RELIABILITY
and
RELIABLE DESIGN

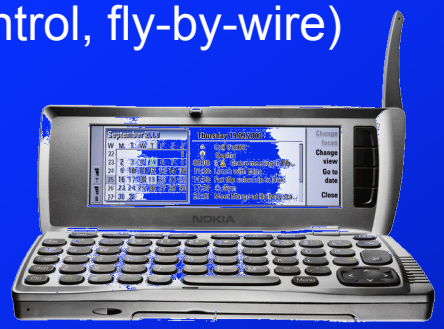
Giovanni De Micheli
Centre Systèmes Intégrés

Outline

- Introduction to reliable design
- Design for reliability
  - Component redundancy
  - Communication redundancy
  - Data encoding and error correction
  - Dealing with variability
- Summary and conclusions

Reliable design:
where do we need it ?

- **Traditional applications**
  - Long-life applications (space missions)
  - Life-critical, short-term applications (aircraft engine control, fly-by-wire)
  - Defense applications (aircraft, guidance & control)
  - Nuclear industry
  - Telecommunications

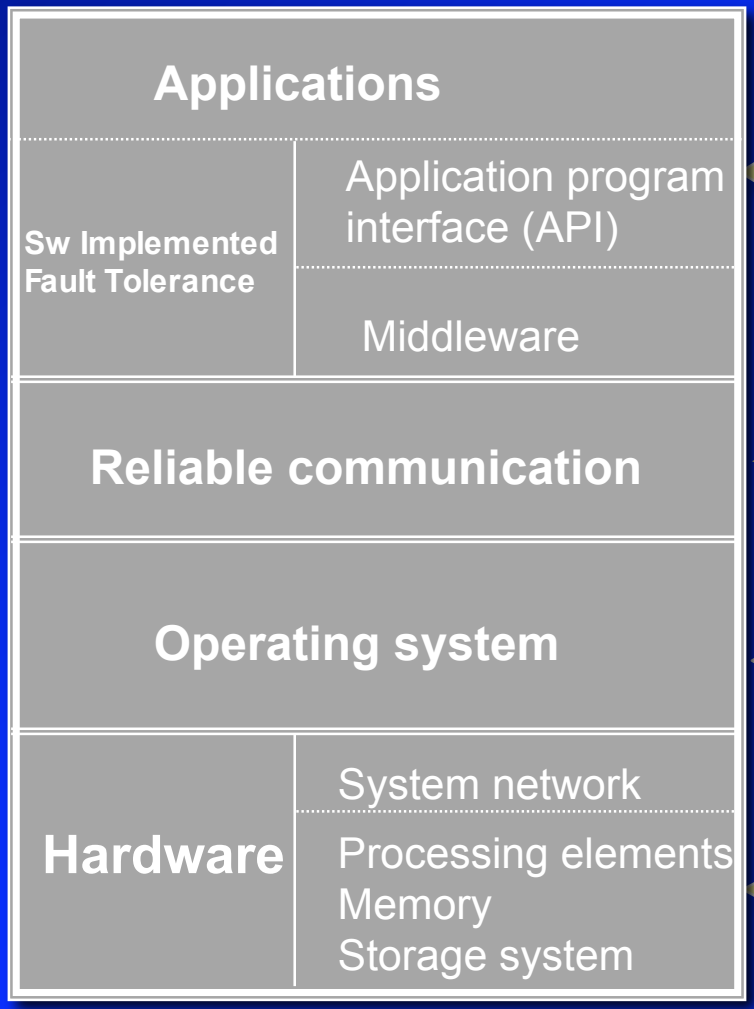


- **New computation-critical applications**
  - Health industry
  - Automotive industry
  - Industrial control systems and production lines
  - Banking, reservations, commerce

The economic perspective

- Availability is a critical business metric for commercial systems and services
  - Nearly 100% availability (“five nines+”) is almost mandatory
- Service outages are frequent
  - 65% website managers report outages over a 6-month period
  - 25% report three or more outages [Internet week 2000]
- High cost of downtime of systems providing vital services
  - Lost opportunities and revenues, non-compliance penalties, potential loss of lives
  - Cost per an hour of downtime varies from $89K for cellular services to $6.5M for stock brokerage [Gartner Group 1998]
- Revenue for *high availability* products in the data/telecom/computer server market is over $100B (≈ $15B for servers alone) [IMEX Research 2003]

Reliability is a system issue



Checkpointing and rollback, application replication, software, voting (fault masking), process pairs, robust data structures, recovery blocks, N-version programming,

CRC on messages , acknowledgment, watchdogs, heartbeats, consistency protocols

Memory management and exception handling, detection of process failures, checkpoint and rollback

Error correcting codes, M-out-of-N and standby redundancy , voting, watchdog timers, reliable storage (RAID, mirrored disks)

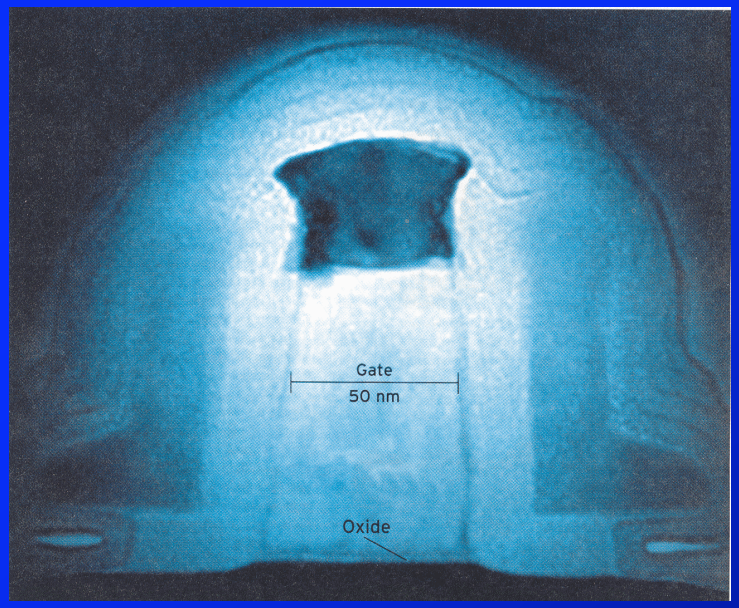
[Iyer]

Malfunctions

- Manufacturing imperfections
  - More likely to happen as lithography scales down
- Approximations during design
  - Uncertainty about details of design
- Aging
  - Oxide breakdown, electromigration
- Environment-induced
  - Soft-errors, electro-magnetic interference
- Operating-mode induced
  - Extremely-low voltage supply

Process variability

- Effects of downscaling
  - Smaller mean values
  - Larger variances
- Worst-case design paradigm fails



Sources of process variations

- Chemical deposition (CD) variation
  - Systematic and random
    - Inter and intra-die
- Width variation
  - Impact on narrow transistors
- Threshold voltage fluctuation
  - Largest impact on short and narrow devices
- Interconnect
  - Dishing and erosion

