Row-by-Row Coding for Mitigation of Bitline Inter-cell Interference in MLC Flash Memories

Sarit Buzaglo, Eitan Yaakobi, and Paul H. Siegel

Center for Magnetic Recording Research, UCSD
Technion - Israel Institute of Technology

Workshop on Coding for Emerging Memories and Storage Technologies

May 3, 2015

Acknowledgement

- This work was supported in part by the National Science Foundation under Grant CCF-1405119, the Israel Science Foundation under Grant No. 1624/14, the ISEF Foundation, and the Weizmann Institute of Science -National Postdoctoral Award Program for Advancing Women in Science.

- P. H. Siegel was supported in part at the Technion by a fellowship from the Lady Davis Foundation and by a Viterbi Leaders Fellowship.

Outline

- Flash Memory Basics
- Inter-cell Interference (ICI)
- ICI-Mitigating Constrained Codes
- Concluding Remarks

References - Flash Memory Error Characterization

[1] Y. Cassuto, M. Schwartz, V. Bohossian, and J. Bruck, "Codes for asymmetric limited-magnitude errors with application to multilevel flash memories," *IEEE Trans. Inf. Theory*, vol. 56, no. 4, pp. 1582-1595, April 2010.

[2] E. Yaakobi, J. Ma, L. Grupp, P. H. Siegel, S. Swanson, and J. K. Wolf, "Error characterization and coding schemes for flash memories," in *Proc. IEEE Globecom Workshops*, Dec. 2010, pp. 1856-1860.

[3] Y. Cai, E.F. Haratsch, O. Mutlu, and K. Mai, ""Error patterns in MLC NAND flash memory: Measurement, characterization, and analysis," in *Proc. DATE*, March 2012, pp. 521-526.

[4] J. Moon, J. Lo, S. Lee, S. Kim, S. Choi, and Y. Song, "Statistical characterization of noise and interference in NAND flash memory," in *IEEE Trans. Circuits and Systems - I: Regular Papers*, vol. 60, no. 8, August 2013, pp. 2153-2164.

[5] Y. Cai, E.F. Haratsch, O. Mutlu, and K. Mai, "Threshold voltage distribution in MLC NAND flash memory: characterization, analysis, and modeling," in *Proc. DATE*, March 2013, pp.1285-1290.

[6] Y. Cai, O. Mutlu, E.F. Haratsch, and K. Mai, "Program interference in MLC NAND flash memory: Characterization, modeling, and mitigation," in *Proc. IEEE ICCD*, June 2013, pp. 123-130.

References - Constrained Codes for Flash Memory

[1] A. Berman and Y. Birk, "Constrained flash memory programming," *Proc. IEEE Symp. Inf. Theory,* July–August, 2011,pp. 2128–2132.

[2] R. Motwani, "Hierarchical constrained coding for floating-gate to floating-gate coupling mitigation in flash memory," in *Proc. IEEE Globecom*, December 2011.

[3] A. Berman and Y. Birk, "Low-complexity two-dimensional data encoding for memory inter-cell interference reduction," *Proc. 27th Conv. IEEE Israel (IEEEI)*,November 2012.

[4] Y. Kim, B. Kumar, K. L. Cho, H. Son, J. Kim, J. J. Kong, and J. Lee, "Modulation coding for flash memories," in *Proc. ICNC*, Jan. 2013, pp. 961-967.

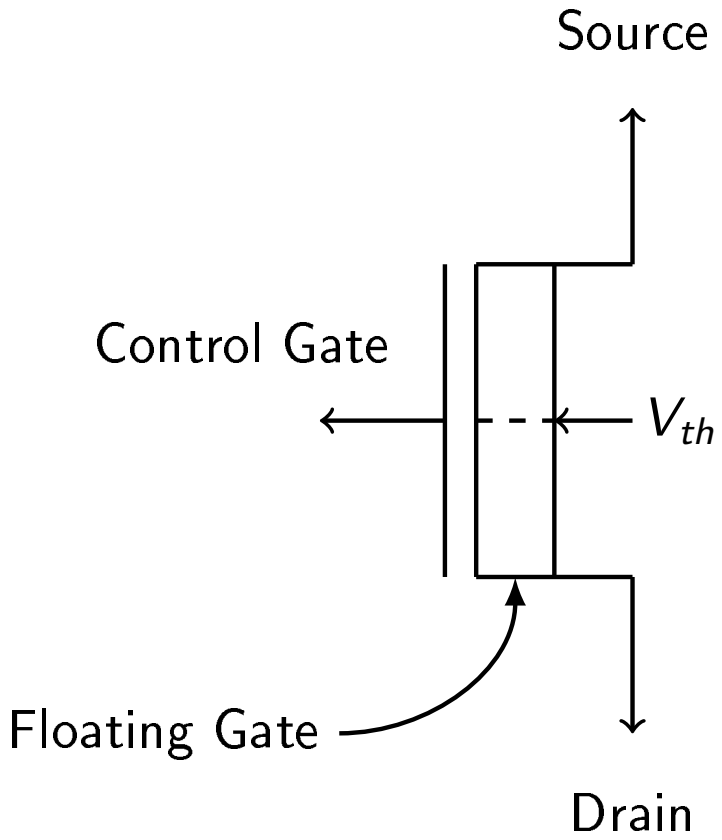
[5] M. Qin, E. Yaakobi, and P. H. Siegel, "Constrained codes that mitigate inter-cell interference in read/write cycles for flash memories," *IEEE JSAC*, vol. 32, no. 5, pp. 836-846, May 2014.

[6] V. Taranalli, H. Uchikawa, and P.H. Siegel, "Error analysis and inter-cell interference mitigation in multi-level cell flash memories," in *Proc. IEEE ICC*, June 2015.

Flash Memory Basics

Flash Memory Structure

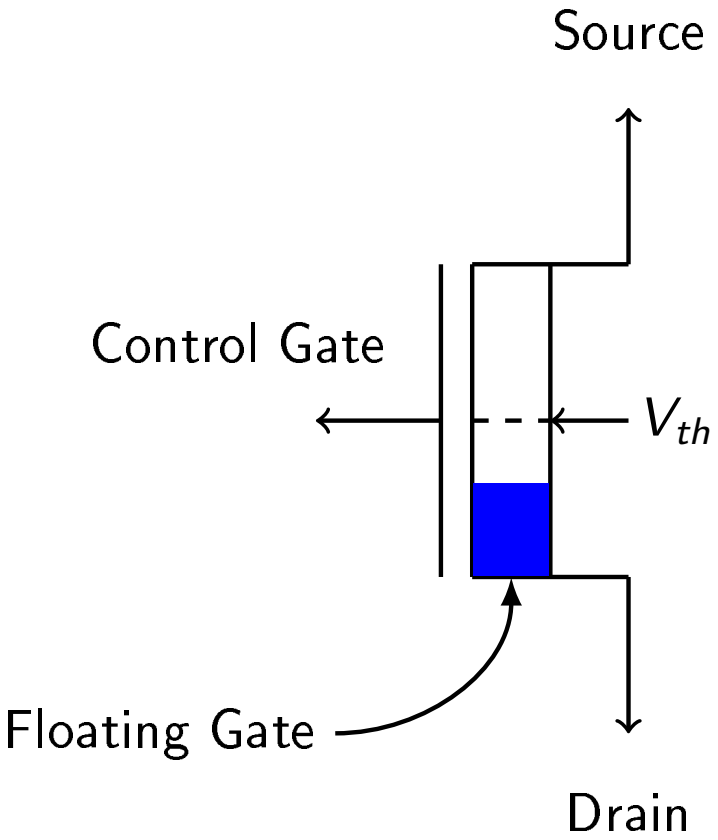
- Floating gate transistor (Cell) – Fundamental data storing unit
- Program a cell to different voltage levels to represent stored bits.



- Incremental Step Pulse Program (ISPP)

Flash Memory Structure

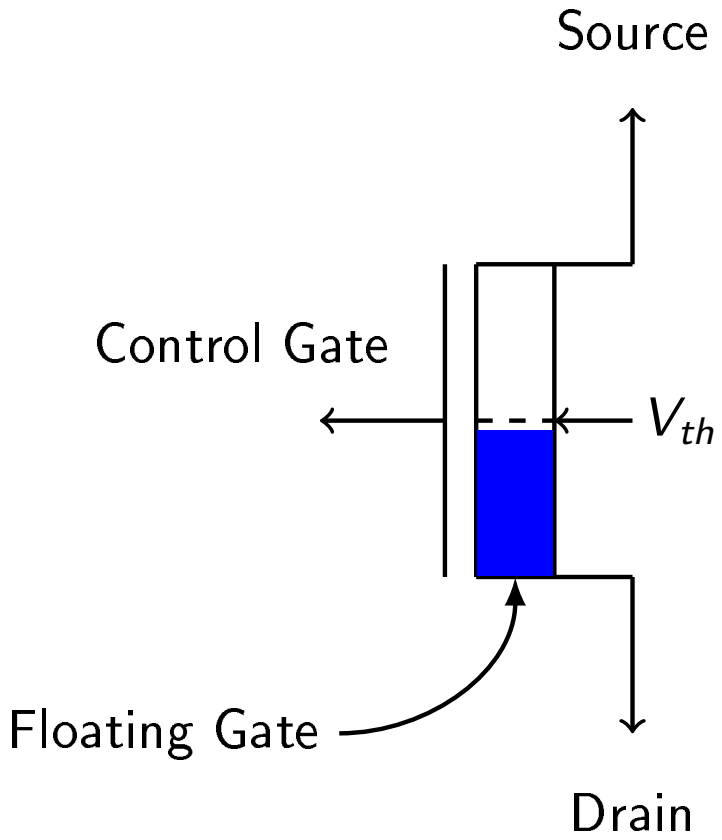
- Floating gate transistor (Cell) – Fundamental data storing unit
- Program a cell to different voltage levels to represent stored bits.



