Dependable Systems

Hardware Dependability - Diagnosis

Dr. Peter Tröger

Sources:

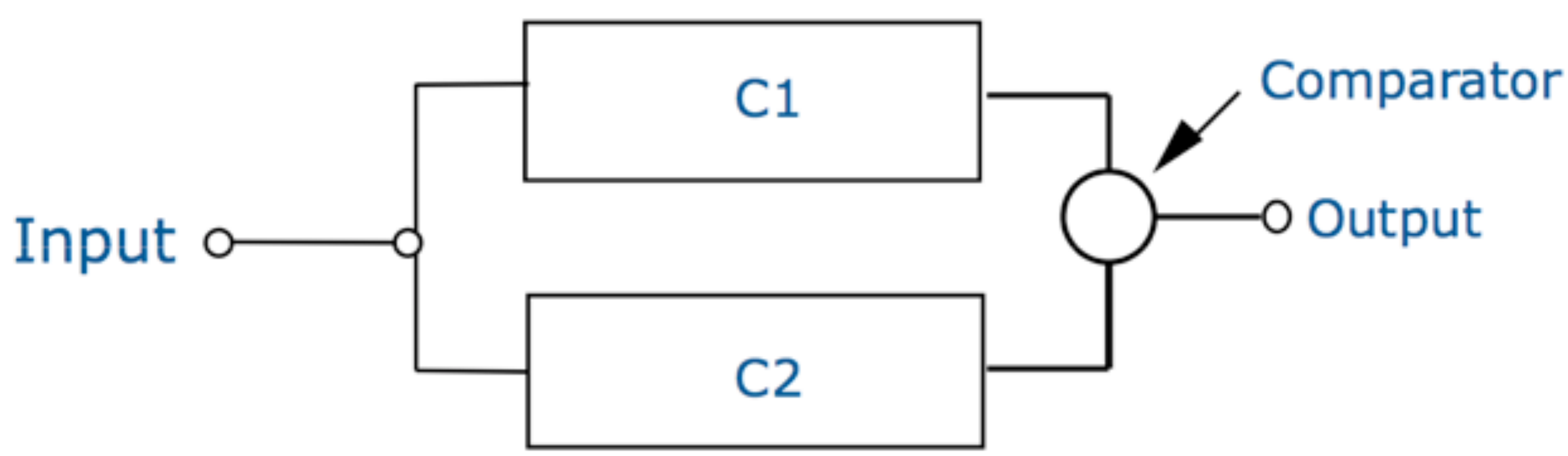
Siewiorek, Daniel P.; Swarz, Robert S.:
Reliable Computer Systems. third. Wellesley, MA : A. K. Peters, Ltd., 1998. ,
156881092X

Fault Diagnosis

- Contains of fault detection (what ?) and fault location (where ?)
- Fault detection
  - Replication checks - Test operation / execution against alternate implementation
  - Timing checks - Test operation / execution against timing constraints
  - Reversal checks - Make use of operation reversibility
  - Coding checks - Utilize redundant (but different) representation of data
  - Reasonableness checks - Check against known system / data properties
  - Structural checks - Ensure consistent structure of data, diagnostic tests, ...
  - Diagnostic checks - Use set of inputs for which the outputs are known
  - Algorithmic checks - Check invariants of an algorithm