Circuit-level mitigation techniques

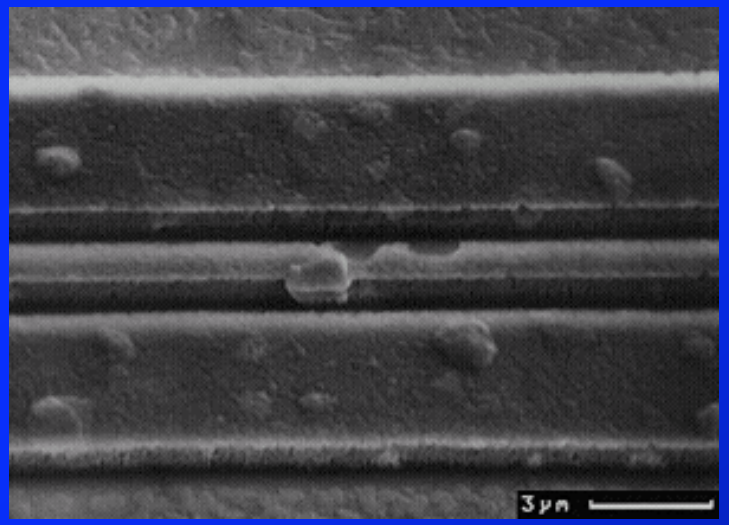
- For sizing:
  - Guardbanding, layout design rules
  - Device matching design rules
  - Regular fabric
- For threshold variation:
  - Graded wells
  - Upsizing devices
- For voltage variations:
  - Dynamic voltage control
  - Thermal management

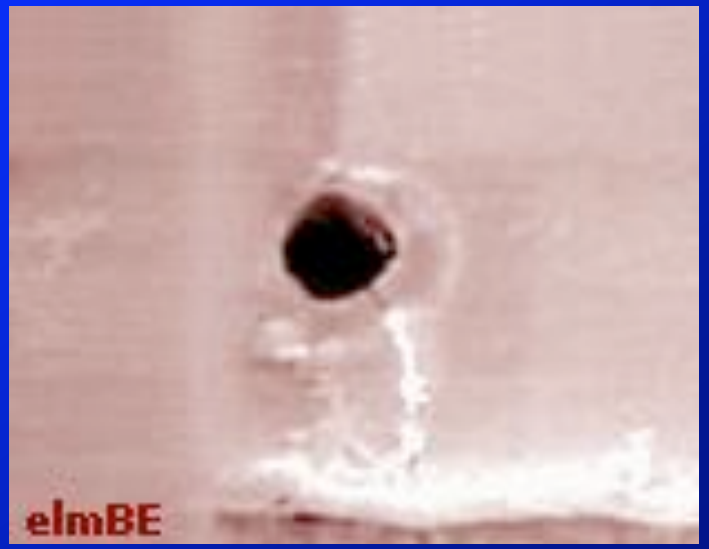
Malfunctions and faults

- Malfunctions can be:
  - Permanent, transient, intermittent
- Malfunctions are captured by:
  - Faults
    - Abstractions of the malfunctions
  - Failure modes
    - Way in which the malfunction manifests
  - Failure rates
    - Related to failure probability

Aging of materials (Permanent malfunctions)

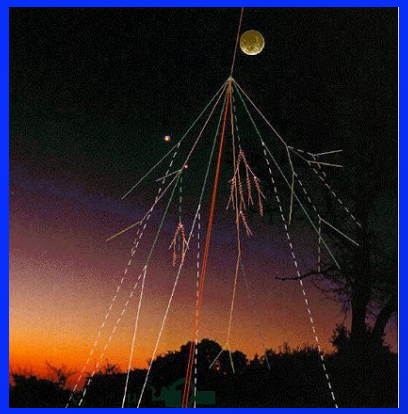
- Failure mechanisms
  - Electromigration
  - Oxide breakdown
  - Thermo-mechanical stress
- Temperature dependence
  - Arrhenius law

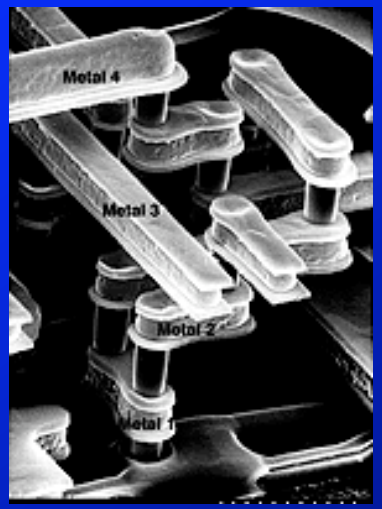




Sources of transient malfunctions

- Soft errors
  - Data corruption due external radiation exposure
- Crosstalk
  - Data corruption due to internal field exposure
- Both malfunctions manifest themselves as timing errors
  - Error containment

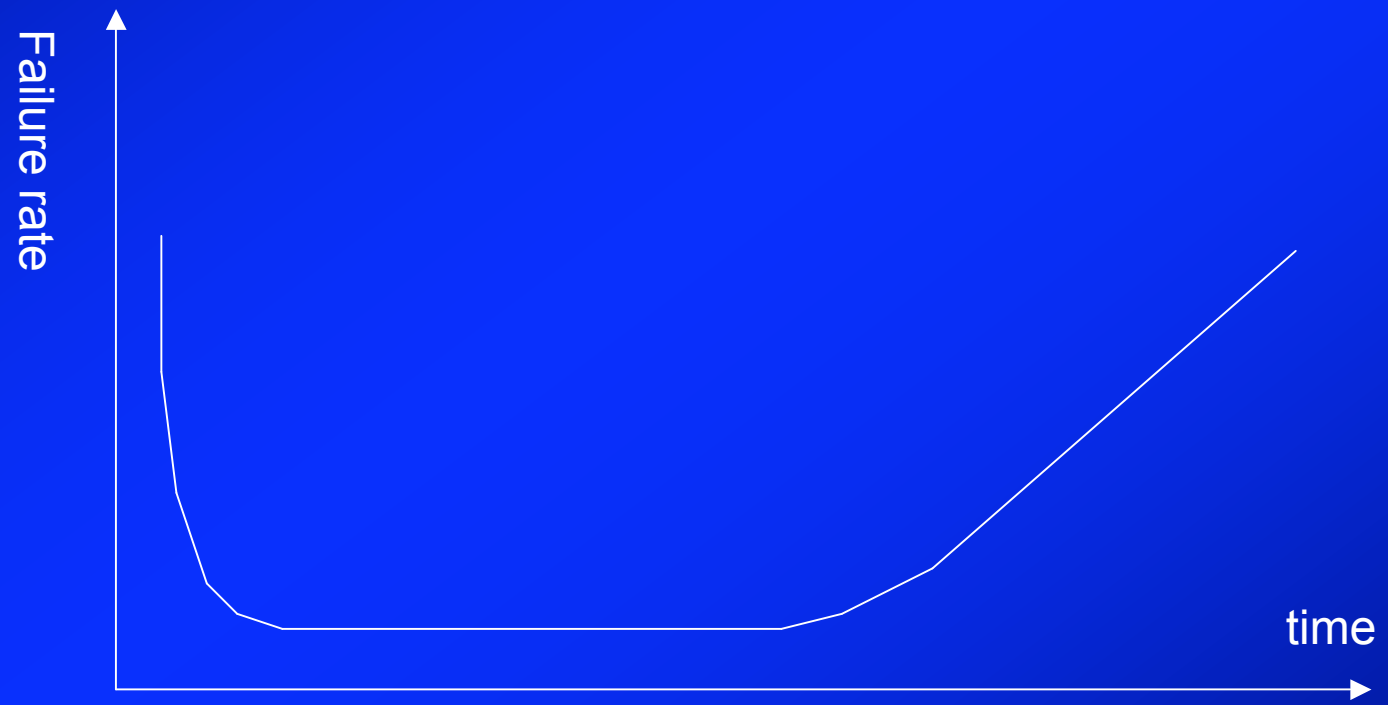




Defining the problems…

- Failure rate:
  - Assuming a unit works correctly in [0,t], the conditional probability $\lambda(t)$ that a unit fails in $[t, t+\Delta t]$
  - Typically the failure λ rate depends on
    - Temperature
    - Time (burn-in and aging)
    - Environmental exposure
      - Soft errors, EMI
  - Often the component failure rate is assumed to be constant for simplicity

