
DO NOT SHARE
SLIDES AND CLASS MATERIALS
ON ONLINE SITES
Course Hero

Advanced Logic Design
Codes for Error Detection and Correction

Mingoo Seok
Columbia University

Slide sources: Digital Design by John F. Wakerly (1999)

Error

- An error in a digital system is the corruption of data from its correct value to some other values
  - For example, cosmic rays can cause temporary failure in memory (DRAM and SRAM) circuits
  - Imperfect hardware fabrics, e.g., magnetic materials in HDD, can induce errors
  - Communication channels can distort data transmission

Error Models

- An error model predicts the effects of failures on data
- Simplest model: **independent error model**
  - A single physical failure is assumed to affect only a single bit of data
  - Example: an alpha particle striking a memory cell
- Multiple error: significantly less likely than single error

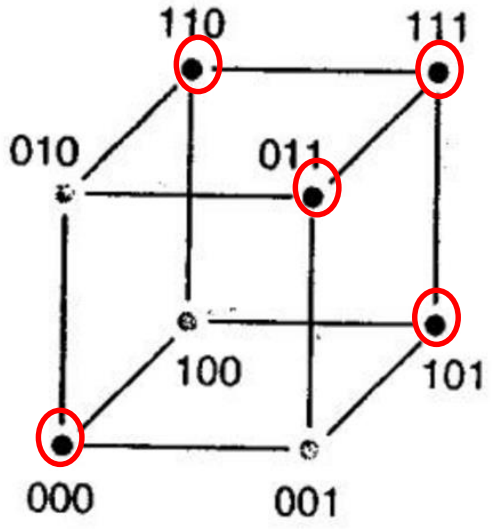
An Error Detecting Code

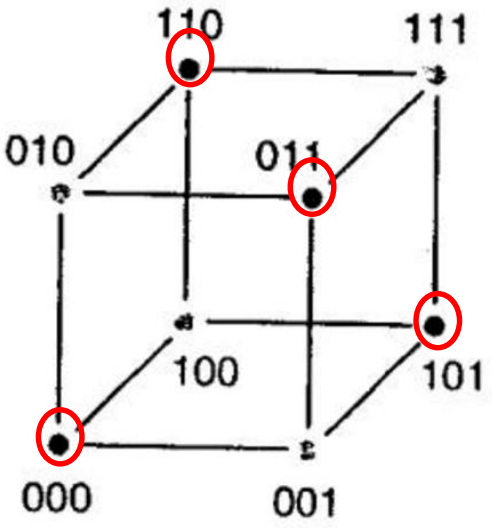
- An error detecting code has a property that corrupting a code will likely produce a bit string that is *not* a code word (a *noncode word*)
- A system that uses only code words can detect errors by a simple rule: if the bit string is a code word it is assumed to be correct; if not, it contains an error

