

# **DESIGNING ROBUST SYSTEMS with UNCERTAIN INFORMATION**

**Giovanni De Micheli    CSL - Stanford University**

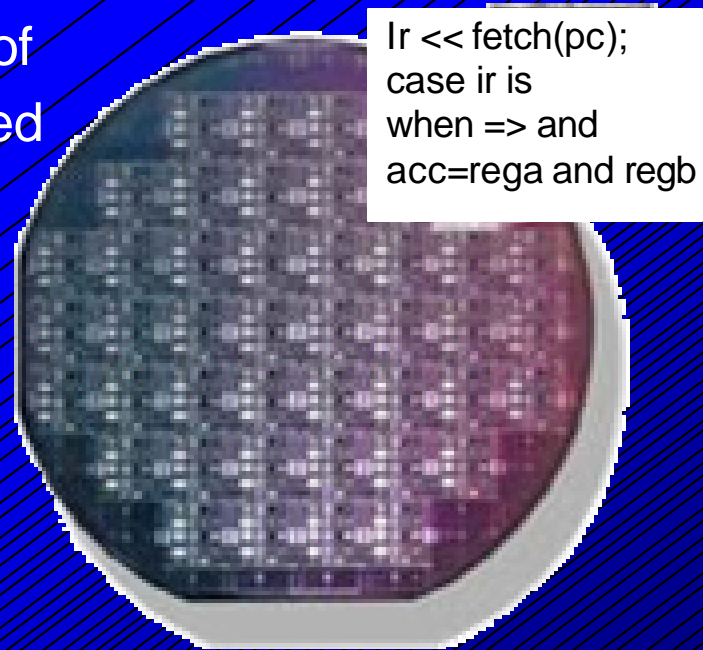
# The philosophical paradigm

- Science at the onset of the XX century
  - *Laplacian determinism*
    - The future state of the universe can be determined from its present state
  - *Quantum theory and uncertainty*
    - We can neither observe nor control microscopic features with accuracy



# The philosophical paradigm

- Design technology at the onset of the XXI century
  - *Design determinism*
    - The complete behavior and features of a microelectronic circuit can be derived from a hardware model
    - Synthesis technology
  - *Design uncertainty* with nanoscale technologies
    - Need for high-level abstractions
    - Inaccuracy of low-level models




# The economic perspective


- System on Chip (SoC) design:
  - Increasingly more complex:
    - Many detailed electrical problems
    - Integration of different technologies
  - Increasingly more expensive and risky
    - A mask set may cost over a million dollars
    - A single functional error can kill a product
  - Fewer design starts
- Large volume needed to recapture hw costs
  - Software solutions are more desirable

# The SoC market

- SoCs find application in many embedded systems
- Concerns:



Correctness  
Reliability and safety  
Robustness



Performance  
Energy consumption  
Cost



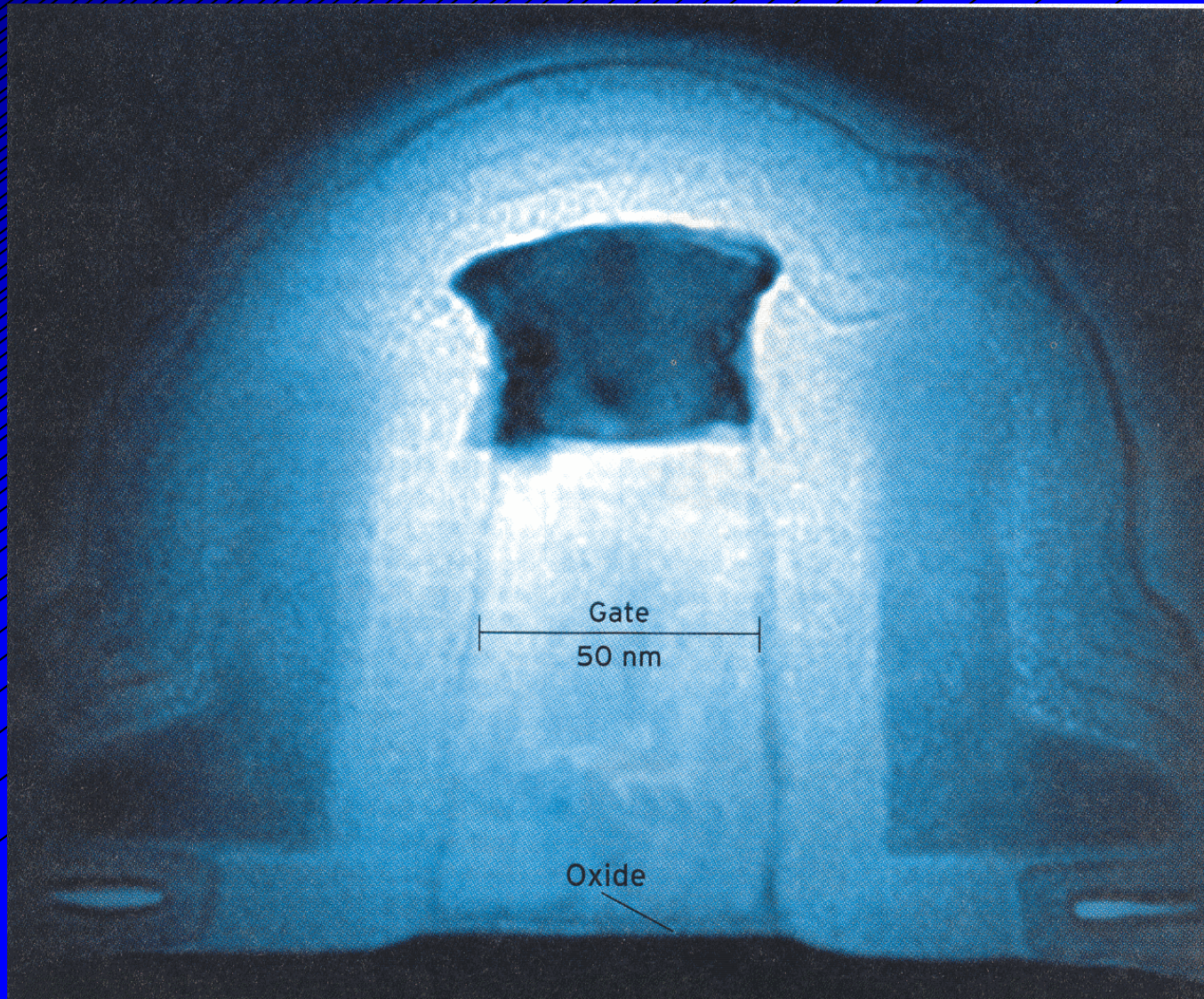
# Robust design

- SoCs must preserve correct operation and performance:
  - Under varying environmental conditions
  - Under changes of design assumptions
- Designing correct and performing circuits becomes increasingly harder
  - Too many factors to take into account
- Paradigm shift needed
  - Design error-tolerant and adaptive circuits