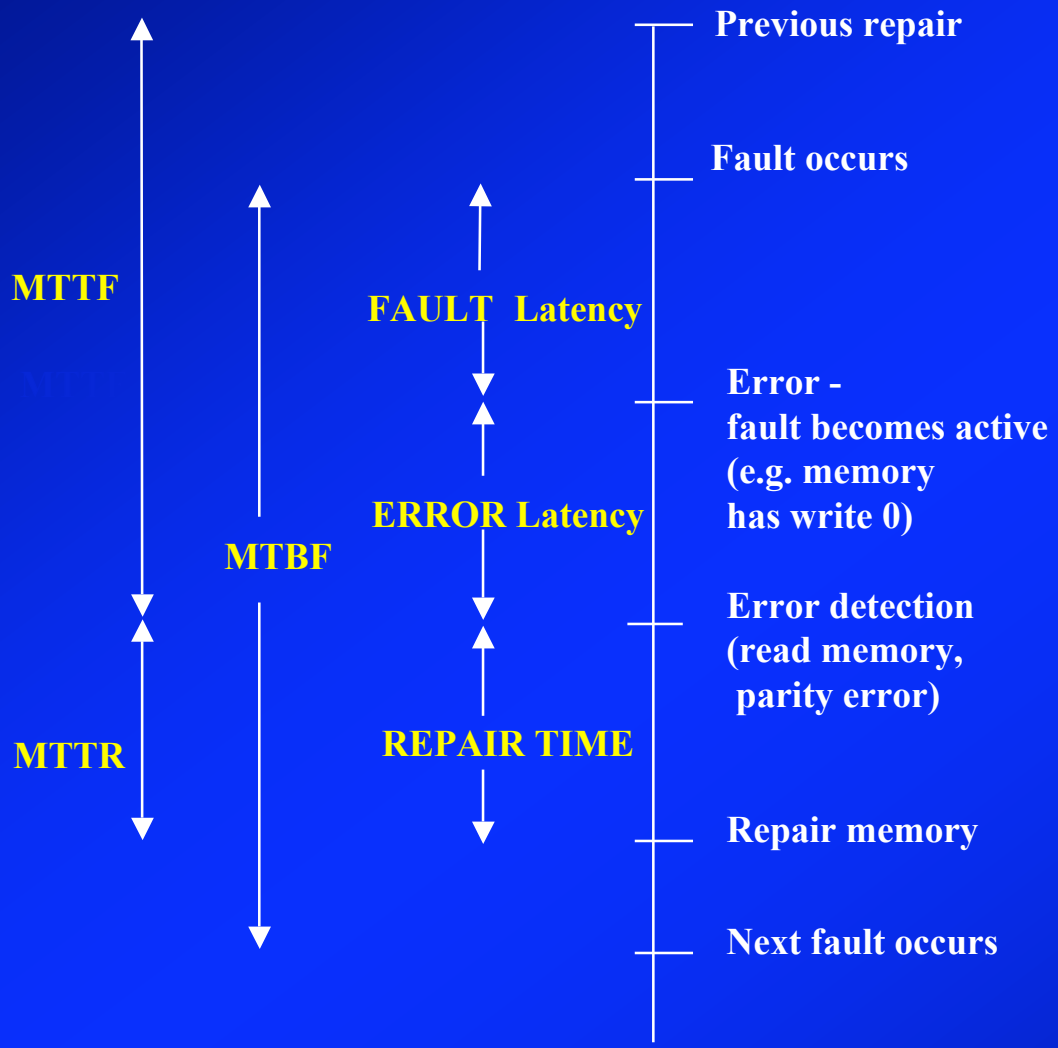
Failure rate
the bathtub curve



Reliability

- The probability function $R(t)$ that a system works correctly in $[0, t]$ without repairs
- Reliability is a function of time
  - If the system consist of a single component with constant failure rate λ, then
    - $R(t) = \exp(-\lambda t)$
  - The mean time to failure is $MTTF = 1/\lambda$
- In general, the MTTF is $E[t] = \int R(t)dt$

Dependability Concepts



Reliability:
a measure of the continuous delivery of service;
R(t) is the probability that the system survives (does not fail) throughout [0, t];
expected value: ***MTTF(Mean Time To Failure)***

Maintainability:
a measure of the service interruption
M(t) is the probability that the system will be repaired within a time less than t;
expected value: ***MTTR (Mean Time To Repair)***

Availability:
a measure of the service delivery with respect to the alternation of the delivery and interruptions
A(t) is the probability that the system delivers a proper (conforming to specification)service at a given time t.
expected value: ***EA = MTTF / (MTTF + MTTR)***

Safety:
a measure of the time to catastrophic failure
S(t) is the probability that no catastrophic failures occur during [0, t];
expected value:
MTTCF(Mean Time To Catastrophic Failure)

Reliability of complex systems

- A system is a connection of components
- System reliability depends on the topology
  - Series/parallel configurations
  - N out of K configurations
  - General topologies
- Common mode failures
  - Failure mode that affects all components
  - Examples:
    - Failure of voltage regulator for SoC
    - Failure of scheduler to process exception routines

