Dependable Systems

Hardware Dependability - Diagnosis

Dr. Peter Tröger

Sources:

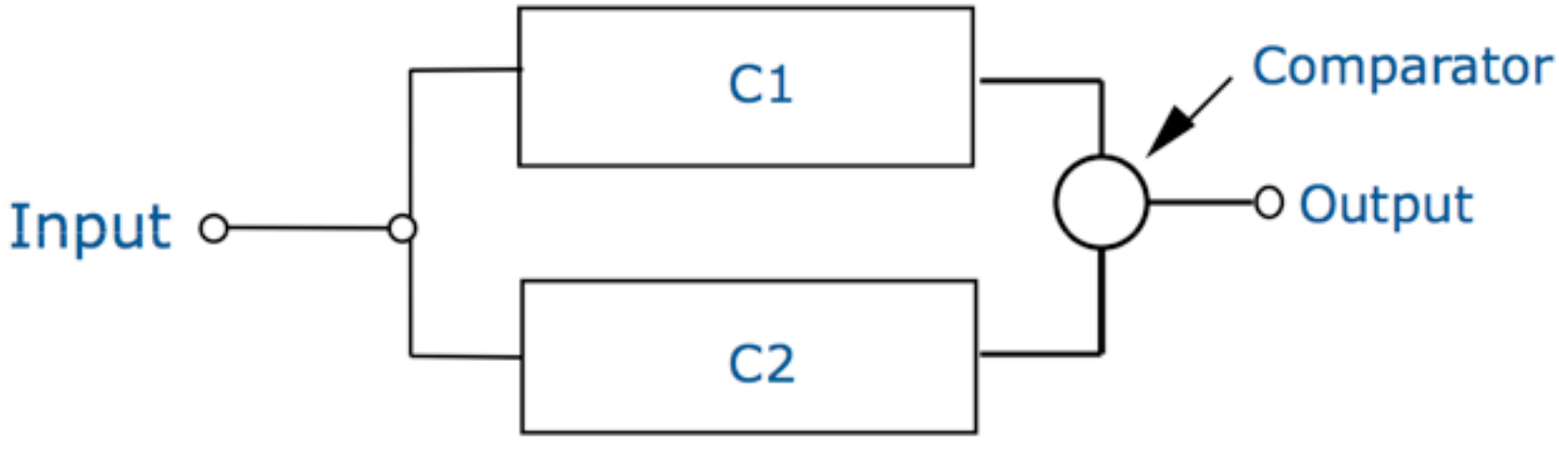
Siewiorek, Daniel P.; Swarz, Robert S.:
Reliable Computer Systems. third. Wellesley, MA : A. K. Peters, Ltd., 1998. ,
156881092X

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 reversability
  - 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

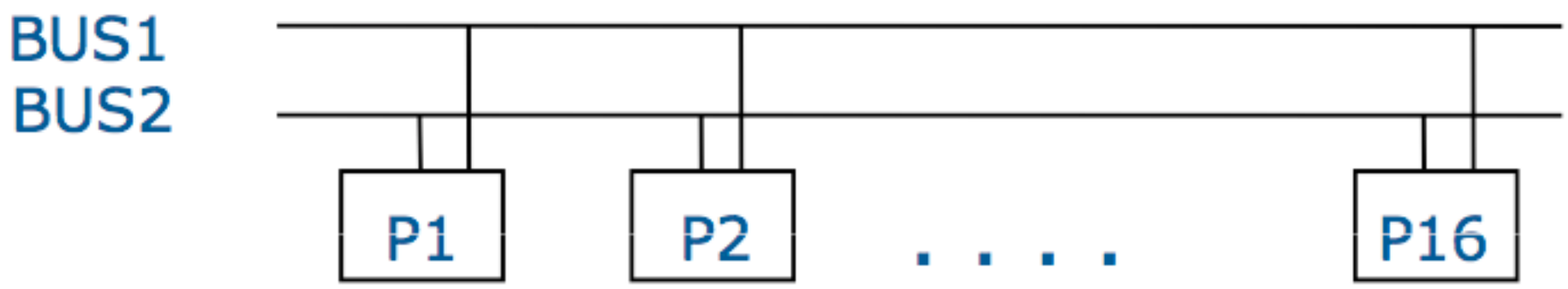
Fault Detection

- **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 (see last lecture)
  - Load tests - run at saturation levels, trigger abnormal environmental conditions

