STAIR Codes: A General Family of Erasure Codes for Tolerating Device and Sector Failures in Practical Storage Systems

Patrick P. C. Lee
The Chinese University of Hong Kong

Coding: From Practice to Theory, February 2015

This is a joint work with Dr. Mingqiang Li.

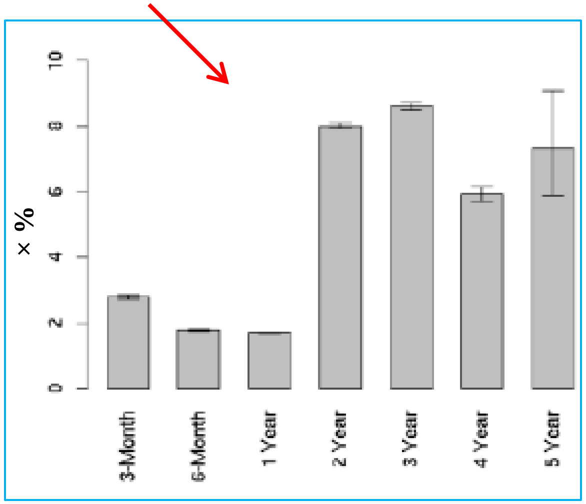
Device and Sector Failures

- Hierarchy of failures in disk arrays:
  - *Device failure*: data loss in an entire device
  - *Sector failure (latent sector error)*: data loss in a sector

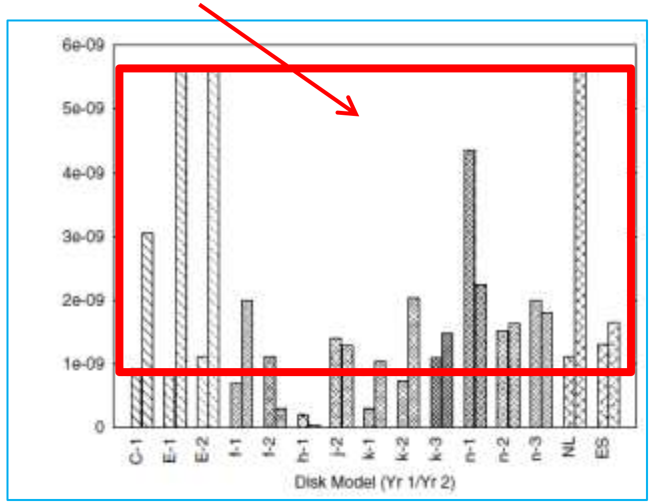
(a) Annual disk failure rate: 1~10%

(b) Sector failures can be more frequent than disk failures

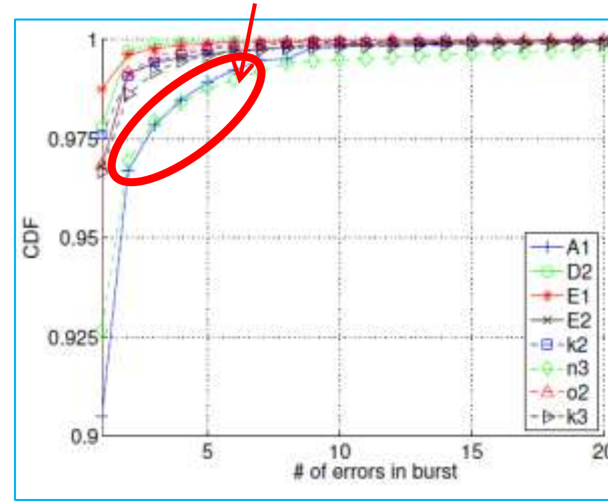
(c) Sector failure bursts can be long (> 5)



Annual disk failure rate
[Pinheiro et al., FAST'07]



Annual sector failure rate
[Bairavasundaram et al., SIGMETRICS '07]



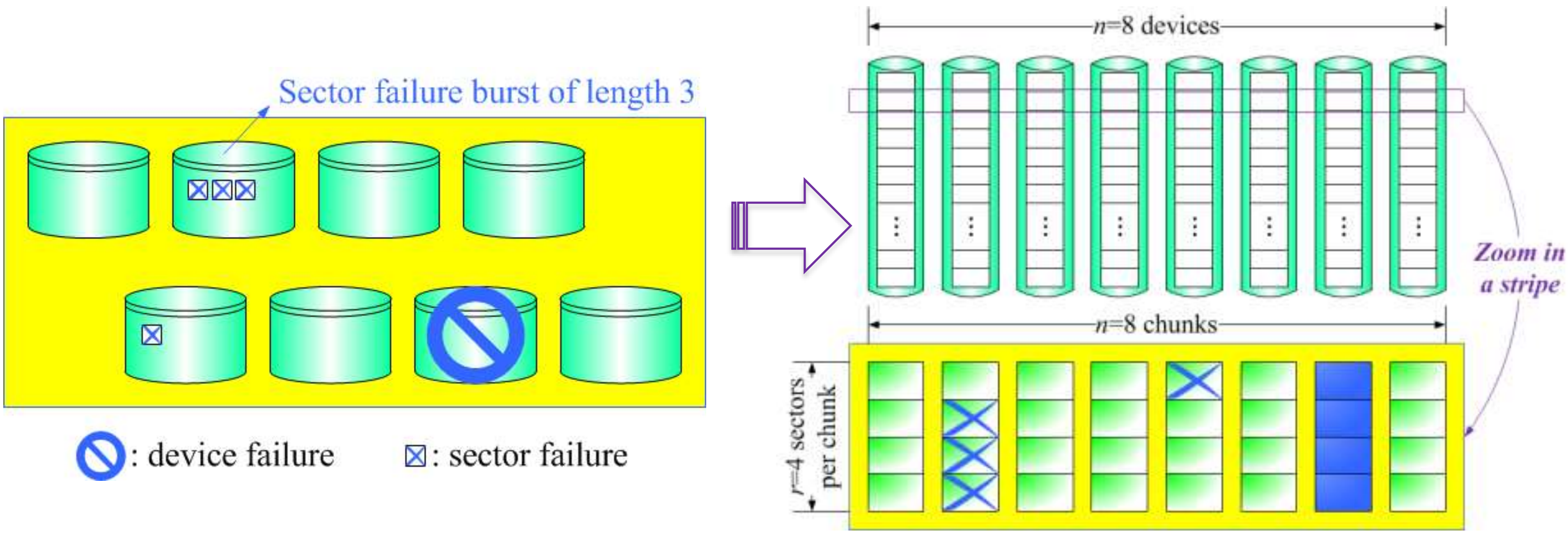
Burstiness of sector failures
[Schroeder et al., FAST '10]

Erasure Coding

- ***(N,K)* systematic MDS codes**
  - Encode K data symbols to create N-K parity symbols
  - Distribute a stripe of the N symbols across disks
  - Any K out of N symbols can recover original K data symbols
- Symbols are mapped to sectors
- RAID is one specific implementation of erasure coding

Mixed Failure Scenario

- Consider a worst-case failure scenario with
  - $m=1$ entirely failed device, and
  - $m'=2$ partially failed devices with **1** and **3** sector failures



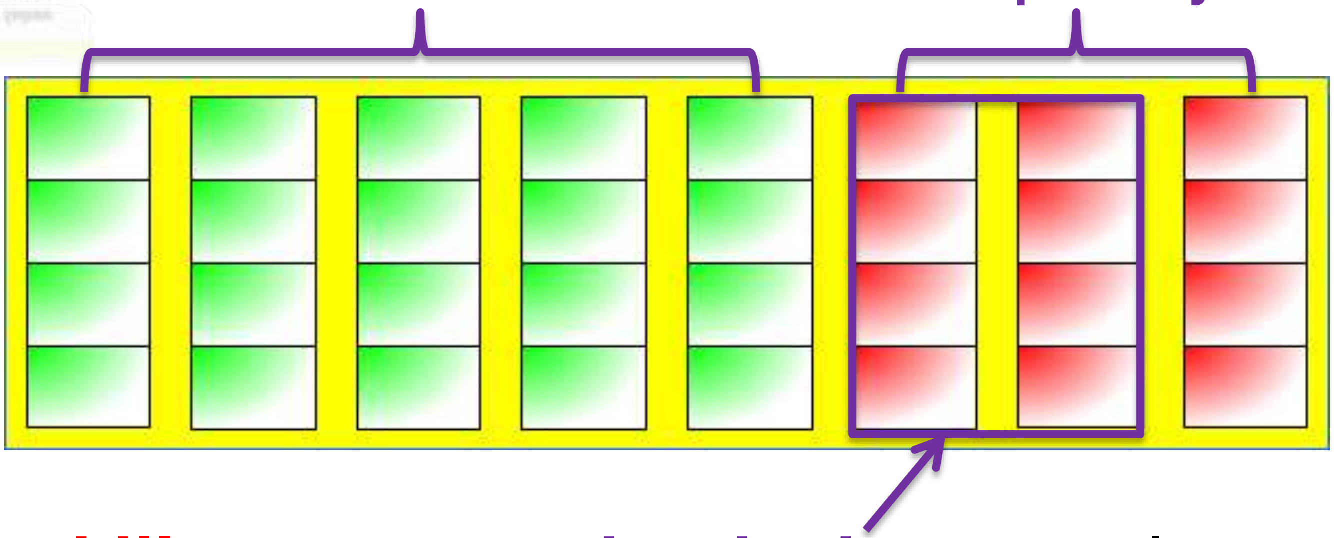
Question: ***How can we efficiently tolerate such a mixed failure scenario via erasure coding?***

RAID

3 parity devices to tolerate

- m=1 **entirely failed device**
- m'=2 **partially failed devices**

5 data devices

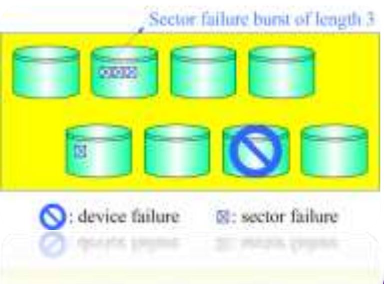


➢ **Overkill** to use **2 parity devices** to tolerate m'=2 partially failed devices

- Device-level tolerance only

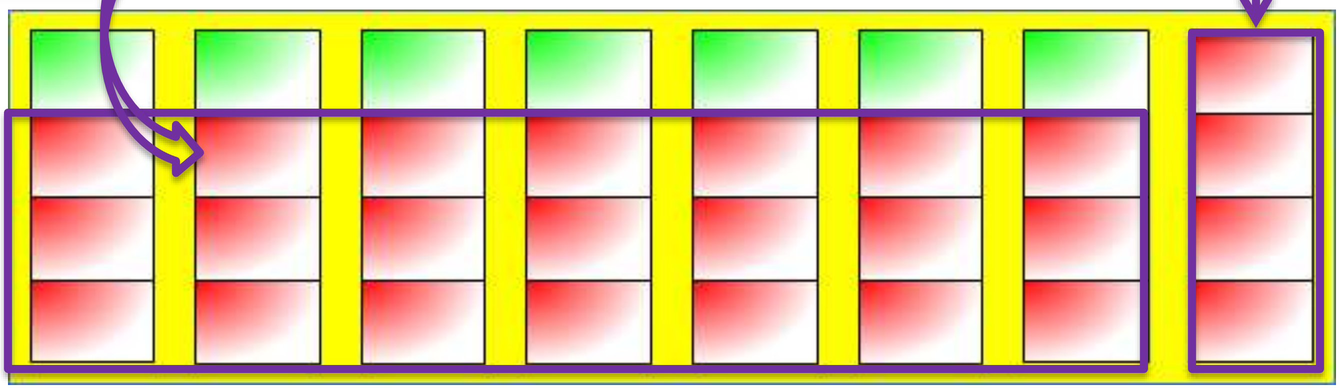
Intra-Device Redundancy (IDR)

[Dholakia et al., TOS 2008]



3 parity sectors per data device to tolerate a sector failure burst of length 3

m=1 parity device

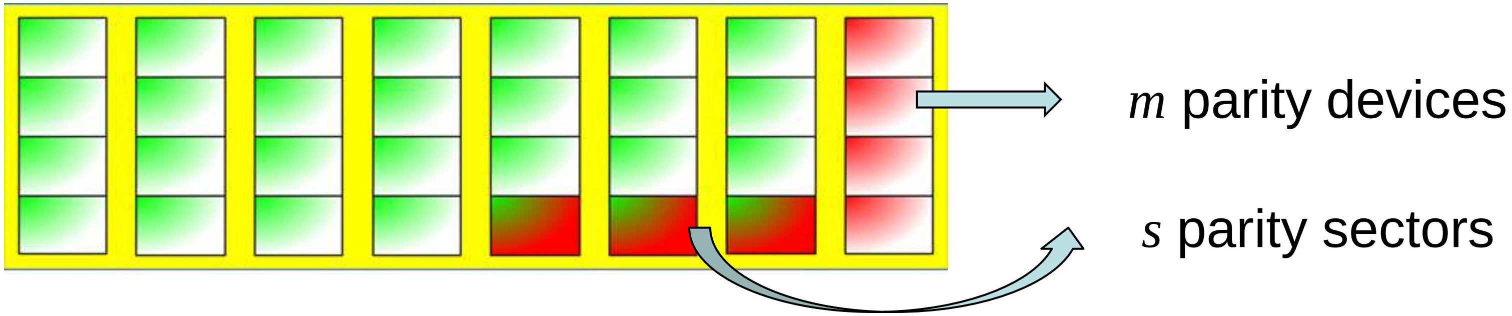


➢ Still **overkill** to add parity sectors per data device

Sector-Disk (SD) Codes

[*Plank et al., FAST '13, TOS'14*]

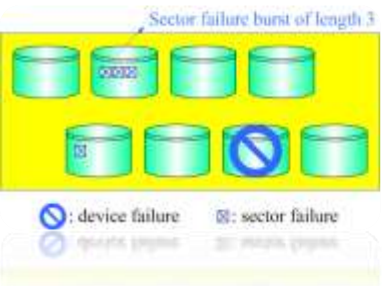
- Simultaneously tolerate
  - m entirely failed devices
  - s failed sectors (per stripe) in partially failed devices
- Construction currently limited to $s \leq 3$