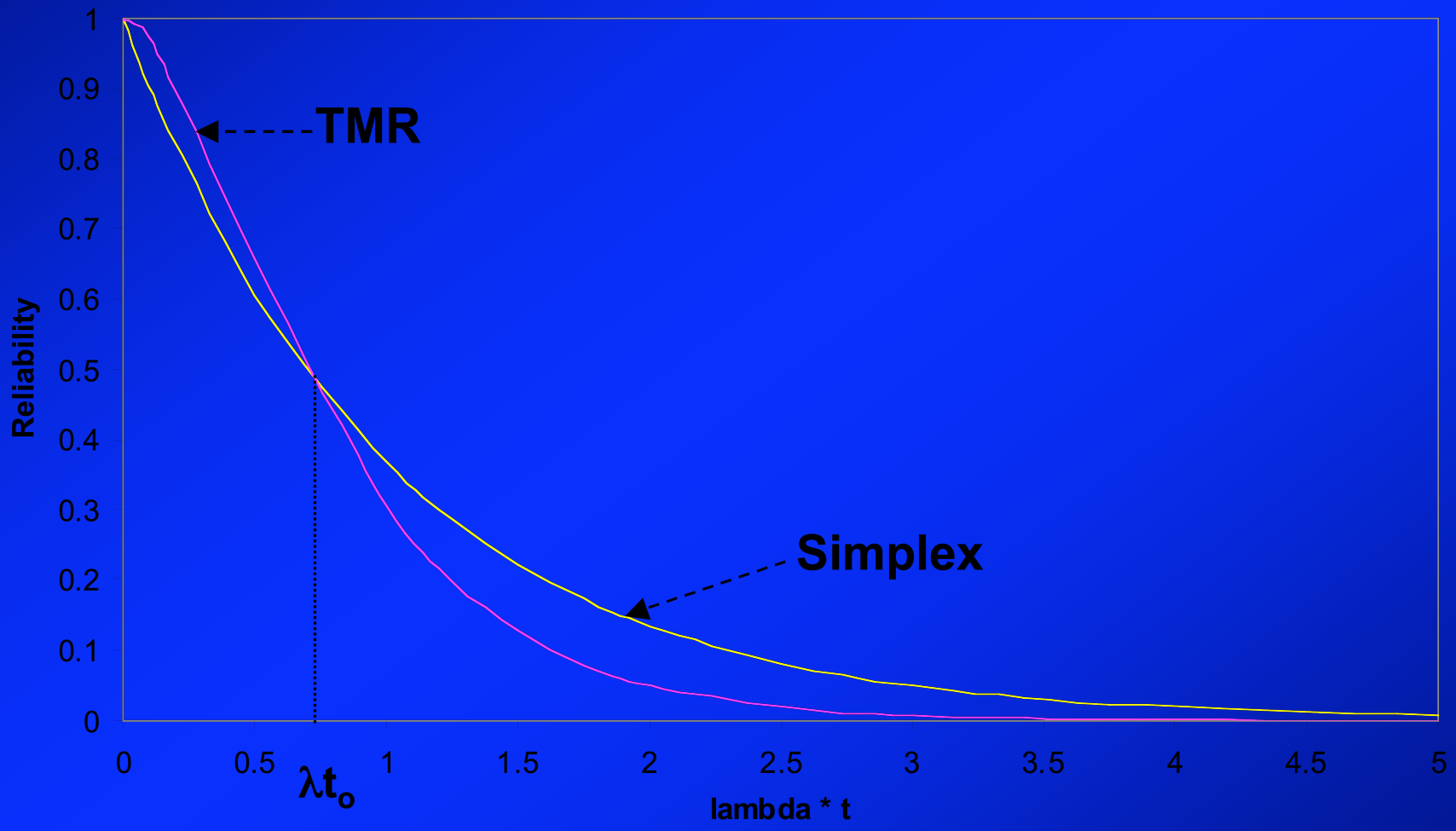
Very simple example

- For reliability analysis, a system consists of three components:
  - Processor, memory, bus
- All components have to be up at the same time to accomplish the mission
- The three components form a series configuration
- The system reliability is the product of the component reliabilities (if the failure rates are independent)
- Assume failure rates constant:
  - The system failure rate is the sum of the failure rates
  - The MTTF is its inverse

Example (2)

- For reliability analysis, a system consists of two processors:
  - A working processor suffices to accomplish the mission
- The two components form a parallel configuration
- The system unreliability is the product of the component unreliabilities (if the failure rates are independent)
  - $R(t) = 1 - [1-R_1(t)]\,[1-R_2(t)]$
  - Assume failure rates constant
  - The MTTF is $1/\lambda_1 + 1/\lambda_2 + 1/(\lambda_1 + \lambda_2)$
- Other relevant configurations:
  - Standby
  - Triple modular redundancy

TMR vs simplex reliability



Outline

- Introduction to reliable design
- Design for reliability
  - Component redundancy
  - Communication redundancy
  - Data encoding and error correction
  - Dealing with variability
- Summary and conclusions

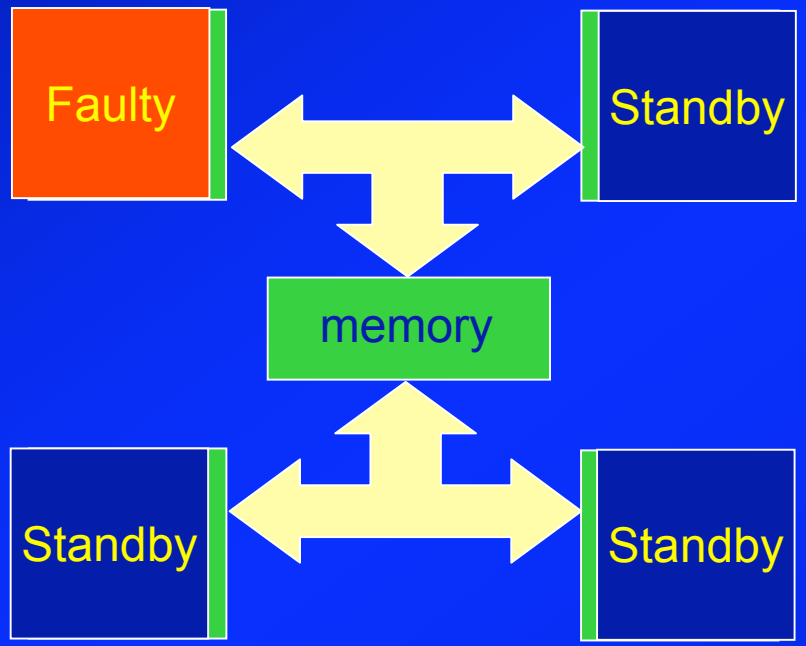
Design for reliability

- Hard failures
  - Exploit redundancy:
    - Components
    - Interconnect
- Soft failures
  - Encoding
  - Containment and rollback
- Variability
  - Timing-error tolerant circuits
  - Self-calibrating circuits

Providing component redundancy

- Component redundancy for enhanced reliability
  - Energy consumption penalty may be severe
- Power-managed standby components
  - Provide for temporary/permanent back-up
  - Provide for load and stress sharing
- Power management and reliability are intertwined:
  - PM allows reasonable use of redundancy on chip
  - Failure rates depend on effect of PM on components
- A programmable and flexible interconnection means is required

Example



When core operates
failure rate is higher as compared to standby unit

When core fails,
it is replaced by standby core

System management may alternate cores at high frequency, voltage and failure rate, to optimize long term reliability

Issues

- Analyze system-level reliability
  - as a function of a power management policy
- Determine a system management policy
  - to maximize reliability (over a time interval) and minimize energy consumption
- Determine a system management policy and system topology
  - to maximize reliability (over a time interval) and minimize energy consumption

Outline

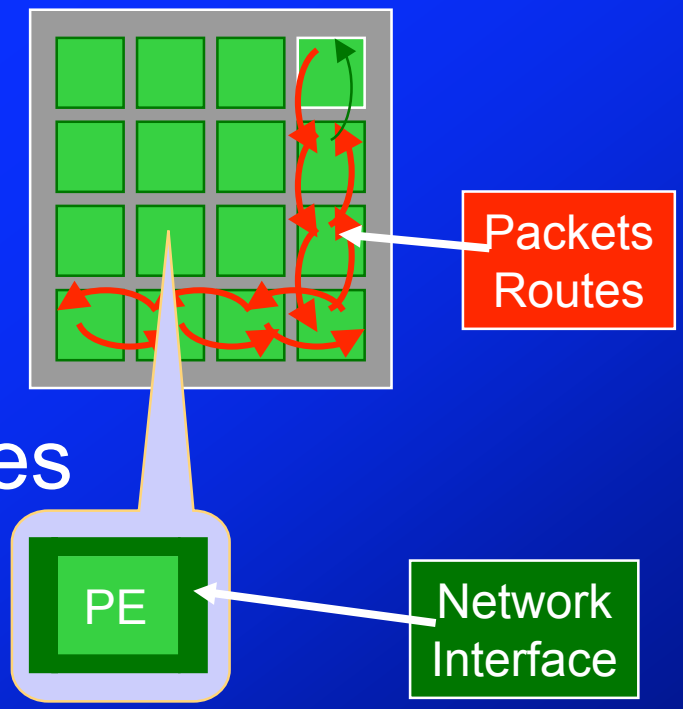
- Introduction to dependable design
- Design for reliability
  - Component redundancy
  - Communication redundancy
  - Data encoding and error correction
  - Dealing with variability
- Summary and conclusions

Why on-chip networking ?