- Incremental Step Pulse Program (ISPP)
- Program, Verify

Flash Memory Structure

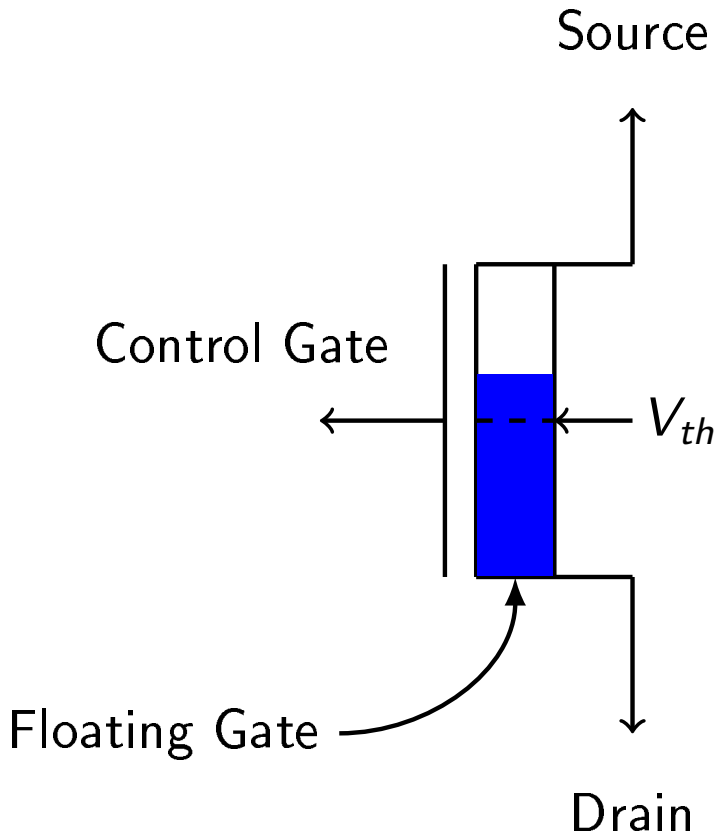
- Floating gate transistor (Cell) – Fundamental data storing unit
- Program a cell to different voltage levels to represent stored bits.



- Incremental Step Pulse Program (ISPP)
- Program, Verify
- Program, Verify

Flash Memory Structure

- Floating gate transistor (Cell) – Fundamental data storing unit
- Program a cell to different voltage levels to represent stored bits.



- Incremental Step Pulse Program (ISPP)
- Program, Verify
- Program, Verify
- Program, Verify

Flash Memory Structure

- Common types of NAND Flash
  - Single-Level Cell (SLC) – 1 bit/cell
  - Multi-Level Cell (MLC) – 2 bits/cell
  - Three-Level Cell (TLC) – 3 bits/cell
- Cells $\rightarrow$ Pages $\rightarrow$ Smallest unit for program and read operations
- Pages $\rightarrow$ Blocks $\rightarrow$ Smallest unit for the erase operation

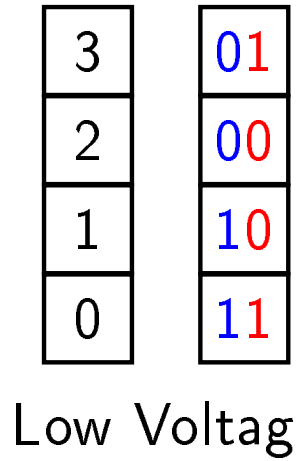
Flash Memory Structure

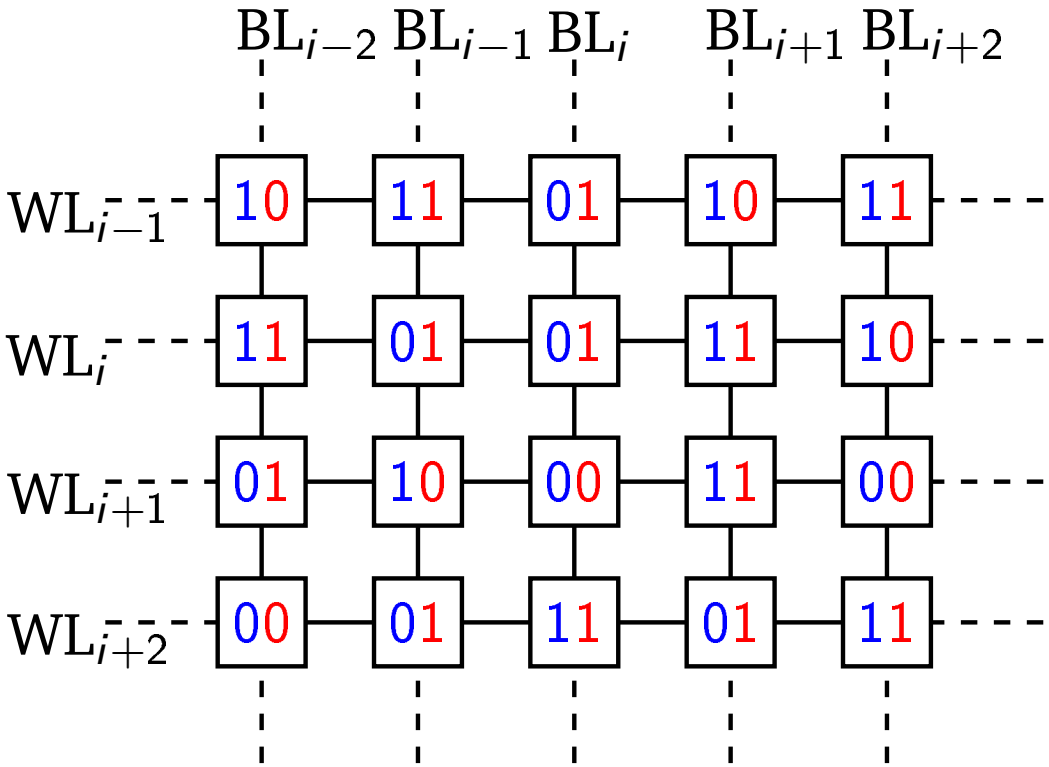
- Common types of NAND Flash
  - Single-Level Cell (SLC) – 1 bit/cell
  - Multi-Level Cell (MLC) – 2 bits/cell
  - Three-Level Cell (TLC) – 3 bits/cell
- Cells $\rightarrow$ Pages $\rightarrow$ Smallest unit for program and read operations
- Pages $\rightarrow$ Blocks $\rightarrow$ Smallest unit for the erase operation

MLC Flash Memory Structure

Lower Page

Upper Page



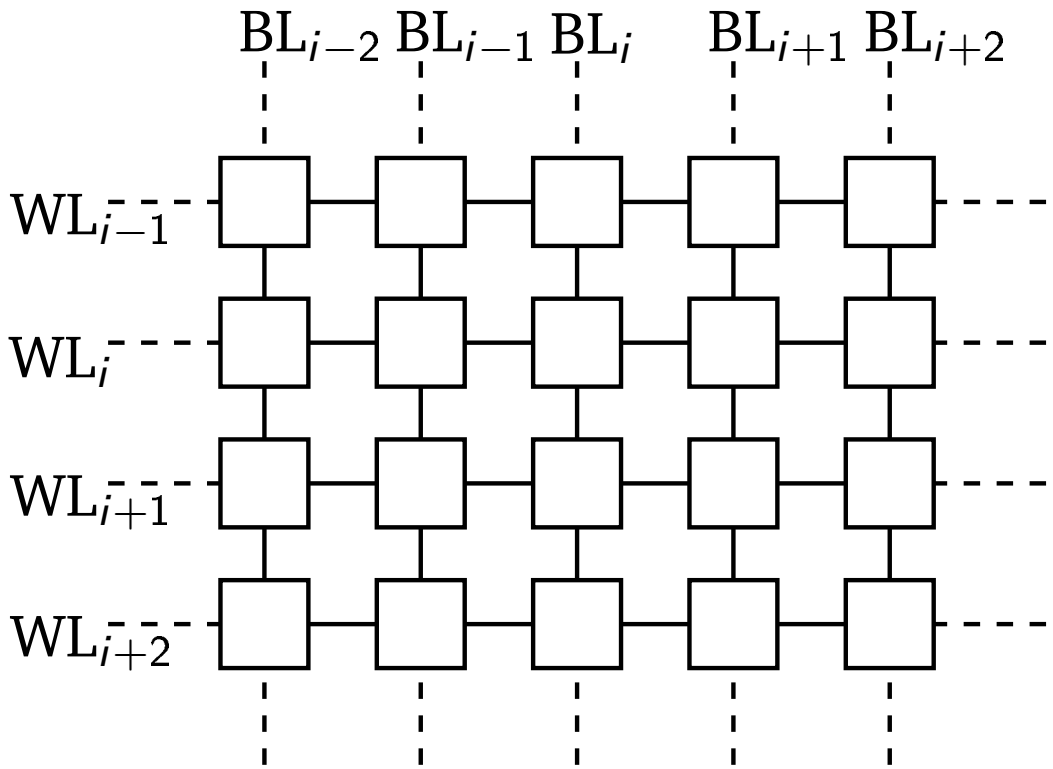


- Typical MLC flash page holds 4K - 16K bytes
- Typical block holds 64 - 128 pages