Fault Detection

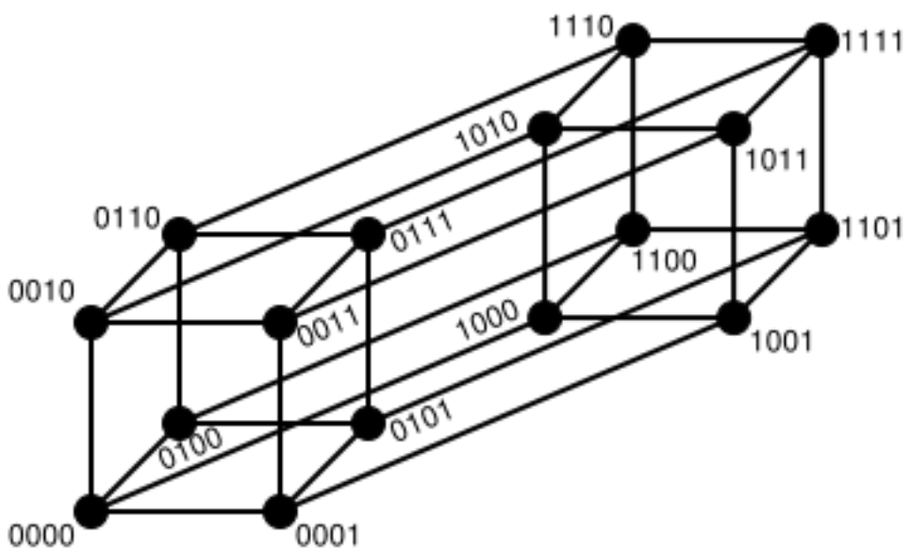
- **Plausibility checks**
  - Based on knowledge about system design and data types
  - Range checks
  - Consistency checks
  - Type checks

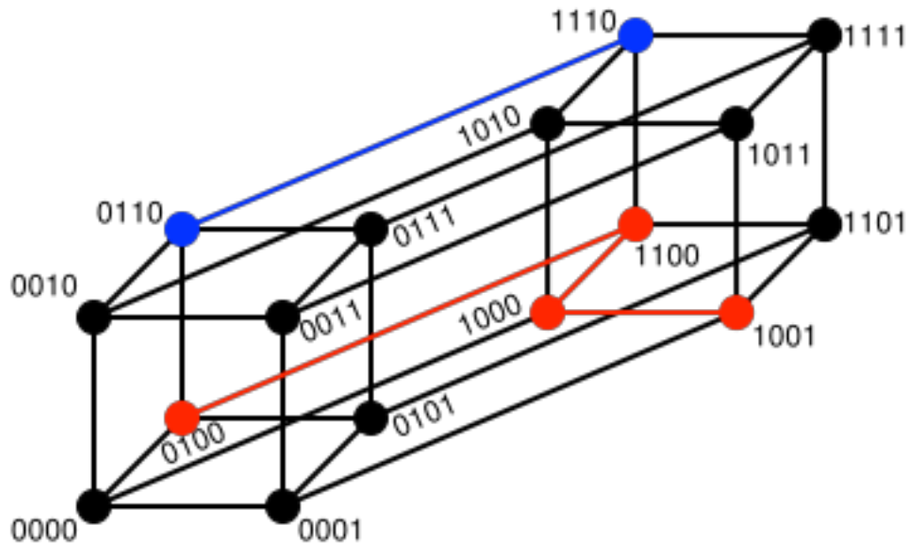
Fault Detection - Coding Checks

- Coding techniques help with fault detection, diagnosis and tolerance
- General idea: Apply redundancy to information
  - 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 one
- 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
- Application in data storage. ALU, communication busses, self-checking hardware, ...
- Ranking codes through their detection coverage, overhead, complexity added, ...