- Provide a structured methodology for realizing on-chip communication schemes
  - Modularity
  - Flexibility
- Cope with inherent limitations of busses
  - Performance and power of busses do not scale up
- Support reliable operation
  - Layered approach to error detection and correction

Interconnect design in a multi-processing environment

- Most SoCs are multi-processors
  - Homogeneous
    - High performance computation
  - Heterogeneous
    - Application specific solutions
- Classic and *ad hoc* topologies
- Different QoS requirements
  - Best-effort services
  - Guaranteed performance



Providing communication reliability

- Some network topologies support multiple source/destination paths
  - Tolerate transient congestion, transient and permanent link malfunctions
- Error detection and correction
  - Physical links
    - Timing-errors detection by shadow latches
  - Switches and routers
    - Flit-level error detection and correction with CRCs
  - Network interface
    - Packet integrity check
  - Processor cores
    - Software data correctness check

Outline

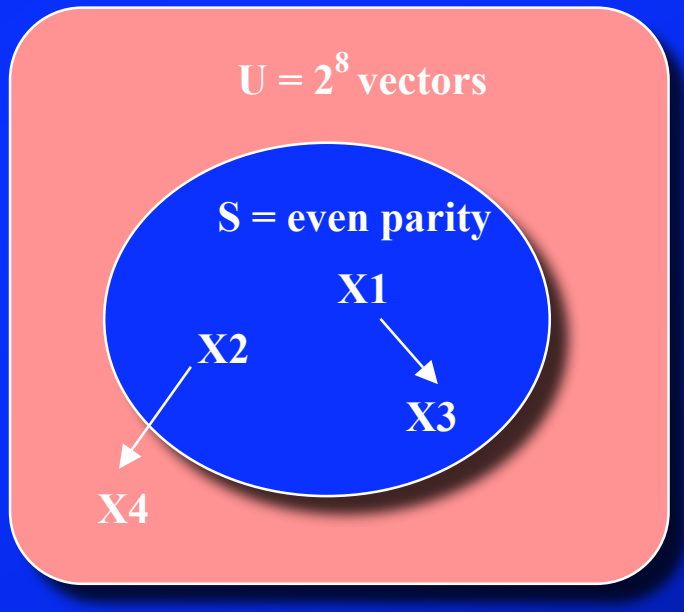
- Introduction to dependable design
- Design for reliability
  - Component redundancy
  - Communication redundancy
  - Data encoding and error correction
  - Dealing with variability
- Summary and conclusions

Encoding

- At logic level, codes provide means of masking and detecting errors
- Formally, a code is a subset S of universe U of possible vectors
- A noncode word is a vector in set U-S

X1 is a codeword <10010011>
Due to multiple bit error, becomes
X3 = <10011100>
not detectable

X2 is a codeword, becomes X4 noncode
detectable



Basic Concepts

- Consider 2^k messages (i.e. k bits)
- Encode messages with 2^k codewords using n-bit vectors
  - (n, k) code
  - Fraction k/n is called rate of code
- Hamming distance properties:
  - ***Hamming distance*** between two vectors x and y, d(x,y) is number of bits in which they differ.
  - ***Distance of a code*** is a minimum of Hamming distances between all pairs of code words.

Example: x = (1011), y = (0110)

w(x) = 3, w(y) = 2, d(x, y) = 3

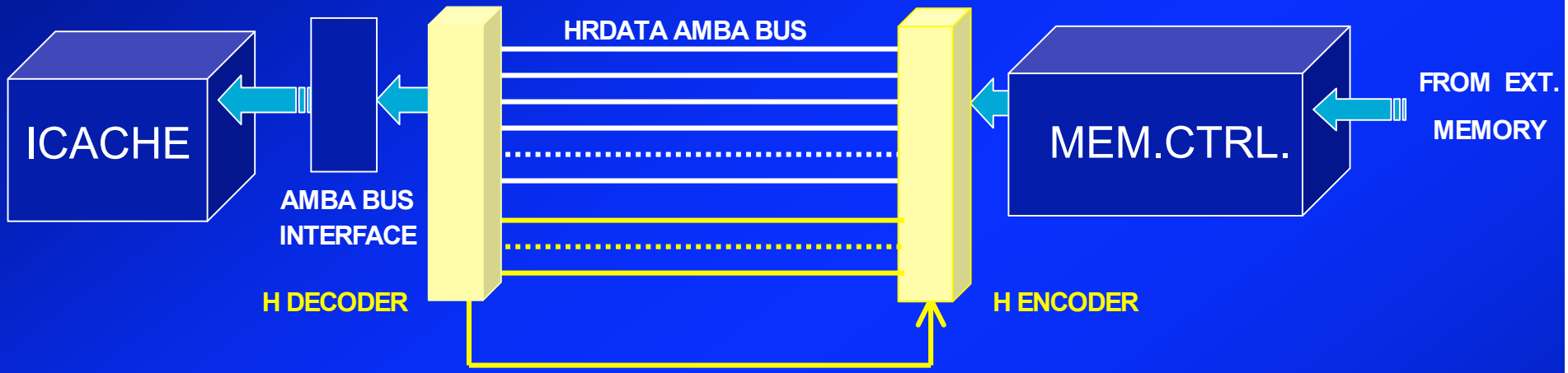
Distance Properties

- To **detect** all error patterns of ***Hamming distance*** ≤ **d, code distance must be** ≥ **d+1**
  - e.g., code with distance 2 can detect single-bit errors
- To **correct** all error patterns of ***Hamming distance*** ≤ **c, code distance must be** ≥ **2c + 1**
  - e.g., code with distance 3 can correct single-bit errors
- To **detect** all patterns of ***Hamming distance*** **d**, and correct all patterns of ***Hamming distance*** **c, code distance must be** ≥ **2c + d + 1**
  - e.g., code with distance 5 can correct double errors and detect quadruple errors

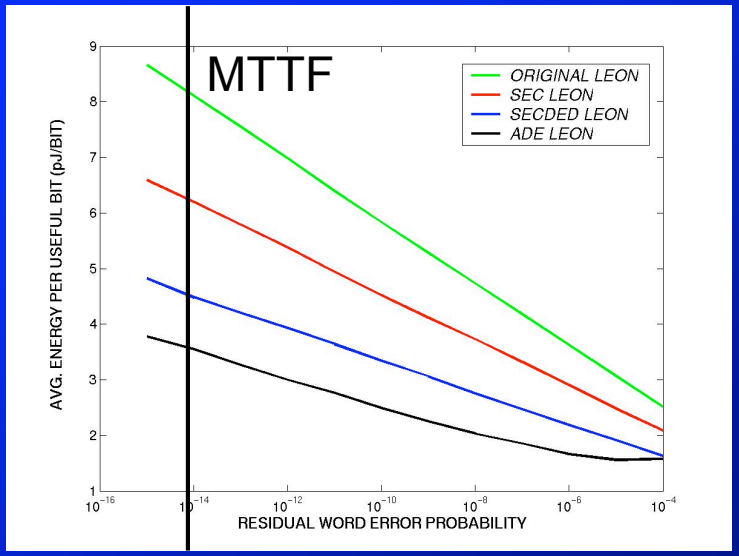
Codes for Storage and Communication
Cyclic Codes

- Cyclic codes are parity check codes with additional property that cyclic shift of codeword is also a codeword
  - if (C_{n-1}, C_{n-1} ... C_1, C_0) is a codeword, (C_{n-2}, C_{n-3}, ... C_0, C_{n-1}) is also a codeword
- Cyclic codes are used in
  - sequential storage devices, e.g. tapes, disks, and data links
  - communication applications
- An (n,k) cyclic code can detect single bit errors, multiple adjacent bit errors affecting fewer than (n-k) bits, and burst transient errors
- Cyclic codes require less hardware
  - Use linear feedback shift registers (LFSR)
  - Parity check codes require complex encoding, decoding circuit using arrays of EX-OR gates, AND gates, etc.