Programming MLC Flash Blocks

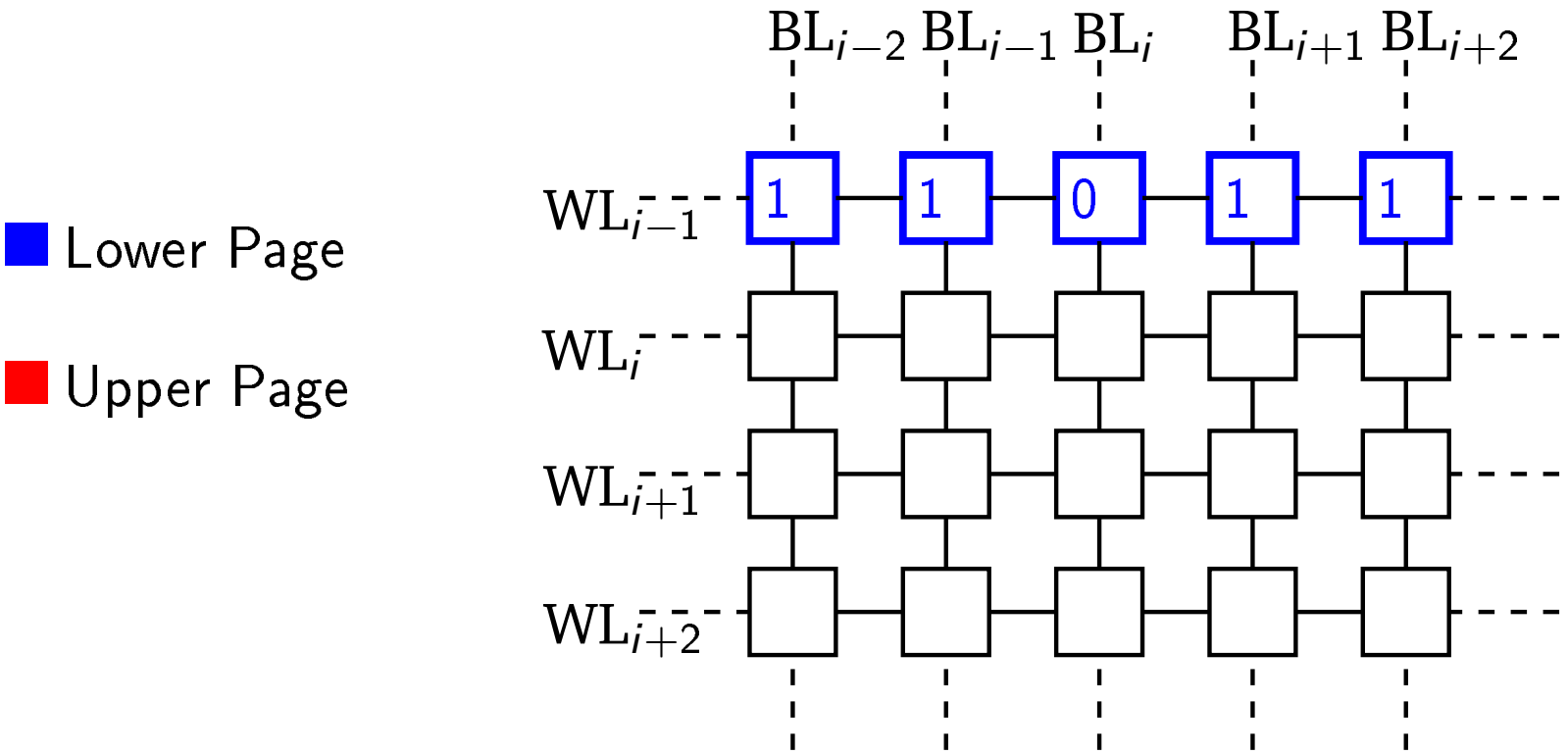
Lower Page

Upper Page



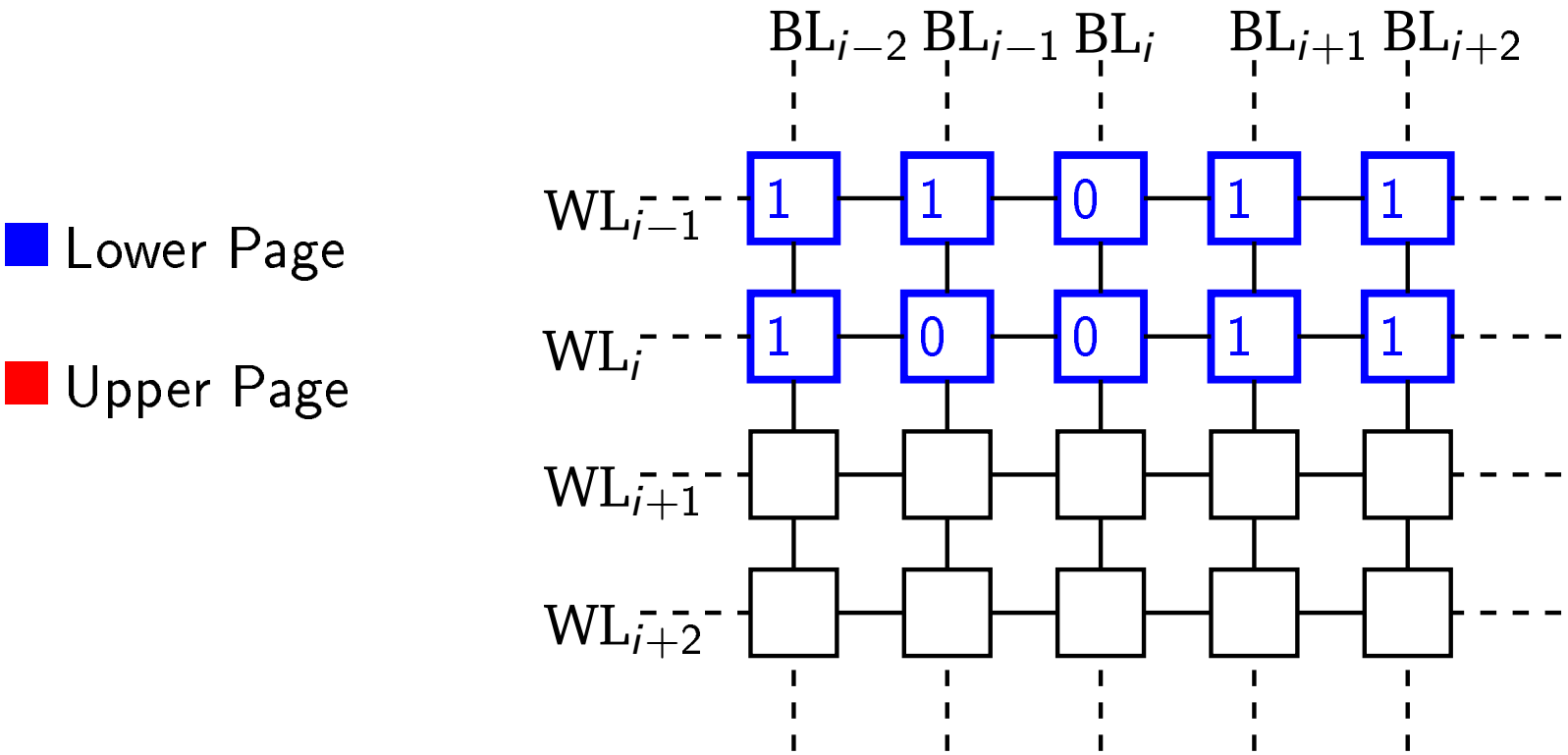
- Upper and lower pages within wordlines are independent.
- Programming of pages is done row-by-row.

Programming MLC Flash Blocks



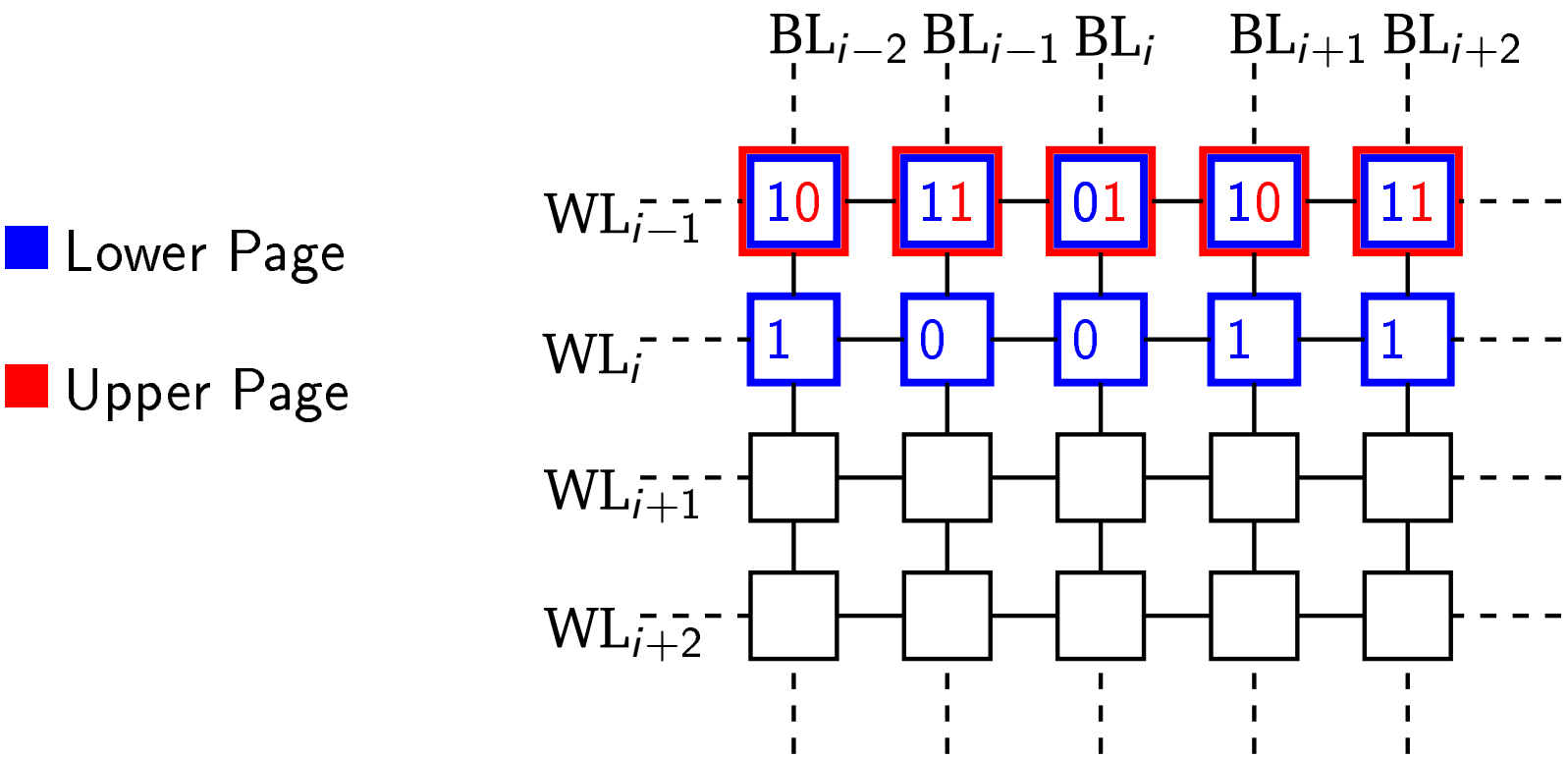
- Upper and lower pages within wordlines are independent.
- Programming of pages is done row-by-row.