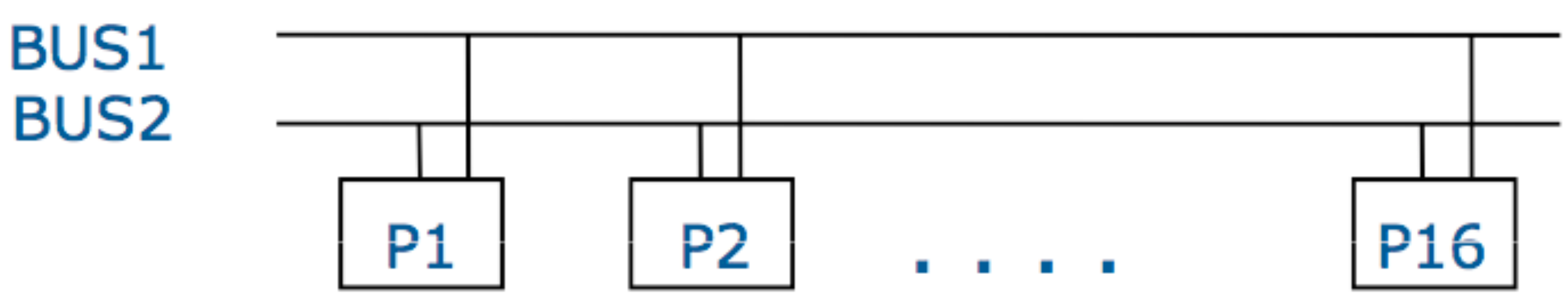
Fault Detection

- **Replication check**
  - Perform test against the same / alternate implementation
    - Identical / partial copies of unit - assumes correct design and independent failure
    - Separate and different copies - assumes design faults
    - Repeated execution - assumes transient fault
  - Works well, but expensive



Fault Detection

- **Timing checks**
  - Monitor execution based on timing constraints
  - Watchdog - Progress resets timer, expired timer assumes broken component
    - Detection of crash, overload, infinite loop; frequency depends on application
  - Broadcast timeout - Component broadcasts message on progress, receivers have deadline for next message
  - Acknowledge timeout - Peer should react on request with answer message



Tandem: Heartbeat broadcast every second, check on every second message

Fault Detection

- **Reversal checks**
  - One-to-one relationship between inputs and outputs
    - Calculates inputs from output by reverse function, comparison
      - Re-read data after write, mathematical functions

$$\sqrt{x}^2 = x$$

- **Diagnostic checks**
  - Check known output for some test inputs
  - Typical approach in built-in hardware testing
  - Load tests - run at saturation levels, trigger abnormal environmental conditions
- **Plausibility checks**
  - Based on knowledge about system design and data types - range checks, consistency checks, type checks

Fault Detection - Coding Checks

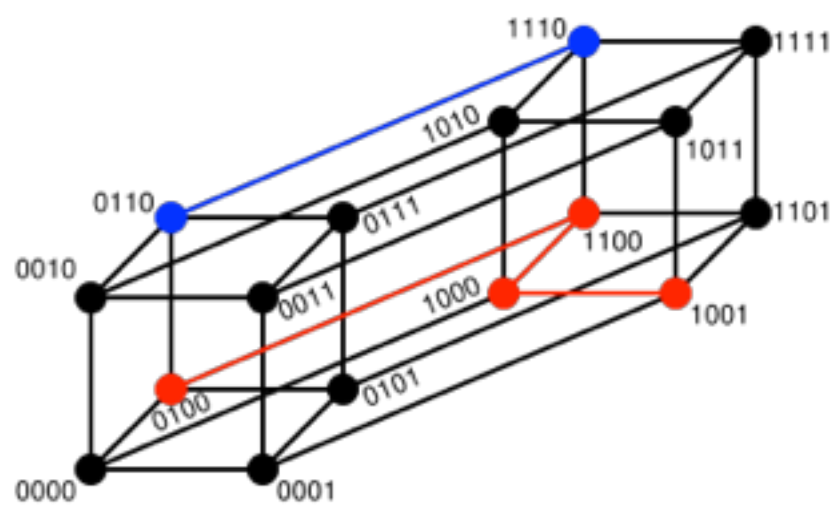
- Coding techniques help with fault detection, diagnosis and tolerance
- General idea: Apply redundancy to information word
  - Code defines a subset of all possible information words as valid information
  - Each valid information is a **code word**
  - Error detection: Is the given word a (valid) code word
- Number space - All number that can be represented in a numerical system
  - Code space - Subset of a number space
  - Separable code vs. non-separable code
- Applications: Data storage, ALU, communication busses, self-checking hardware, ...
- Ranking of codes: detection coverage, overhead, complexity added, ...