# Issues

- **Extremely small size**
  - Coping with deep submicron (DSM) technologies
    - Spreading of parameters
- **Extremely large scale**
  - System complexity
    - Changing environmental conditions
- **New fabrication materials**
  - Novel technologies
    - How to make the leap

# Extremely small size





# Silicon technology roadmap

Year	Gate length ( <i>nm</i> )	Transistor density ( <i>million/cm<sup>2</sup></i> )	Clock rate ( <i>GHz</i> )	Supply voltage ( <i>V</i> )
2002	75	48	2.3	1.1
2007	35	154	6.7	0.7
2013	13	617	19.3	0.5

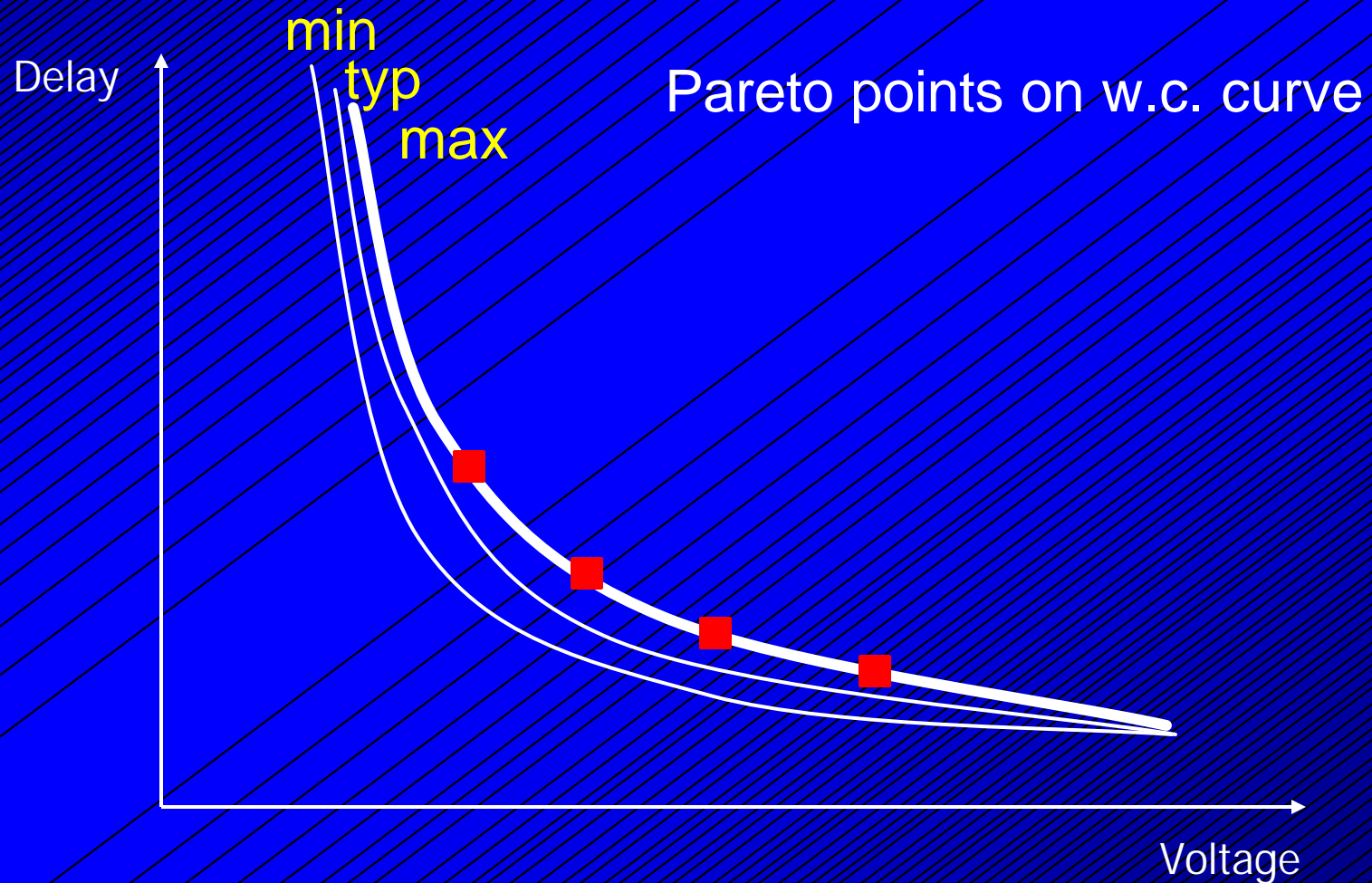
# Qualitative trends

- Continued gate downscaling
- Increased transistor density and frequency
  - ✍ Power and thermal management
- Lower supply voltage
  - ✍ Reduced noise immunity
- Increased spread of physical parameters
  - ✍ Inaccurate modeling of physical behavior