Hamming Distance

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





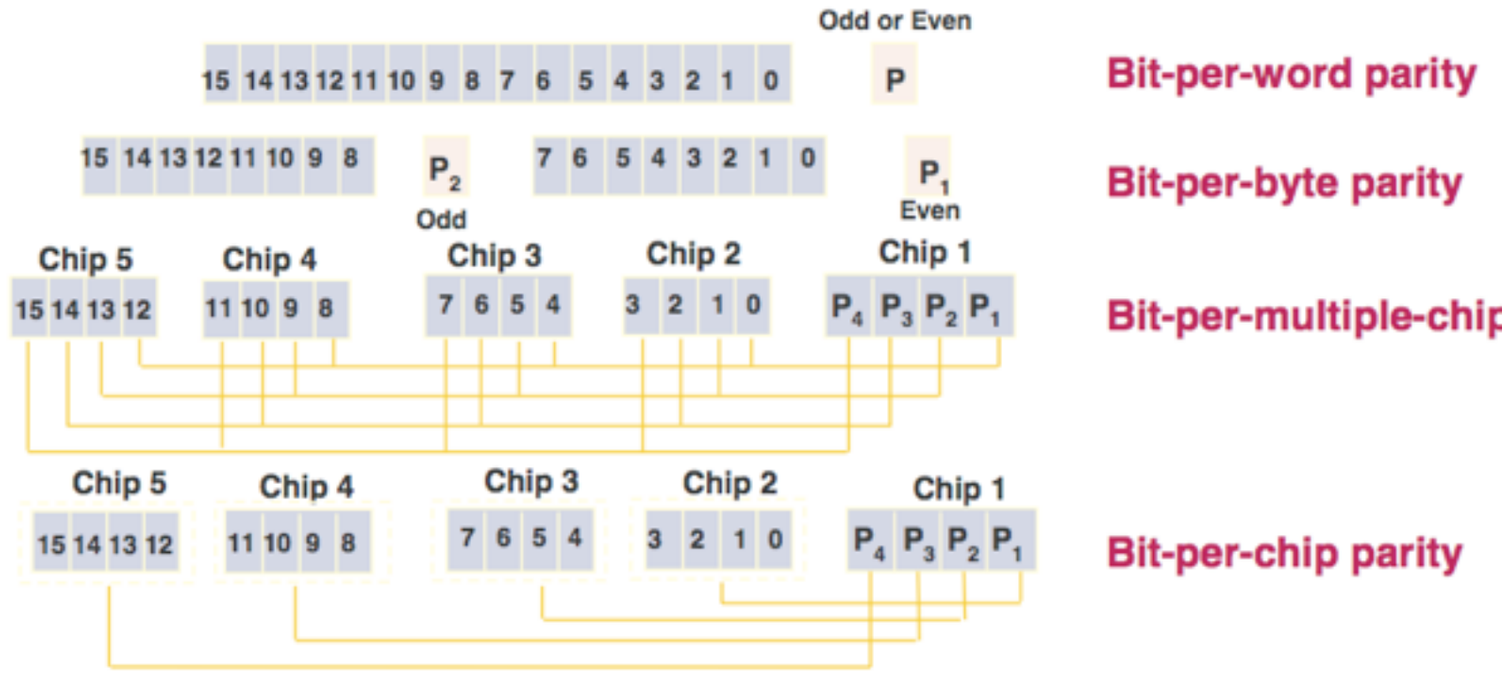
taken from Wikipedia

Coding Checks in Memory Hardware

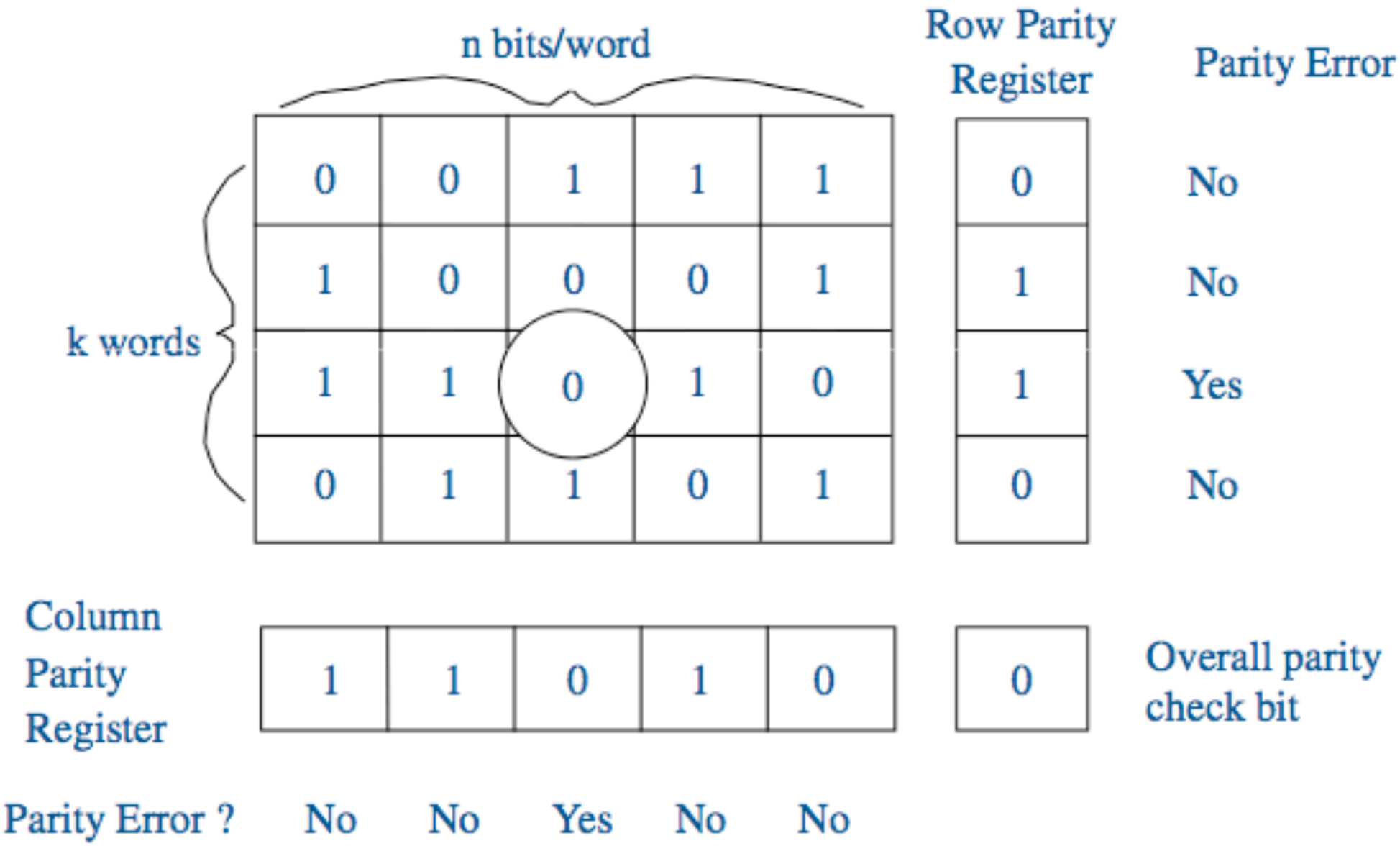
- Primary memory
  - Parity code
  - m-out-of-n resp. m-of-n resp. m/n code
  - Checksumming
    - Berger Code
    - Hamming code
- Secondary storage
  - RAID codes
  - Reed-Solomon code

Parity Codes

- Add parity bit to the information word
  - Even parity: Odd number of ones -> add one to make the number of ones even
  - Odd parity: 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



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, so decimal digits are representable