Coding Checks in Memory Hardware

- **Primary memory**
  - Parity code
  - m-out-of-n resp. m-of-n resp. m/n code
  - Checksumming
  - Berger Code
  - Hamming code
- **Secondary storage**
  - RAID codes
  - Reed-Solomon code

Hamming Distance

- Number of positions on which two words differ (Richard Hamming, 1950)
    - Alternative definition: Number of necessary substitutions to make A to B
    - Alternative definition: Number of errors that transform A to B

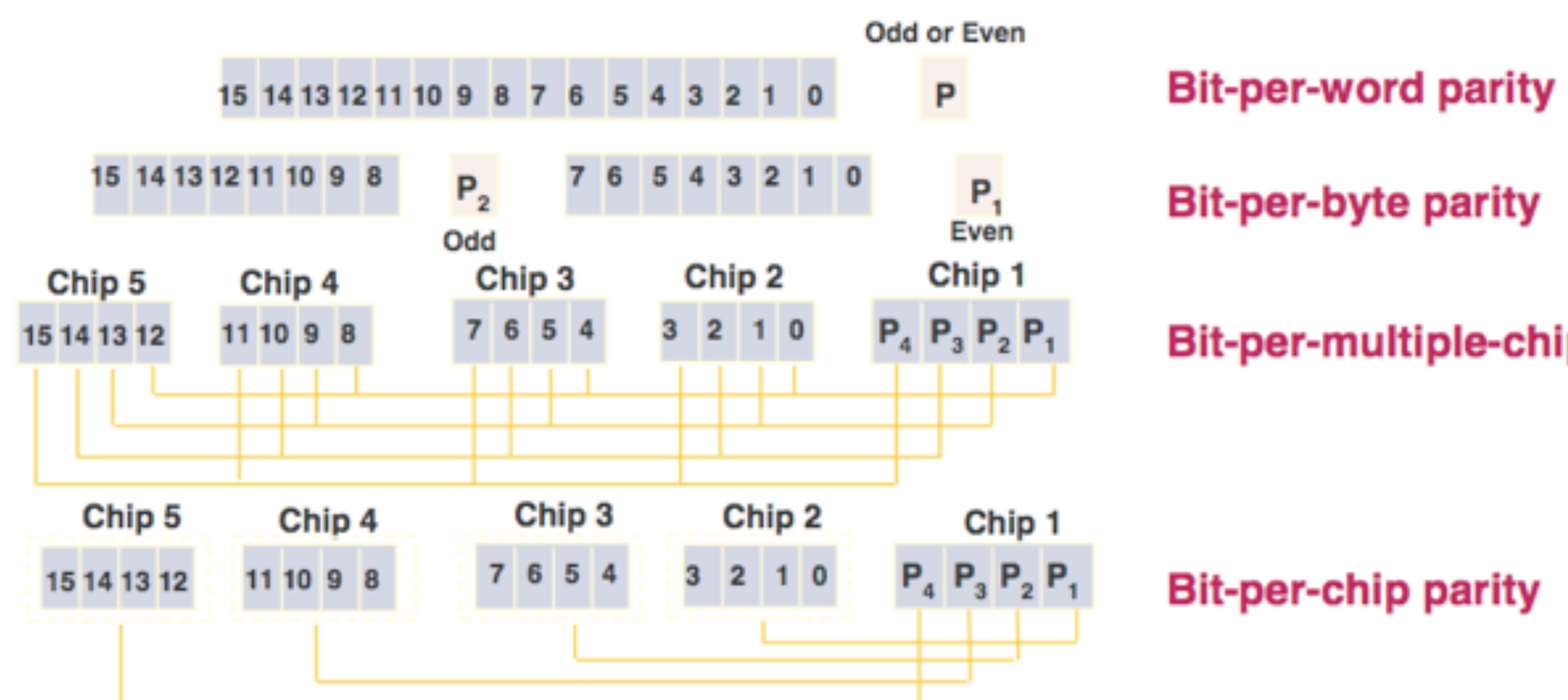


(C) Wikipedia

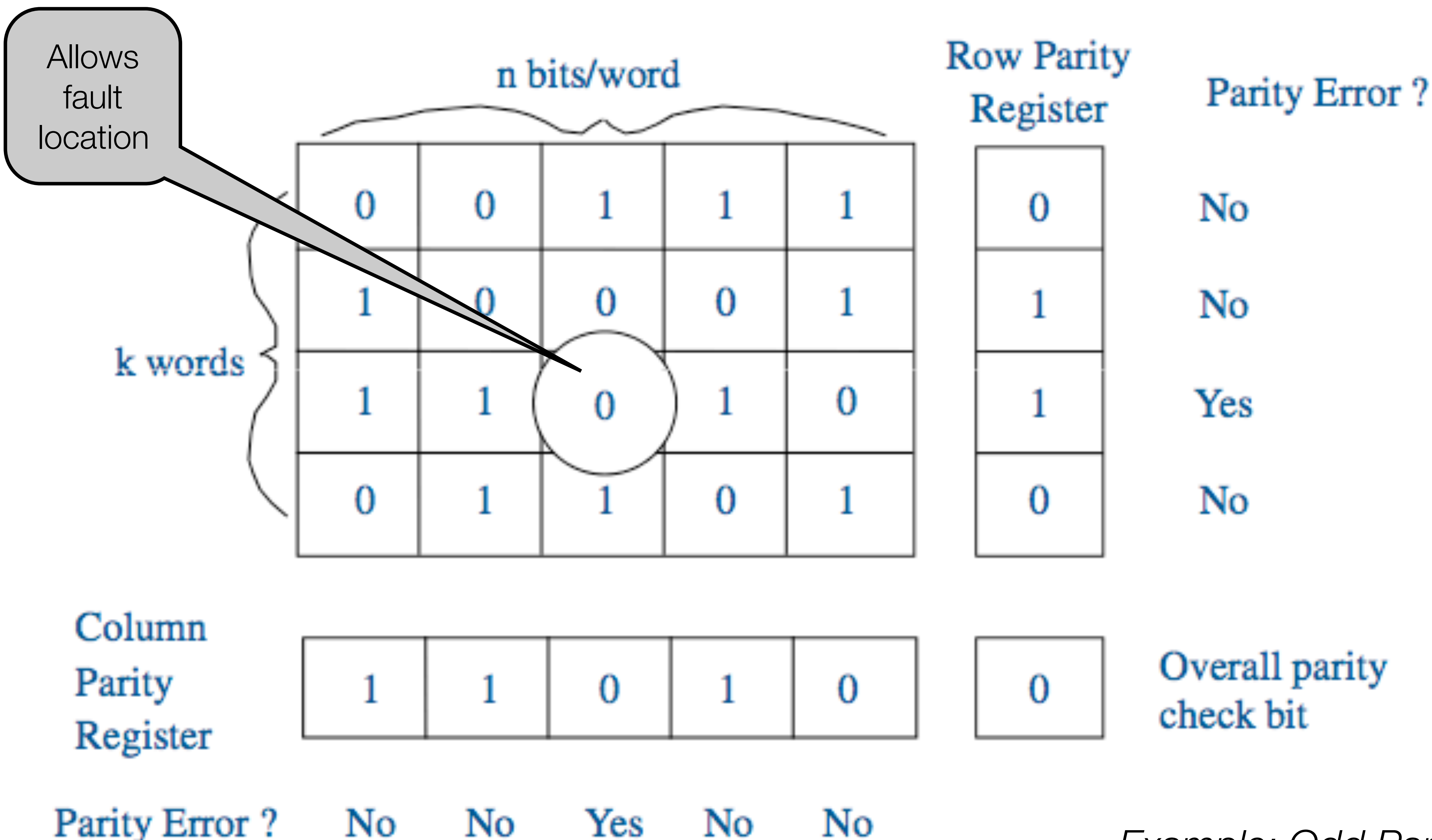
- Minimum / code Hamming distance: Minimum distance between any two code words
- To *detect* **d** single-bit errors, a code must have a min. distance of at least **d + 1**
    - If at least d+1 bits change in the transmitted code, a new (valid) code appears
- To *correct* **d** single-bit errors, the minimum distance must be **2d+1**

