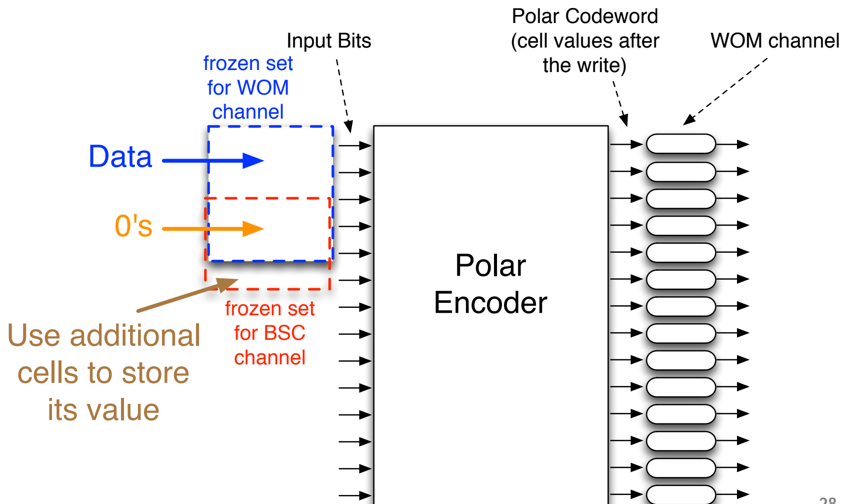
Consider two channels:

1. **WOM channel**. Let its frozen set be $F_{WOM(\alpha,\epsilon)}$.
2. **BSC channel**. Let its frozen set be $F_{BSC(p)}$.

General Coding Scheme



General Coding Scheme



Rate of the Code

Analyze the rate of a single write step:

- Let $N \to \infty$ be the size of the polar code.
- The size of $F_{WOM(\alpha,\epsilon)}$ (the frozen set for the WOM channel) is $\alpha H(\epsilon) N$.
- The size of $F_{BSC(p)}$ (the frozen set for the BSC) is $H(p)N$.
- The number of bits in the written data is $|F_{WOM(\alpha,\epsilon)} - F_{BSC(p)}|$.
- The number of additional cells we use to store the value in $F_{BSC(p)} - F_{WOM(\alpha,\epsilon)}$ is $\frac{|F_{BSC(p)} - F_{WOM(\alpha,\epsilon)}|}{1 - H(p)}$.
- For $i = 1, 2, \cdots, t$, let M_i be the number of bits written in the ith write, and let $N_{additional,i}$ be the number of additional cells we use to store the value in $F_{BSC(p)} - F_{WOM(\alpha,\epsilon)}$ in the ith write. Then the sum-rate is

$$R_{sum} = \frac{\sum_{i=1}^{t} M_i}{N + \sum_{i=1}^{t} N_{additional,i}}.$$

When is $F_{BSC(p)}$ a subset of $F_{WOM(\alpha,\epsilon)}$?

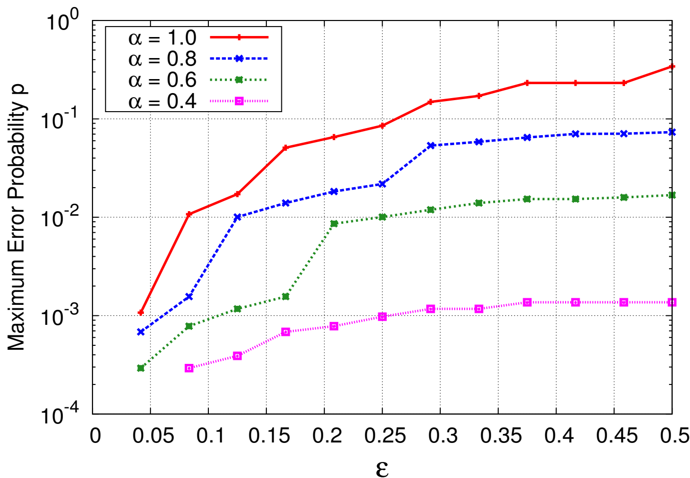


Fig. 8. The maximum value of p found for which $F_{\mathrm{BSC}(p)} \subseteq F_{\mathrm{WOM}(\alpha,\epsilon)}$.

Theoretical Analysis

It is interesting to know how much $F_{WOM(\alpha,\epsilon)}$ and $F_{BSC(p)}$ intersects.

Degrading WOM Channel to BSC

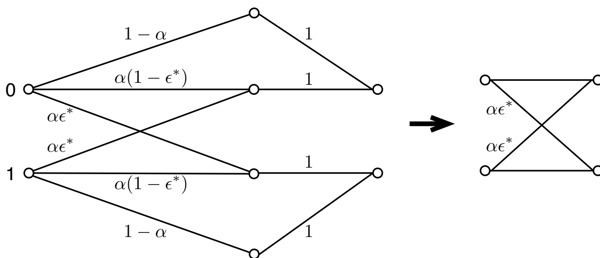


Fig. 3. Degrading the channel $WOM(\alpha, \epsilon^*)$ to $BSC(\alpha\epsilon^*)$. The two channels on the left and on the right are equivalent.