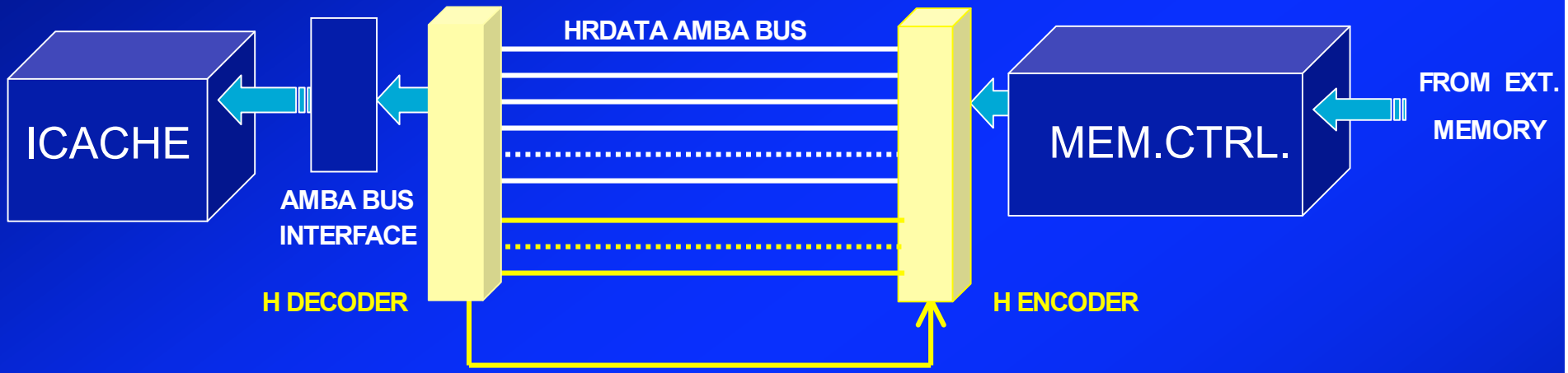
Error-resilient coding



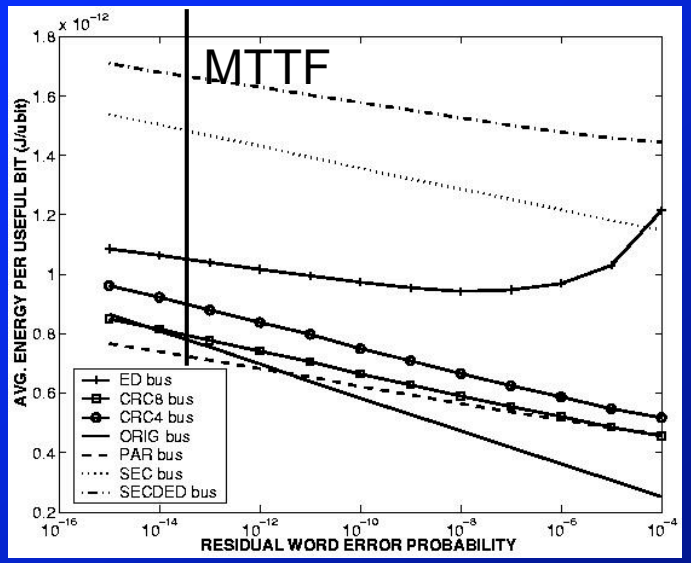
- Compare original AMBA bus to extended bus with error detection and correction or retransmission
  - SEC coding
  - SEC-DED coding
  - ED coding
- Explore energy efficiency [Bertozzi]



Error-resilient coding



- Compare original AMBA bus to extended bus with error detection and correction or retransmission
  - SEC, SEC-DEC, ED coding
  - CRC4 and CRC8 coding
- On shorter links, CRC become competitive when ENC/DEC power is accounted for [Bertozzi]



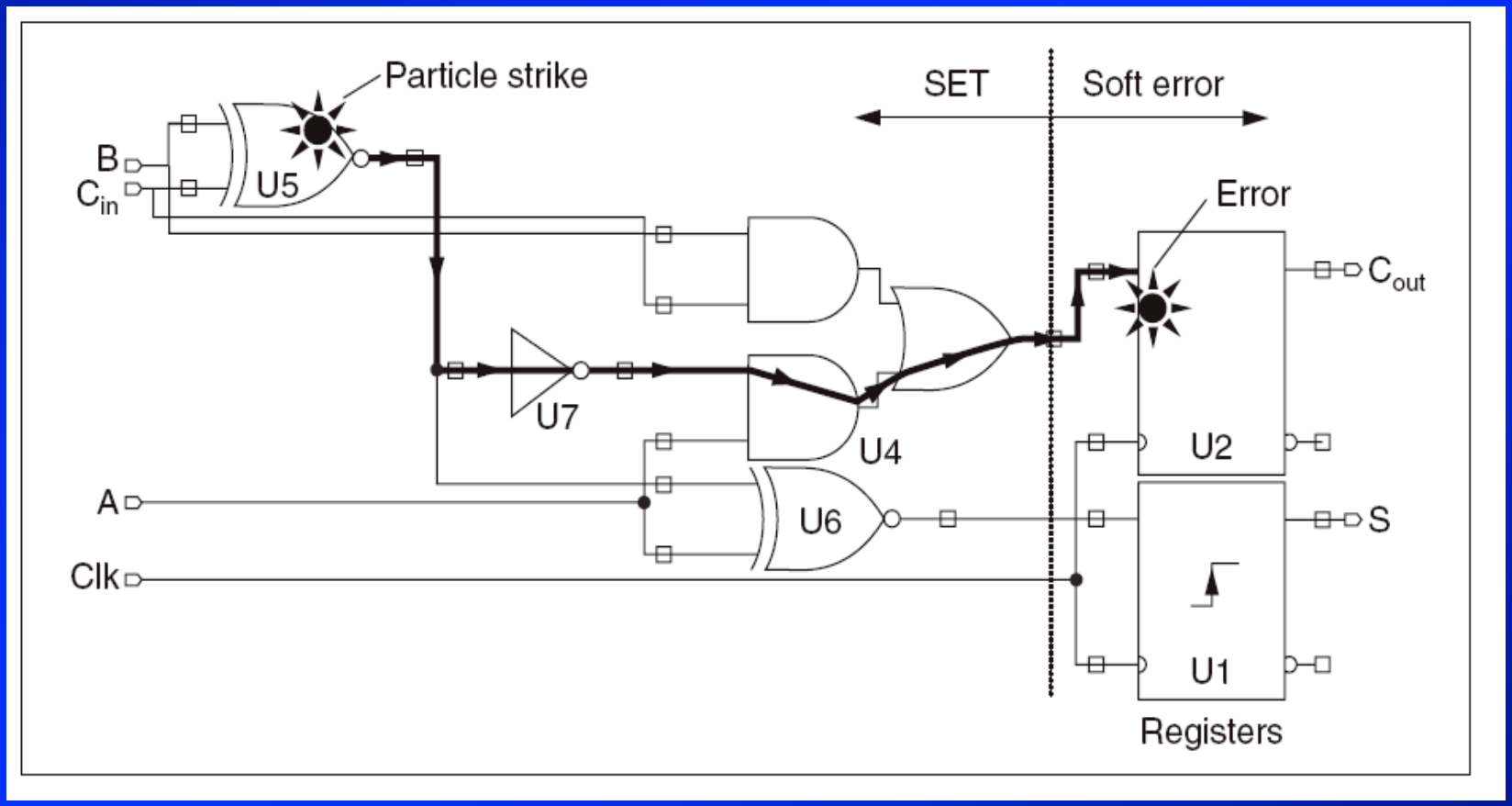
Outline

- Introduction to reliable design
- Design for reliability
  - Component redundancy
  - Communication redundancy
  - Data encoding and error correction
  - Dealing with variability
- Summary and conclusions