Parity Codes

- Add parity bit to the information word
  - Even parity: If odd number of ones, add one to make the number of ones even
  - Odd parity: If even number of ones, add one to make the number of ones odd
- Variants
  - Bit-per-word parity
  - Bit-per-byte parity (Example: Pentium data cache)
  - Bit-per-chip parity
  - ...



Two-Dimensional Parity



Example: Odd Parity

m-out-of-n Code

- Data word of length **n** (including code bits), make sure that **m** ones are in the word
- Example: 2-out-of-5 code
  - Often used in telecommunication and in the US postal system
  - 5 bits allow 10 combinations with 2-out-of-5, so all decimal digits are representable
  - Supports more than only single-bit errors

Decimal Digit	2-out-of-5
0	0 0 0 1 1
1	0 0 1 0 1
2	0 0 1 1 0
3	0 1 0 0 1
4	0 1 0 1 0
5	0 1 1 0 0
6	1 0 0 0 1
7	1 0 0 1 0
8	1 0 1 0 0
9	1 1 0 0 0

Code Properties

Code	Bits / Word	Number of possible words	Hamming Distance	Coverage
Even Parity	4	8	2	Any single bit error, no double-bit error not all-0s or all 1s errors
Odd Parity	4	8	2	Any single bit error, no double-bit error all-0s or all 1s errors
2/4	4	6	2	Any single bit error, 33% of double bit errors