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

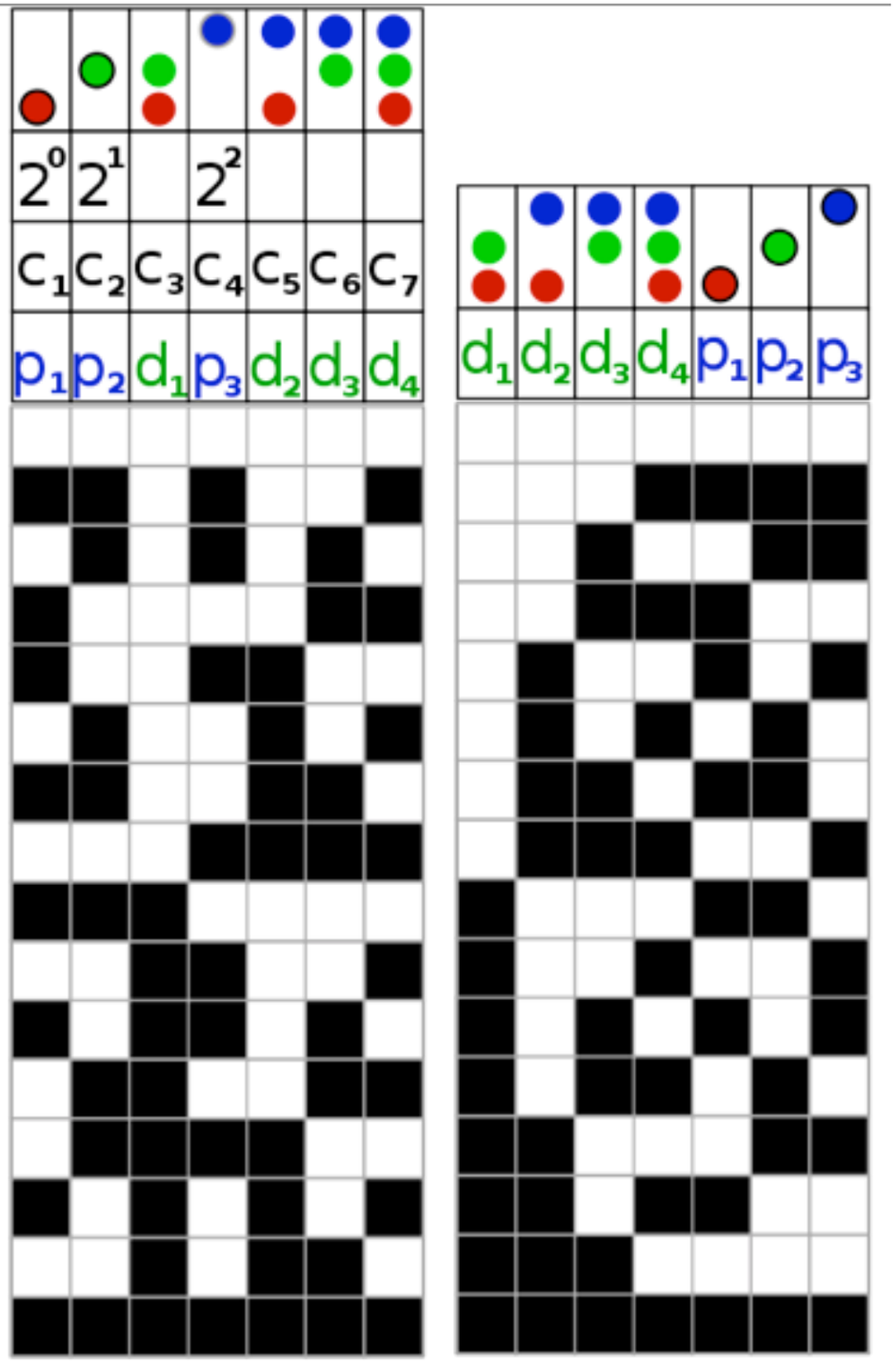
- General idea: Multiply components of a data word and add them up to a checksum
  - Good for large blocks of data, low hardware overhead
  - Might require substantial time, memory overhead
  - 100% coverage for single faults
- Example: ISBN 10 check digit
  - Final number of 10 digit ISBN number is 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