Programming MLC Flash Blocks



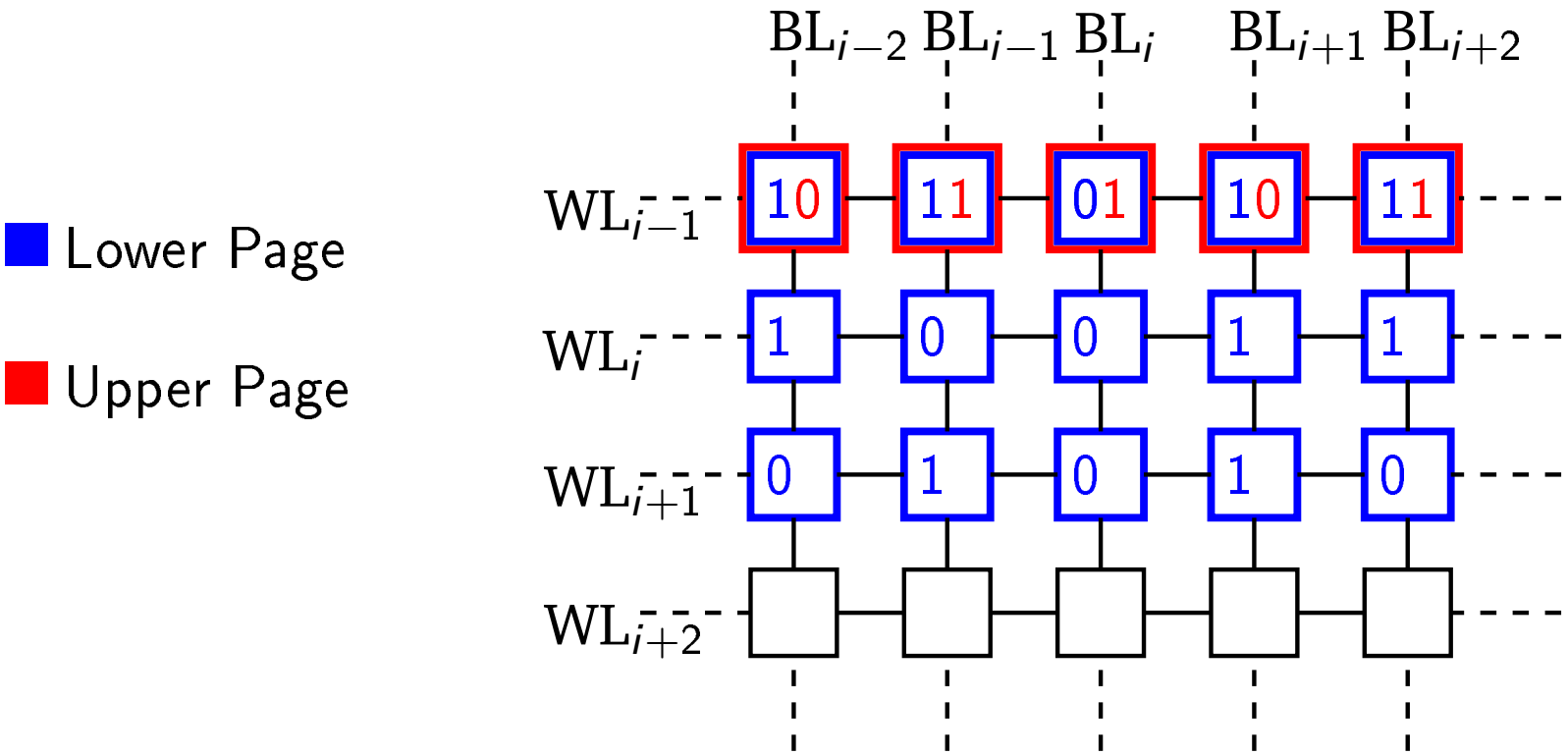
- Upper and lower pages within wordlines are independent.
- Programming of pages is done row-by-row.

Programming MLC Flash Blocks



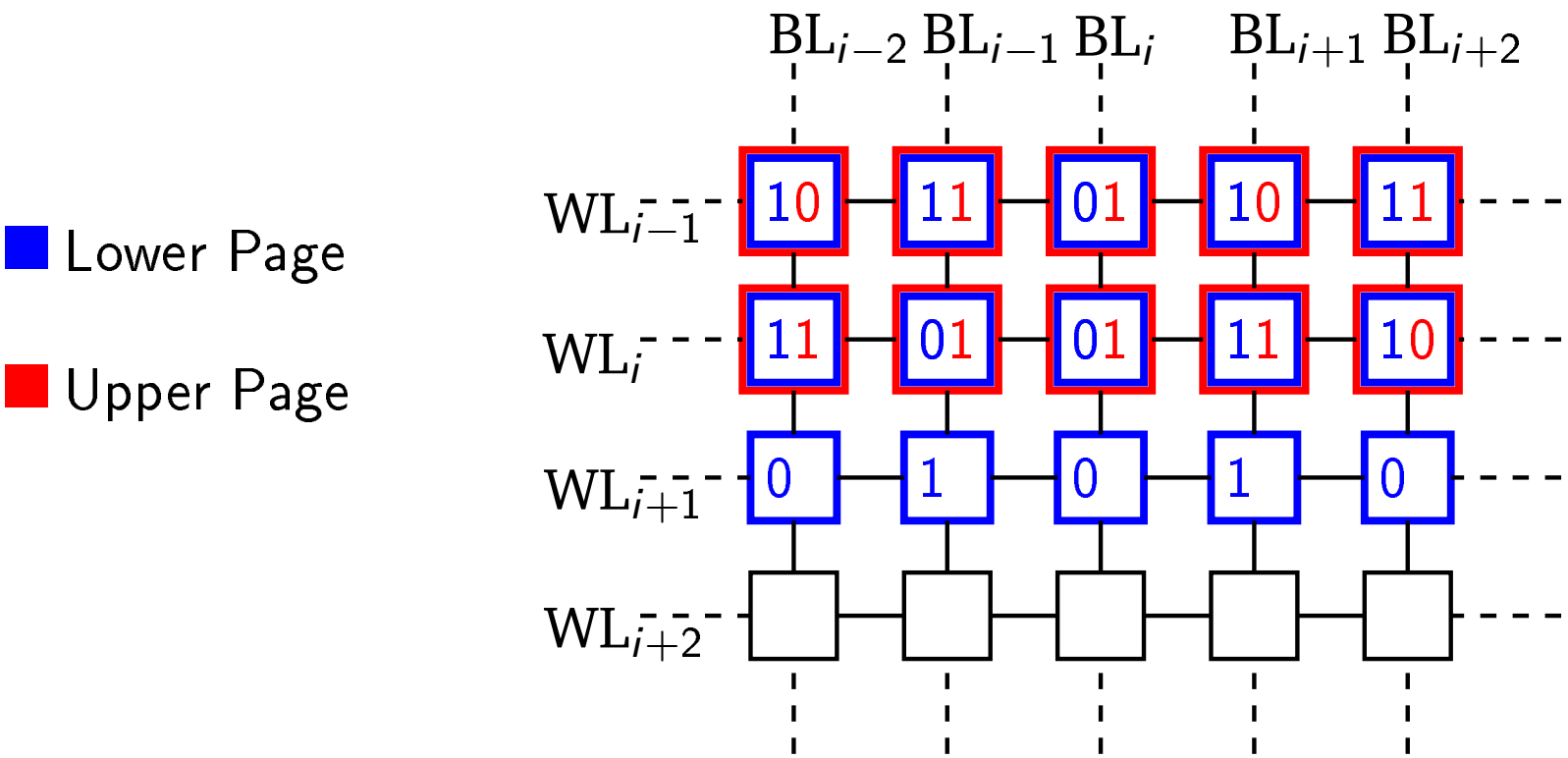
- Upper and lower pages within wordlines are independent.
- Programming of pages is done row-by-row.