Dealing with variability

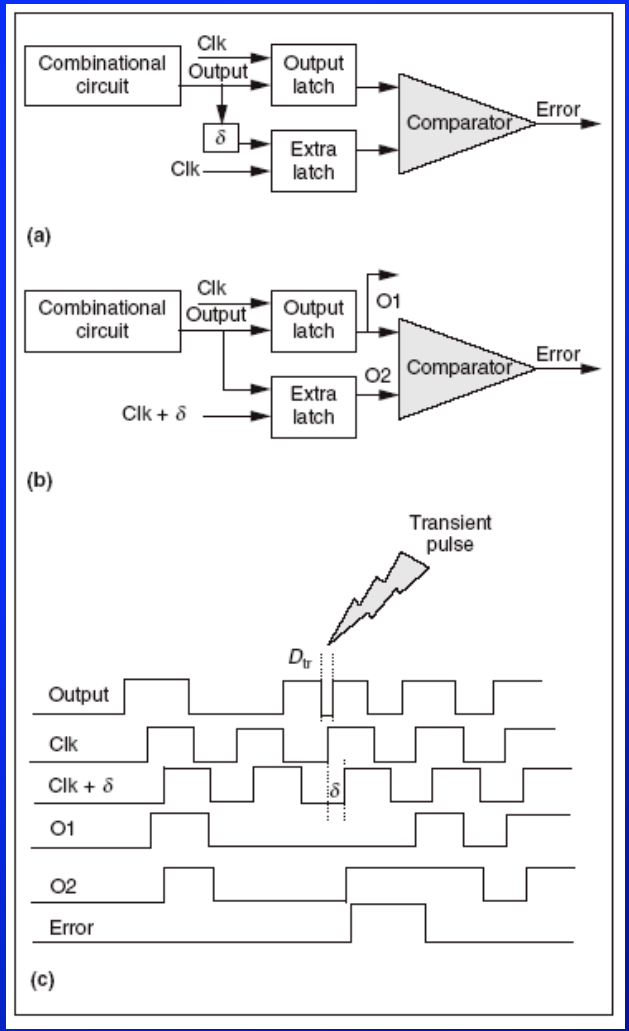
- Most variability problems induce timing errors
  - Power supply variation
  - Wire length estimation
  - Crosstalk
  - Soft errors
- Timing errors can be contained while using an aggressive operating frequency
  - Timing errors are rare
  - Micro rollback
  - Delayed clocks

Propagation of soft error



Radiation-hardened registers

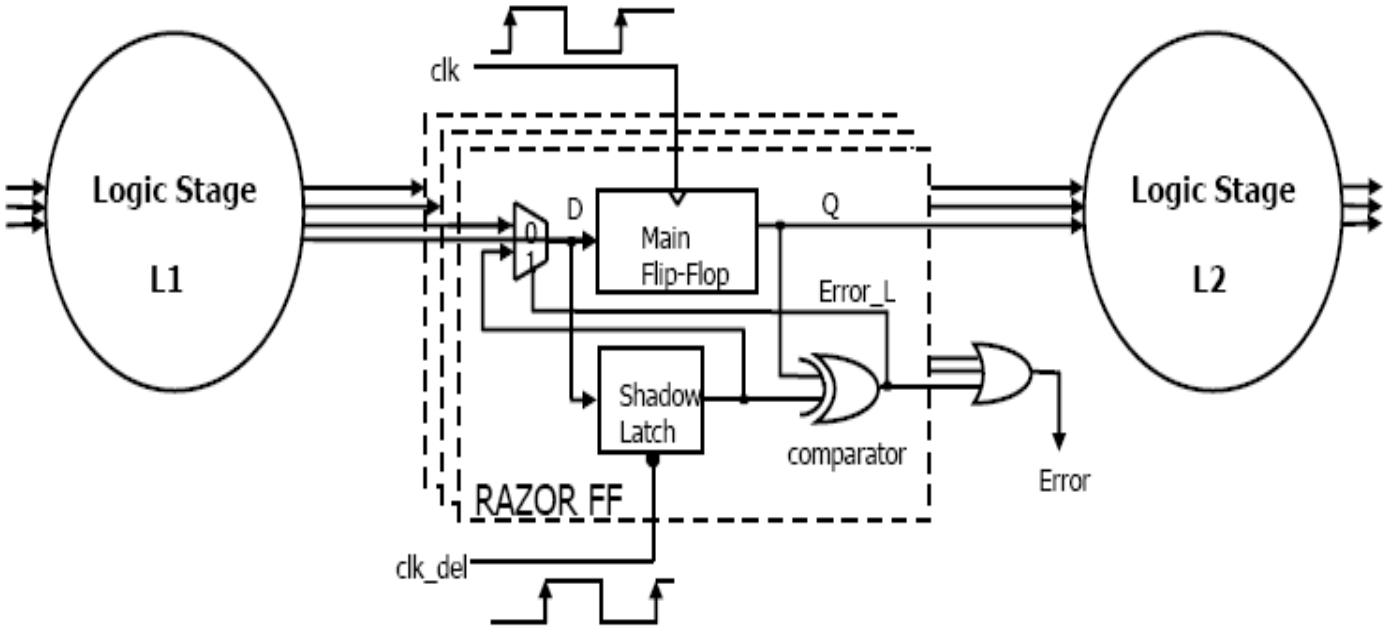
- Protection against soft errors
  - Timing errors
- Each latch is duplicated
  - Shadow latch has delayed clock
- Comparison between original and shadow latch detects error
  - Error correction is possible



[IROC Technologies]