Checksumming

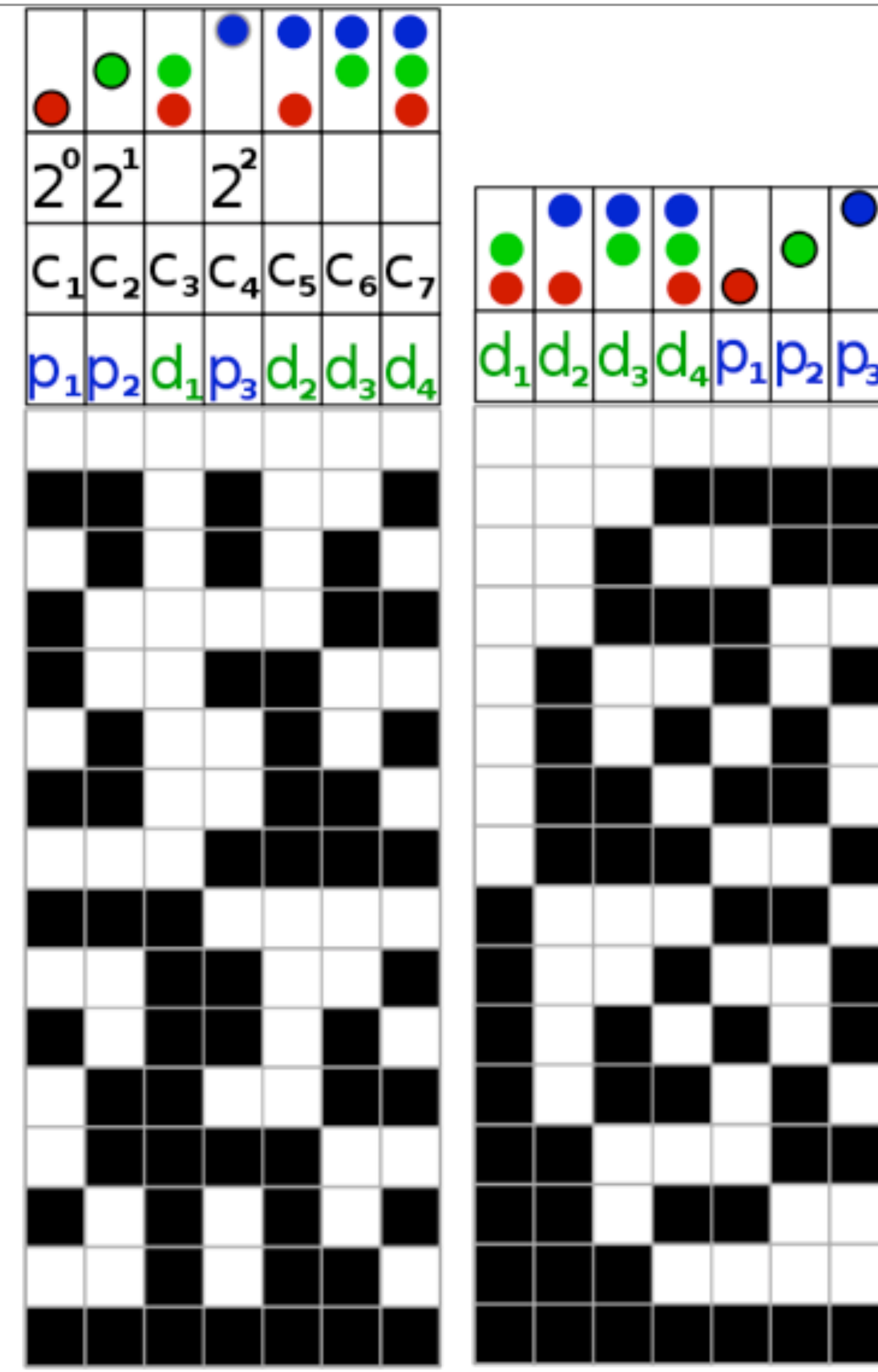
- General idea: Multiply components of a data word and add them up to a checksum
  - Good for large blocks of data, low hardware overhead
  - Might require substantial time, memory overhead
  - 100% coverage for single faults
- Example: ISBN 10 check digit
  - Final number of 10 digit ISBN number is a check digit, computed by this approach
    - Multiply all data digits by their position number, starting from right
    - Take sum of the products, and check digit so that results MOD 11 = 0
    - Check digit „01“ is represented by X
- More complex approaches get better coverage (e.g. CRC) for more CPU time

Berger / Hamming Code

- Berger code
  - Number of 0‘s is appended to the data word, *log n +1* check bits
  - Low hardware overhead, typically used for memory and communication
  - Can detect all unidirectional errors (only 0->1 or only 1->0 flips), no repair
- Hamming code
  - *log2n* check bits, allow both detection and location
  - Class of block codes with different length, hamming distance 3
    - 1-bit error correctable, 2-bit error detectable
    - Can be extended to provide k-error correction with 2k-error detection
  - High overhead (15% - 70%), but location detection pays of in MTTF