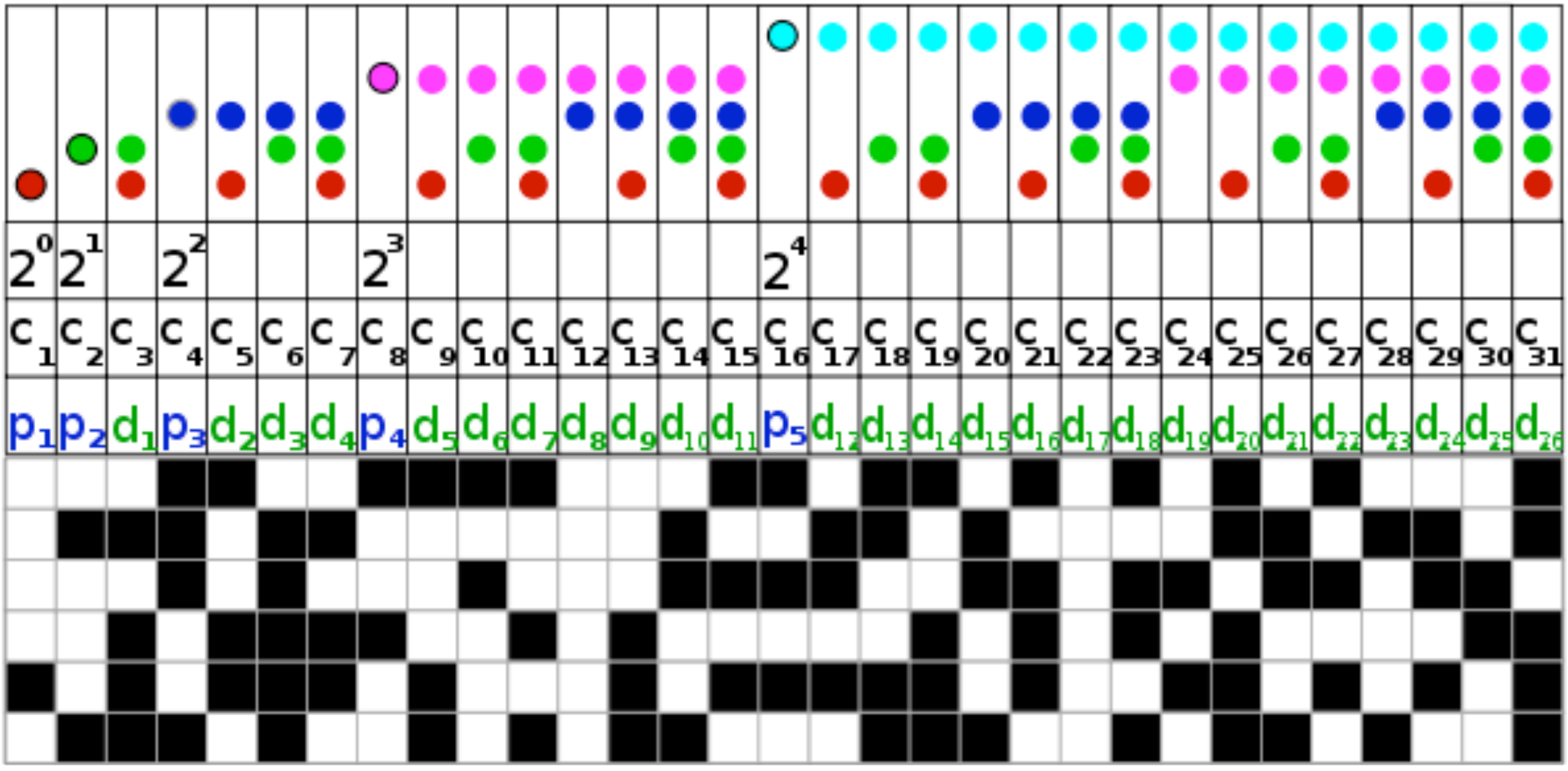
Hamming Code

- 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

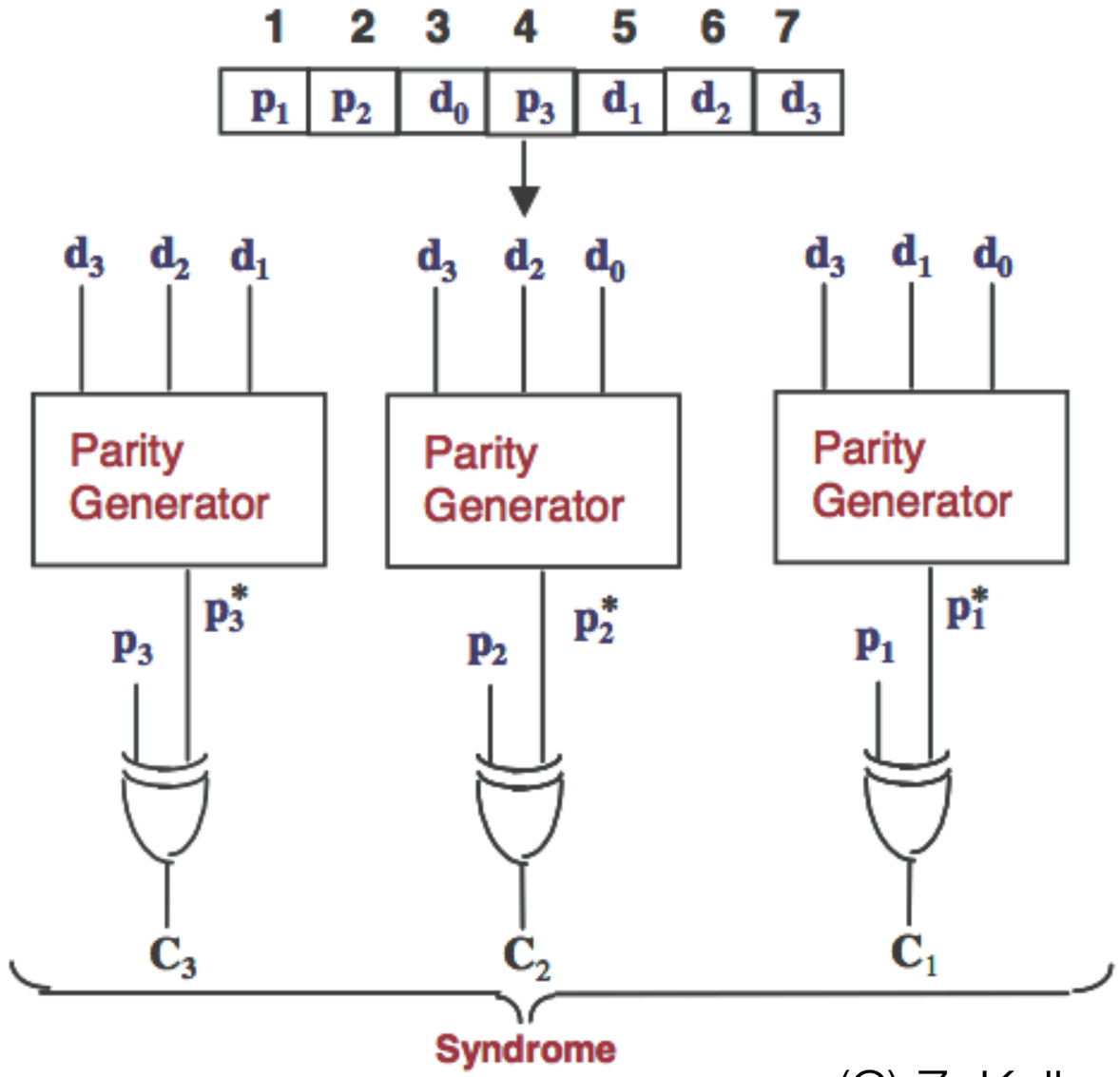
Hamming Code for 26bit word size



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

System-Level Diagnosis

- Complex hardware uses multiple approaches in different locations
  - Processor hardware: Signature analysis, watchdogs
  - Memory hardware: Error-correcting codes, parity
  - Network hardware: Checksumming, CRC, ...
- Concurrent detection is typically combined with component-level error correction
- Packaging determines required level of fault localability
- Concurrent diagnosis needs to consider component dependencies and relation
  - Parallel vs. localized diagnosis
  - Centralized vs. distributed diagnosis
- Many algorithm proposals for **system-level diagnosis**

System-Level Diagnosis

- Basic idea: Avoid steep costs of NMR and special testing hardware, by letting processing elements test each other for correct functioning
  - Test exists to check component A from component B
  - ‚Bad‘ component can be detected safely, if the tester works
    - Problem: Faulty components do not deliver correct test results
- Definition: A system is **t-diagnosable** if any distribution of t faults is diagnosable (detectable **and** locatable)