Concept of Error-Detecting Code

= code word

= noncode word





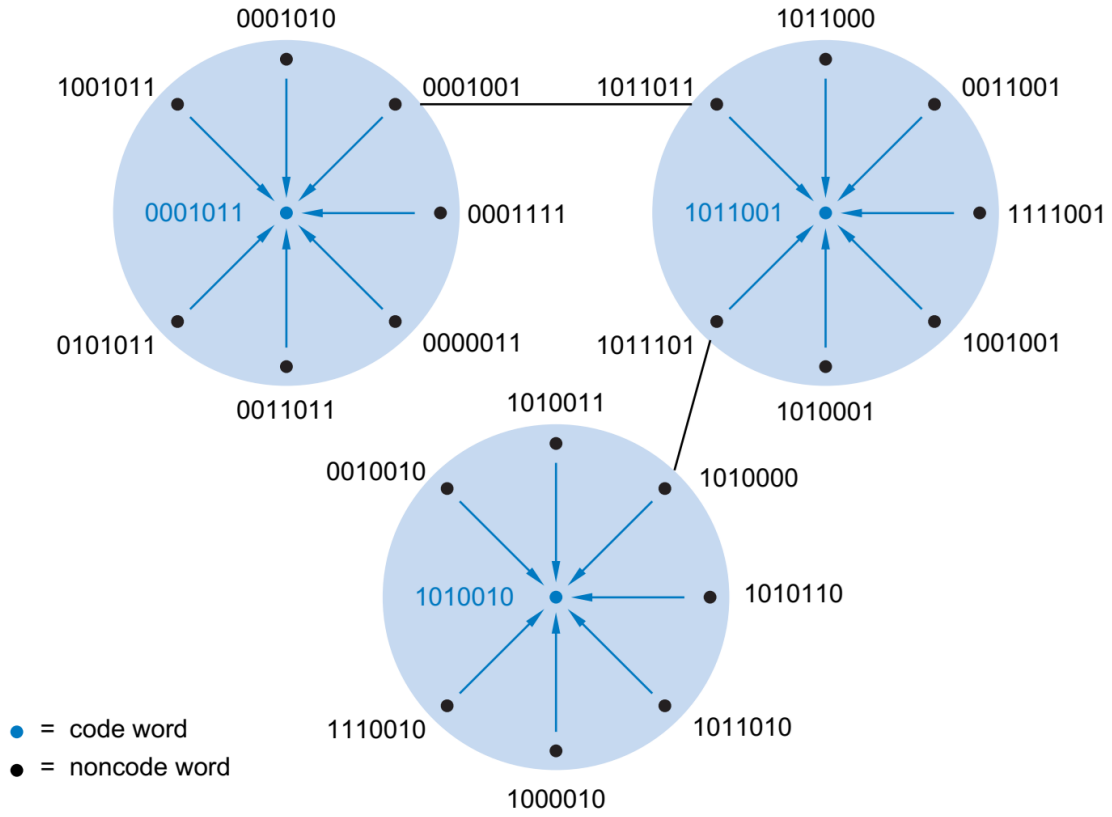
- **Left:** some code words have a distance of 1
- **Right:** all the code words have a distance of 2 or more among them
  - Drawback: out of 8 possible 3-bit words, only 4 can be used; i.e., to store/send 2-bit information w/ error detection capability, we have to commit 3 bits

Parity Bit Code

Information Bits	*Even-parity Code*	*Odd-parity Code*
000	000 0	000 1
001	001 1	001 0
010	010 1	010 0
011	011 0	011 1
100	100 1	100 0
101	101 0	101 1
110	110 0	110 1
111	111 1	111 0

- **Even parity bit:** we can add 1 bit, called a parity bit, that is set to
  - 0 if there are an even number of 1s among the information bits, and to
  - 1 otherwise
- Odd parity bit
- *Cannot detect* two bit errors; the parity bit doesn't change
- No correction capability: it *cannot locate* which bit is corrupted