Programming MLC Flash Blocks



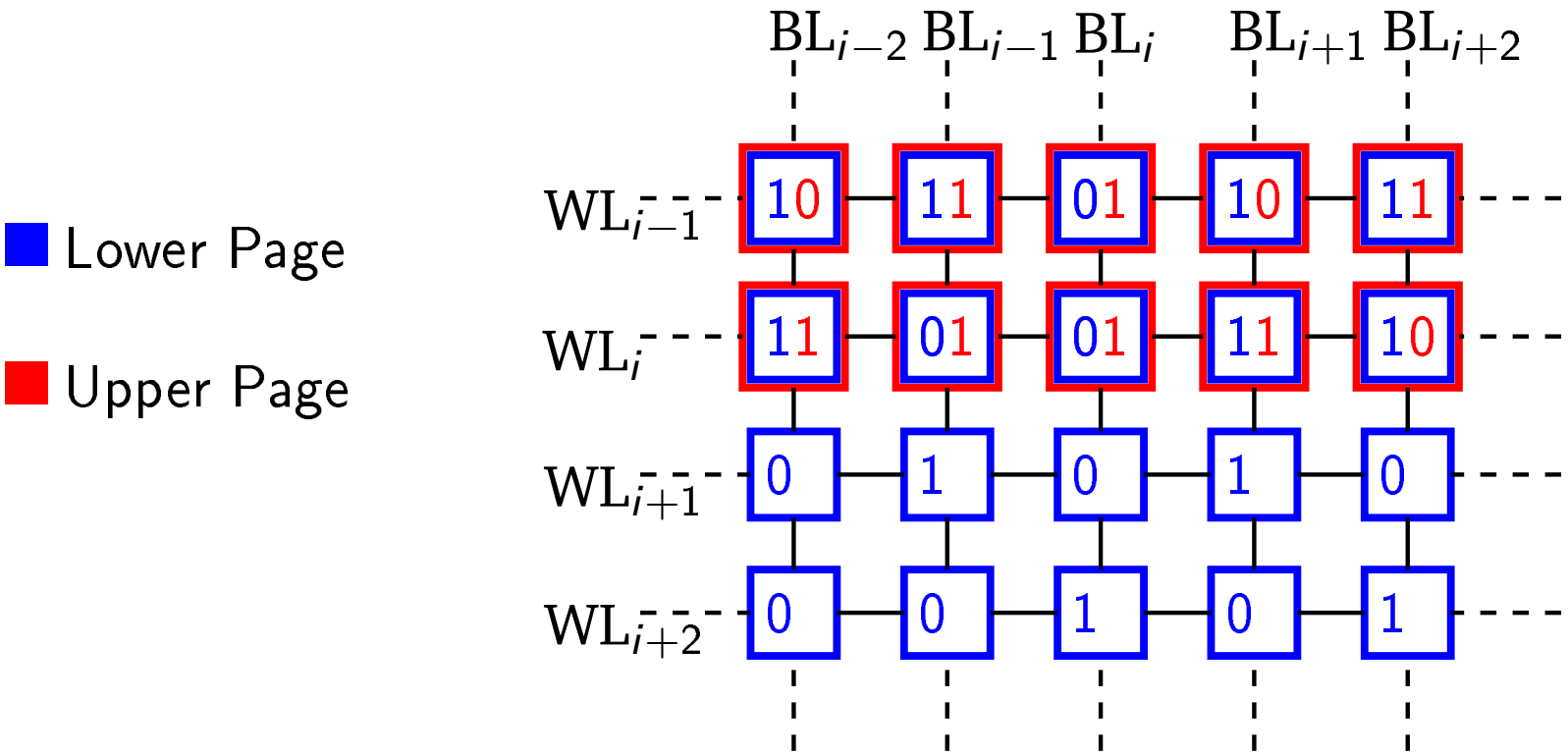
- Upper and lower pages within wordlines are independent.
- Programming of pages is done row-by-row.

Programming MLC Flash Blocks



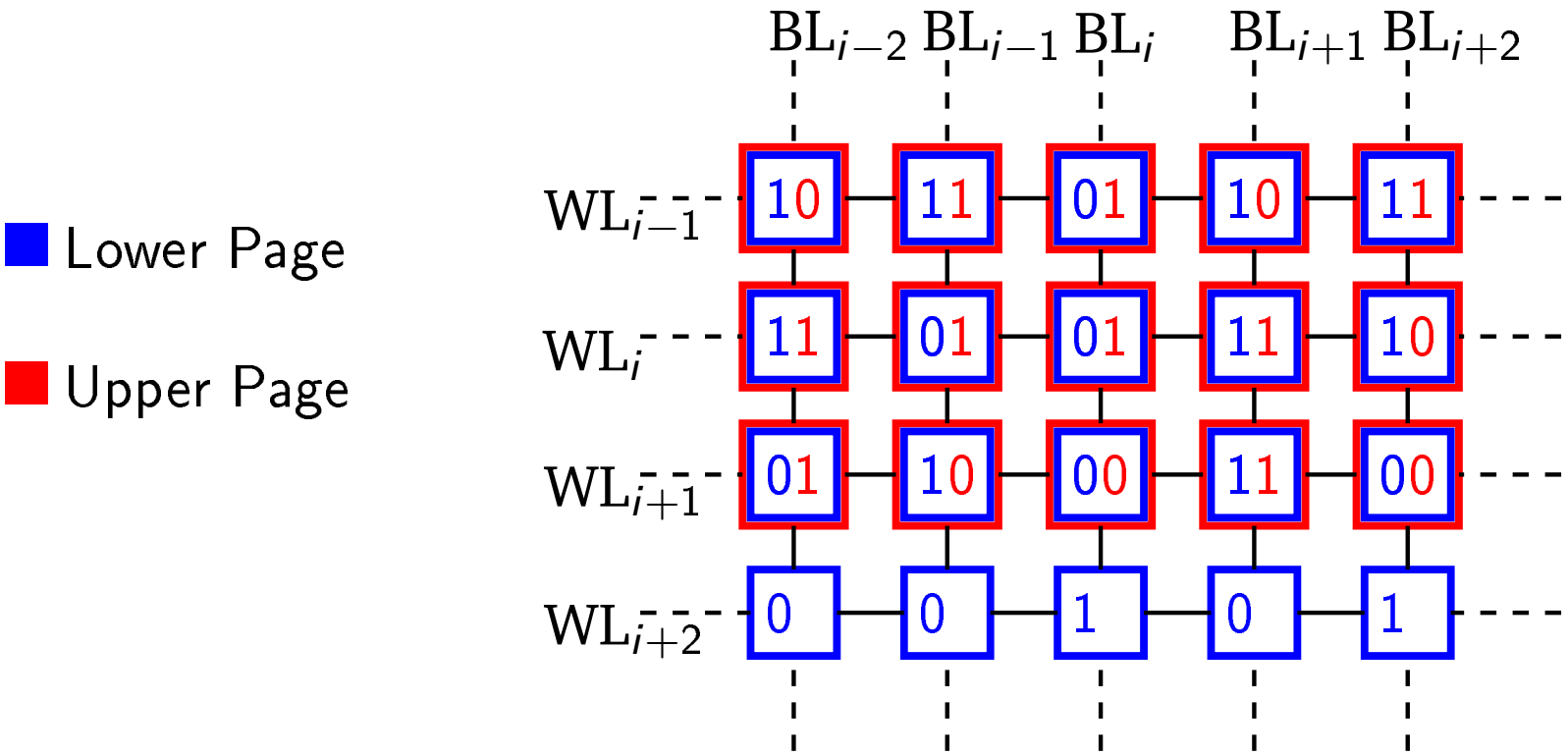
- Upper and lower pages within wordlines are independent.
- Programming of pages is done row-by-row.

Programming MLC Flash Blocks



- Upper and lower pages within wordlines are independent.
- Programming of pages is done row-by-row.

Programming MLC Flash Blocks



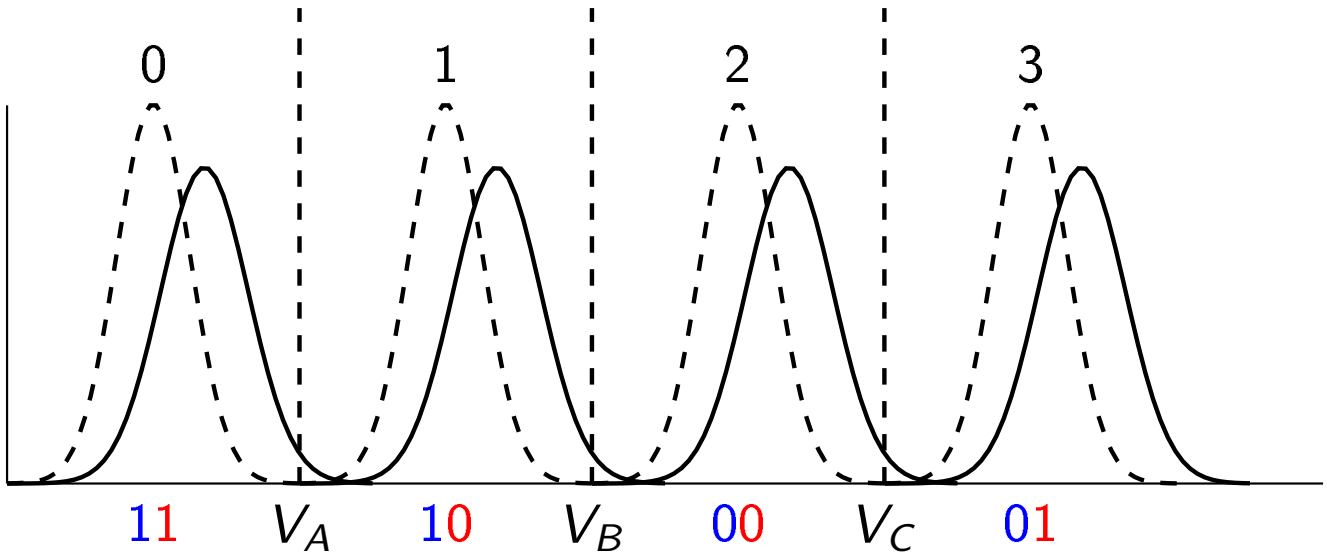
- Upper and lower pages within wordlines are independent.
- Programming of pages is done row-by-row.

Inter-cell Interference (ICI)

Inter-cell Interference (ICI)

- Parasitic capacitance coupling occurs among neighboring cells.
- This causes random, data dependent errors after cells are programmed.
- ICI-induced errors are a dominant problem for small feature size technology.