  - Assumes existence of an external supervisor that safely collects the results
  - Also used for systems with time dependabilities
- Tests are assumed to be exhaustive for the given fault model
- Faults are assumed to be permanent (until the diagnosis is finished)

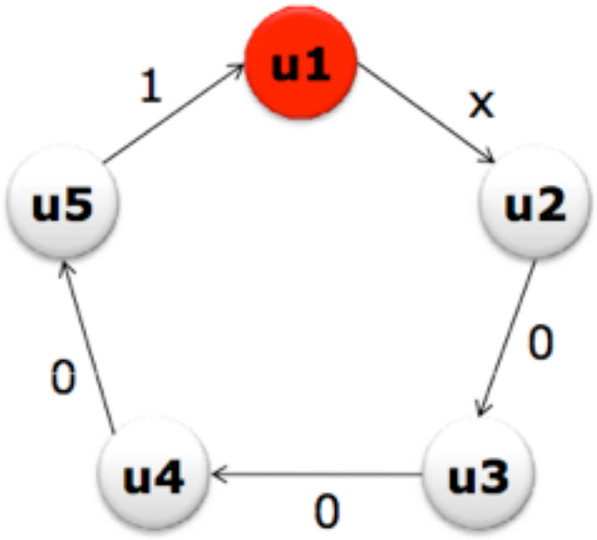
Preparata Metze Chien (PMC) Model

- Published in 1967 as algorithm for diagnosis in multiprocessor systems
- System is decomposed into n (replaceable) units
  - Units don‘t have to be identical, $U=u_1, u_2, u_3, ..., u_n$
  - Each unit is either ‚faulty‘ (1) or ‚fault-free‘ (0), nodes test each other
  - Centralized component collects all test results for the system diagnosis
- Result of all pair-wise tests is called **syndrome -** the base for fault location
  - If the testing node is fault-free, it returns the correct status
  - If the testing node is faulty, the result can be arbitrary
  - No unit tests itself

PMC Model

- Non-failing supervisor collects syndrome
- All syndroms are different, so fault can be detected and located

Faulty Unit	Possible Syndrom				
1	0	0	0	0	1
	1	0	0	0	1
2	1	0	0	0	0
	1	1	0	0	0
3	0	1	0	0	0
	0	1	1	0	0
4	0	0	1	0	0
	0	0	1	1	0
5	0	0	0	1	0
	0	0	0	1	1