Degrading WOM Channel to Another WOM Channel

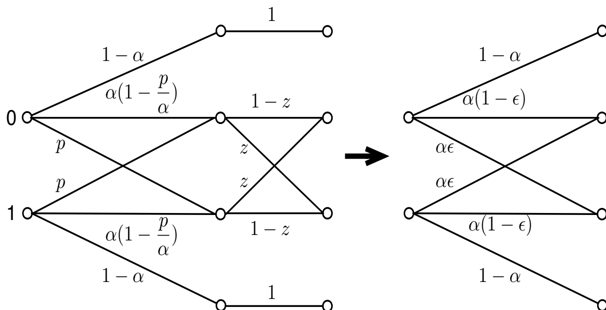


Fig. 4. Degrading channel $WOM(\alpha, \frac{p}{\alpha})$ to $WOM(\alpha, \epsilon)$. Here $z = \frac{\alpha\epsilon - p}{\alpha - 2p}$. The two channels on the left and on the right are equivalent.

Common Upgrading/Degrading of WOM-channel and BSC

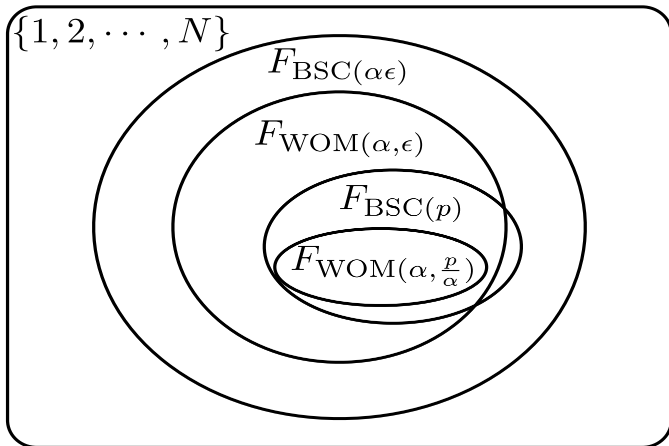
Lemma 2. *When* $p \leq \alpha\epsilon$,

$$F_{\mathrm{WOM}(\alpha,\frac{p}{\alpha})} \subseteq \left(F_{\mathrm{BSC}(p)} \cap F_{\mathrm{WOM}(\alpha,\epsilon)}\right),$$

and

$$\left(F_{\mathrm{WOM}(\alpha,\epsilon)} \cup F_{\mathrm{BSC}(p)}\right) \subseteq F_{\mathrm{BSC}(\alpha\epsilon)}.$$

Common Upgrading/Degrading of WOM-channel and BSC



Lower Bound to Achievable Sum-Rate

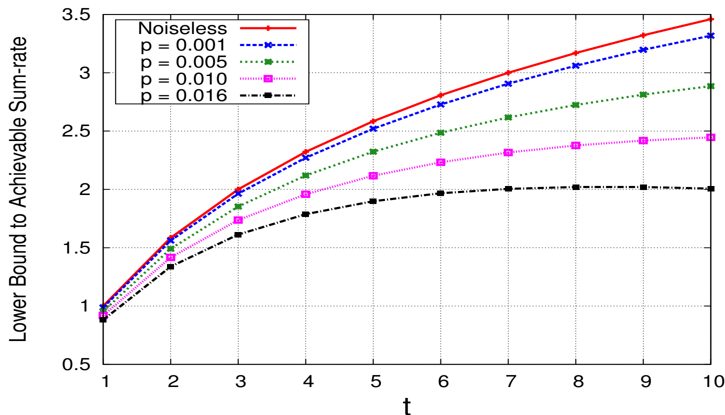


Fig. 6. Lower bound to achievable sum-rates for different error probability p.

Rank Modulation

Definition of Rank Modulation [1-2]

Rank Modulation:

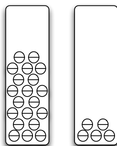
We use the relative order of cell levels (instead of their absolute values) to represent data.

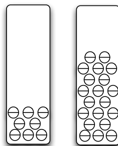
[1] A. Jiang, R. Mateescu, M. Schwartz and J. Bruck, "Rank Modulation for Flash Memories," in *Proc. IEEE International Symposium on Information Theory (ISIT)*, pp. 1731–1735, July 2008.

[2] A. Jiang, M. Schwartz and J. Bruck, "Error-Correcting Codes for Rank Modulation," in *Proc. IEEE International Symposium on Information Theory (ISIT)*, pp. 1736–1740, July 2008.

Examples and Extensions of Rank Modulation

- Example: Use 2 cells to store 1 bit.

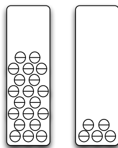




Examples and Extensions of Rank Modulation

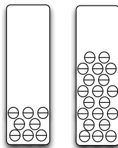
- Example: Use 2 cells to store 1 bit.

Relative order: (1,2)
Value of data: 0



cell 1 cell 2

Relative order: (2,1)
Value of data: 1

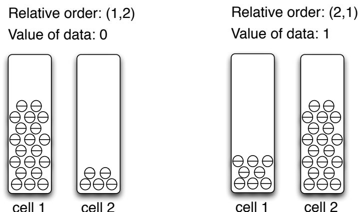


cell 1 cell 2

- Example: Use 3 cells to store $\log_2 6$ bits. The relative orders $(1, 2, 3), (1, 3, 2), \cdots, (3, 2, 1)$ are mapped to data $0, 1, \cdots, 5$.