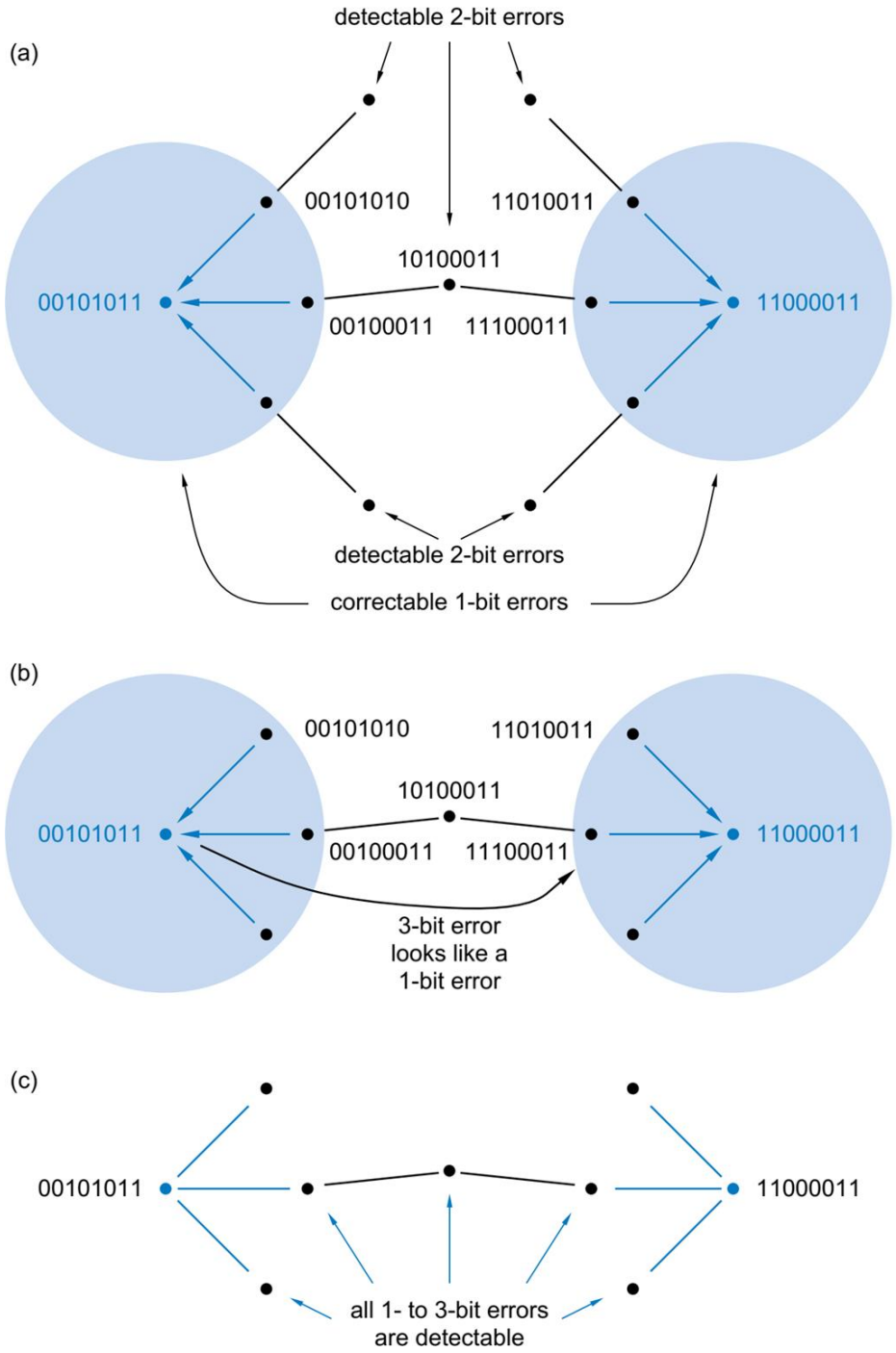
Error-Correcting Code



- If the distance b/w two code words is 3 or more, we can find the code word that is the closest to each non code word

Error-Correcting Code

- In general, if a code has minimum distance $2c+1$, it can be used to correct errors that affect up to c bits ($c=1$ in the prev. example)
- If a code's minimum distance is $2c+d+1$, it can correct errors up to c bits and detect errors up to $c+d$ bits



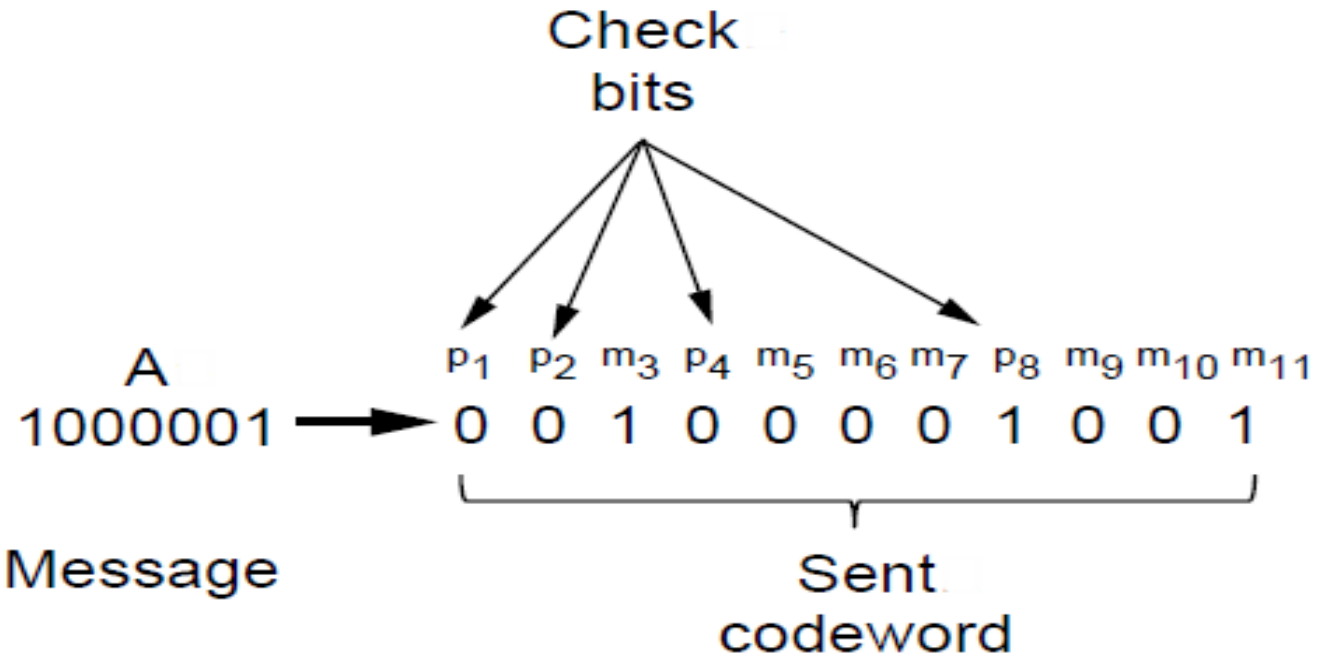
- Distance = 4 = 2·1 +1+1 (c=1, d=1)
- 00101010 → can be corrected
- 10100011 → detected but not corrected