# Critical design issue

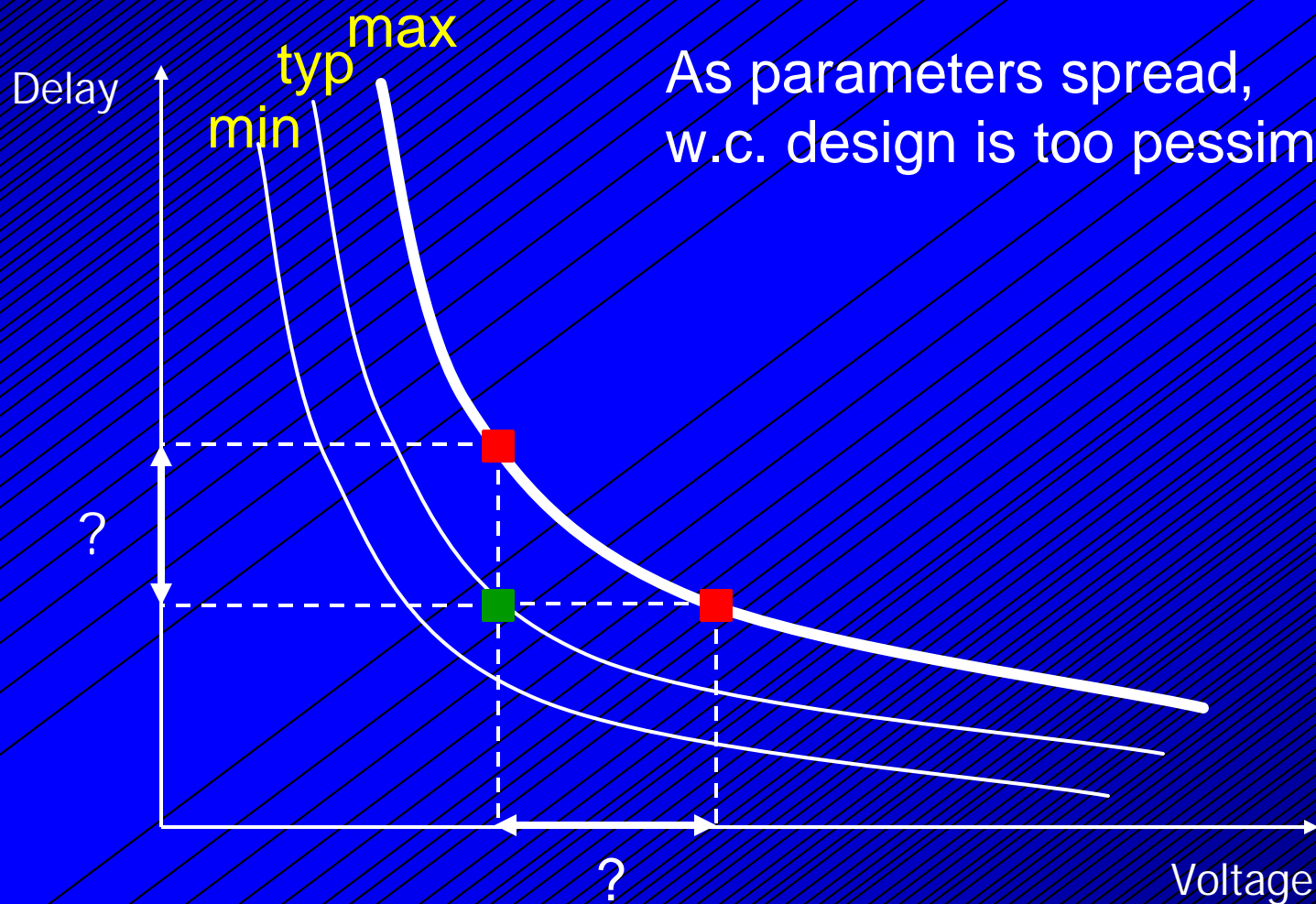
- Achieve desired performance levels with limited energy consumption
- Dynamic power management (DPM)
  - Component shut off
  - Frequency and voltage downscaling
- Explore (at run time) the voltage/delay trade off curve

# Design space exploration worst case analysis



# Adaptive design space

## worst case analysis



# Self-calibrating circuits

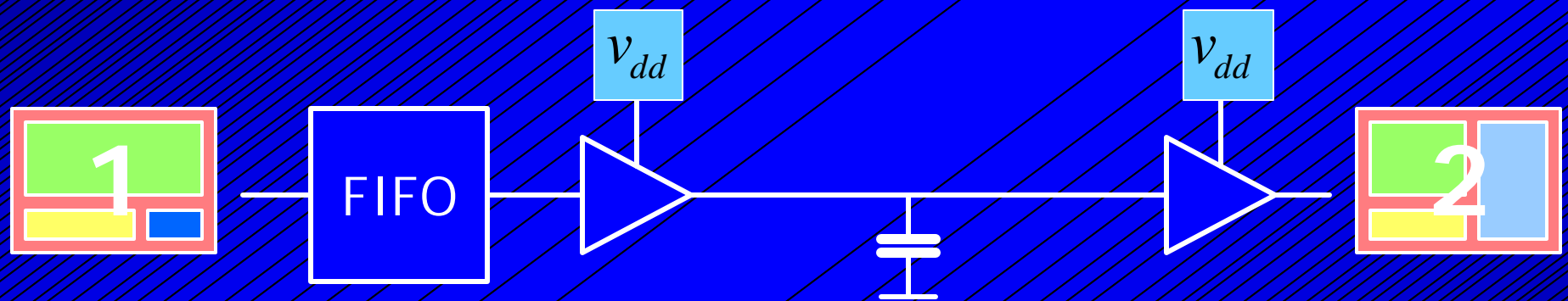
- The operating points of a circuit should be determined on-line
  - Variation from chip to chip
  - Operation at the edge of failure
- Analogy
  - Sailing boat tacking against the wind
  - Max gain when sailing close to wind
    - When angle is too close, large loss of speed



# How to calibrate?

- General paradigm
  - A circuit may be in *correct* or *faulty* operational state, depending on a parameter (e.g., voltage)
  - Computed/transmitted data need checks
    - If data is faulty, data is recomputed and/or retransmitted
  - Error rate is monitored on line
  - Feedback loop to control *operational state parameter* based on error rate
- **Circuits can generate errors:**
  - Errors must be detected and corrected
  - Correction rate is used for calibration