Examples and Extensions of Rank Modulation

- Example: Use 2 cells to store 1 bit.



- Example: Use 3 cells to store $\log_2 6$ bits. The relative orders $(1,2,3),(1,3,2),\cdots,(3,2,1)$ are mapped to data $0,1,\cdots,5$.
- In general, k cells can represent $\log_2(k!)$ bits.

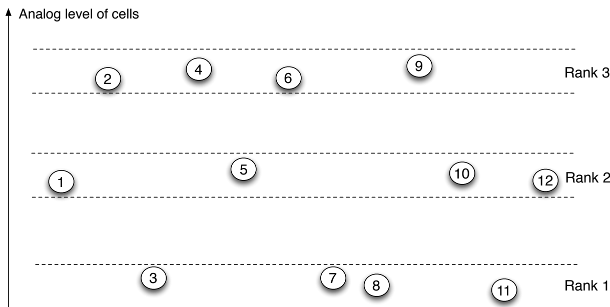
Rank Modulation using Multi-set Permutation

Extension: Let each rank have m cells.

Example

Let $m = 4$. The following is a multi-set permutation

$$(\{2,4,6,9\},\{1,5,10,12\},\{3,7,8,11\}).$$



Error-Correcting Codes for Rank Modulation

Error Correcting Codes for Rank Modulation

Error Models and Distance between Permutations

Based on the error model, there are various reasonable choices for the distance between permutations:

- Kendall-tau distance. (To be introduced in detail.)
- L_∞ distance.
- Gaussian noise based distance.
- Distance defined based on asymmetric errors or inter-cell interference.

We should choose the distance appropriately based on the type and magnitude of errors.

Kendall-tau Distance for Rank Modulation ECC [1]

When errors happen, the smallest change in a permutation is the local exchange of two adjacent numbers in the permutation. That is,

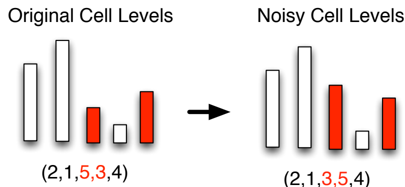
$$(a_1, \cdots, a_{i-1}, \underbrace{a_i, a_{i+1}}_{\text{adjacent pair}}, a_{i+2}, \cdots, a_n) \rightarrow (a_1, \cdots, a_{i-1}, \underbrace{a_{i+1}, a_i}_{\text{adjacent pair}}, a_{i+2}, \cdots, a_n)$$

Kendall-tau Distance for Rank Modulation ECC [1]

When errors happen, the smallest change in a permutation is the local exchange of two adjacent numbers in the permutation. That is,

$$(a_1, \cdots, a_{i-1}, \underbrace{a_i, a_{i+1}}_{\text{adjacent pair}}, a_{i+2}, \cdots, a_n) \rightarrow (a_1, \cdots, a_{i-1}, \underbrace{a_{i+1}, a_i}_{\text{adjacent pair}}, a_{i+2}, \cdots, a_n)$$

Example:

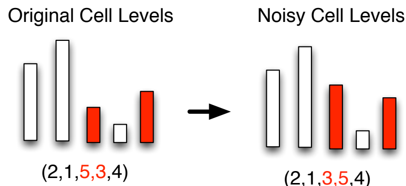


Kendall-tau Distance for Rank Modulation ECC [1]

When errors happen, the smallest change in a permutation is the local exchange of two adjacent numbers in the permutation. That is,

$$(a_1, \cdots, a_{i-1}, \underbrace{a_i, a_{i+1}}_{\text{adjacent pair}}, a_{i+2}, \cdots, a_n) \rightarrow (a_1, \cdots, a_{i-1}, \underbrace{a_{i+1}, a_i}_{\text{adjacent pair}}, a_{i+2}, \cdots, a_n)$$

Example:



We can extend the concept to multiple such "local exchanges" (for larger errors).

[1] A. Jiang, M. Schwartz and J. Bruck, "Error-Correcting Codes for Rank Modulation," in *Proc. IEEE* International Symposium on Information Theory (ISIT), pp. 1736–1740, July 2008.

Kendall-tau Distance for Rank Modulation ECC

Definition (Adjacent Transposition)

An adjacent transposition is the local exchange of two neighboring numbers in a permutation, namely,

$$(a_1, \cdots, a_{i-1}, a_i, a_{i+1}, a_{i+2}, \cdots, a_n) \rightarrow (a_1, \cdots, a_{i-1}, a_{i+1}, a_i, a_{i+2}, \cdots, a_n)$$

Kendall-tau Distance for Rank Modulation ECC

Definition (Adjacent Transposition)

An adjacent transposition is the local exchange of two neighboring numbers in a permutation, namely,

$$(a_1, \cdots, a_{i-1}, a_i, a_{i+1}, a_{i+2}, \cdots, a_n) \to (a_1, \cdots, a_{i-1}, a_{i+1}, a_i, a_{i+2}, \cdots, a_n)$$

Definition (Kendall-tau Distance)

Given two permutations A and B, the Kendall-tau distance between them, $d_\tau(A, B)$, is the minimum number of adjacent transpositions needed to change A into B. (Note that $d_\tau(A, B) = d_\tau(B, A)$.)