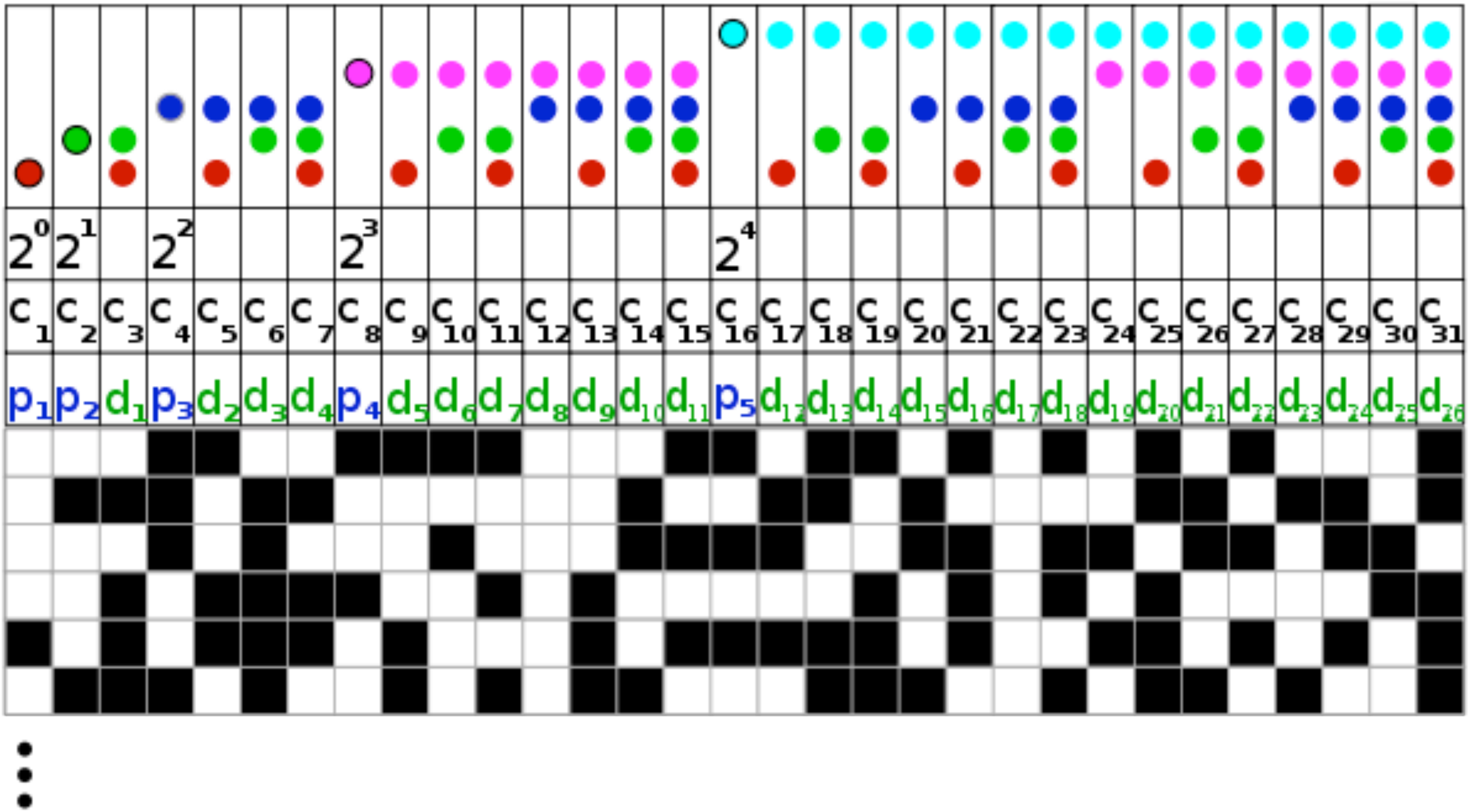
Hamming Code

- Every data word with n bits is extended by k control bits
- Control bits by standard parity approach (even parity), implementable by XOR
  - Inserted at power-of-two positions (not necessary for code implementation)
  - For parity bit p_X, use data bit groups of length 2^{X-1} to the right
  - Data bits are controlled by overlapping groups in the parity bits
- Application in DRAM memory