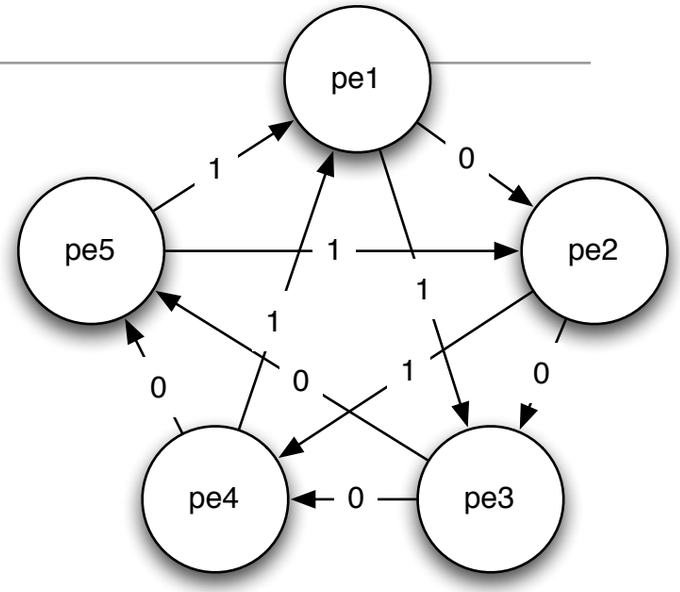


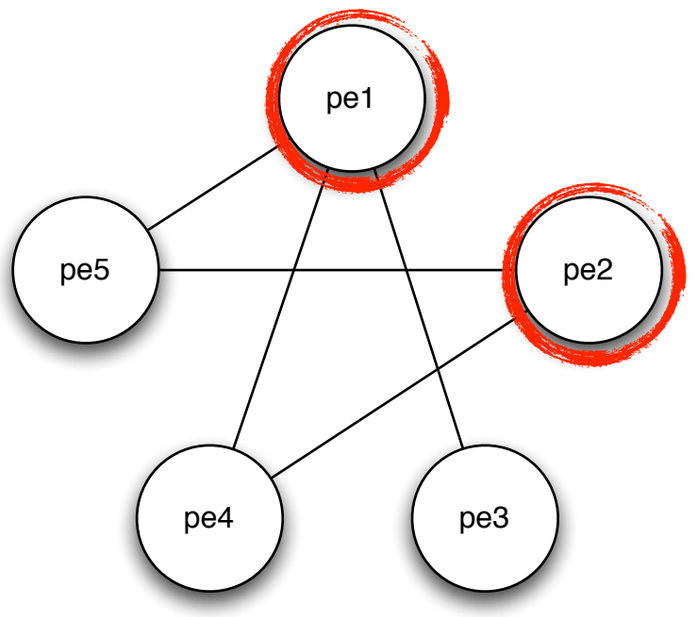
PMC Model

- A system is **t-diagnosable** with **PMC** if each node is tested by at least **t** other nodes and **n >= 2t+1**
- Example: Ring test structure is 1-diagnosable
- Strict assumptions in PMC
  - All faults are permanent faults
  - A fault free processor can always determine accurately the test result
  - A central perfect arbitration unit exists
  - Not more than t processors may be faulty in one diagnosis round

Diagnosis with the PMC Model

- Diagnosis algorithm by Dahbura & Masson (1984)
  - Best solution in terms of worse-case efficiency $O(n^{2.5})$
  - Convert test graph into *disagreement graph*
    - Choose a processing element, and assume it is fault free
    - Draw edge to processing elements that are then understood as faulty
    - Repeat for all nodes
- Edges represent *implied faulty sets* of each PE
- Find *minimum vertex cover* of the result
  - Minimal set of nodes that touches all edges in the graph
  - Represent the faulty units





t-Diagnosability

- **Barsi Grandoni Maestrini (BGM) Model** - Published in 1976
  - Change in comparison to PMC: Tests are typically comparisons, and two identical incorrect values are unlikely
    - Faulty processor is always detected as faulty, regardless of the testers state
    - Necessary condition: A system is t-diagnosable with BGM if **n >= t+2**
- **Kreutzer and Hakimi** - Extension of PMC resp. BGM
  - HK1: As PMC, but a faulty tester would report a working processor as faulty
  - HK2: As BMC, but a faulty tester always reports a faulty processor
- All extensions limit the possible syndrom to occur, which makes diagnosis easier
- Further PMC extensions to cope with imperfect tests, intermittent faults, test scheduling, arbiter-free distributed diagnosis (see Barborak et al., ACMCS, June 93)