Kendall-tau Distance for Rank Modulation ECC

Definition (Adjacent Transposition)

An adjacent transposition is the local exchange of two neighboring numbers in a permutation, namely,

$$(a_1, \cdots, a_{i-1}, a_i, a_{i+1}, a_{i+2}, \cdots, a_n) \rightarrow (a_1, \cdots, a_{i-1}, a_{i+1}, a_i, a_{i+2}, \cdots, a_n)$$

Definition (Kendall-tau Distance)

Given two permutations A and B, the Kendall-tau distance between them, $d_\tau(A, B)$, is the minimum number of adjacent transpositions needed to change A into B. (Note that $d_\tau(A, B) = d_\tau(B, A)$.)

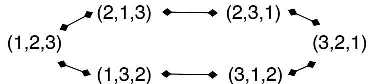
If the minimum Kendall-tau distance of a code is 2t+1, then it can correct t adjacent transposition errors.

Kendall-tau Distance for Rank Modulation ECC

Definition (State Diagram)

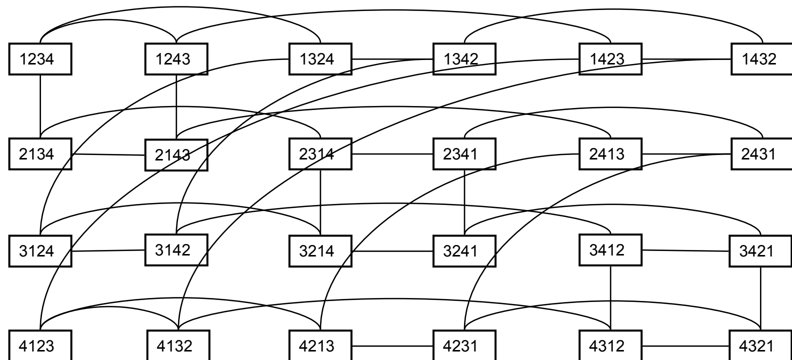
Vertices are permutations. There is an undirected edge between two permutations $A, B \in S_n$ iff $d_\tau(A, B) = 1$.

Example: The state diagram for $n = 3$ cells is



Kendall-tau Distance for Rank Modulation ECC

Example: The state diagram for $n = 4$ cells is



One-Error-Correcting Code

We introduce an error-correcting code of minimum Kendall-tau distance 3, which corrects one Kendall (i.e., adjacent transposition) error.

One-Error-Correcting Code

We introduce an error-correcting code of minimum Kendall-tau distance 3, which corrects one Kendall (i.e., adjacent transposition) error.

Definition (Inversion Vector)

Given a permutation $(a_1, a_2, \cdots, a_n)$, its inversion vector $(x_1, x_2, \cdots, x_{n-1}) \in \{0,1\} \times \{0,1,2\} \times \cdots \times \{0,1,\cdots,n-1\}$ is determined as follows:

- For $i = 1, 2, \cdots, n-1$, x_i is the number of elements in $\{1, 2, \cdots, i\}$ that are behind $i+1$ in the permutation $(a_1, \cdots, a_n)$.

One-Error-Correcting Code

We introduce an error-correcting code of minimum Kendall-tau distance 3, which corrects one Kendall (i.e., adjacent transposition) error.

Definition (Inversion Vector)

Given a permutation $(a_1, a_2, \cdots, a_n)$, its inversion vector $(x_1, x_2, \cdots, x_{n-1}) \in \{0,1\} \times \{0,1,2\} \times \cdots \times \{0,1,\cdots,n-1\}$ is determined as follows:

- For $i = 1, 2, \cdots, n-1$, x_i is the number of elements in $\{1, 2, \cdots, i\}$ that are behind $i+1$ in the permutation $(a_1, \cdots, a_n)$.

Example: The inversion vector for $(1,2,3,4)$ is $(0,0,0)$. The inversion for $(4,3,2,1)$ is $(1,2,3)$. The inversion vector for $(2,4,3,1)$ is $(1,1,2)$.

One-Error-Correcting Code [1]

By viewing the inversion vector as coordinates, we embed permutations in an $(n-1)$-dimensional space.

One-Error-Correcting Code [1]

By viewing the inversion vector as coordinates, we embed permutations in an $(n-1)$-dimensional space.

Fact: For any two permutations $A, B \in S_n$, $d_\tau(A, B)$ is no less than their L_1 distance in the $(n-1)$-dimensional space.

One-Error-Correcting Code [1]

By viewing the inversion vector as coordinates, we embed permutations in an $(n-1)$-dimensional space.

Fact: For any two permutations $A, B \in S_n$, $d_\tau(A, B)$ is no less than their L_1 distance in the $(n-1)$-dimensional space.

Idea: We can construct a code of minimum L_1 distance D in the $(n-1)$-dimensional array of size $2 \times 3 \times \cdots \times n$. Then it is a code of Kendall-tau distance at least D for the permutations.