- How to tolerate our mixed failure scenario?
  - m=1 entirely failed device, and
  - m'=2 partially failed devices with **1** and **3** sector failures

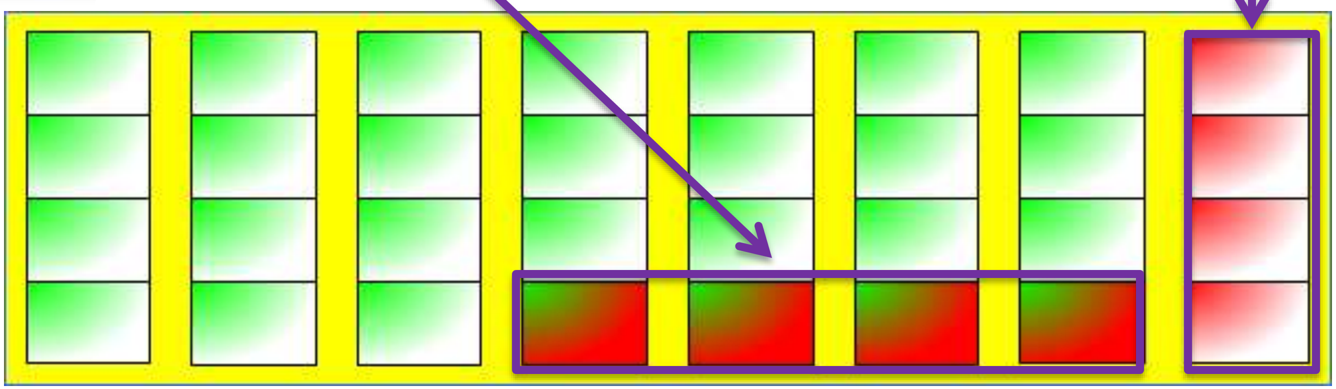
Sector-Disk (SD) Codes

[*Plank et al., FAST ’13, TOS’14*]



s=4 **global parity sectors to tolerate any 4 sector failures**

m=1 **parity device**



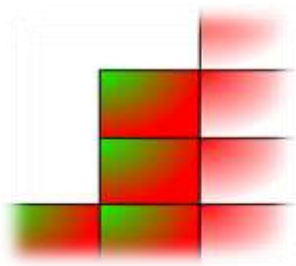
- Such an SD code is **unavailable**

[FAST’14]
[ACM Trans. Storage]

Our Work

- Construct a **general, space-efficient** family of erasure codes to tolerate both device and sector failures
  - a) **General**: without any restriction on
    - *size of a storage array,*
    - *number of tolerable device failures, or*
    - *number of tolerable sector failures*
  - b) **Space-efficient**:
    - Use parity sectors to tolerate sector failures (like SD codes)

STAIR Codes



Failure Scenarios

- RAID reconstruction performance preserved for m disk failures

- Fault tolerance for the worst-case m disk failures and a "coverage" of sector failures

Key Properties of STAIR Codes

- Sector failure coverage vector $\mathbf{e}$
  - Defines a pattern of how sector failures occur, rather than how many sector failures would occur

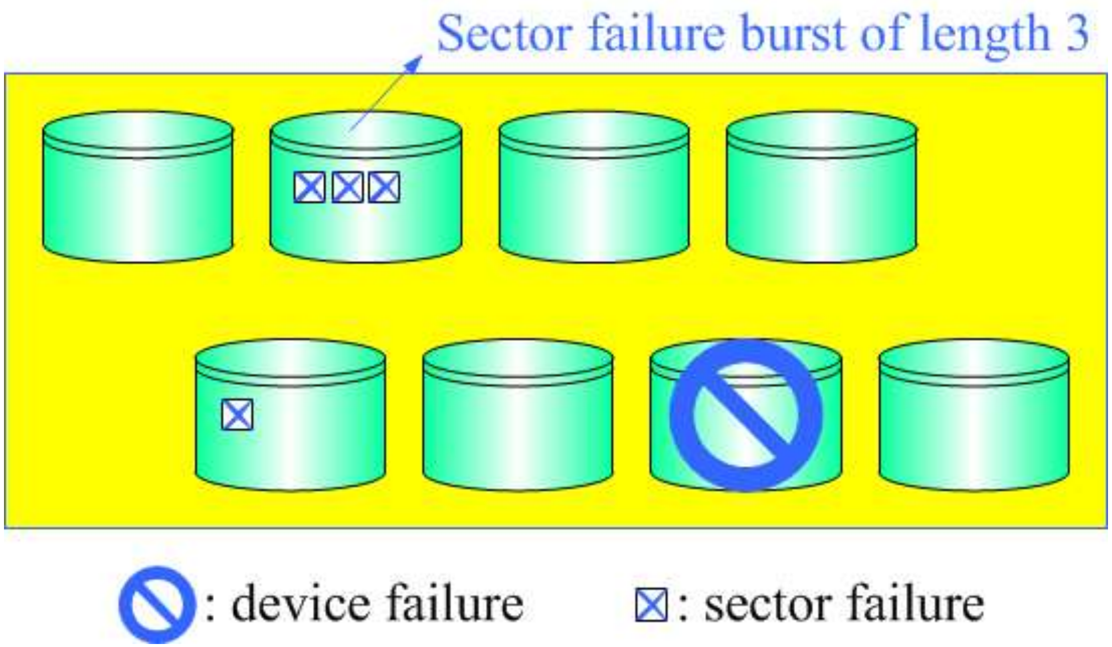
- Code structure based on two encoding phases
  - Each phase builds on an MDS code

- Two encoding methods: upstairs and downstairs encoding
  - Maintain regularity of data placement
  - Reuse computed parity results in encoding
  - Provide complementary performance gains

Sector Failure Coverage Vector

- $\mathbf{e} = (e_0, e_1, e_2, \ldots, e_{m'-1})$
  - Bounds # of partially failed devices m'
  - Bounds # of sector failures per device e_l $(0 \le l \le m'-1)$
    - $\sum e_l = s$
  - Rationale: sector failures come in small bursts
    → Can define small m' and reasonable size e_l for bursts

Sector Failure Coverage Vector

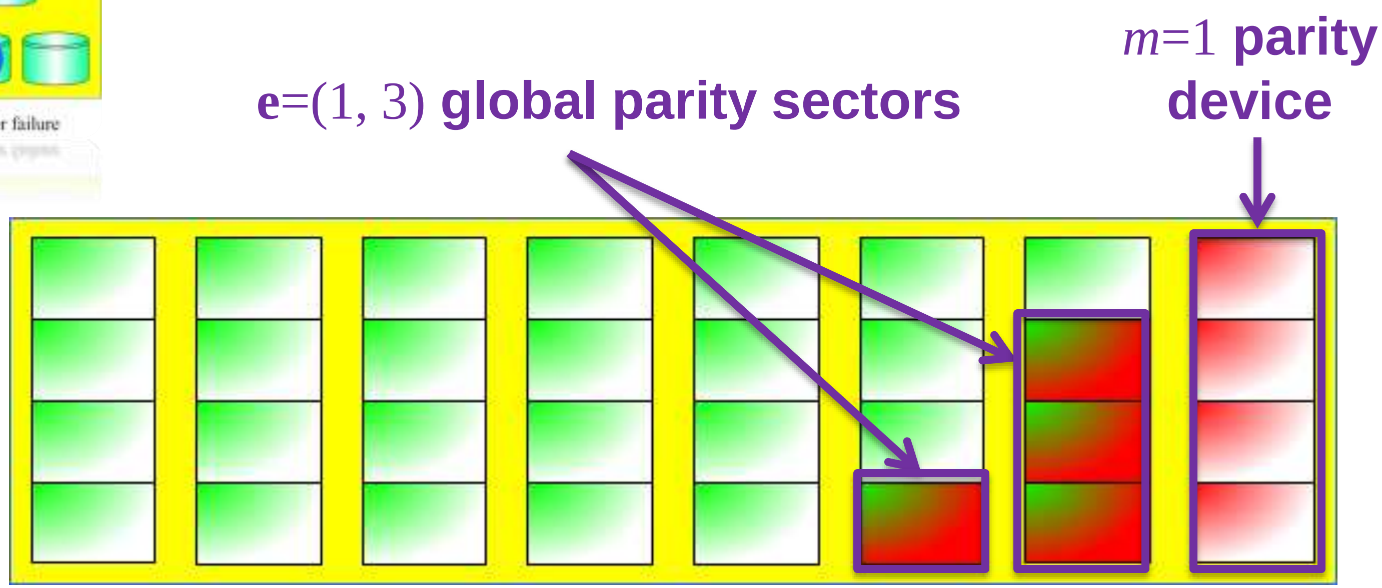


- Set $\mathbf{e}=(1, 3)$:
  - At most **2** devices (aside entirely failed devices) have sector failures
    - One device has at most **3** sector failures, and
    - Another one has at most **1** sector failure

Examples of e

- $\mathbf{e} = (1)$
  - PMDS and SD codes with $s = 1$
- $\mathbf{e} = (r)$
  - $(n, n - m - 1)$ codes (r = number of rows of a stripe)
- $\mathbf{e} = (\varepsilon, \varepsilon, \ldots, \varepsilon)$
  - Note: $m' = n - m$
  - Intra-Disk Redundancy code with ε parity symbols per column

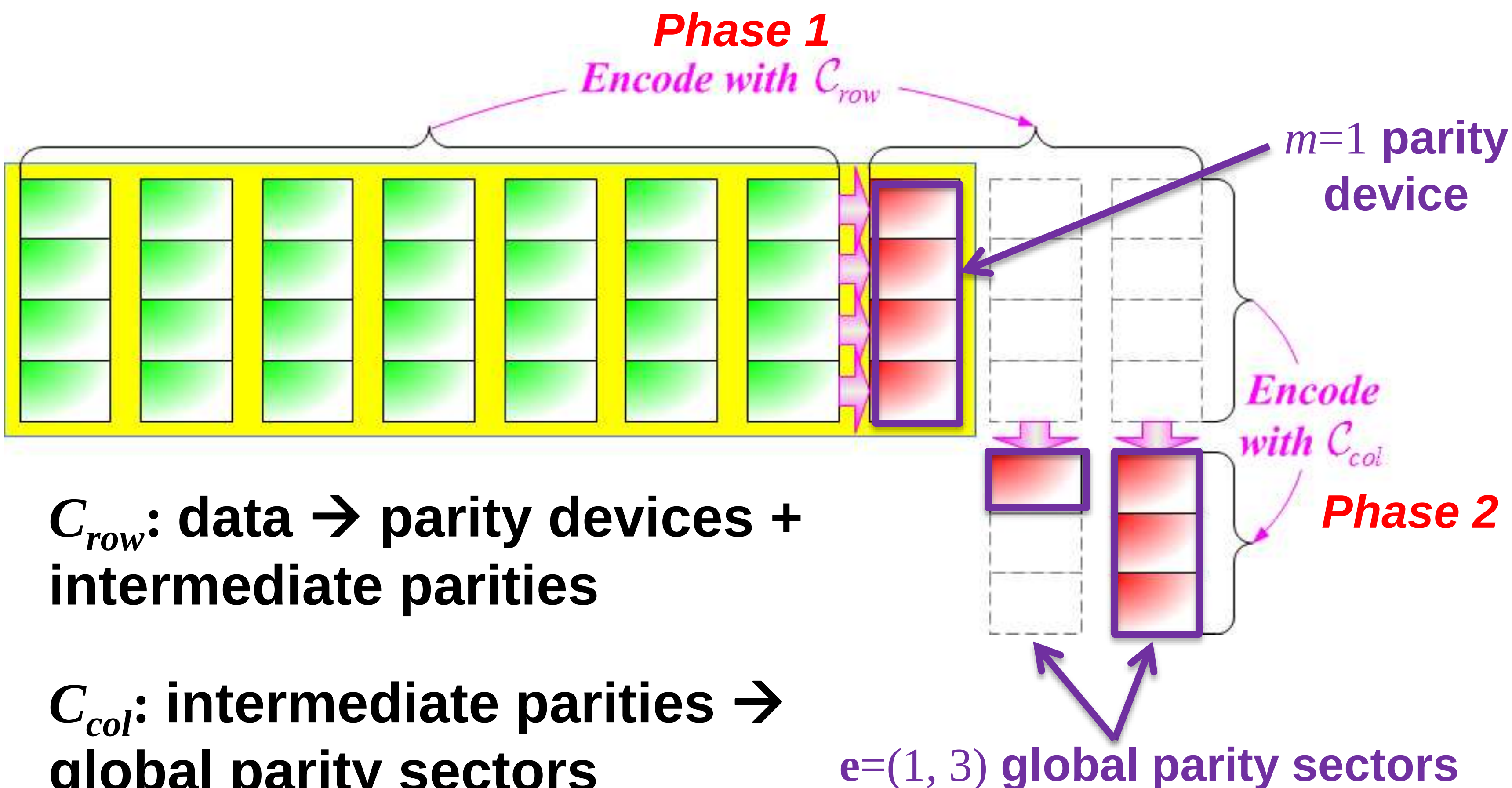
Parity Layout



- **Q:** How to generate $\mathbf{e}=(1, 3)$ global parity sectors and $m=1$ parity device?