Hamming Code

- A general method for constructing codes with a minimum distance of 3
- Can be extended for a minimum distance of 4
  - Double-error detection and single-error correction
  - Widely used for computer memory systems
  - Note that memory bit error is one of the main errors in computer systems

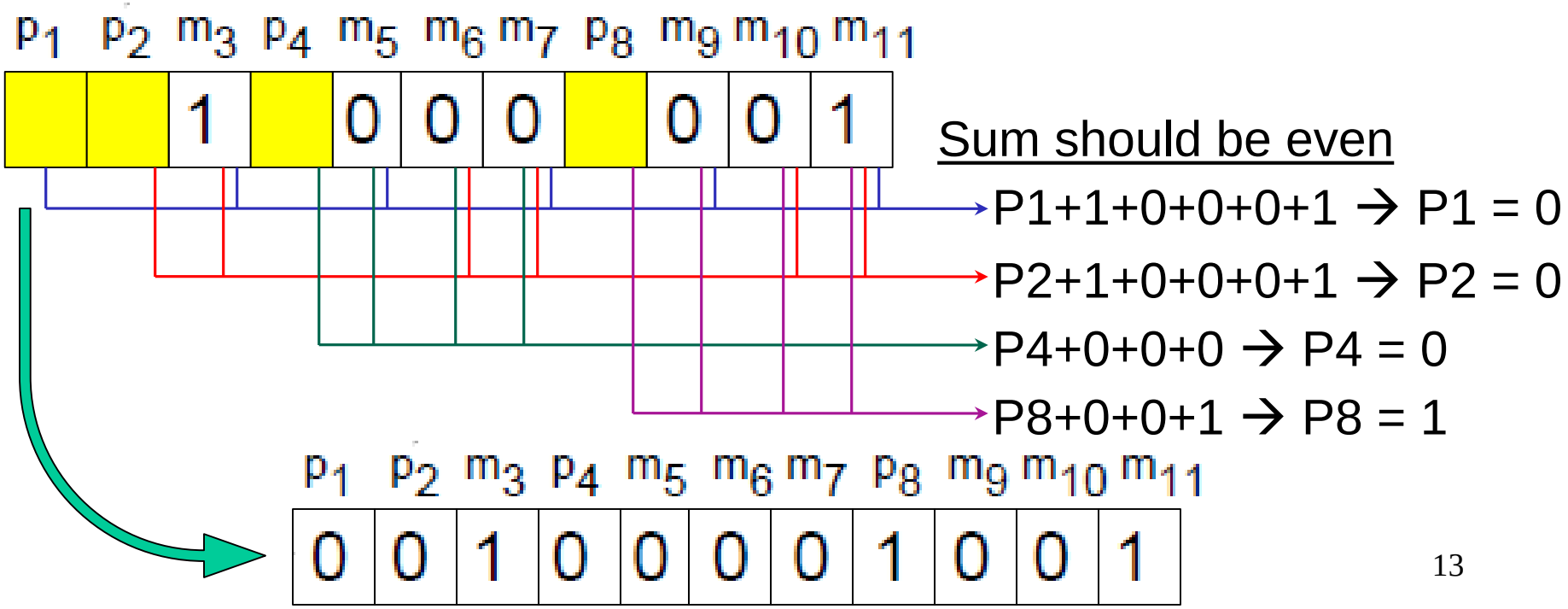
Hamming Code

- Linear systematic block code
- For *m*-bit message we need *r*-bit redundancy, where $(m + r + 1) \leq 2^r$
- Redundancy bits are placed in the position of the power of twos
- Example: If $m = 7$, then $r = 4$