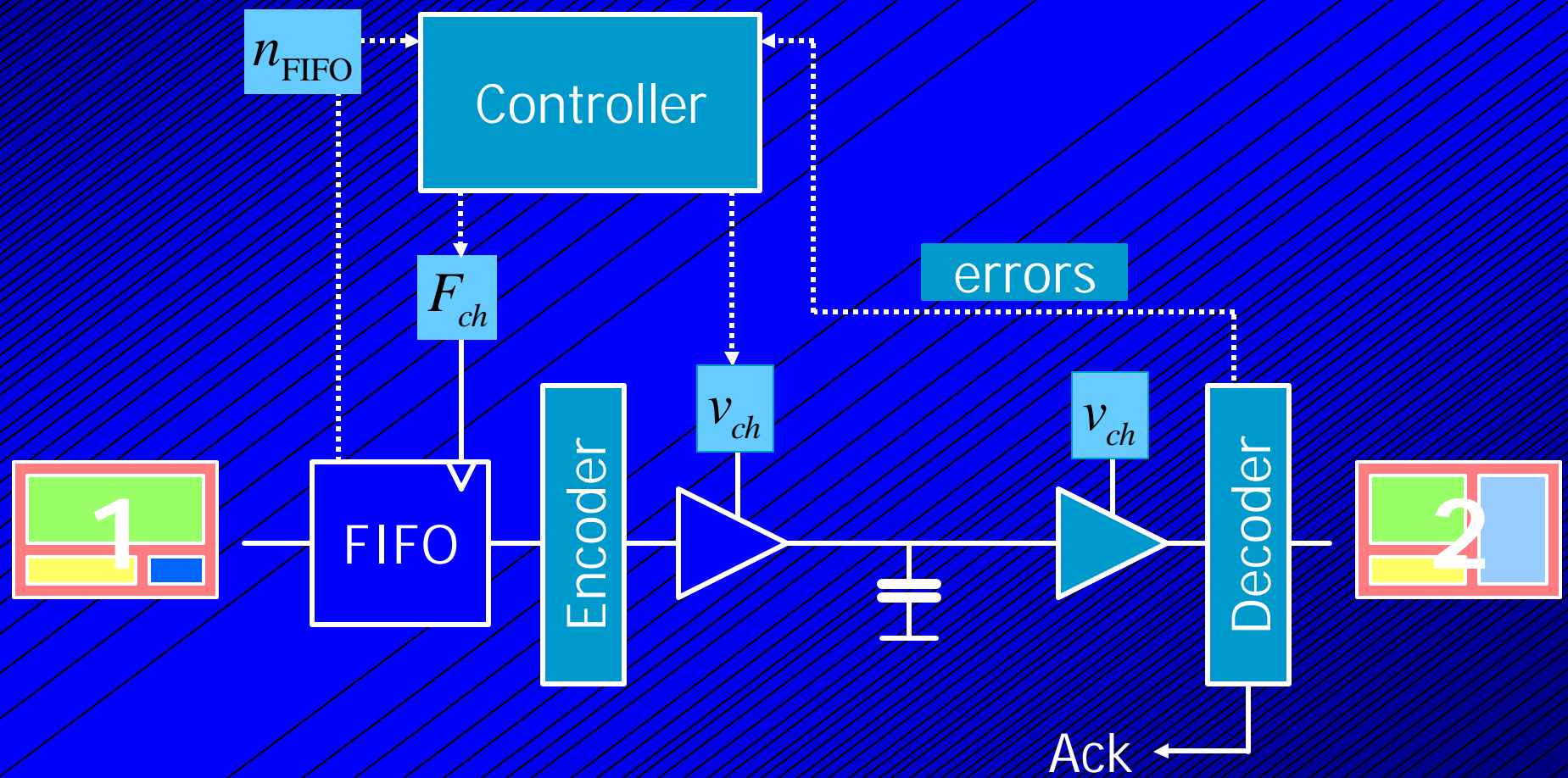
# Example: on chip transmission scheme



- Globally asynchronous, locally synchronous (GALS)
- FIFO for decoupling
- Variable transmission frequency



# Adaptive low-power transmission scheme



# Self-calibration

- Self-calibration makes circuit robust against:
  - Design process variations
  - External disturbances
    - E.g., soft errors, EM interference, environment
- Self-calibration may take different embodiments
  - May be applied during normal operation
    - To compensate for environmental changes
  - May be used at circuit boot time
    - To compensate for manufacturing variations
- **General paradigm to cope with DSM problems**

# Extremely large scale

- Engineers will always attempt to design chips at the edge of human capacity
- Challenges:
  - Large scale: billion transistor chips
  - Heterogeneity: digital, analog, RF, optical, MEMS, sensors, micro-fluidics
- Many desiderata: high performance, low power, low cost, fast design, small team, ...

# Component-based design

- SoCs are designed (re)-using large macrocells
  - Processors, controllers, memories...
  - *Plug and play* methodology is very desirable
  - Components are qualified before use
- Design goal:
  - Provide a functionally-correct, reliable operation of the interconnected components
- **Critical issues:**
  - Properties of the physical interconnect
  - Achieving robust system-level assembly

# Physical interconnection

- Electrical-level information transfer is **unreliable**
  - **Timing errors**
    - Delay on global wires and delay uncertainty
    - Synchronization failure across different islands
    - Crosstalk-induced timing errors
  - **Data errors:**
    - Data upsets due to EM interference and soft errors
- **Noise** is the abstraction of the error sources
- The problem will get more and more acute as geometries and voltages scale down

# Systems on chips: *a communication-centric view*

- Design component interconnection under:
  - **Uncertain** knowledge of physical medium
  - **Incomplete** knowledge of environment
    - Workload, data traffic, ...
- Design interconnection as a **micro-network**
  - Leverage network design technology
  - Manage information flow
    - To provide for performance
  - Power-manage components based on activity
    - To reduce energy consumption

# Micro-network characteristics

- Micro-networks require:
  - Low communication **latency**
  - Low communication **energy consumption**
  - Limited adherence to **standards**
- SoCs have some physical parameters that:
  - Can be predicted accurately
  - Can be described by stochastic distributions

# Micro-network stack

## Software

- application
- system

## Architecture and control

- transport
- network
- data link

## Physical

- wiring

Design choices at each stack level affect:

- Communication speed
- Reliability
- Energy

Control Protocols:

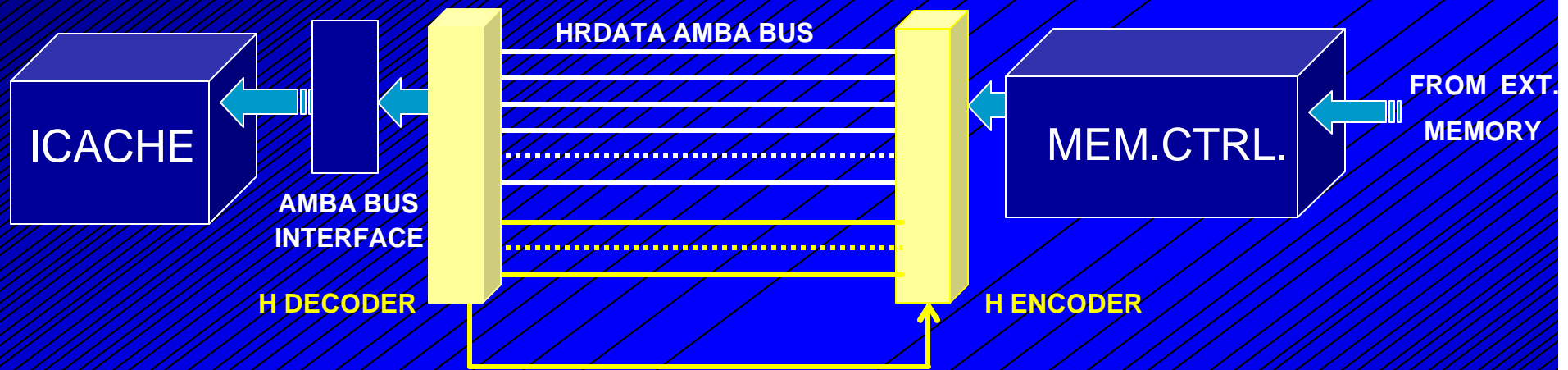
- Layered
- Implemented in Hw or Sw
- Providing error correction



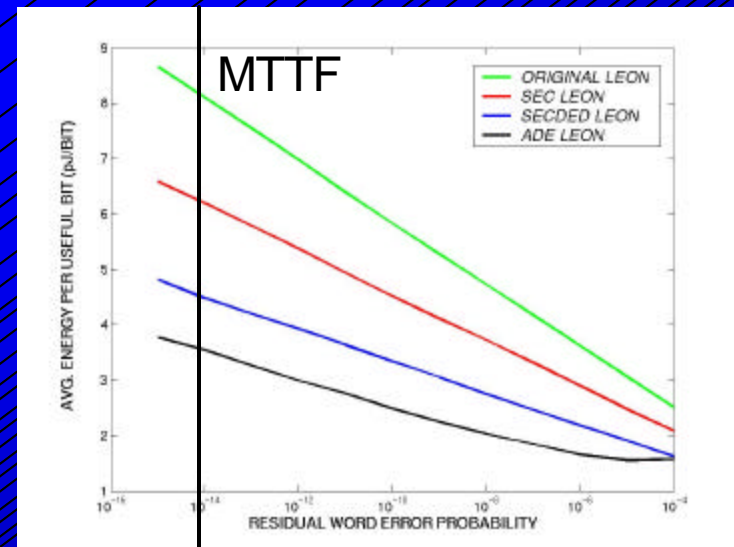
# Achieving robustness in micro-networks

- Error detection and correction is applied at various layers in micro-networks
- Paradigm shift:
  - Present design methods reduce noise
    - Physical design (e.g., sizing, routing)
  - Future methods must cope with noise
    - Push solution to higher abstraction levels

# Data-link protocol example: error-resilient coding

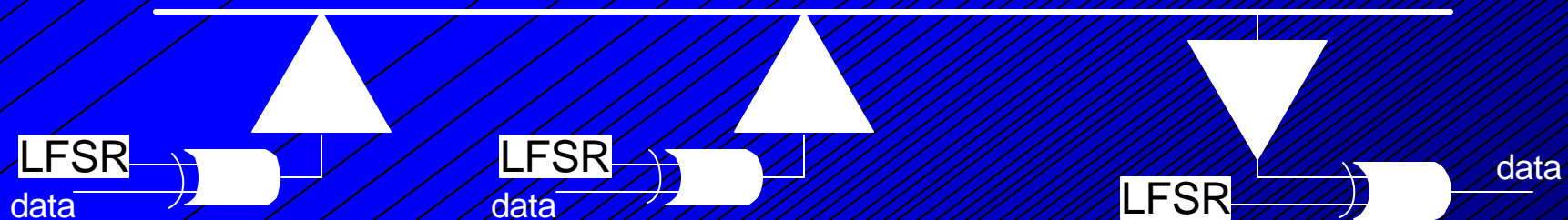


- Compare original AMBA bus to extended bus with error detection and correction or retransmission
  - SEC coding
  - SEC-DED coding
  - ED coding
- Explore energy efficiency



# Advanced bus techniques: CDMA on bus

- Motivation: many data sources
  - Support multiple concurrent write on bus
  - Discriminate against background noise
- Spread spectrum of information
  - Driver/receiver multiply data by random sequence generated by LFSR
    - LFSR signature is key for de-spreading



# Going beyond buses

- Buses:
  - Pro: simple, existing standards
  - Contra: performance, energy-efficiency, arbitration
- Other network topologies:
  - Pro: higher performance, experience with MP
  - Contra: physical routing, need for network and transport layers
- **Challenge: exploit appropriate network architecture and corresponding protocols**

# Network and transport layers

- Information is in **packets**
- **Network** issues:
  - Network **switching**
    - Circuit, packet, cut-through, wormhole
  - Network **routing**
    - Deterministic and adaptive routing
- **Transport** issues:
  - Decompose and reconstruct information
  - Packet granularity
  - Admission/congestion control