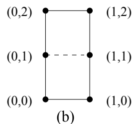
[1] A. Jiang, M. Schwartz and J. Bruck, "Error-Correcting Codes for Rank Modulation," in *Proc. IEEE* International Symposium on Information Theory (ISIT), pp. 1736–1740, July 2008.

One-Error-Correcting Code

Example: When $n = 3$ or $n = 4$, the embedding is as follows. (Only the solid edges are the edges in the state graph of permutations.)

Permutation		Coordinates
1 2 3	→	(0,0)
1 3 2	→	(0,1)
2 1 3	→	(1,0)
2 3 1	→	(1,1)
3 1 2	→	(0,2)
3 2 1	→	(1,2)

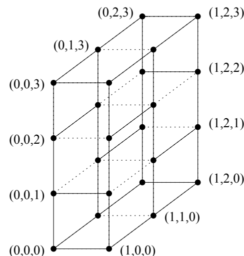
(a)



(b)

Permutation		Coordinates	Permutation		Coordinates
1 2 3 4	→	(0,0,0)	3 1 2 4	→	(0,2,0)
1 2 4 3	→	(0,0,1)	3 1 4 2	→	(0,2,1)
1 3 2 4	→	(0,1,0)	3 2 1 4	→	(1,2,0)
1 3 4 2	→	(0,1,1)	3 2 4 1	→	(1,2,1)
1 4 2 3	→	(0,0,2)	3 4 1 2	→	(0,2,2)
1 4 3 2	→	(0,1,2)	3 4 2 1	→	(1,2,2)
2 1 3 4	→	(1,0,0)	4 1 2 3	→	(0,0,3)
2 1 4 3	→	(1,0,1)	4 1 3 2	→	(0,1,3)
2 3 1 4	→	(1,1,0)	4 2 1 3	→	(1,0,3)
2 3 4 1	→	(1,1,1)	4 2 3 1	→	(1,1,3)
2 4 1 3	→	(1,0,2)	4 3 1 2	→	(0,2,3)
2 4 3 1	→	(1,1,2)	4 3 2 1	→	(1,2,3)

(c)



(d)

One-Error-Correcting Code

Construction (One-Error-Correcting Rank Modulation Code)

Let $C_1, C_2 \subseteq S_n$ denote two rank modulation codes constructed as follows. Let $A \in S_n$ be a general permutation whose inversion vector is $(x_1, x_2, \cdots, x_{n-1})$. Then A is a codeword in C_1 iff the following equation is satisfied:

$$\sum_{i=1}^{n-1} i x_i \equiv 0 \pmod{2n-1}$$

A is a codeword in C_2 iff the following equation is satisfied:

$$\sum_{i=1}^{n-2} i x_i + (n-1) \cdot (-x_{n-1}) \equiv 0 \pmod{2n-1}$$

Between C_1 and C_2, choose the code with more codewords as the final output.

One-Error-Correcting Code

For the above code, it can be proved that:

- The code can correct one Kendall error.
- The size of the code is at least $\frac{(n-1)!}{2}$.
- The size of the code is at least half of optimal.

Codes Correcting More Errors [1]

- The above code can be generalized to correct more errors.

$$\mathcal{C} = \{(x_1, x_2, \cdots, x_{n-1}) \mid \sum_{i=1}^{n-1} h_i x_i \equiv 0 \mod m\}$$

- Let $A(n, d)$ be the maximum number of permutations in S_n with minimum Kendall-tau distance d. We call

$$C(d) = \lim_{n \to \infty} \frac{\ln A(n, d)}{\ln n!}$$

capacity of rank modulation ECC of Kendall-tau distance d.

$$C(d) = \begin{cases} 1 & \text{if } d = O(n) \\ 1 - \epsilon & \text{if } d = \Theta(n^{1+\epsilon}),\ 0 < \epsilon < 1 \\ 0 & \text{if } d = \Theta(n^2) \end{cases}$$

[1] A. Barg and A. Mazumdar, "Codes in Permutations and Error Correction for Rank Modulation," *ISIT'10*.

Variable Level Cell (VLC)

What is the right number of levels?

Performance of SLC, MLC and TLC:

- SLC: 2 levels, endurance of $\sim 10^6$ Program/Erase cycles.
- MLC: 4 levels, endurance of $\sim 10^5$ Program/Erase cycles.
- TLC: 8 levels, endurance of $\sim 10^4$ Program/Erase cycles.

Question: Is there a way to adaptively choose the number of levels, based on the cells' quality and random programming performance?

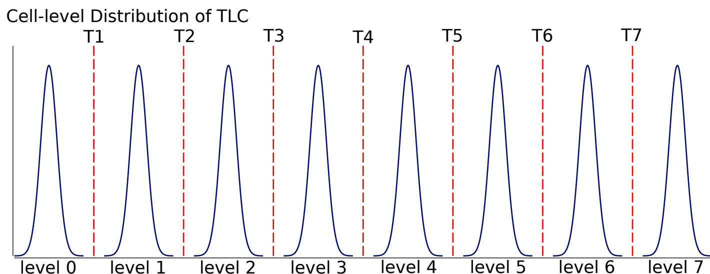
Variable Level Cell (VLC) [1]

Main Idea of VLC:

- Set thresholds dynamically.
- Do not fix the number of levels in advance.