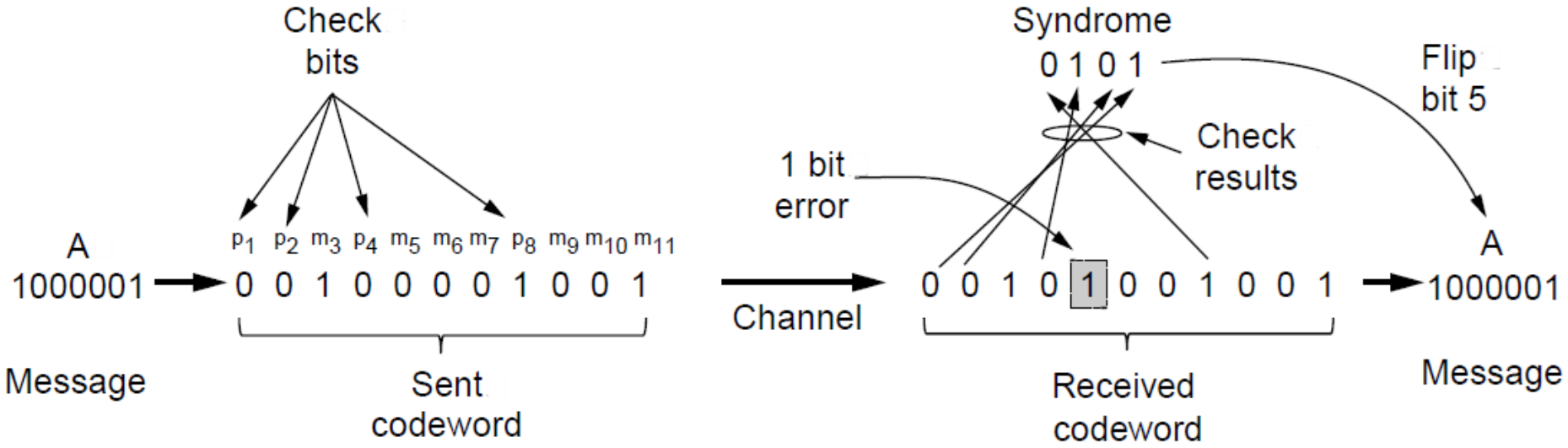
Hamming Code Example

- How to construct codeword
  - message: m = 7 bits
  - 1. find r using the formula → r = 4
  - 2. place data bits in codeword
  - 3. calculate P bits: even parity



Hamming Code Example

- How to correct error and retrieve message
  - 1. Compute *syndrome* similar to codeword construction → error position
  - 2. Flip the bit in the error position
  - 3. Remove P bits from codeword



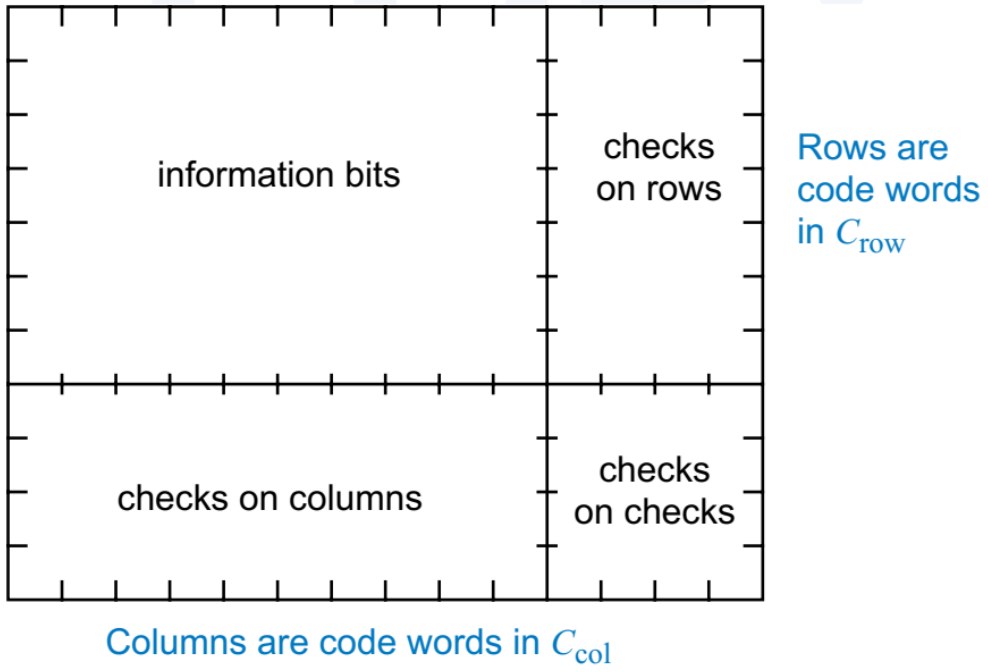
Example of an (11, 7) Hamming code correcting a single-bit error