The *razor* approach

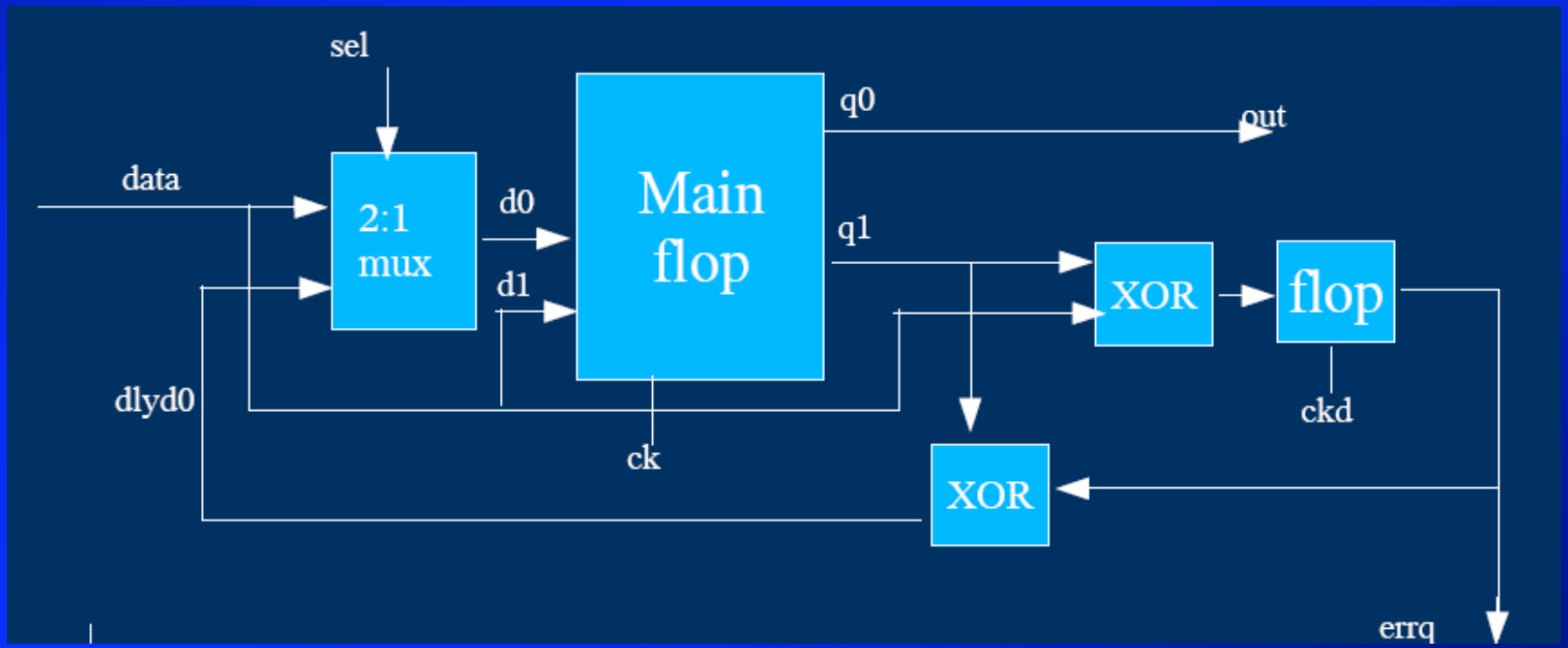
- Applicable to processor design
- Try to shave off power consumption
  - Reduce voltage margins with in situ error detection and correction for delay faults
- Compare two samples of data



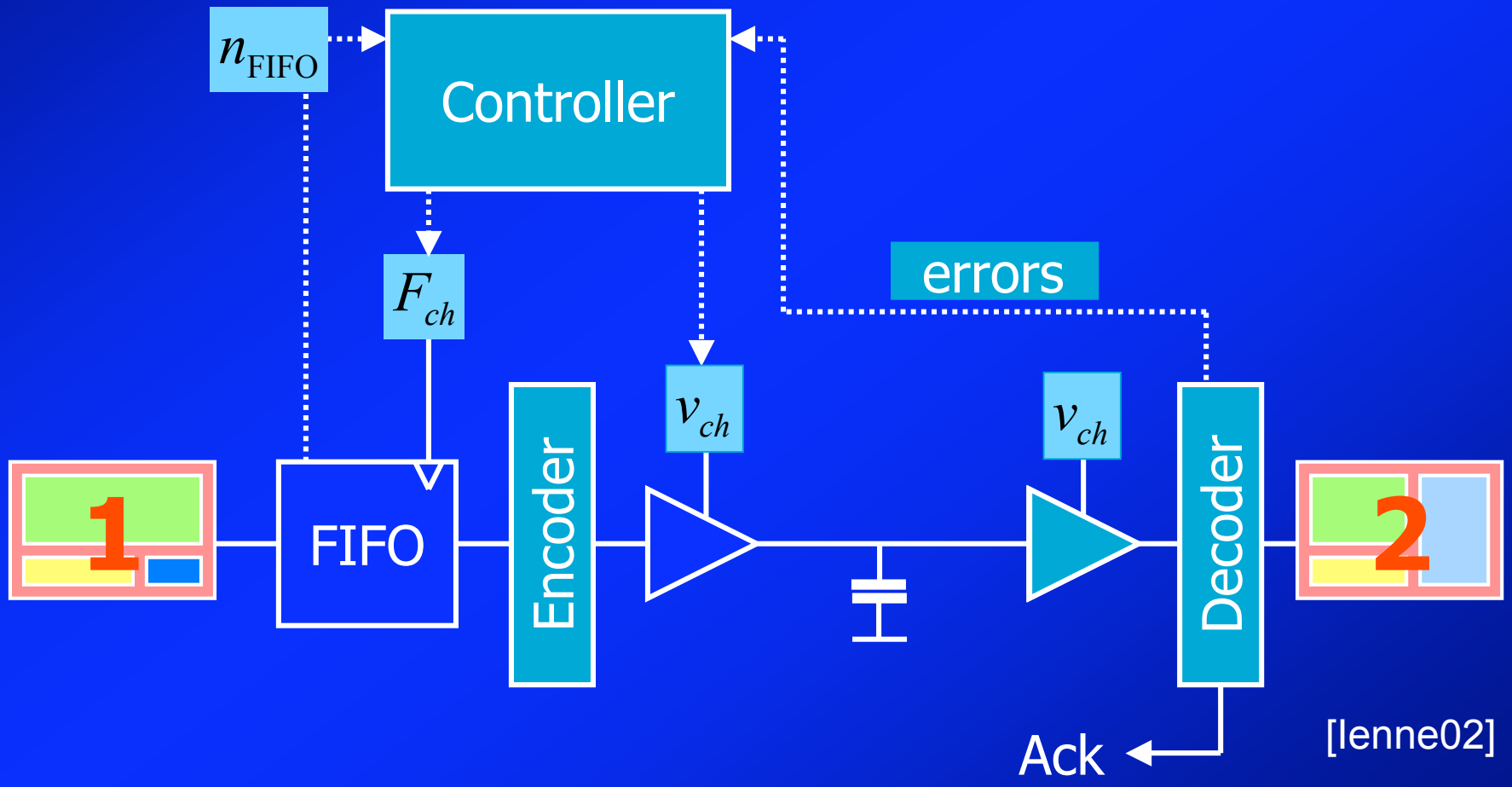
[Austin 03]

The *t-error* approach

- Applicable to NoC communication
- Use aggressive clocking frequency
  - Address data-dependent wire propagation delay
  - Compare two samples of data
  - Correct data and propagate with one cycle delay penalty



Adaptive low-power transmission scheme



[Ienne02]

Outline

- Introduction to reliable design
- Design for reliability
  - Component redundancy
  - Communication redundancy
  - Data encoding and error correction
  - Dealing with variability
- Summary and conclusions

Achieving reliable SoCs Summary

- Exploit redundancy
  - Component-level redundancy
    - Supported by modularity of micro-networks
    - Requires energy management
  - Communication link redundancy
    - Supported by path diversity of micro-networks
- Error detection and correction
  - Encoding, CRCs, self-checking circuits
- Dealing with variability
  - Detect and correct timing errors

Conclusions

- Reliable design is important in many application domains
- Reliable MPSOC design can be achieved with system-level techniques to obviate the limitations of the materials and environment
- Structured design methodologies and structured interconnect design support reliable design


Thank You!
Merci!