[1] A. Jiang, H. Zhou and J. Bruck, Variable-level cells for nonvolatile memories, in *Proc. ISIT*, pp. 2489-2493, 2011.

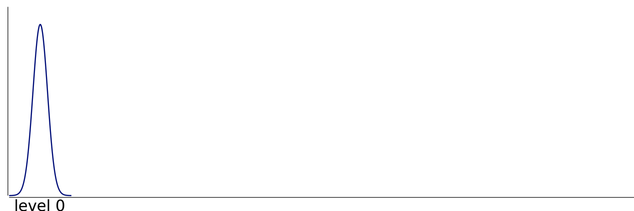
Existing Technology: Fixed Thresholds and Levels



Variable Level Cell (VLC)

Main Idea of VLC:

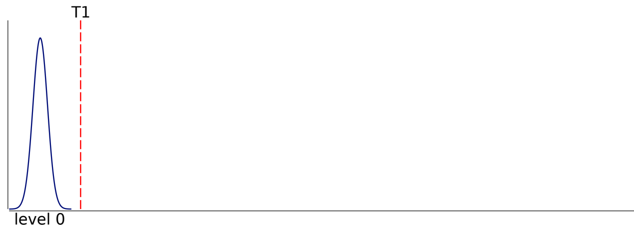
- Set thresholds dynamically.
- Do not fix the number of levels in advance.



Variable Level Cell (VLC)

Main Idea of VLC:

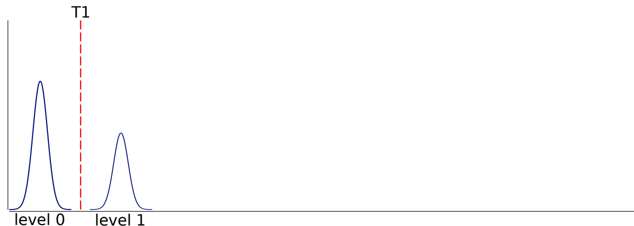
- Set thresholds dynamically.
- Do not fix the number of levels in advance.



Variable Level Cell (VLC)

Main Idea of VLC:

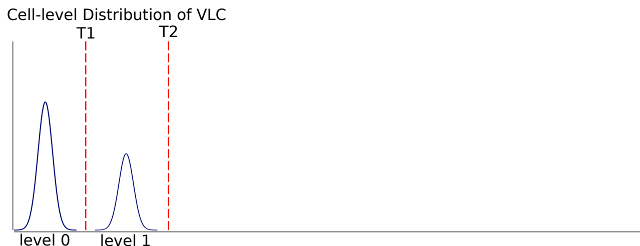
- Set thresholds dynamically.
- Do not fix the number of levels in advance.



Variable Level Cell (VLC)

Main Idea of VLC:

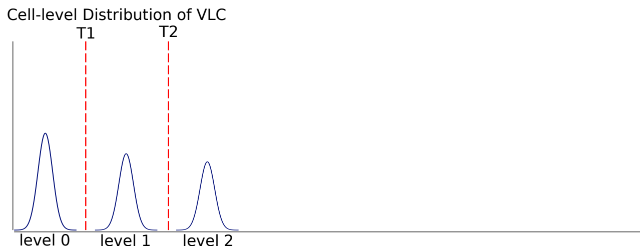
- Set thresholds dynamically.
- Do not fix the number of levels in advance.



Variable Level Cell (VLC)

Main Idea of VLC:

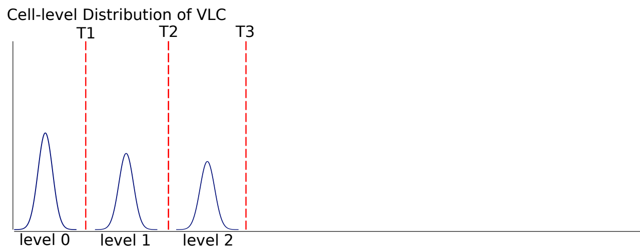
- Set thresholds dynamically.
- Do not fix the number of levels in advance.



Variable Level Cell (VLC)

Main Idea of VLC:

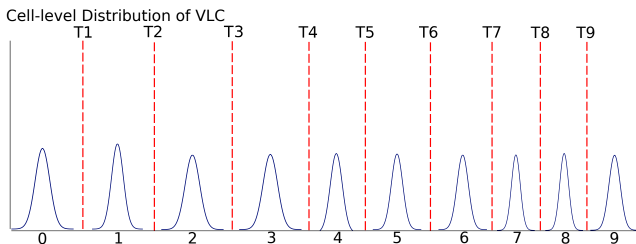
- Set thresholds dynamically.
- Do not fix the number of levels in advance.



Variable Level Cell (VLC)

Main Idea of VLC:

- Set thresholds dynamically.
- Do not fix the number of levels in advance.