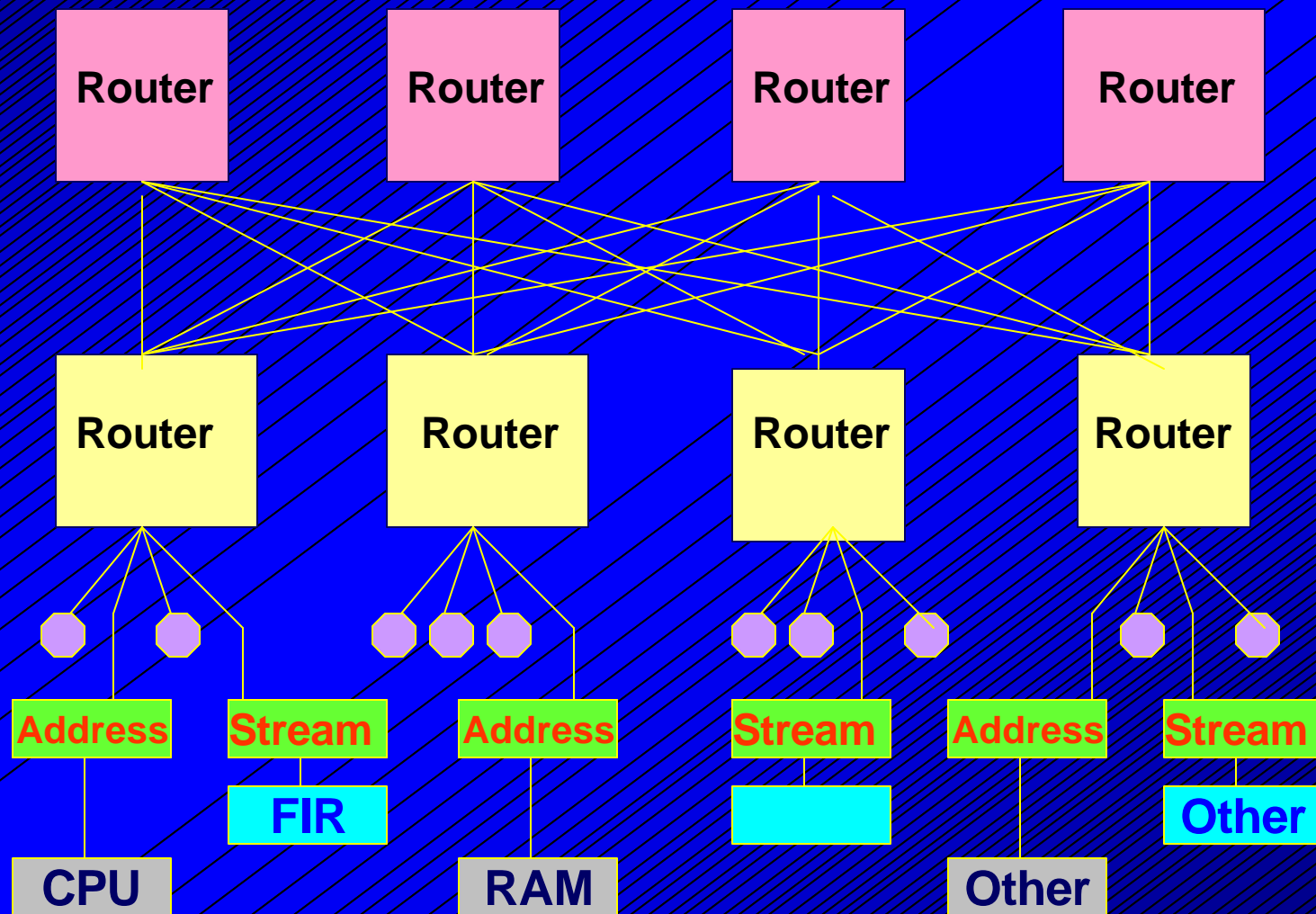
# SPIN micro-network

- Applied to SoCs
- 36-bit packets
  - Header: destination
  - Trailer: checksum



- **Fat-tree** network architecture
- **Cut-through** switching
- Deterministic *tree* routing

# SPIN micro-network



# Benefits of packets

- Reliable error-control mechanism
  - With small overhead
- Exploit different routing paths
  - Spread information to avoid congestion
- Several user-controllable parameters
  - Size, retransmission schemes, ...
- Use retransmission rate for calibrating parameters



# System assembly around micro-network

- Network architecture provides backbone
- Component *plug and play*:
  - Programmable network interface
  - Reconfigurable protocols
  - Recognize network and self configure
- **Self-assembly of SoCs addresses the issue of component reuse and heterogeneity**

# Extremely large scale design

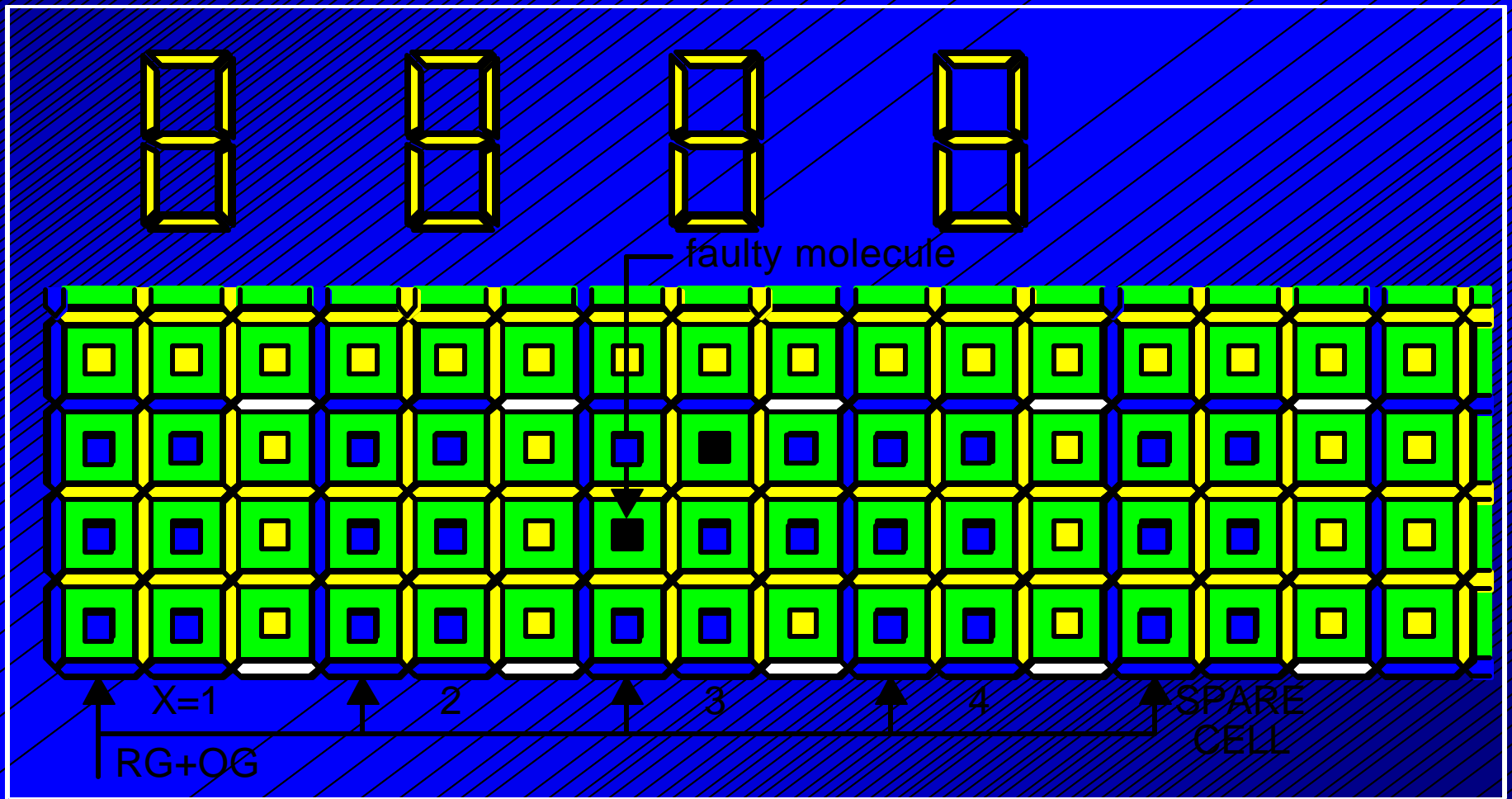
- **Heterogeneous** components with malleable interfaces
- **Macroscopic self-assembly**
  - Exploit degrees of freedom in component/interface specifications
  - Self-configuration realizes interfacing details abstracted by designers
  - Self-configuration, together with redundancy, addresses self-correction of some possible design errors
- **Self-healing**
  - Correcting for run-time failures
  - Method to increase availability and robustness