Hamming Code Overhead

	Minimum-distance-3 Codes		Minimum-distance-4 Codes	
Information Bits	Parity Bits	Total Bits	Parity Bits	Total Bits
1	2	3	3	4
≤ 4	3	≤ 7	4	≤ 8
≤ 11	4	≤ 15	5	≤ 16
≤ 26	5	≤ 31	6	≤ 32
≤ 57	6	≤ 63	7	≤ 64
≤ 120	7	≤ 127	8	≤ 128

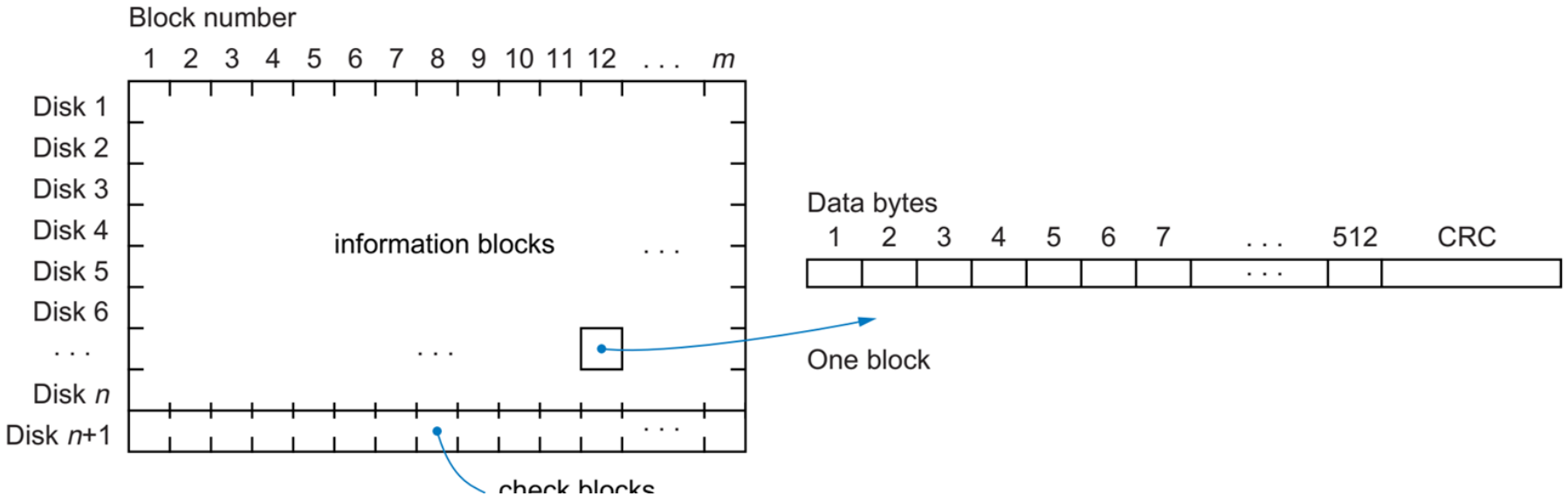
- The relative overhead reduces w/ more information bits
- DRAM word size

2-Dim. Code (aka Product Code)



- The distance of 2D code is the product of d_{row} and d_{col}
- Example: having 1-b parity code for column and the same for row produces a distance of 4 (= 2·2)
- This is more economical in terms of storage overhead
- But you need to read rows as well as columns for error detection and correction

2-Dim. Code (aka Product Code)



- RAID (redundant array of inexpensive disks) uses 2-Dim. code
- Each block has error detection bits (CRC)
- The n+1 Disk contains error correcting codes (e.g., Hamming), each of which takes care of the blocks in the same column of Disks 1 to n
- This requires to read all the N disks; this is overhead but at least feasible

Checksum Code

- The Hamming code is essentially modulo-**2** addition of ***bit****s*
  - If (m1 + m2 + … mX) mod 2 == 1 → 1; otherwise 0
- We can do "modulo-**256** addition of **byte**s"
  - Each byte contains 8 bits (0 to 2^8-1)
  - Can detect any single byte error, since such an error will cause a recomputed sum of bytes to disagree with the checksum