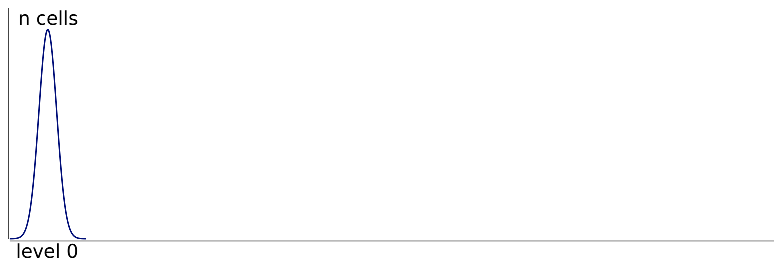


Variable Level Cell (VLC)

- VLC is more adaptive compared to current schemes.
- Programming is more robust to
  - Cell quality degradation/variance;
  - Probabilistic charge injection behavior.
- Multiple levels can be programmed in parallel for higher speed.

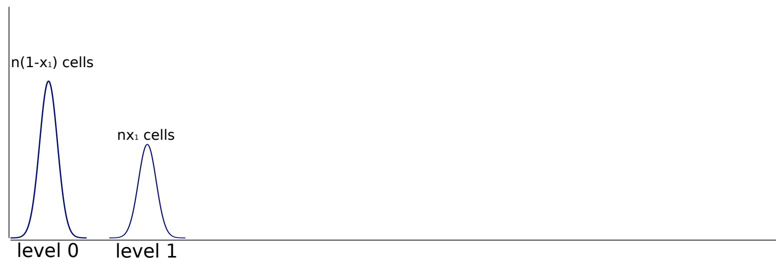
Storing Data in VLC

How to store data? One solution for one-write storage:



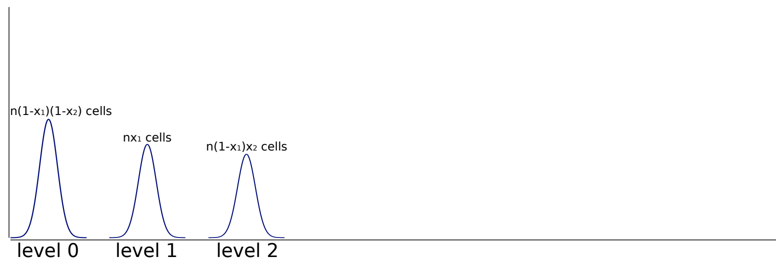
Storing Data in VLC

- Level 1 can store $nH(x_1)$ bits.
- Reading these $nH(x_1)$ bits will require two threshold comparisons.



Storing Data in VLC

- Level 2 can store $n(1 - x_1)H(x_2)$ bits.
- Reading these $n(1 - x_1)H(x_2)$ bits will require one additional threshold comparison.



Capacity of VLC

Assume

- Level 1 can be programmed with probability p_1;
- Level 2 can be programmed with probability $p_1 p_2$;
- Level 3 can be programmed with probability $p_1 p_2 p_3$;
- $\cdots$;
- Level q can be programmed with probability $p_1 p_2 \cdots p_q$, where q is the maximum possible level number.

Capacity of VLC

Define $A_1, A_2, \cdots, A_{q-1}$ recursively:

- Let $A_{q-1} = 2^{p_{q-1}}$;
- For $i = q-2, q-3, \cdots, 1$, let $A_i = (1 + A_{i+1})^{p_i}$.

Theorem

The capacity (expected value) of VLC is

$$C_{VLC} = \log_2 A_1$$

bits per cell.

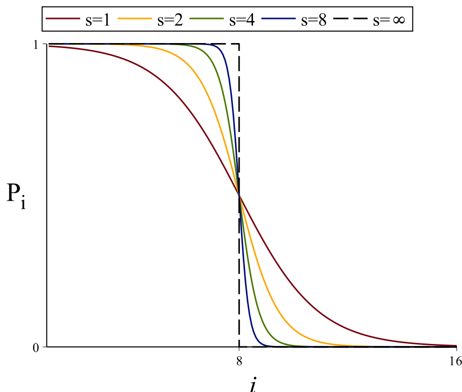
- For the capacity region of rewriting codes, see:

[1] A. Jiang, H. Zhou and J. Bruck, Variable-level cells for nonvolatile memories, in *Proc. ISIT*, pp. 2489-2493, 2011.

Comparison of Capacity between VLC and MLC

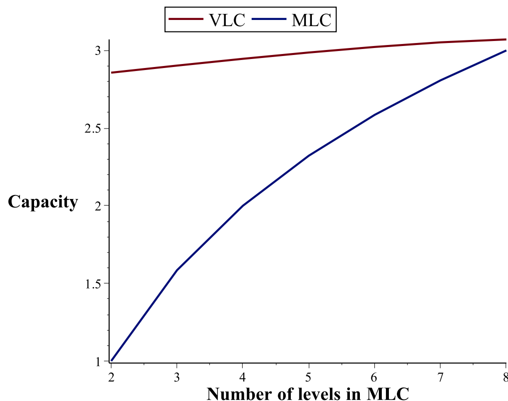
For $i = 1, \cdots, q-1$, let P_i be the probability that level i can be programmed.

Let s be a constant. Let $P_i = \frac{1}{1+2^{(i-8)s}}$ for $i = 1, 2, \cdots, 16$.