- **A:** Use two MDS codes C_{row} and C_{col}

Two Encoding Phases

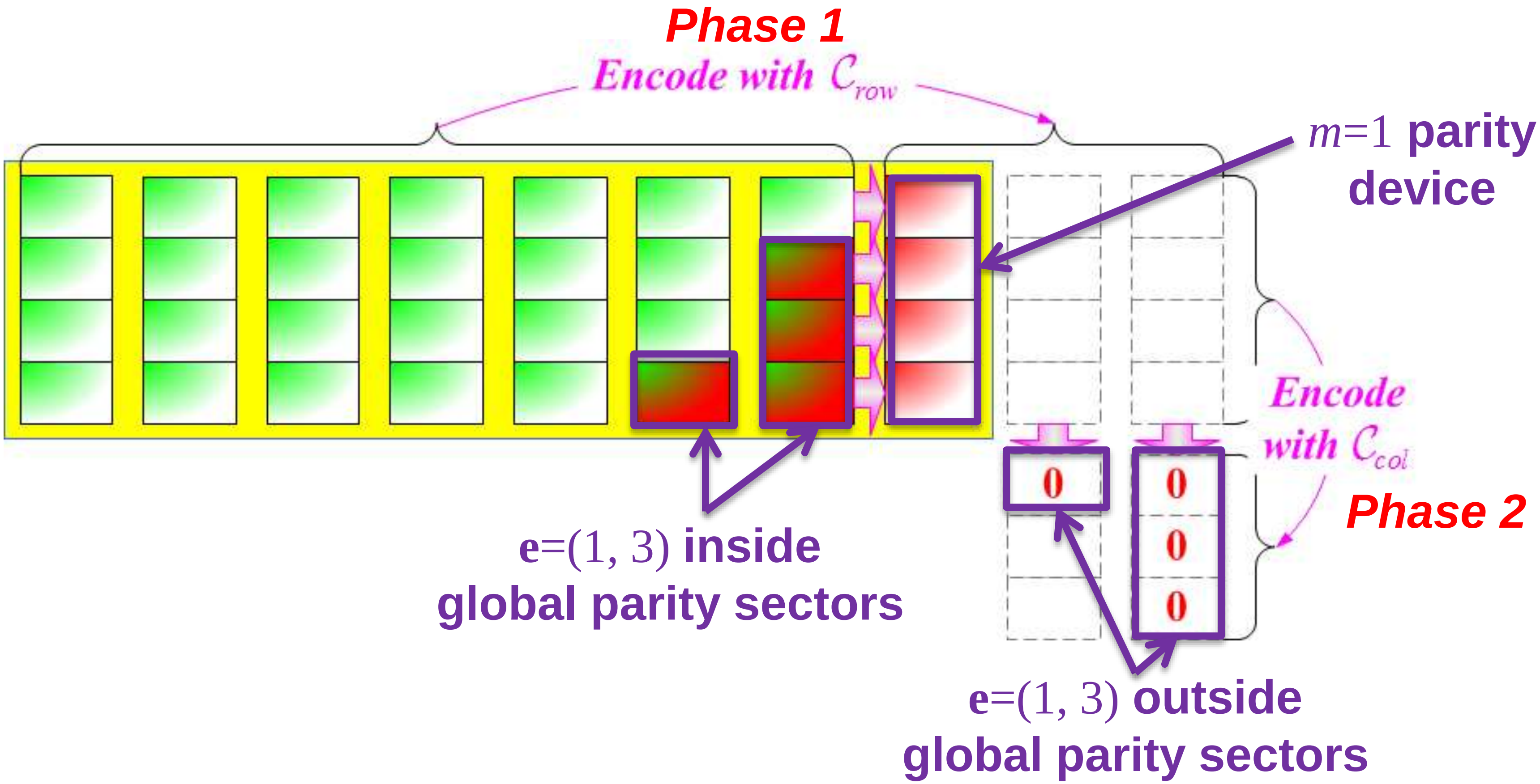


C_{row}: data → parity devices + intermediate parities

C_{col}: intermediate parities → global parity sectors

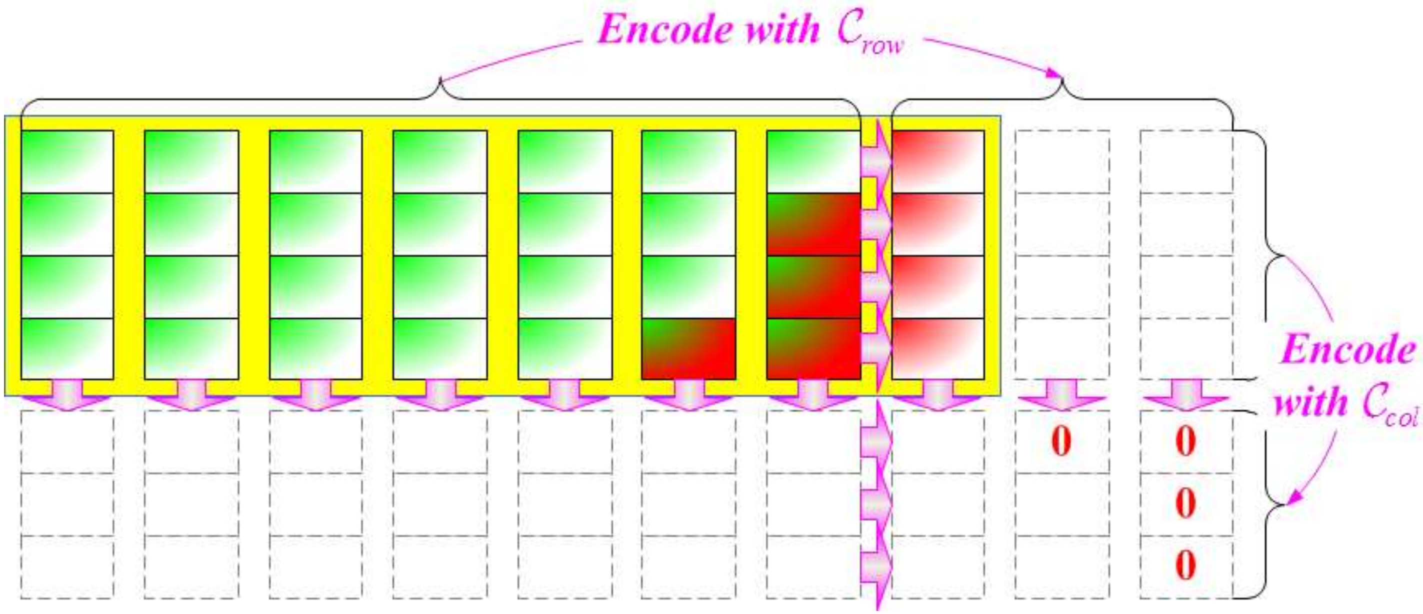
***Q:** How to keep the global parity sectors inside a stripe?*

Two Encoding Phases



- ***A:*** set outside global parity sectors as **zeroes**; reconstruct inside global parity sectors

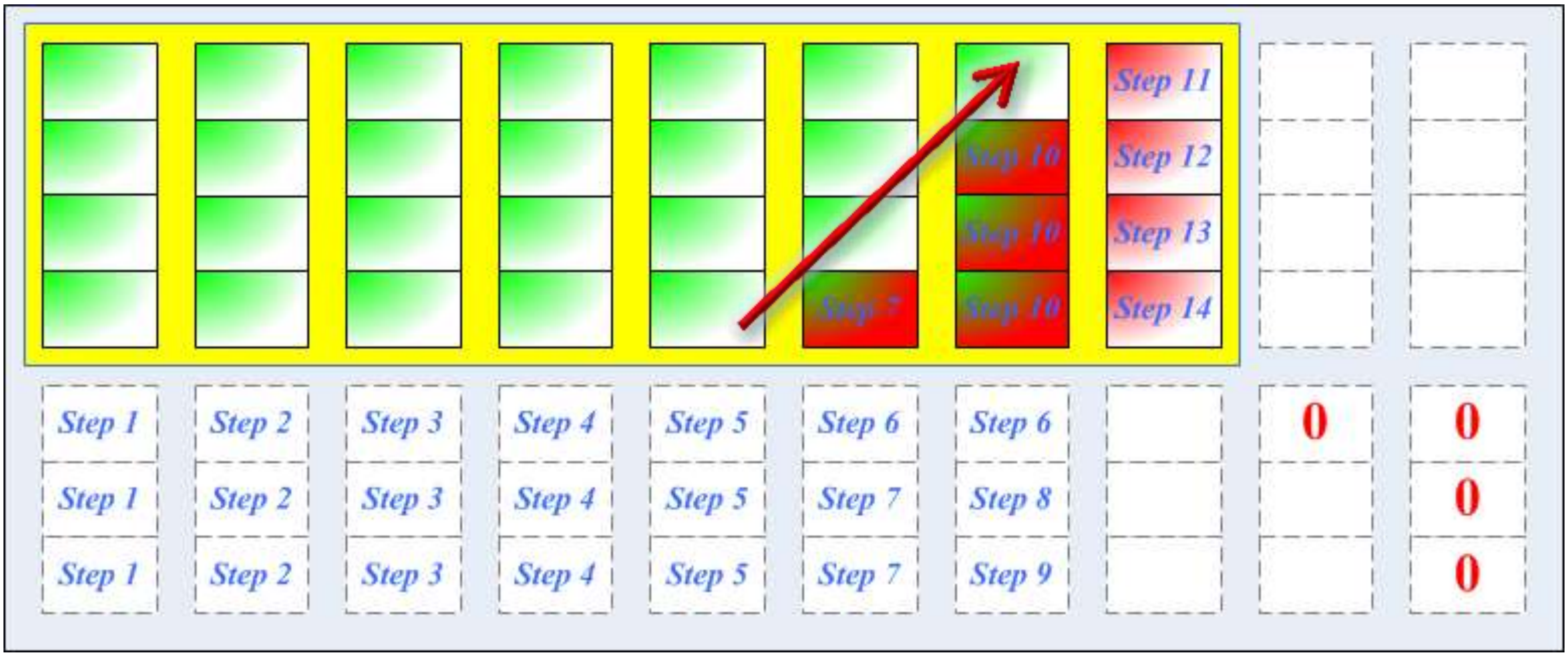
Augmented Rows



- How do we compute inside parity sectors?
  - Form a canonical stripe
- Encode each column with C_{col} to form **augmented rows**
  - Generate virtual parities in augmented rows
- Each augmented row is a codeword of C_{row}

Upstairs Encoding

- Idea: Generate parities in upstairs direction



- Can be generalized as **upstairs decoding** for recovering failures

Upstairs Encoding

- Detailed steps:

C_{row}: **(10,7) code** C_{col}: **(7,4) code**

Upstairs Encoding

- Detailed steps:

C_{row}: (10,7) code **C_{col}: (7,4) code**

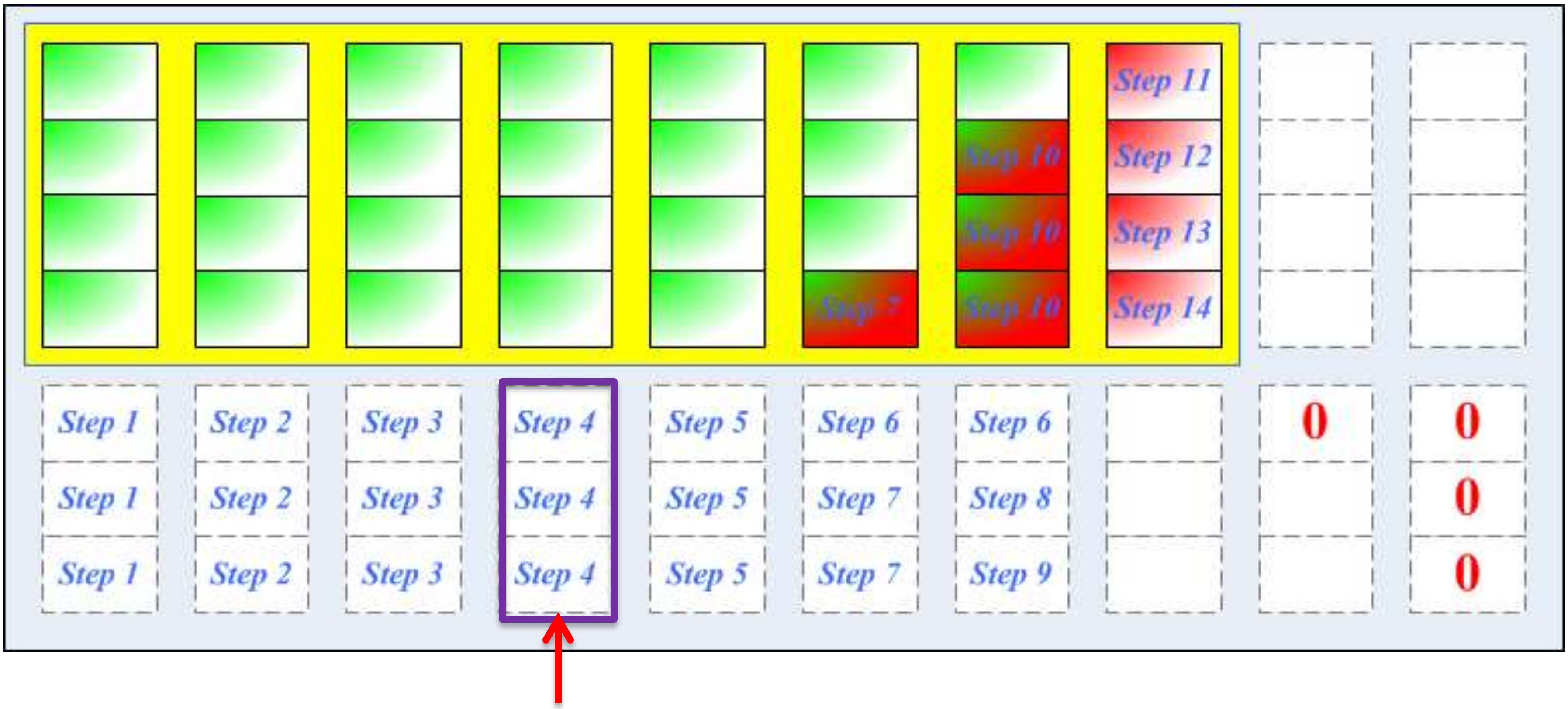
Upstairs Encoding

➢ Detailed steps:

C_{row}: **(10,7) code** C_{col}: **(7,4) code**

Upstairs Encoding

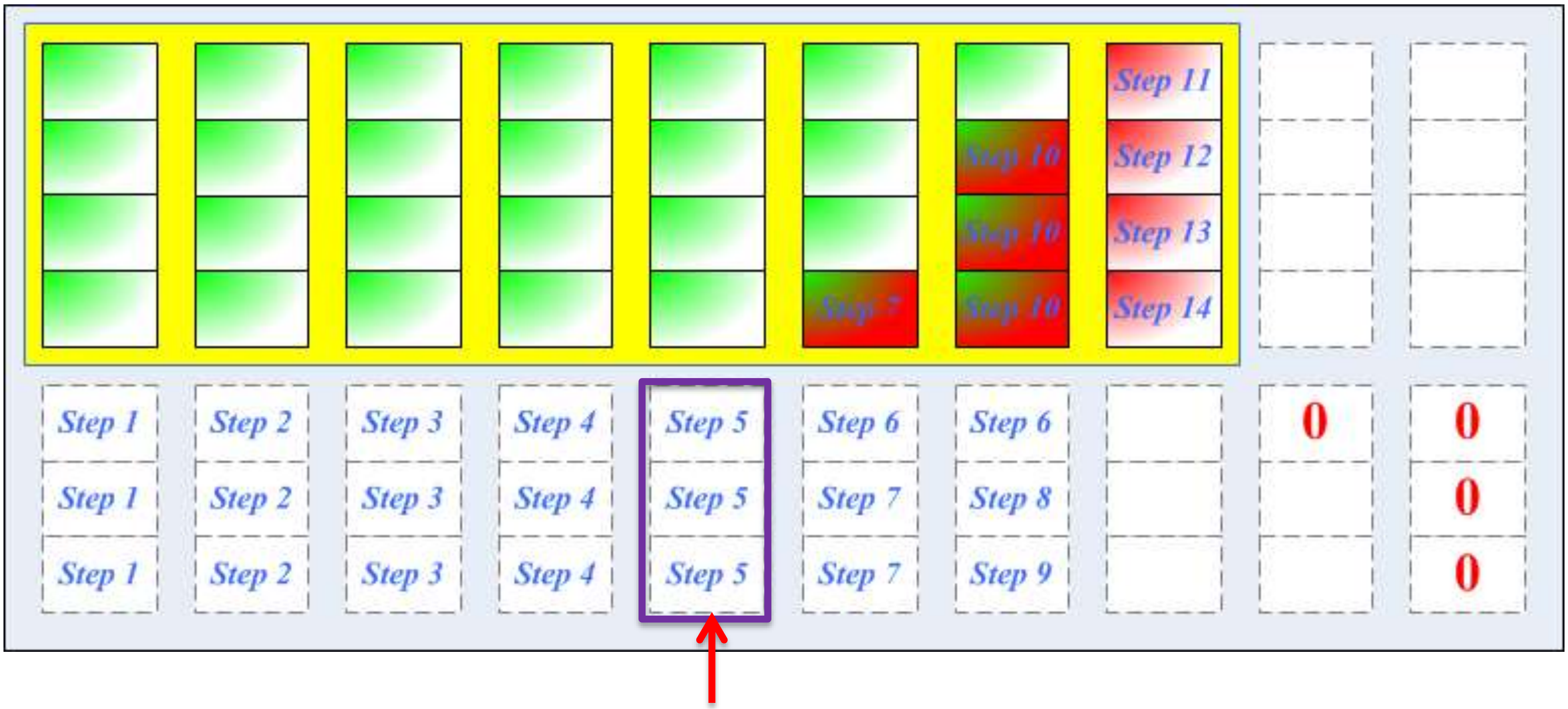
- Detailed steps:



C_{row}: **(10,7) code** C_{col}: **(7,4) code**

Upstairs Encoding

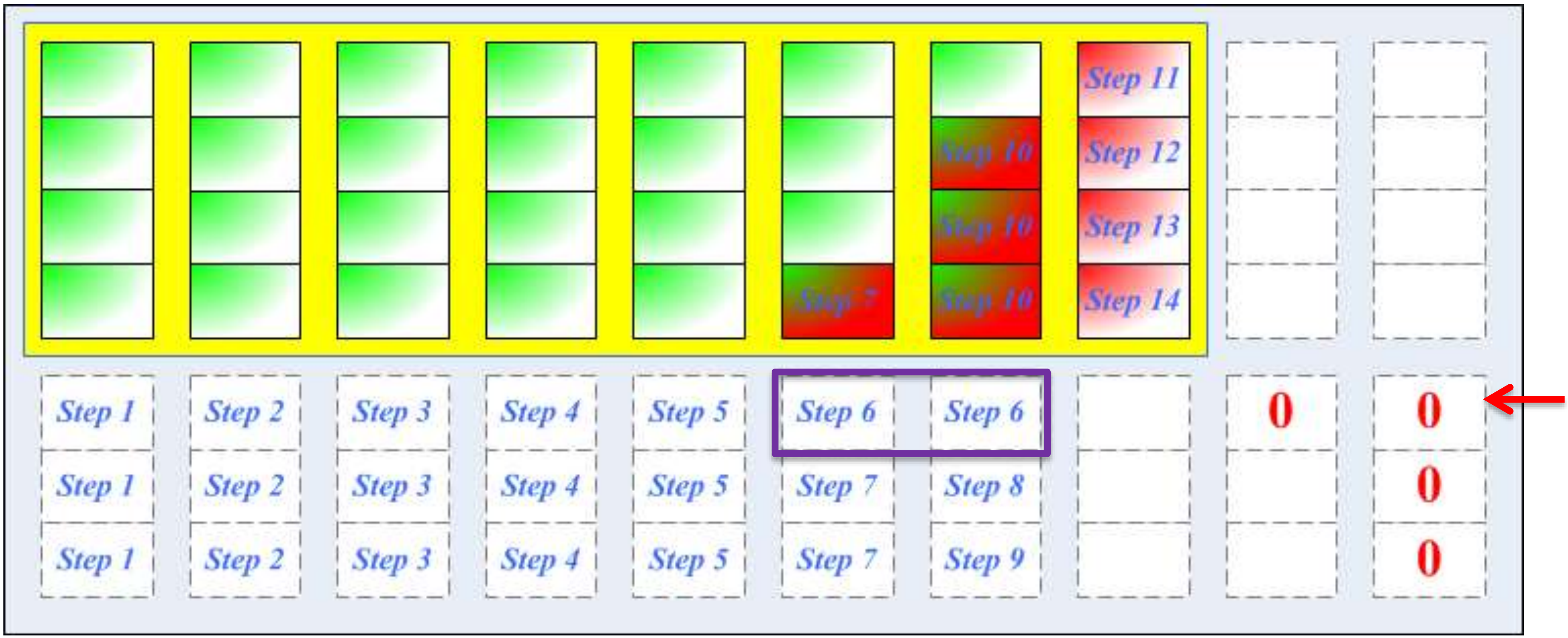
- Detailed steps:



C_{row}: (10,7) code **C_{col}: (7,4) code**

Upstairs Encoding

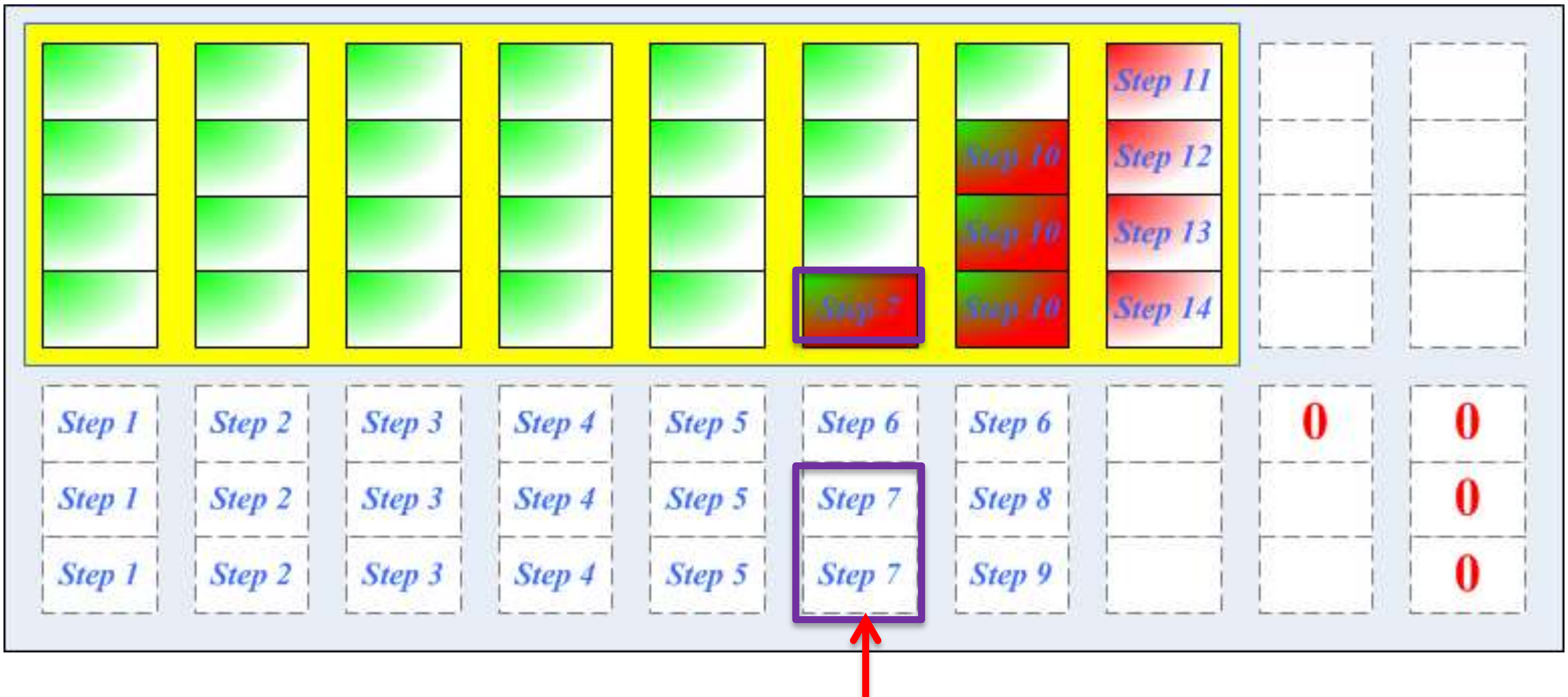
- Detailed steps:



C_{row}: **(10,7) code** C_{col}: **(7,4) code**

Upstairs Encoding

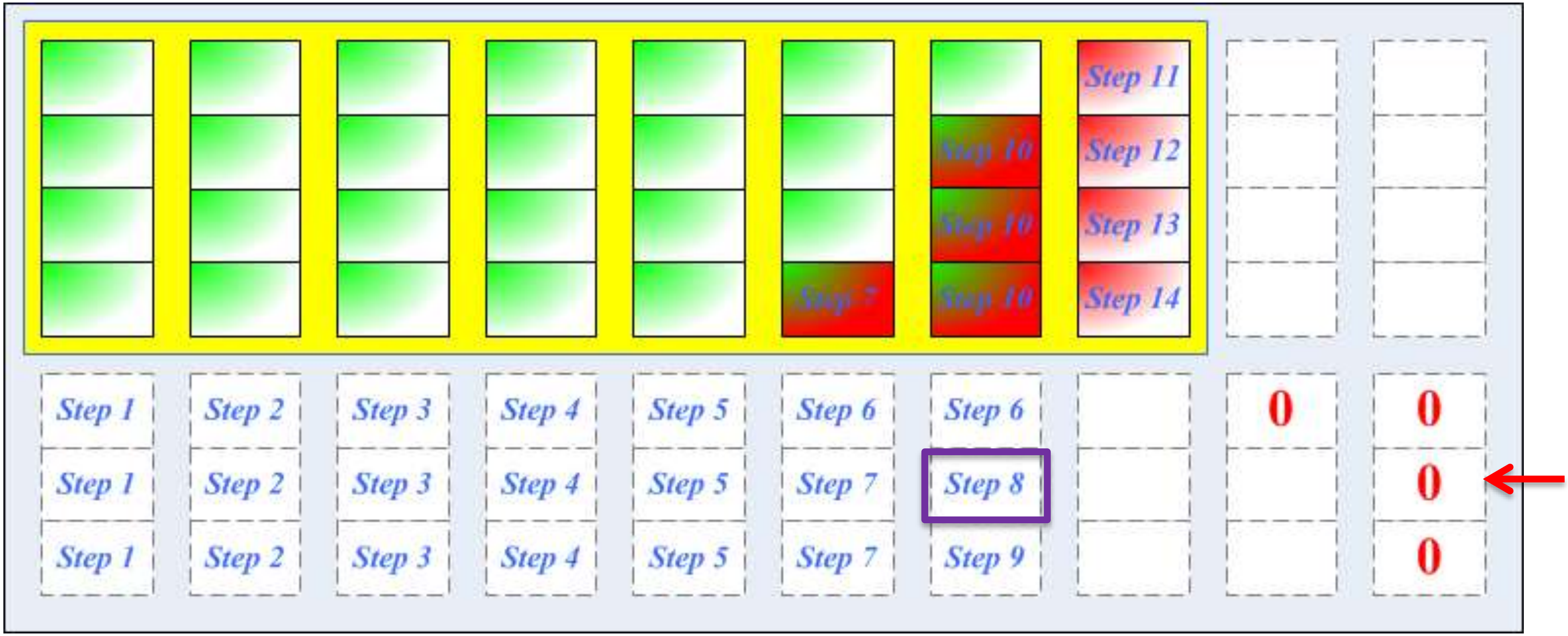
- Detailed steps:



C_{row}: (10,7) code **C_{col}: (7,4) code**

Upstairs Encoding

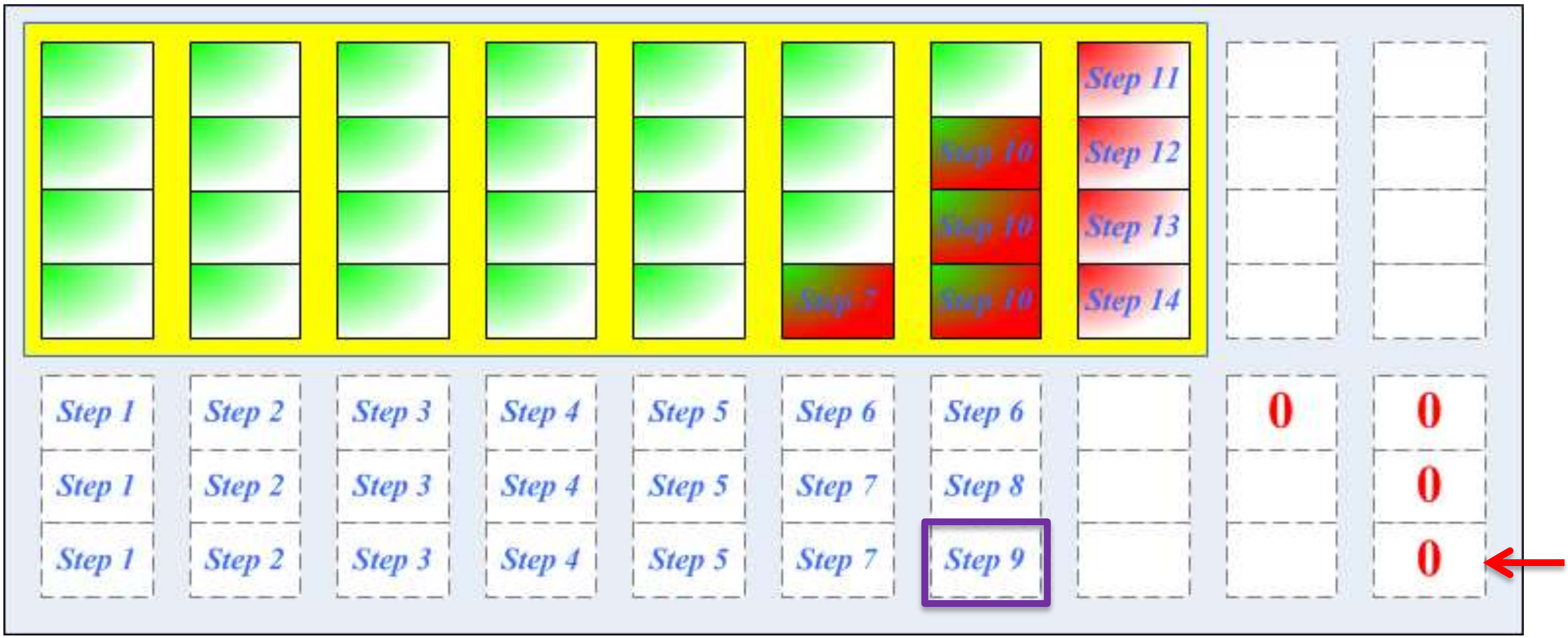
- Detailed steps:



C_{row}: **(10,7) code** C_{col}: **(7,4) code**

Upstairs Encoding

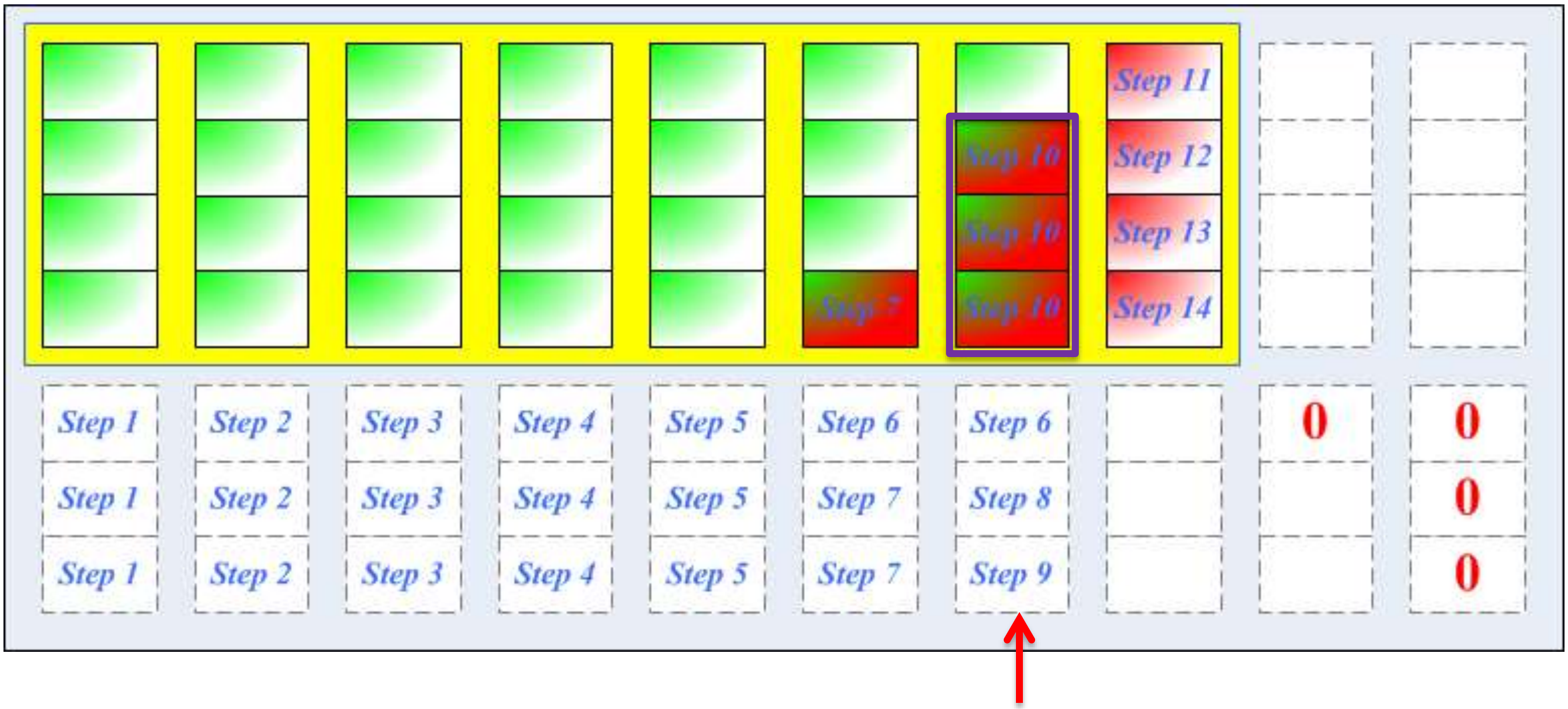
- Detailed steps:



C_{row}: **(10,7) code** C_{col}: **(7,4) code**

Upstairs Encoding

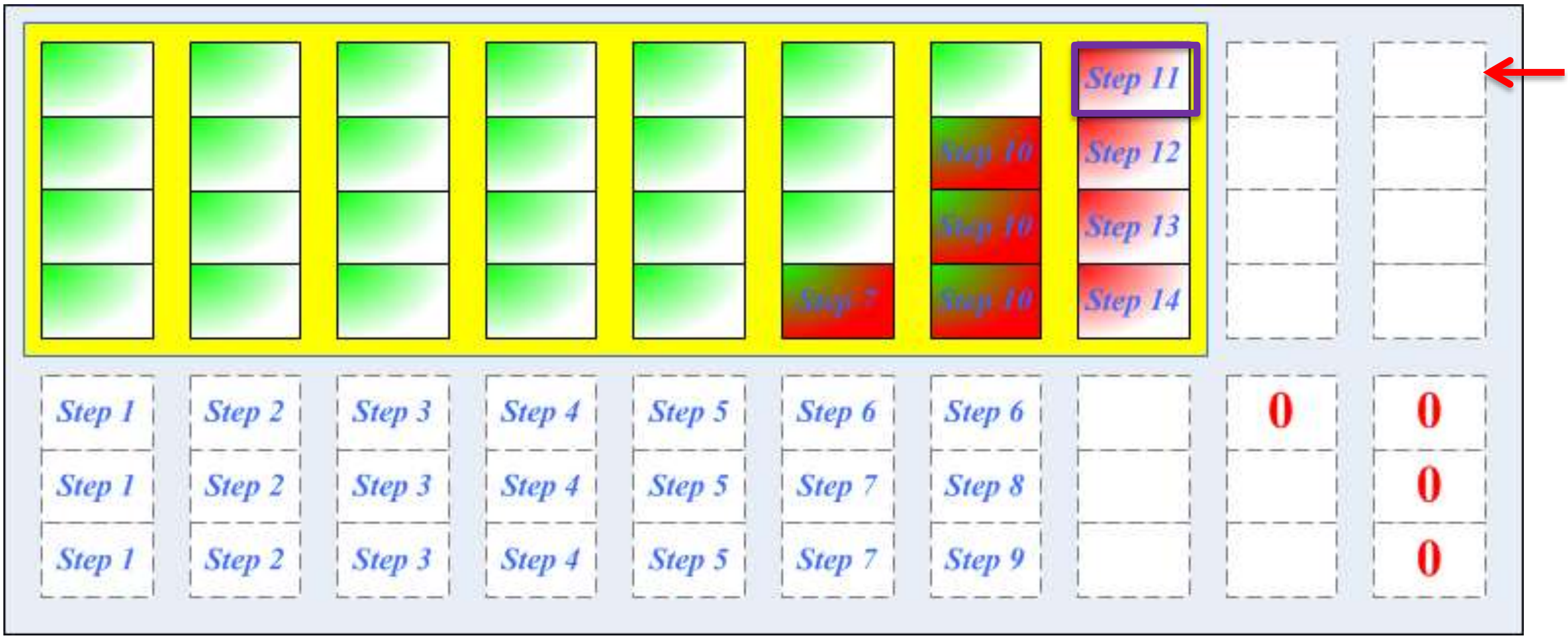
- Detailed steps:



C_{row}: **(10,7) code** C_{col}: **(7,4) code**

Upstairs Encoding

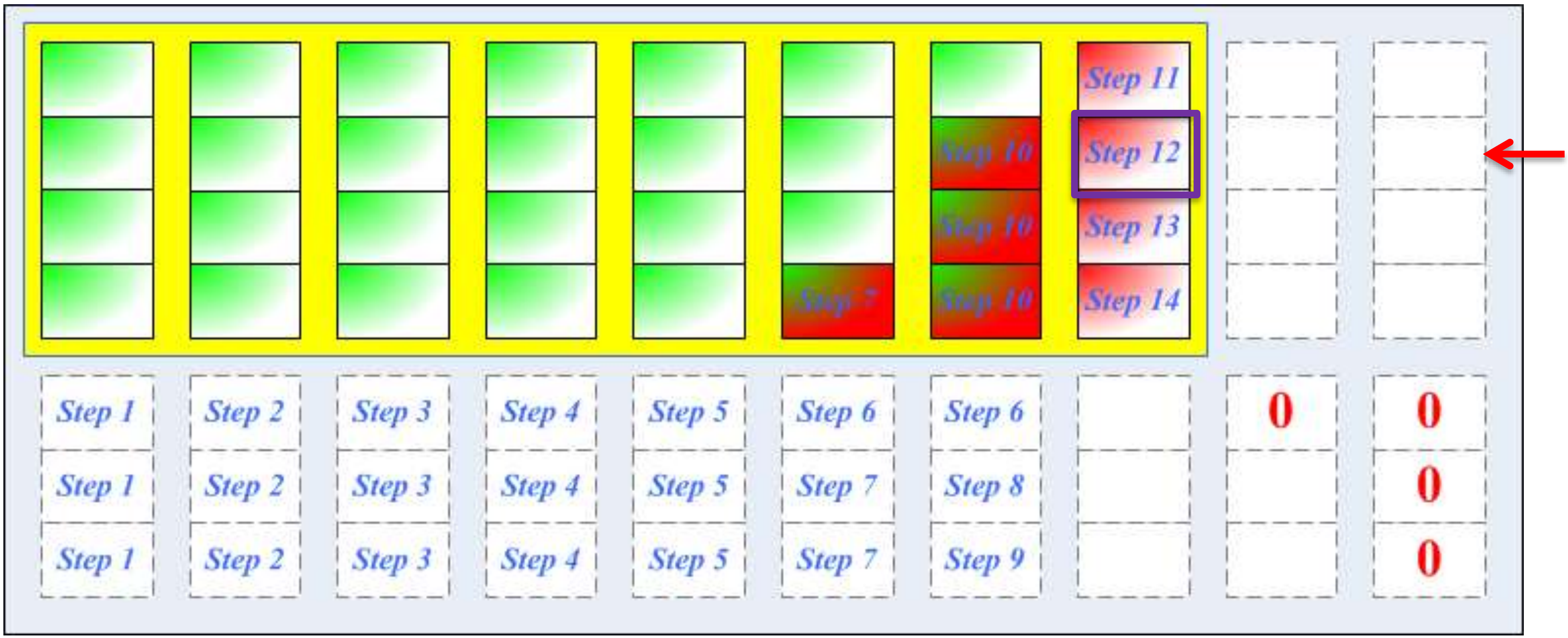
➢ Detailed steps:



C_{row}: **(10,7) code** C_{col}: **(7,4) code**

Upstairs Encoding

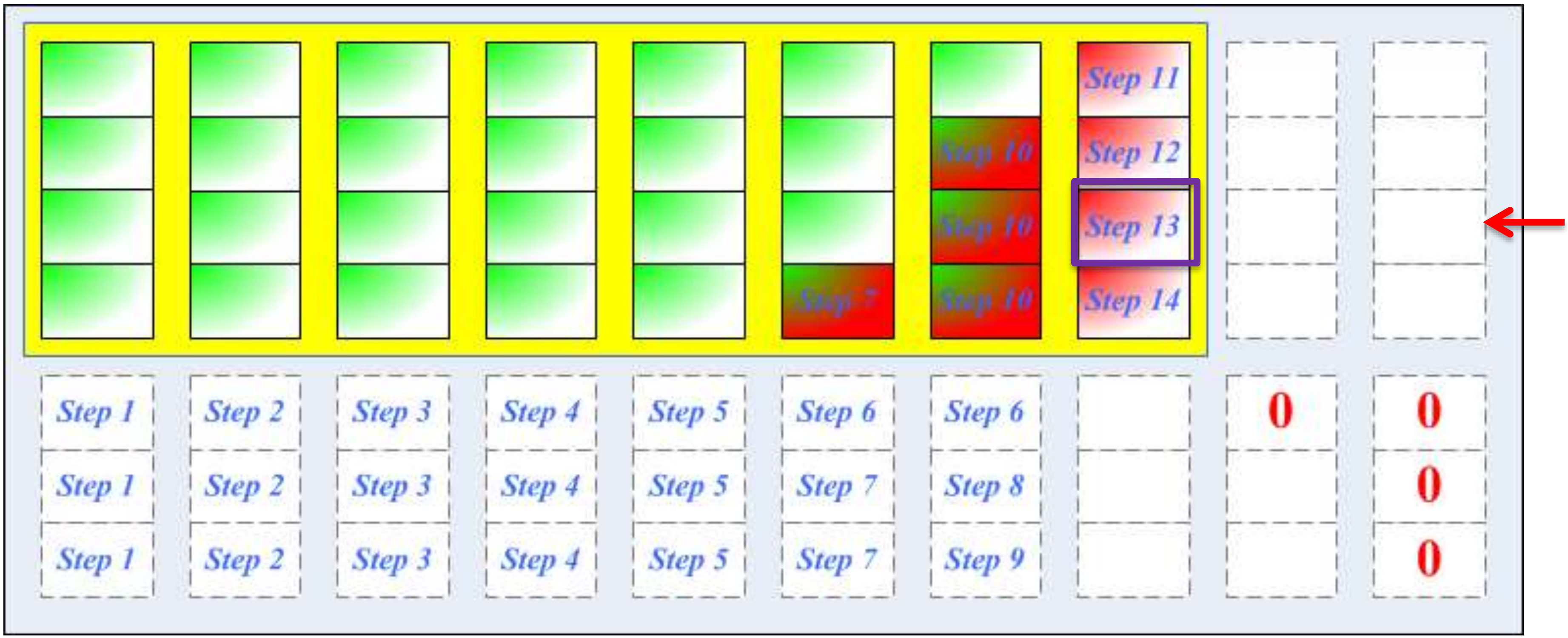
- Detailed steps:



C_{row}: (10,7) code **C_{col}: (7,4) code**

Upstairs Encoding

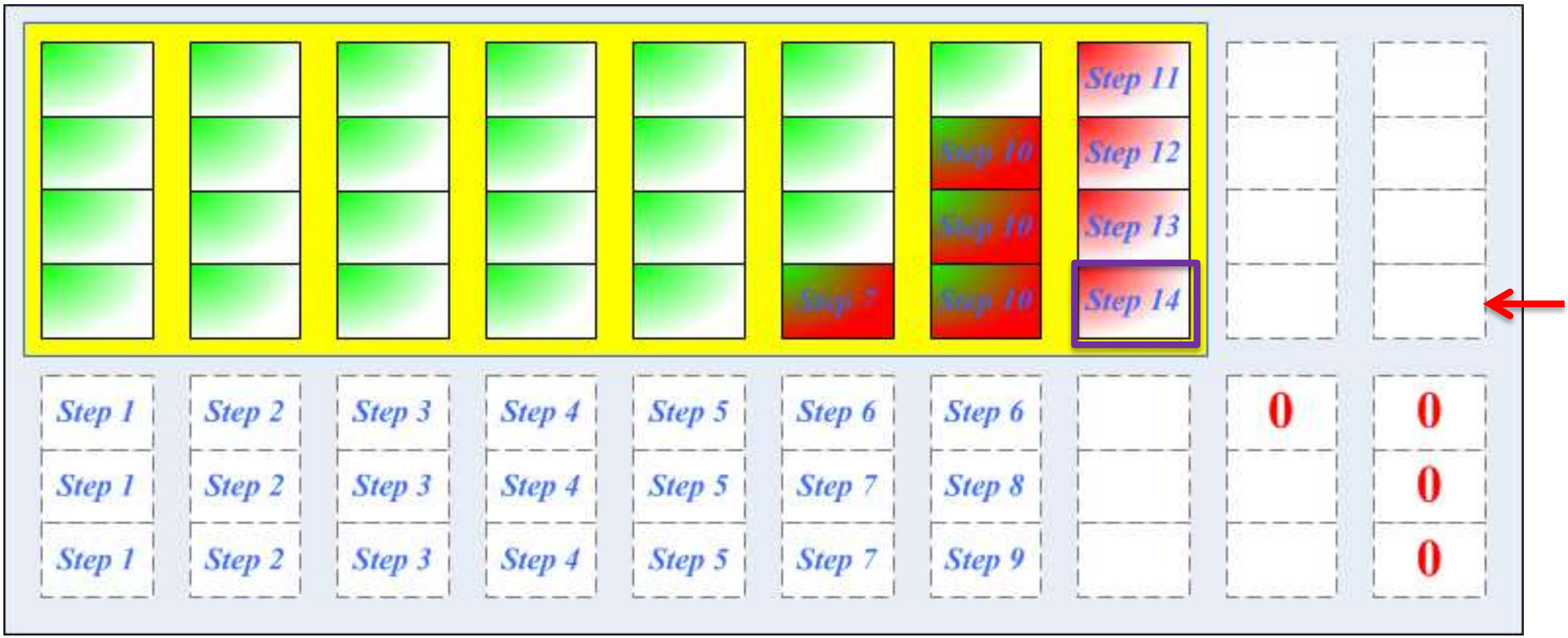
➢ Detailed steps:



C_{row}: **(10,7) code** C_{col}: **(7,4) code**

Upstairs Encoding

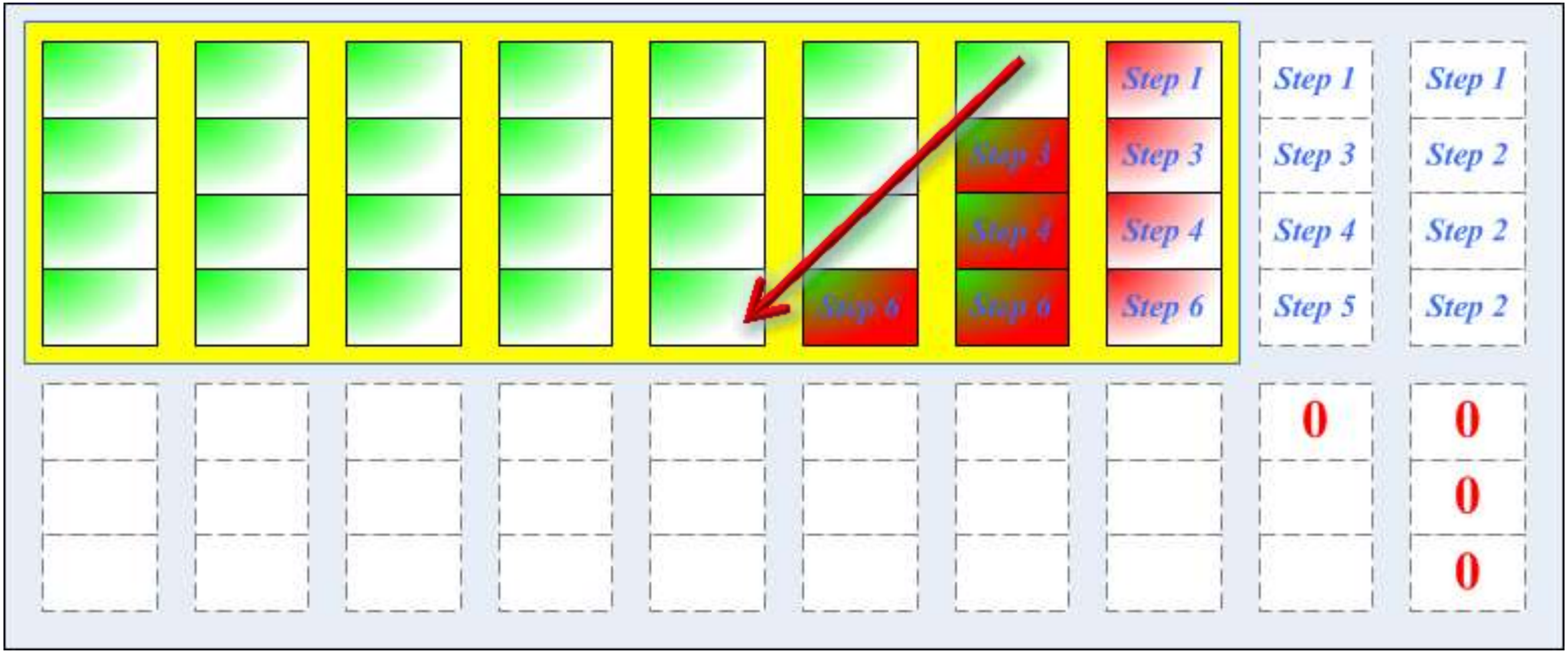
- Detailed steps:



Notes: parity computations reuse previously computed parities

Downstairs Encoding

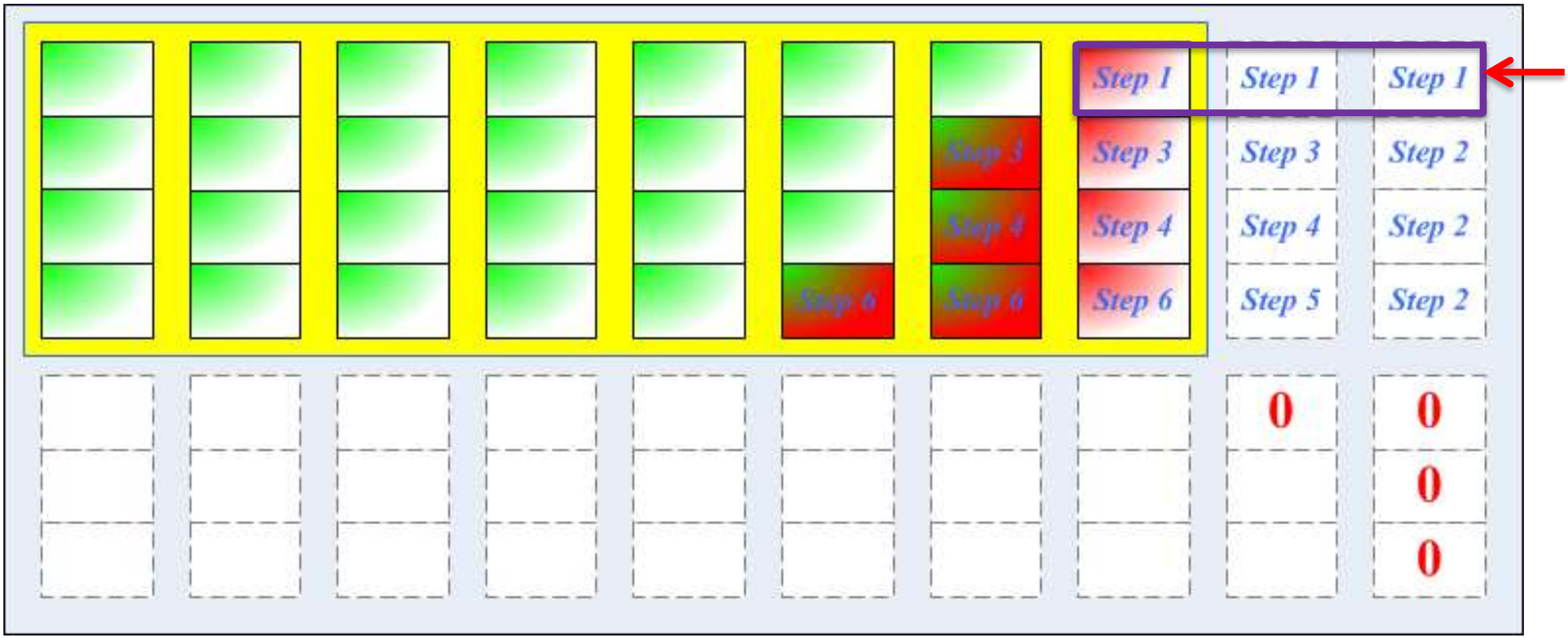
- ➢ Another idea: Generate parities in downstairs direction