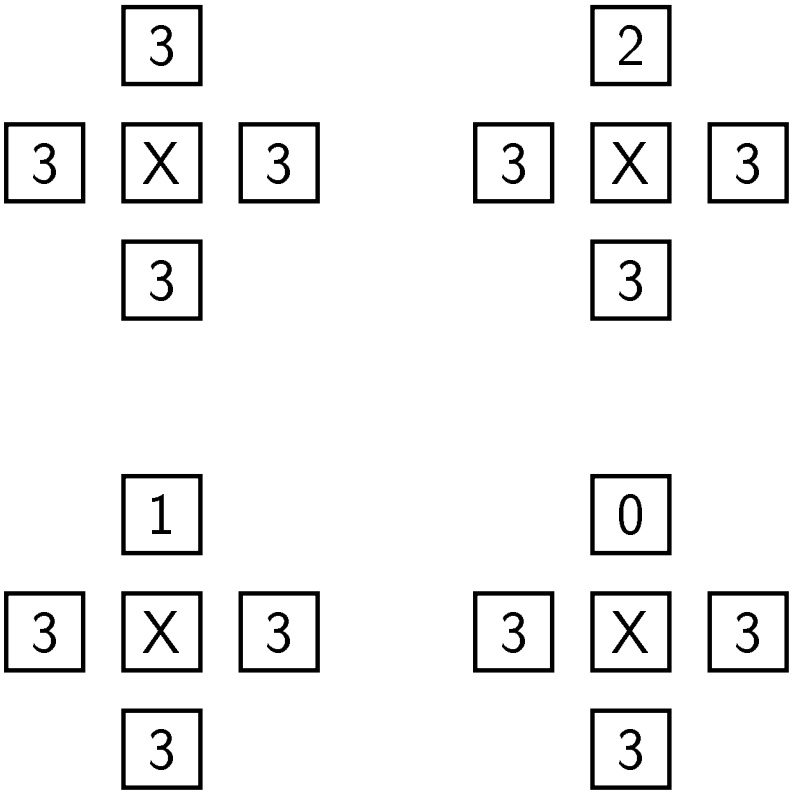
Dominant Cell Errors



Most cell errors are adjacent cell-level errors in the upward direction:

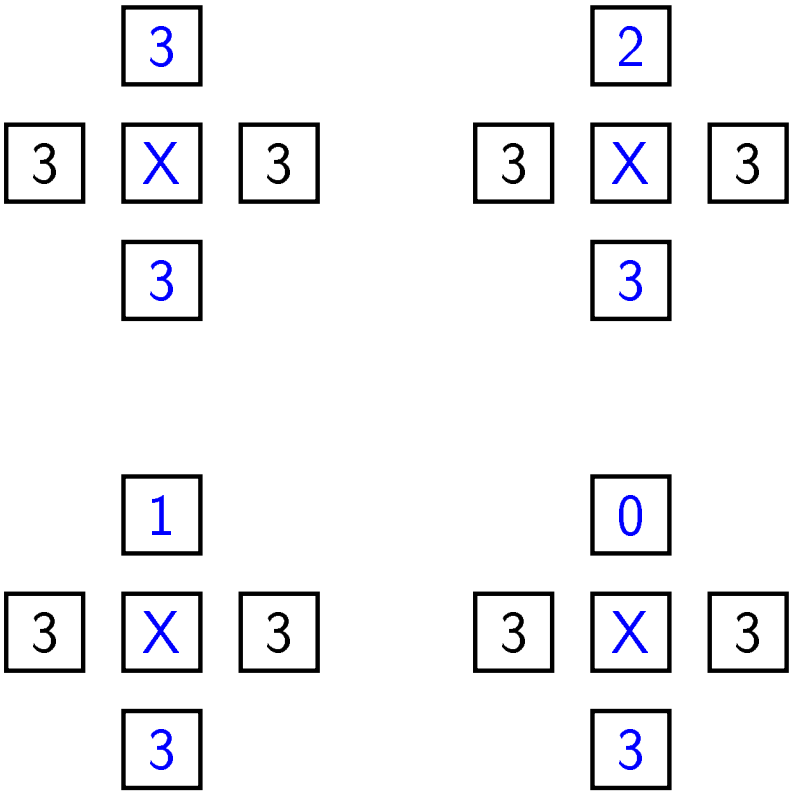
$$0 \rightarrow 1 \quad \text{(upper page error)}$$
$$1 \rightarrow 2 \quad \text{(lower page error)}$$
$$2 \rightarrow 3 \quad \text{(upper page error)}$$

Dominant Cell Error Patterns



- Neighbor cells programmed to the highest level '3' cause the most ICI.
- Wordline (horizontal) ICI effect is symmetric.
- Bitline (vertical) ICI effect is asymmetric.
- Bitline ICI causes more errors.

Dominant Cell Error Patterns



- Neighbor cells programmed to the highest level '3' cause the most ICI.
- Wordline (horizontal) ICI effect is symmetric.
- Bitline (vertical) ICI effect is asymmetric.
- Bitline ICI causes more errors.

How can we control bitline ICI-induced errors under the constraints of page-oriented, row-by-row programming?

ICI-Mitigating Constrained Codes

Dominant Error Patterns - Binary Representations

- Binary representation of dominant cell error patterns:

Cell levels	LU-LU-LU
3-0-3	01-11-01
3-1-3	01-10-01
3-2-3	01-00-01
2-0-3	00-11-01
2-1-3	00-10-01

Dominant Error Patterns - Binary Representations

- Lower pages of dominant cell error patterns:

Cell levels	LU-LU-LU
3-0-3	01-11-01
3-1-3	01-10-01
3-2-3	01-00-01
2-0-3	00-11-01
2-1-3	00-10-01

Dominant Error Patterns - Binary Representations

- Upper pages of dominant cell error patterns:

Cell levels	LU-LU-LU
3-0-3	01-11-01
3-1-3	01-10-01
3-2-3	01-00-01
2-0-3	00-11-01
2-1-3	00-10-01

ICI-Mitigating No-010 Constraint

- All patterns except 3-2-3 contain 0-1-0 in lower pages.

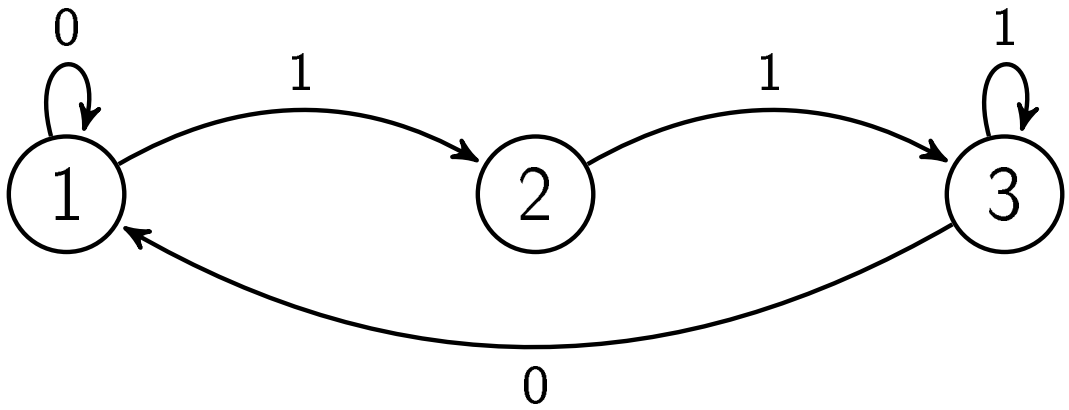
Cell levels	LU-LU-LU
3-0-3	01-11-01
3-1-3	01-10-01
3-2-3	01-00-01
2-0-3	00-11-01
2-1-3	00-10-01

- Forbidding 0-1-0 in bitline lower pages eliminates vertical patterns

3-0-3 3-1-3, 2-0-3, 2-1-3

ICI-Mitigating Constraints

- The set S of binary sequences satisfying the no-010 constraint.



- The maximum possible coding efficiency of a code into these sequences is the capacity $Cap(S)$:

$$Cap(S) = \lim_{n\to\infty} \frac{\log_2 |S \cap \{0,1\}^n|}{n} \approx 0.8114.$$

- This is the highest code rate we could achieve even if we encoded bitline pages directly (i.e., vertically).

Row-by-Row Coding for Bitline ICI Mitigation

- We present a row-by-row code construction that consists of two steps.
  - Step 1: Probabilistic analysis (for target code rate)
  - Step 2: Code construction using constant-weight codes
- The design method yields codes with the following properties:
  - Encoding is row-by-row and fixed rate.
  - Encoding / decoding a row requires the previous two rows.
  - The code rate can approach the capacity $Cap(S) \approx 0.8114$ (as the number of wordlines and bitlines approaches infinity).

Probabilistic Analysis

- We define a stationary Markov process in terms of:
  - Probability mass function $\pi(x_1x_2)$, $x_1x_2 \in \{0,1\}^2$, $0 \leq \pi(x_1x_2) \leq 1$, $\sum_{x_1,x_2 \in \{0,1\}} \pi(x_1x_2) = 1$.
  - For each x_1x_2, a conditional probability mass function $P(x_3|x_1x_2)$, $x_3 \in 0,1$, $P(1|x_1x_2) = 1 - P(0|x_1x_2)$.
- For a binary sequence $\mathrm{b} = b_1 b_2 \ldots b_m$, we define

$$Pr(\mathrm{b}) = \pi(b_1 b_2) \prod_{i=3}^{m} P(b_i|b_{i-2}b_{i-1}).$$

Probabilistic Analysis (cont.)

- By stationarity, we obtain a system of equations, linear in $\pi(x_1x_2)$

$$\begin{aligned}
\pi(00)P(0|00) + \pi(10)P(0|10) &= \pi(00) \\
\pi(00)P(1|00) + \pi(10)P(1|10) &= \pi(01) \\
\pi(01)P(0|01) + \pi(11)P(0|11) &= \pi(10) \\
\pi(01)P(1|01) + \pi(11)P(1|11) &= \pi(11)
\end{aligned}$$

- We can express $\pi(x_1x_2)$ in terms of $P(0|\tilde{x}_1\tilde{x}_2)$, and the information rate of the process can be written as

$$R(P) = H(P(x_3|x_1x_2)).$$

Probabilistic Analysis (cont.)

- To satisfy the no-010 constraint, we set $P(0|01) = 0$.
- Then $P(0|x_1x_2)$, $x_1x_2 \neq 01$ can be chosen to maximize $R(P)$, yielding $R(P) = Cap(S)$.
- For our code construction, choose n and P such that $\pi(x_1x_2)n$ and $P(0|x_1x_2)\pi(x_1x_2)n$ are integers.
- Since $R(P)$ is continuous in $P(0|x_1x_2)$, we can do this for rates arbitrarily close to $Cap(S)$.