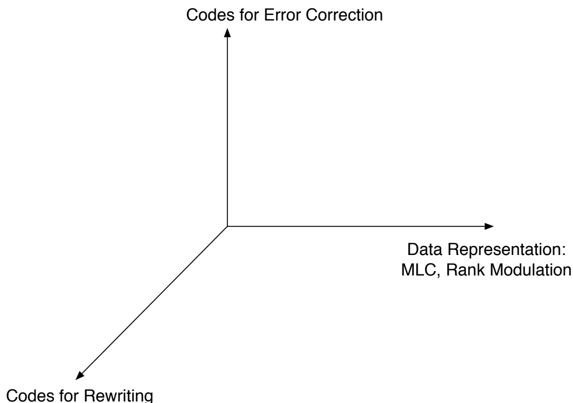


Comparison of Capacity between VLC and MLC

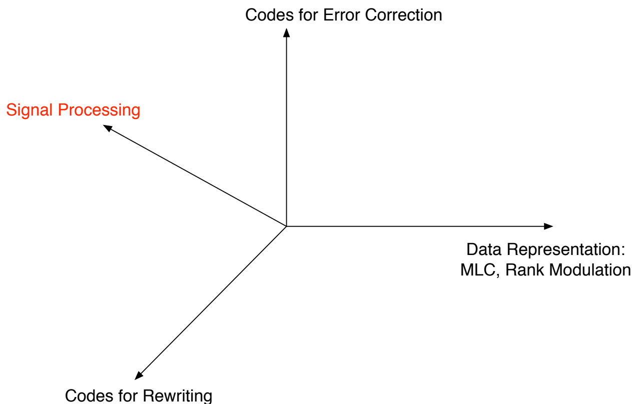
Assume MLC uses levels that can be programmed with probability 0.99 or more.



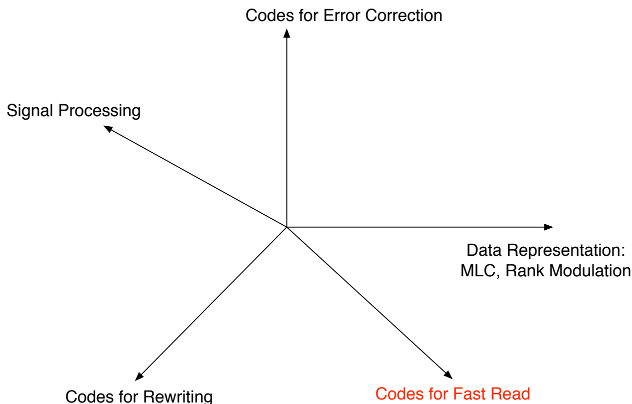
Open Problems on Coding for NVMs



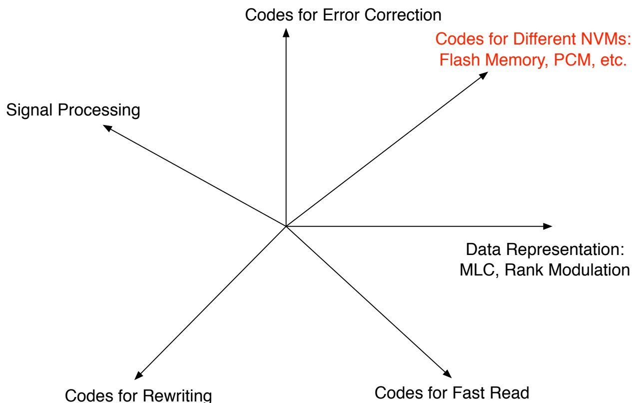
Open Problems on Coding for NVMs



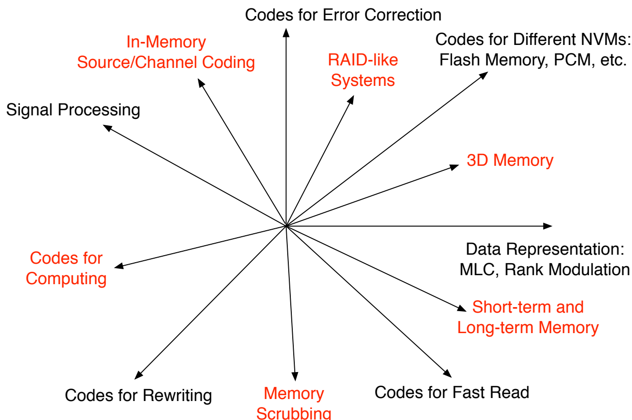
Open Problems on Coding for NVMs



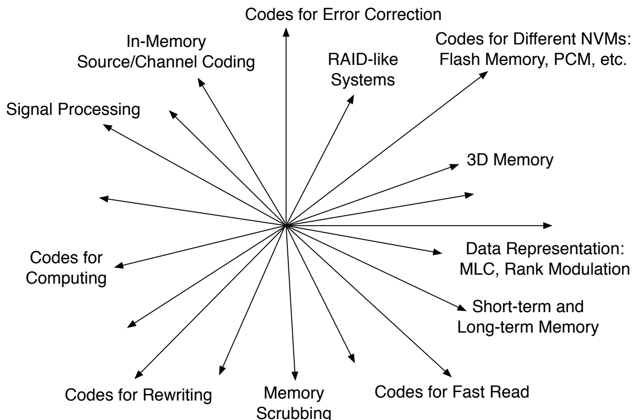
Open Problems on Coding for NVMs



Open Problems on Coding for NVMs



Open Problems on Coding for NVMs



Open Problems on Coding for NVMs