# Example: Biowall

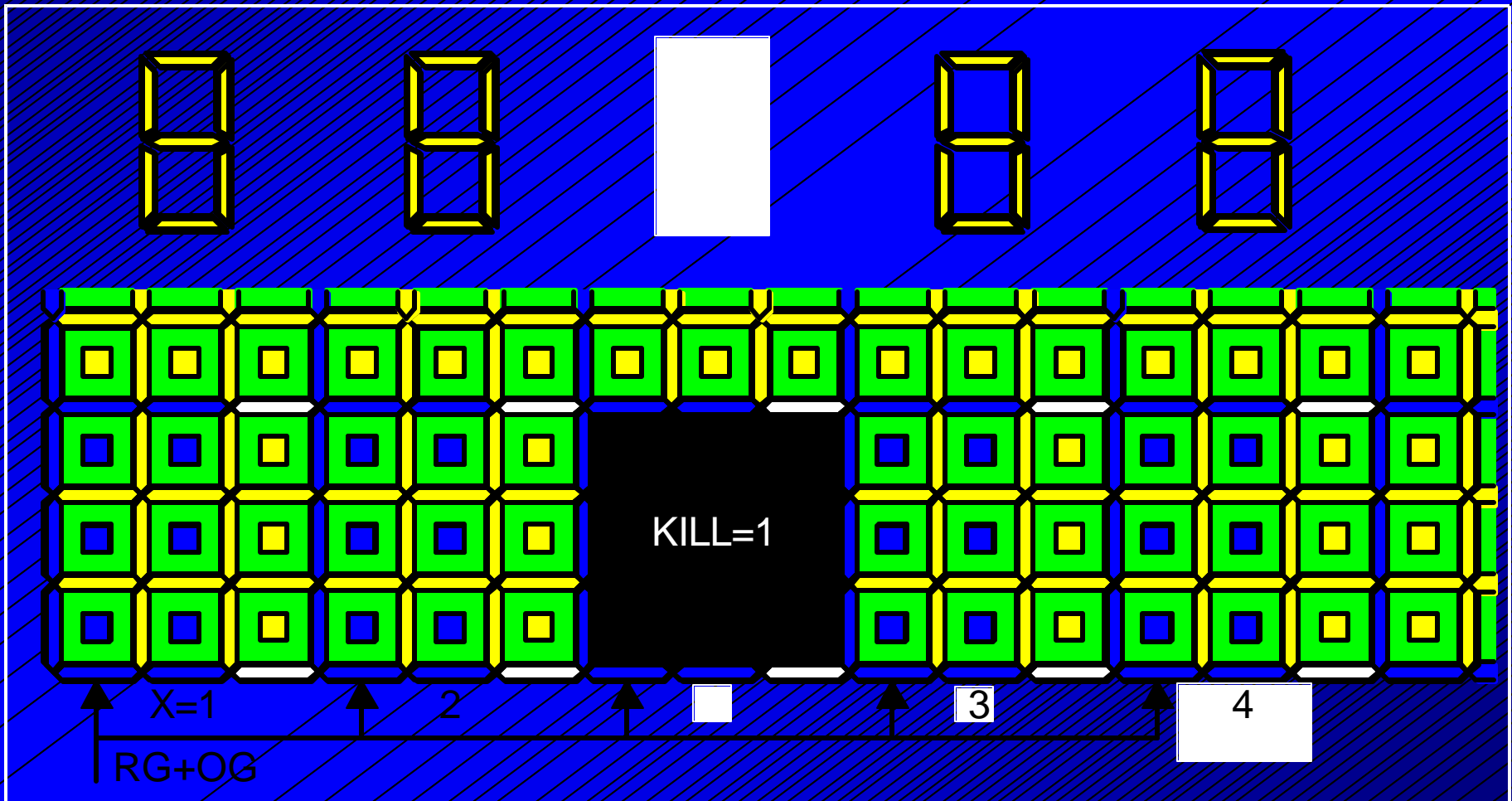


- Embryonics project at EPFL, Switzerland
- Cellular design with redundancy
  - Each cell programmed by a string (gene)
  - FPGA technology
- Self-healing property:
  - Upon cell failure, neighbors reconfigure to take over function

# Cellular self-repair



# Cellular self-repair



# Autonomic computing

- Broad R&D project launched by IBM
- Self-healing
  - Design computer and software that perform self-diagnostic functions and can fix themselves without human intervention
  - Strong analogies to biological systems
- **Reduced cost of design and maintenance**