- ➢ **Cannot** be generalized for decoding

Downstairs Encoding

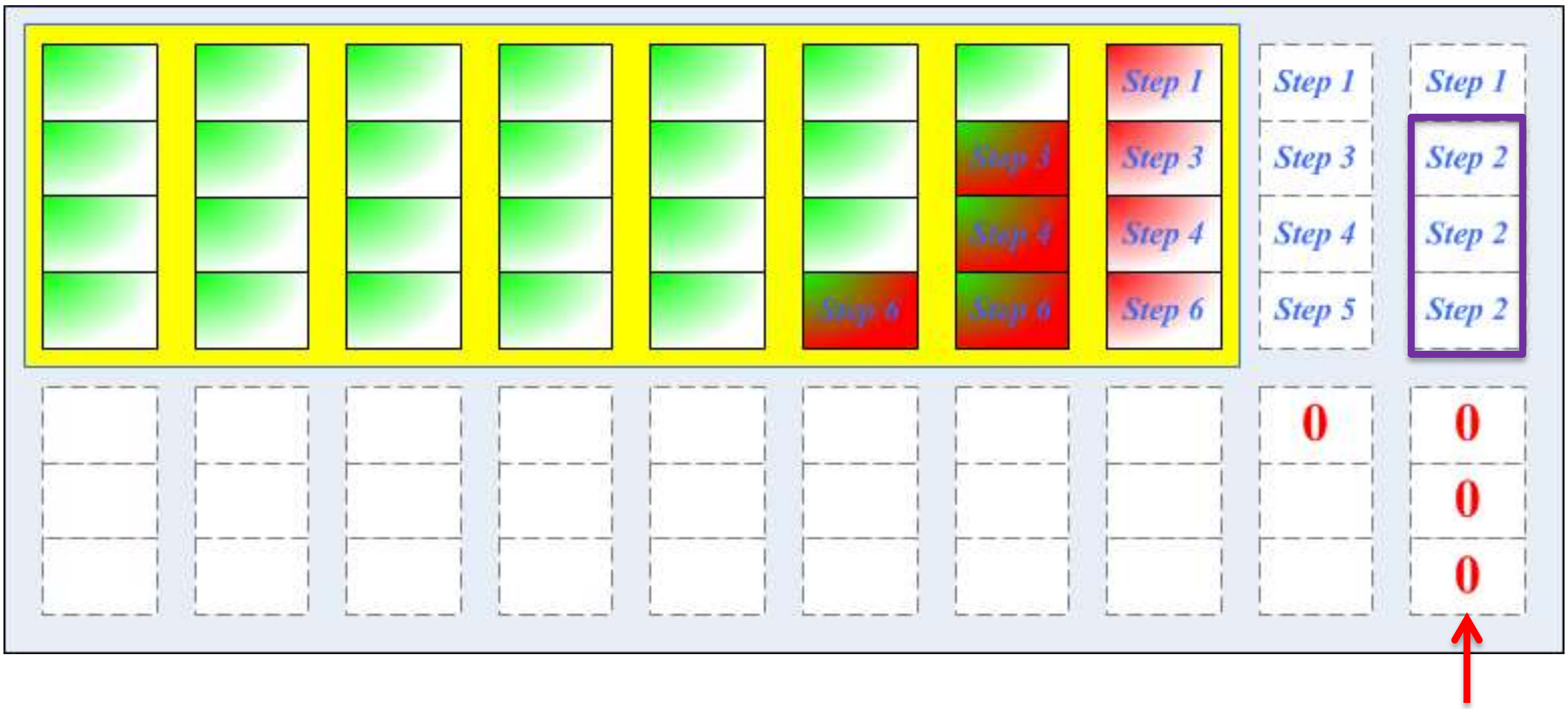
- Detailed steps:



C_{row}: **(10,7) code** C_{col}: **(7,4) code**

Downstairs Encoding

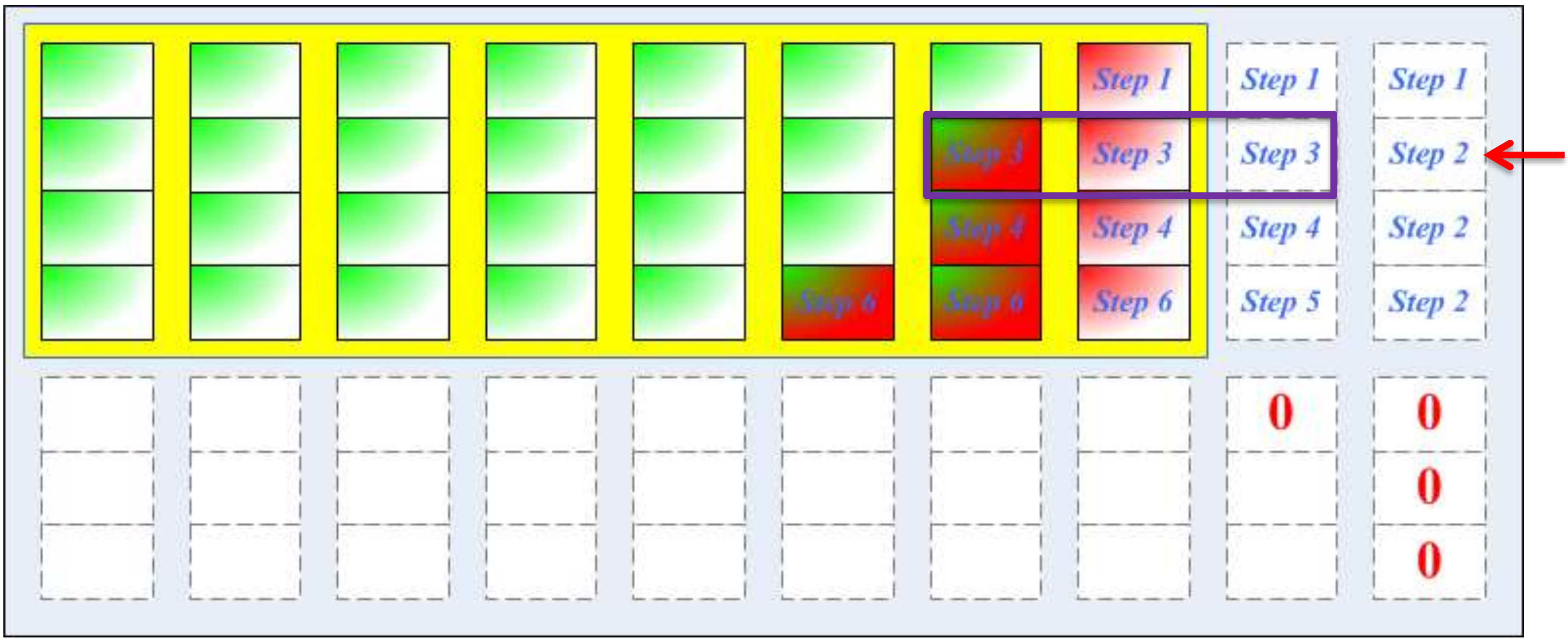
- Detailed steps:



C_{row}: **(10,7) code** C_{col}: **(7,4) code**

Downstairs Encoding

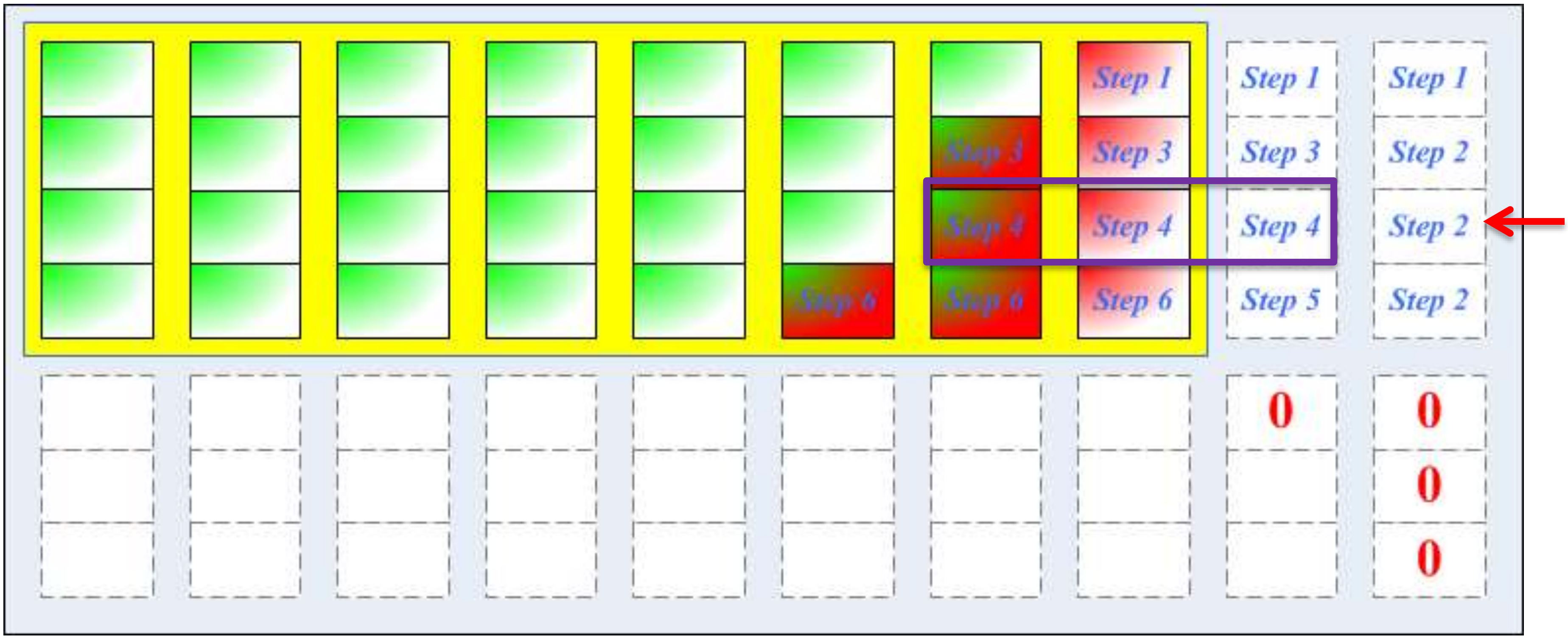
- Detailed steps:



C_{row}: **(10,7) code** C_{col}: **(7,4) code**

Downstairs Encoding

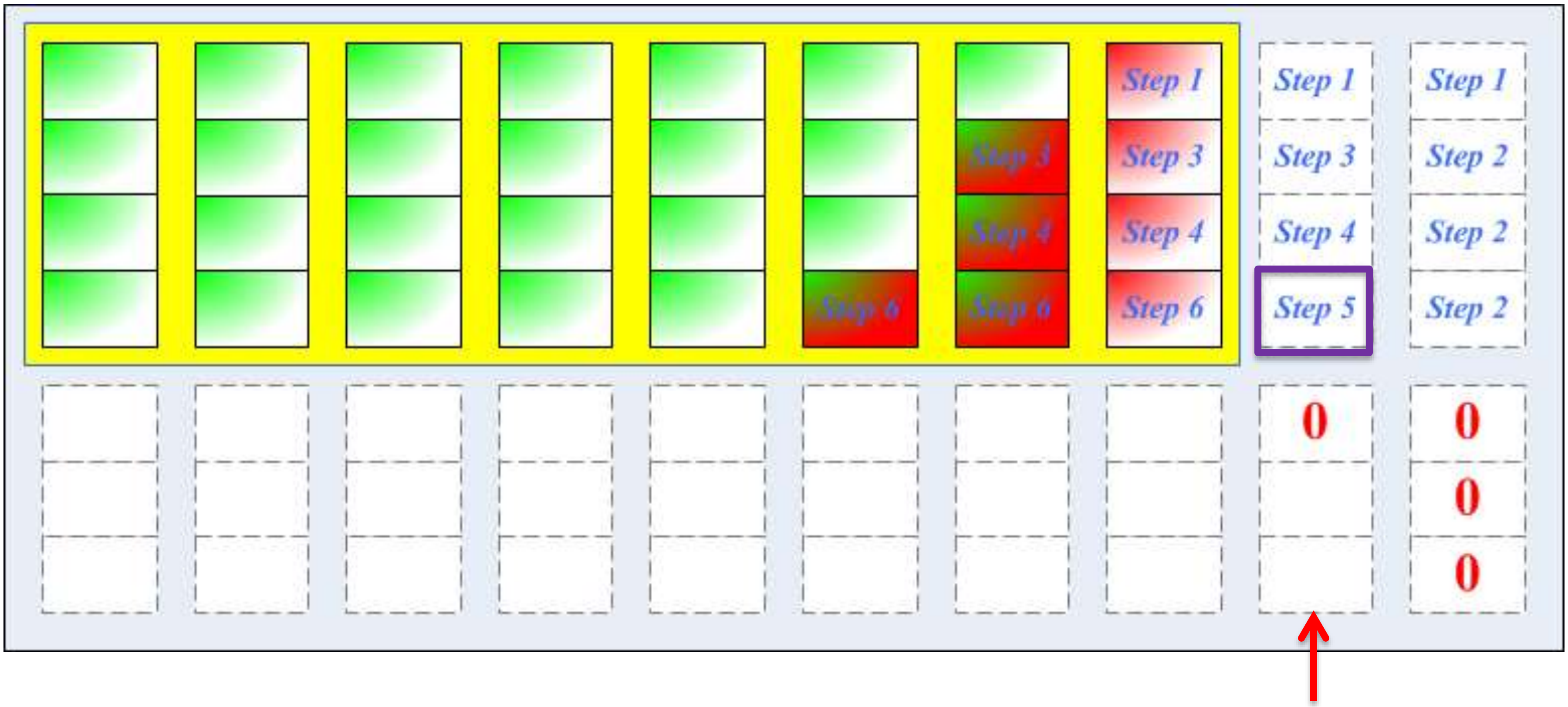
- Detailed steps:



C_{row}: (10,7) code **C_{col}: (7,4) code**

Downstairs Encoding

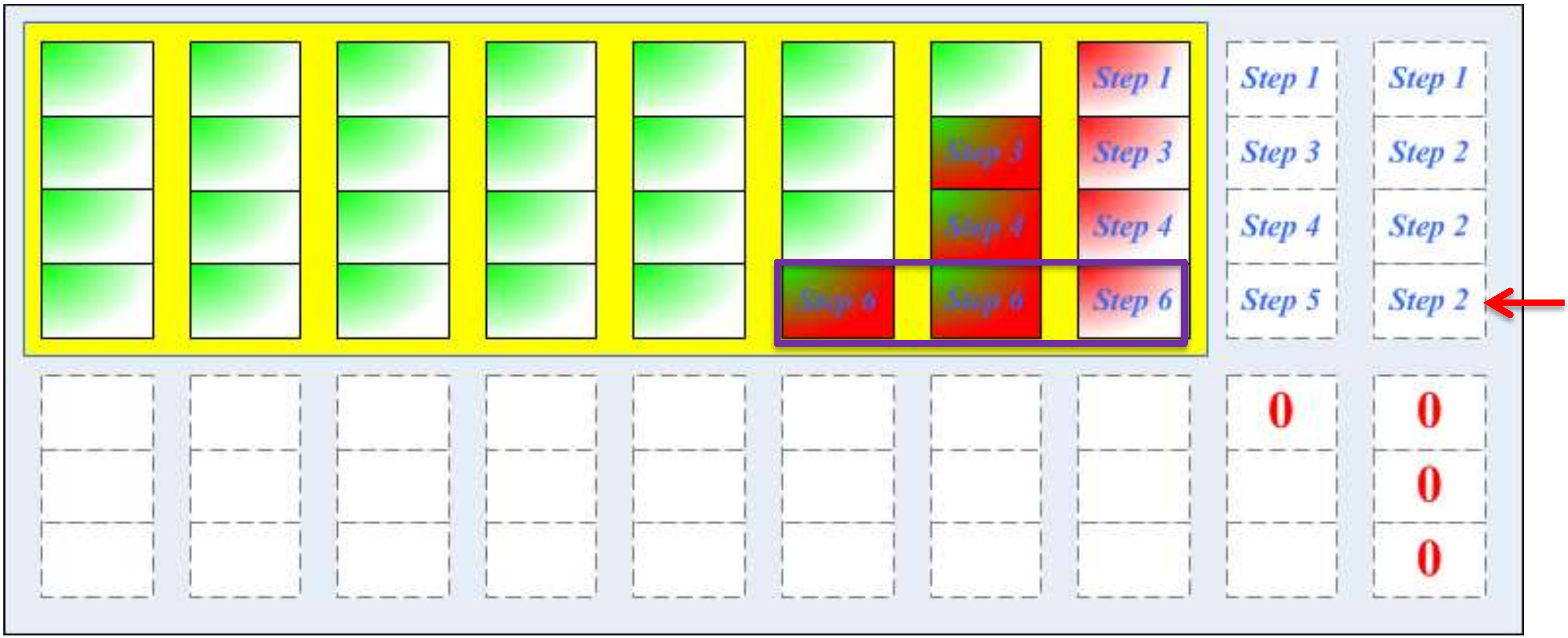
- Detailed steps:



C_{row}: **(10,7) code** C_{col}: **(7,4) code**

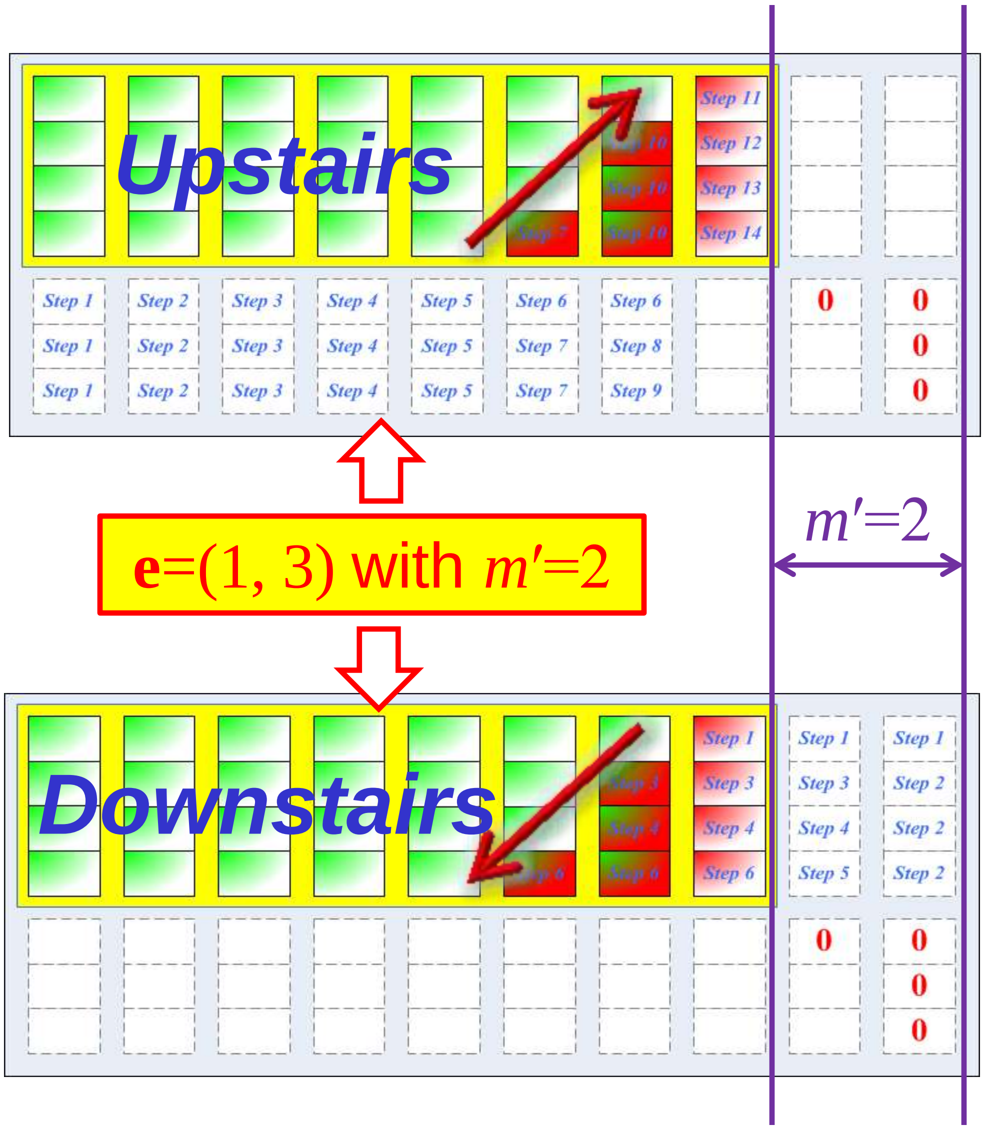
Downstairs Encoding

- Detailed steps:



Like upstairs encoding, parity computations reuse previously computed parities

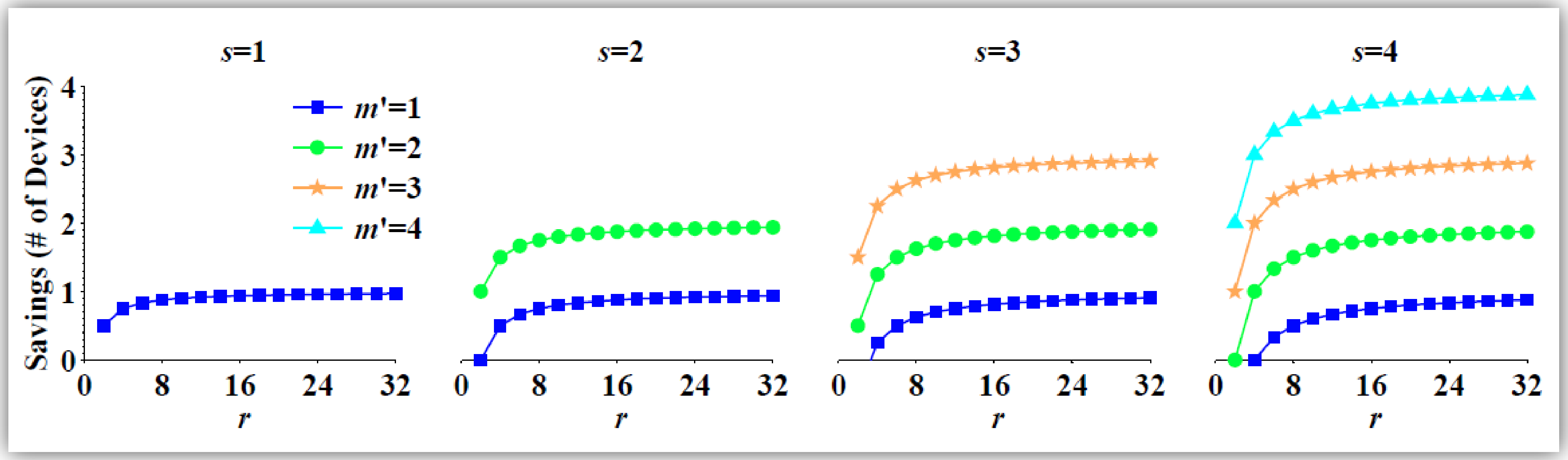
Choosing Encoding Methods



- The two methods are **complementary**
- Intuition:
  - Choose upstairs encoding for large m'
  - Choose downstairs encoding for small m'
- Analysis details in the paper

Storage Space Saving

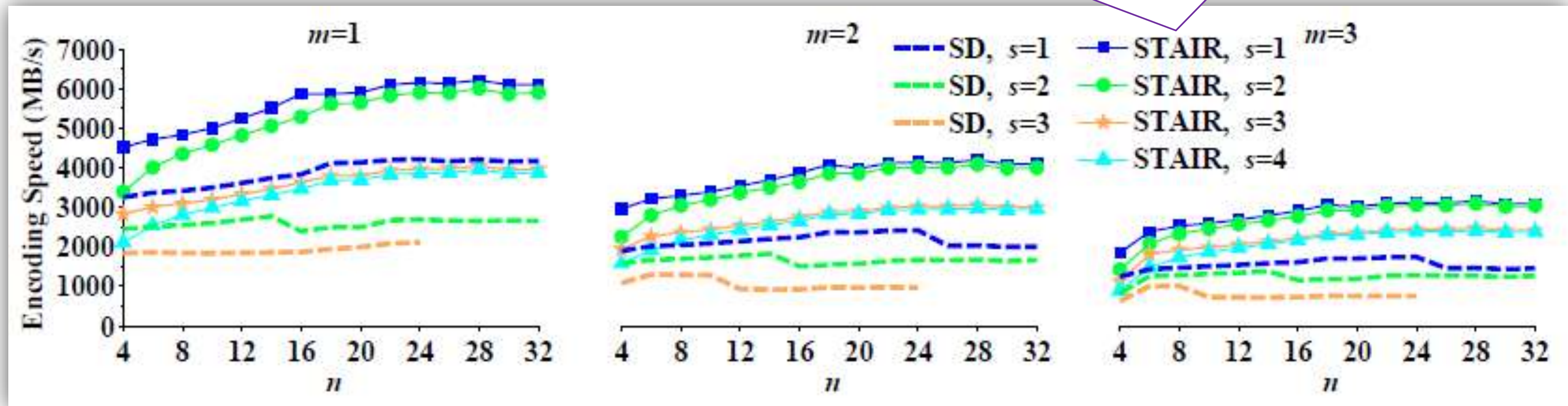
- STAIR codes save $m' - \frac{s}{r}$ devices over RAID
  - s = # of tolerable sector failures
  - m' = # of partially failed devices
  - r = chunk size



As r increases, # of devices saved ~ m'

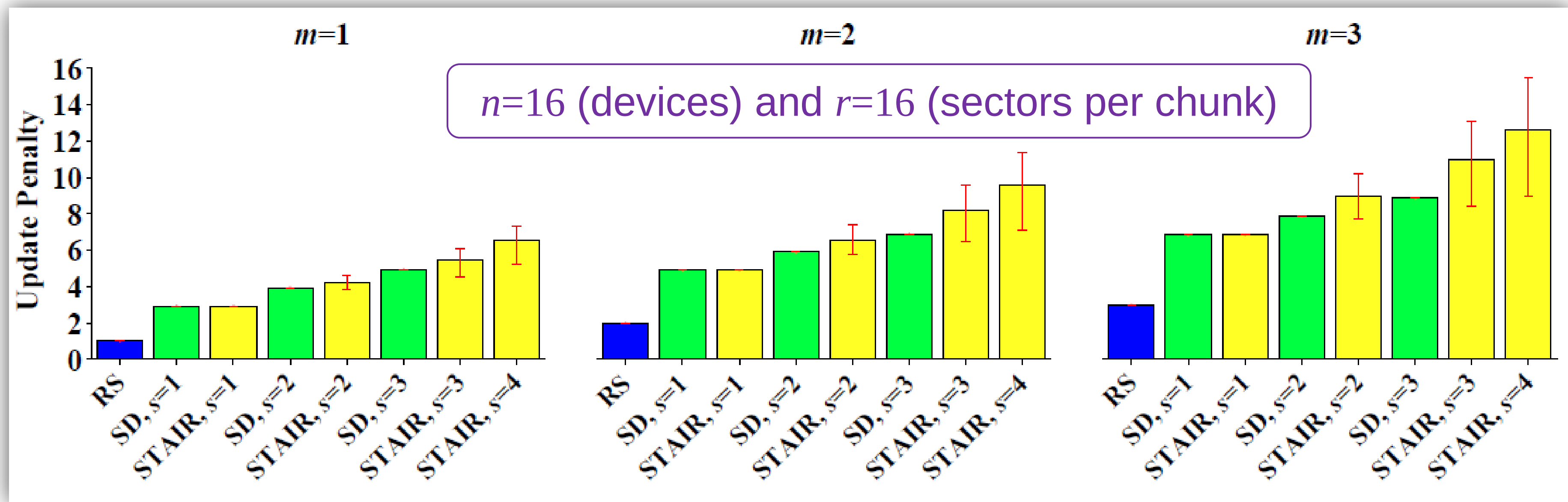
Encoding Speed

n = number of devices
r=16 (sectors per chunk)



- Encoding speed of STAIR codes is on order of **1000MB/s**
- STAIR codes **improve** encoding speed of SD codes **by ~100%**, due to parity reuse
- Similar results for decoding

Update Cost



(Update penalty: average # of updated parity sectors for updating a data sector)

- **Higher update penalty**, due to global parity sectors
- Good for systems with rare updates (e.g., backup) or many full-stripe writes (e.g., SSDs)

[*Plank et al., FAST '13, TOS'14*]

Conclusions

- STAIR codes: a general family of erasure codes for tolerating a hierarchy of failures in a space-efficient manner

- Complementary upstairs encoding and downstairs encoding

- Open source STAIR Coding Library (in C):
  - **http://ansrlab.cse.cuhk.edu.hk/software/stair**

Thank you!

- Contact:
  - Patrick P. C. Lee
    http://www.cse.cuhk.edu.hk/~pclee