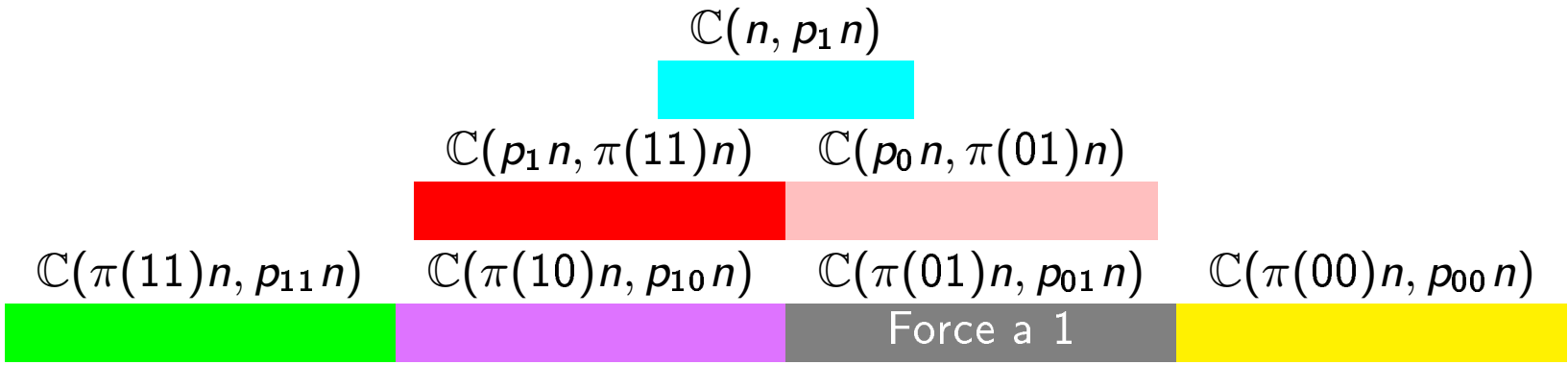
Code Construction

- Define $p_x = \pi(x0) + \pi(x1)$ and $p_{x_1x_2} = P(1|x_1x_2)\pi(x_1x_2)$.
- Let $\mathbb{C}(m, w)$ be the constant-weight code that consists of all binary sequences of length m and weight w.
- The encoding process uses three codes built from various $\mathbb{C}(m, w)$:

$$\begin{aligned}
\mathbb{C}^1 &= \mathbb{C}(n, p_1 n) \\
\mathbb{C}^2 &= \mathbb{C}(p_0 n, \pi(01)n) \times \mathbb{C}(p_1 n, \pi(11)n) \\
\mathbb{C}^3 &= \mathbb{C}(\pi(00)n, p_{00} n) \times \mathbb{C}(\pi(01)n, p_{01} n) \times \\
&\quad \mathbb{C}(\pi(10)n, p_{10} n) \times \mathbb{C}(\pi(11)n, p_{11} n).
\end{aligned}$$

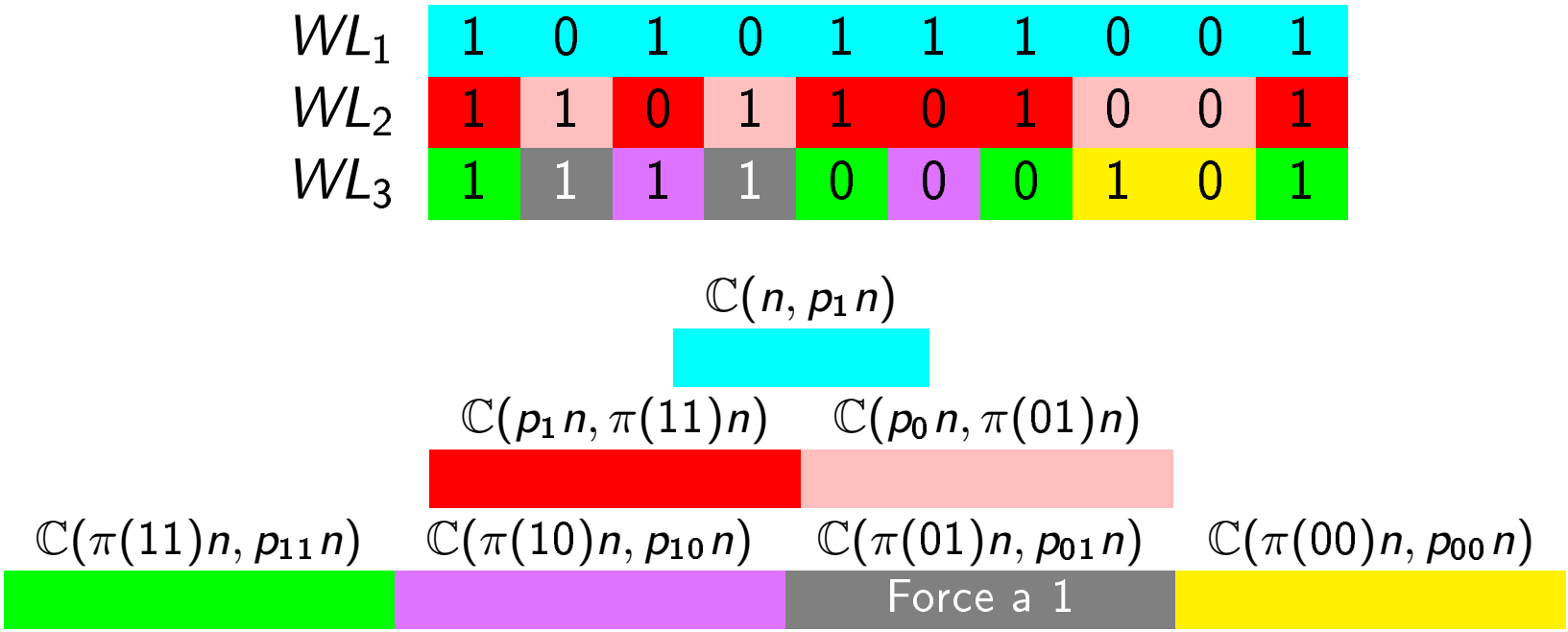
- $P(0|01) = 0$ implies $p_{01} = \pi(01)$, so $\mathbb{C}(\pi(01)n, p_{01} n) = \{\underline{1}\}$.

Encoding



- WL_1: Encode using $\mathbb{C}(n, p_1 n)$.
- WL_2: Find index sets I_0 and I_1 where values in WL_1 are 0 and 1.
  For corresponding index sets in WL_2, encode using $\mathbb{C}(p_0 n, \pi(01)n)$ and $\mathbb{C}(p_1 n, \pi(11)n)$
- WL_i, $i \geq 3$:
  Find index sets $I_{x_1 x_2}$ where value in WL_{i-2}, WL_{i-1} is $x_1 x_2$.
  For corresponding sets of indices in WL_i, encode using the four codes $\mathbb{C}(\pi(x_1 x_2)n, p_{x_1 x_2} n)$, $x_1 x_2 \in \{0,1\}^2$.

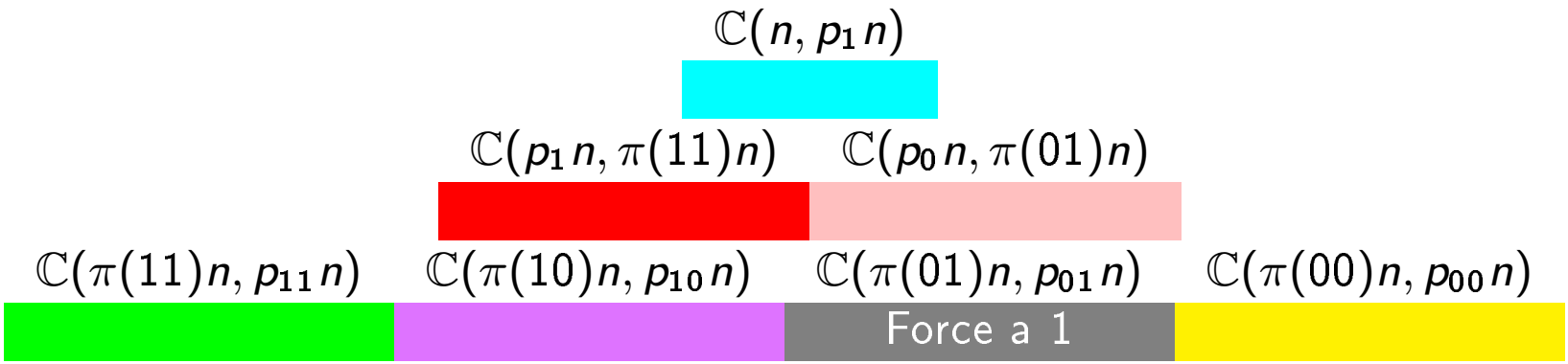
Example: Target rate $R(P) = \frac{4}{5}$



- $p_1 = \frac{3}{5}$; $p_0 = \frac{2}{5}$.
- $\pi(11) = \frac{2}{5}$; $\pi(x_1 x_2) = \frac{1}{5}$, $x_1 x_2 \neq 11$.
- $P(1|11) = P(1|10) = P(1|00) = \frac{1}{2}$ so $\mathbb{C}^2$ and $\mathbb{C}^3$ use balanced codes.
- Since $\lim_{n\to\infty}(1/n)\log_2|\mathbb{C}(n, \lfloor \beta n \rfloor)| = H(\beta)$, the asymptotic encoding rate of the balanced codes is 1.
- The asymptotic rate of code $\mathbb{C}^3$ is therefore 4/5.