taken from Wikimedia Commons

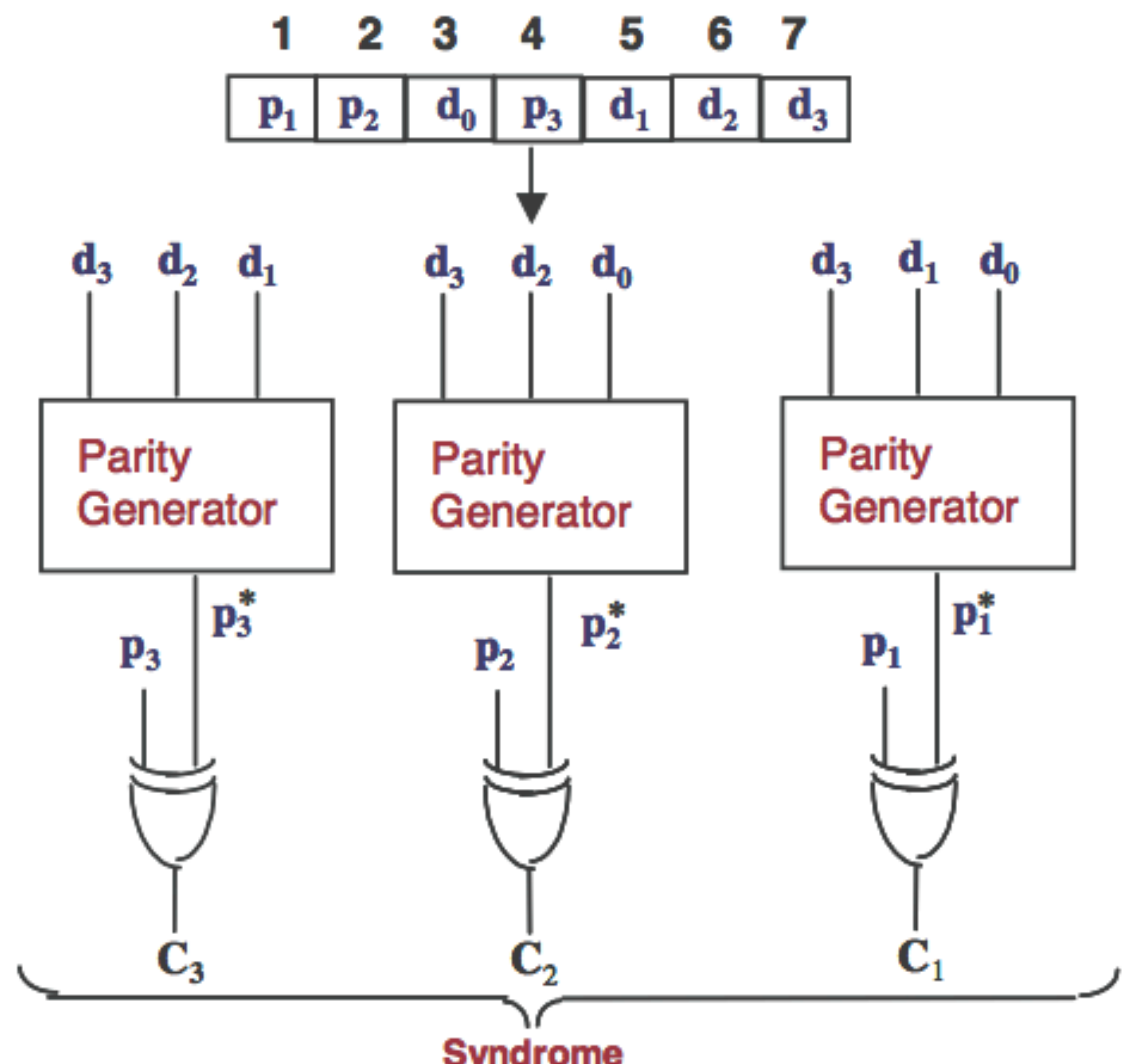
Hamming Code for 26bit word size



taken from Wikimedia Commons

Hamming Code

- Hamming codes are known to produce 10% to 40% data overhead
- Syndrome determines which bit (if any) is corrupted



(C) Z. Kalbarczyk

Example: ECC for Main Memory

- Majority of „one-off“ soft errors in DRAM happens from background radiation
- Denser packages, lower voltage for higher frequencies
- Most common approach is the SECDEC Hamming code
  - "single error correction, double error detection" (SECDEC)
  - Hamming code with additional parity
- Modern BIOS implementations perform thresholf-based warnings on frequent correctable errors