# Autonomics principles

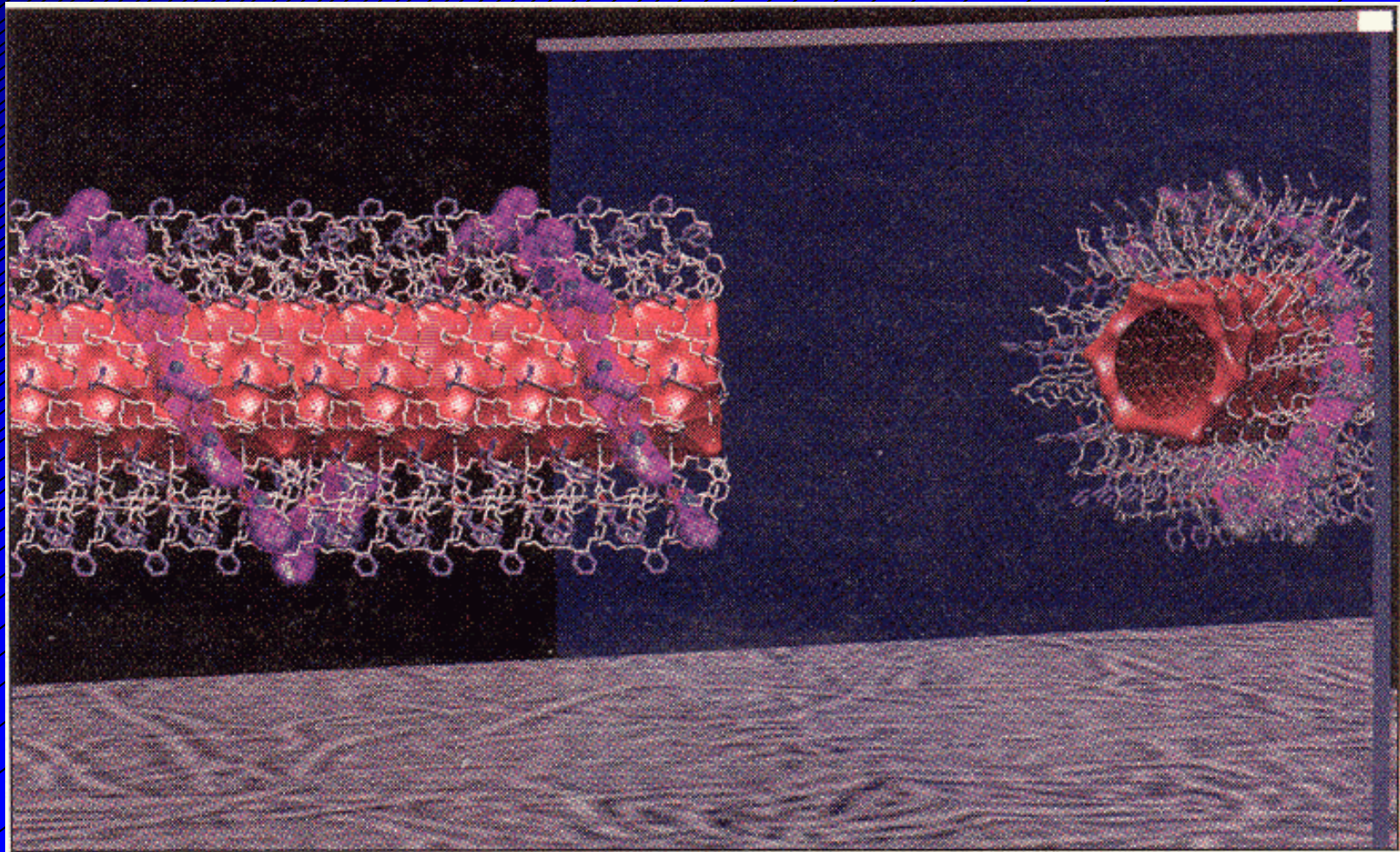
- An autonomic system:
  - must know itself
  - reconfigures itself under varying condition
  - optimizes its operations at run time
  - must support self-healing
  - must defend itself against attacks
  - must know the environment
  - manages and optimizes internal resources without human intervention

# Evolving computing materials

- When will current semiconductor technologies run out of steam?
- What factor will provide a radical change in technology?
  - Performance, power density, cost?
- Several emerging technologies:
  - Carbon nanotubes, nanowires, quantum devices, molecular electronics, biological computing, ...
- Are these technologies compatible with silicon?
  - What is the transition path?
- What are the common characteristics, from a design technology standpoint?



# Rosette nanotubes



# Common characteristics of nano-devices

- Self-assembly used to create structures
  - Manufacturing paradigm is bottom-up
- Significant presence of physical defects
  - Design style must be massively fault-tolerant
- Competitive advantage stems from extreme high density of computing elements
  - $10^{11}$ - $10^{12}$  dev/cm<sup>2</sup> vs.  $3 \times 10^9$  dev/cm<sup>2</sup> for CMOS in 2016
- Some nano-array technologies are compatible with silicon technology and can be embedded in CMOS

# Robust nano-design

- Key ingredients:
  - Massive parallelism and redundancy
  - Exploit properties of crosspoint architectures
    - E.g., Programmable Logic Arrays (PLAs)
  - Local and global reconfiguration
- Some design technologies for robust DSM CMOS design can be applied to nanotechnology

# Summary

## problem analysis

- The electronic market is driven by embedded applications where **reliability** and **robustness** are key figures of merit
- System design has to cope with **uncertainty**
  - Lack of knowledge of details, due to abstraction
  - Physical properties of the material
- As design size scales up, the design challenge is related to **interconnecting** high-level components
- As technology scales down, and as nanotechnologies are introduced, electrical-level information becomes **unreliable**

# Summary

## design strategies

- Robust and reliable design is achieved by:
  - **Self-calibrating** system components
  - **Networking** components on chip with adaptive interfaces
    - Encoding, packet switching and routing provide a new view of logic and interconnect design
  - **Self-healing** components that can diagnose failures and reconfigure themselves
- New emerging technologies will require massive use of error correction and redundancy

**Thank YOU**