Encoding: WL_4

WL_1	1	0	1	0	1	1	1	0	0	1
WL_2	1	1	0	1	1	0	1	0	0	1
WL_3	1	1	1	1	0	0	0	1	0	1
WL_4	1	1	1	0	0	1	1	1	0	0



- Rows WL_{i-2}, WL_{i-1}, $i \geq 3$ have a P-typical distribution of bitline pairs $x_1 x_2$.
- Since the asymptotic rate of $\mathbb{C}(\pi(ab)n, p_{ab}n)$ is $H(P(1|ab))$, and its proportional length is $\pi(ab)$, the asymptotic rate of the coding scheme for general distribution P is $R(P)$.
- The overall rate of the ICI-mitigating scheme is $R = \frac{1}{2}(1 + R(P))$, or R=0.9 in the example.

Decoding

WL_1	1	0	1	0	1	1	1	0	0	1
WL_2	1	1	0	1	1	0	1	0	0	1
WL_3	1	1	1	1	0	0	0	1	0	1
WL_4	1	1	1	0	0	1	1	1	0	0
WL_5	1	0	1	0	0	1	1	0	1	1

- WL_1: Decode using $\mathbb{C}(n, p_1 n)$.
- WL_2: Find index sets I_x where value in WL_1 is x, for $x \in \{0, 1\}$.
  For corresponding index sets in WL_2, decode using $\mathbb{C}(p_x n, \pi(x1)n)$.
- WL_i, $i \geq 3$:
  Find index sets $I_{x_1 x_2}$ where value in WL_{i-2}, WL_{i-1} is $x_1 x_2$.
  For corresponding sets of indices in WL_i, decode using $\mathbb{C}(\pi(x_1 x_2)n, p_{x_1 x_2} n)$

Concluding Remarks

Concluding Remarks

- Our code construction assumes the use of efficient encoding and decoding algorithms for constant-weight codes.
- It can be extended to any finite-memory, bitline ICI-mitigating constraint in vertical lower pages (resp. upper) pages.
- It can also be used independently with a wordline ICI-mitigating constrained code on horizontal upper pages (resp. lower) pages. The overall rate is then the average of the two code rates.

Concluding Remarks

- Our construction is an embodiment of the row-by-row coding method proposed in:

  Ido Tal, Tuvi Etzion, and Ron M. Roth,
  "On Row-by-Row Coding for 2-D Constraints,"
  IEEE Transactions on Information Theory, vol. 55, no. 8,
  August 2009, pp. 3565–3575.

- Extensions to 2-dimensional constrained coding with vertical *data strips* and *merging strips* are described there.

Extensions

- One can allow the bitline ICI pattern 010 to appear with a specified nonzero probability and carry out a similar probabilistic analysis and code construction based upon constant-weight codes.

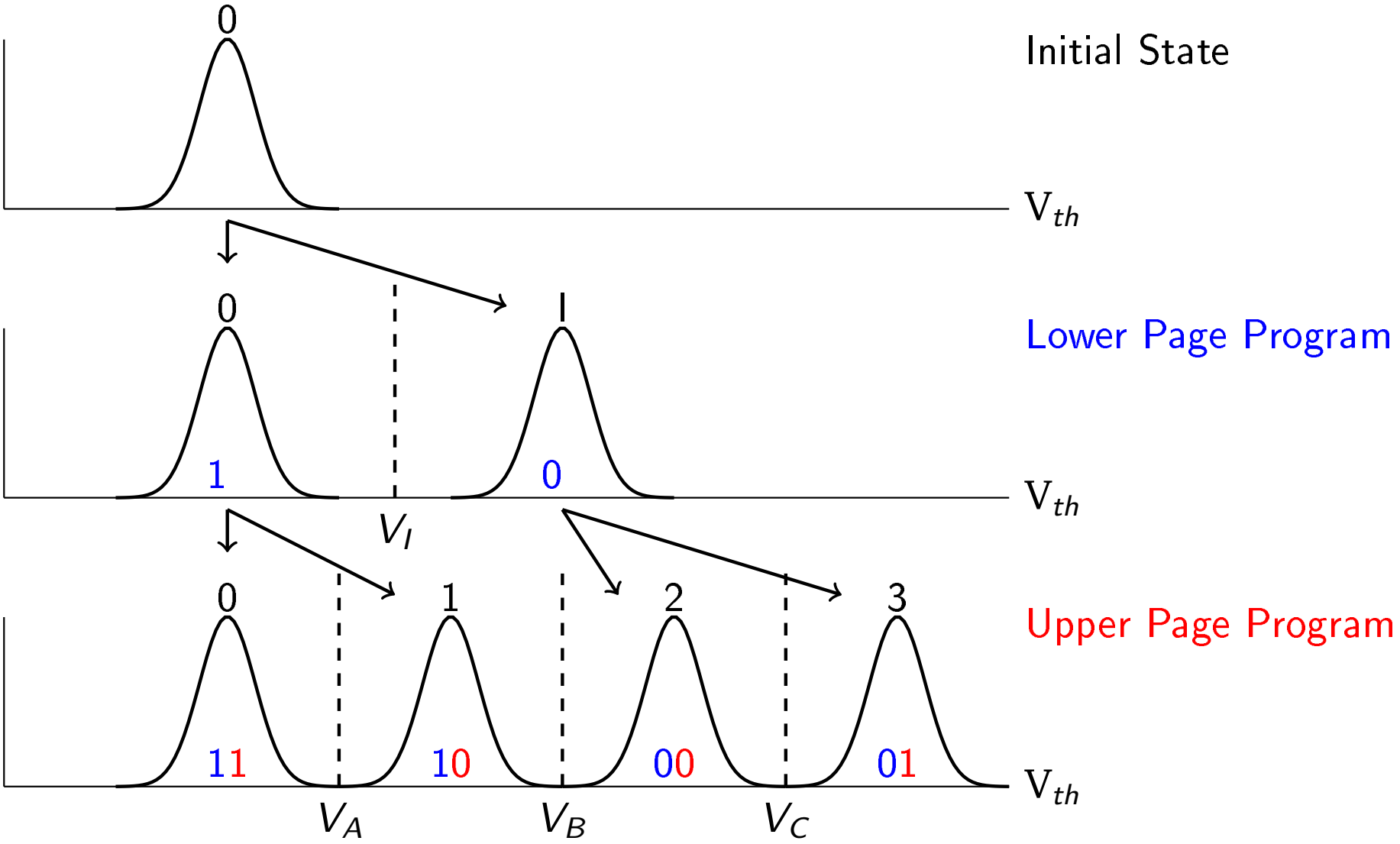
- One can then combine the row-by-row encoding with a systematic error-correcting code, potentially yielding higher rates and good performance, depending on the probability of a bitline ICI error [Buzaglo, et al., ISIT 2015]. (illustrated for SLC no-101 constraint)

WL_1	1	1	0	1	0	0	0	1	PB_1
WL_2	1	0	1	1	0	1	0	1	PB_2
WL_3	1	0	0	0	1	1	0	1	PB_3
WL_4	0	1	0	1	1	1	0	0	PB_4

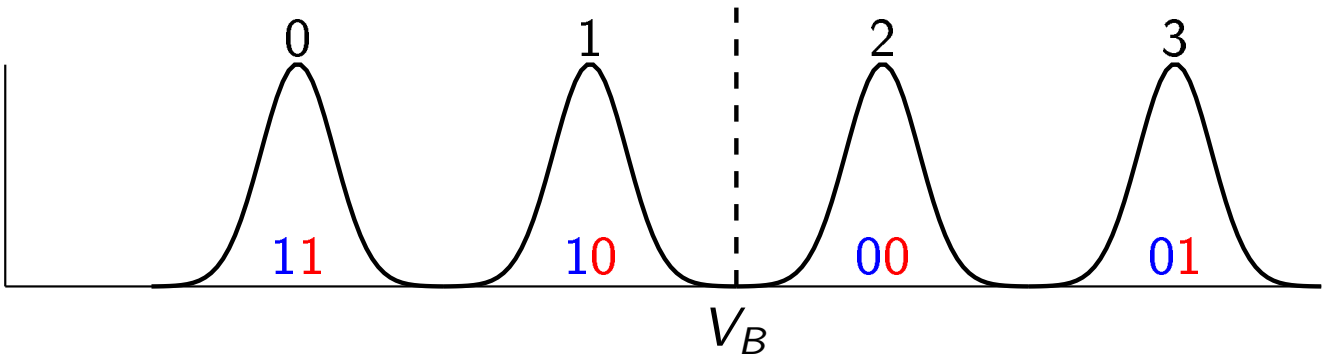
- The analysis of the asymptotic rate and performance of this scheme poses several interesting and challenging problems.

Thank you.

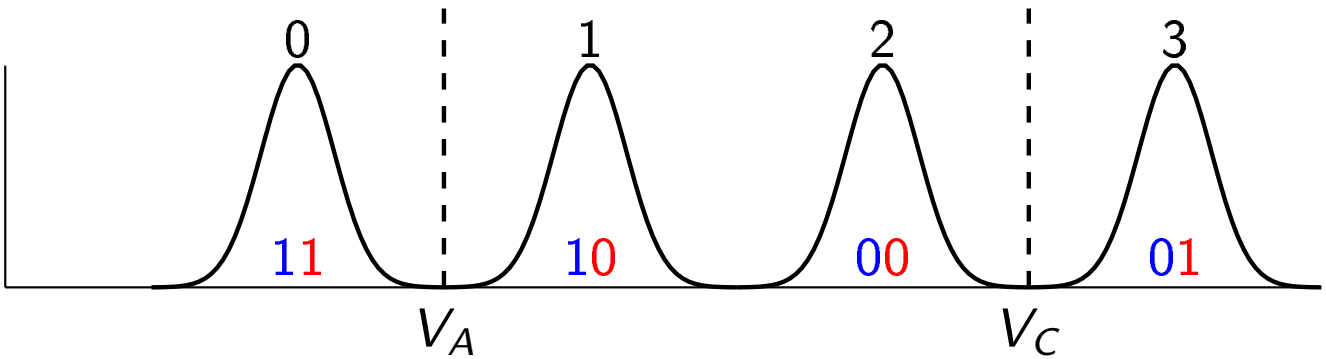
Programming MLC Flash Cells



Reading MLC Flash Cells



Lower Page Read



Upper Page Read