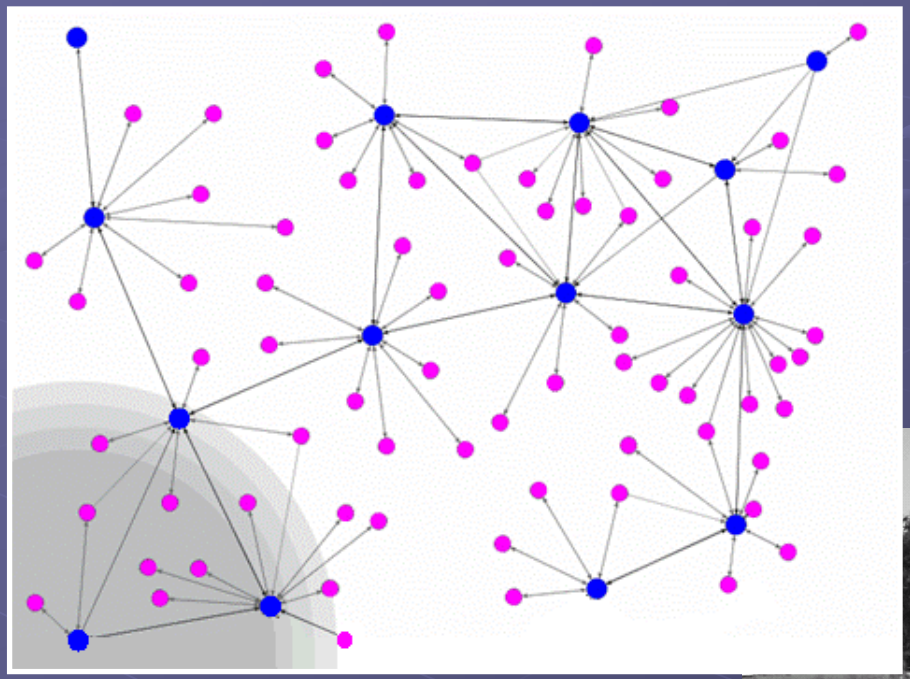
An Ultra-low energy asynchronous processor for Wireless Sensor Networks

L.Necchi, L.Lavagno, D.Pandini, L.Vanzago

Politecnico di Torino ST Microelectronics

Wireless Sensor Networks



- Ad-hoc wireless networks
- Sensing
- Computation
- Actuation

Application areas:

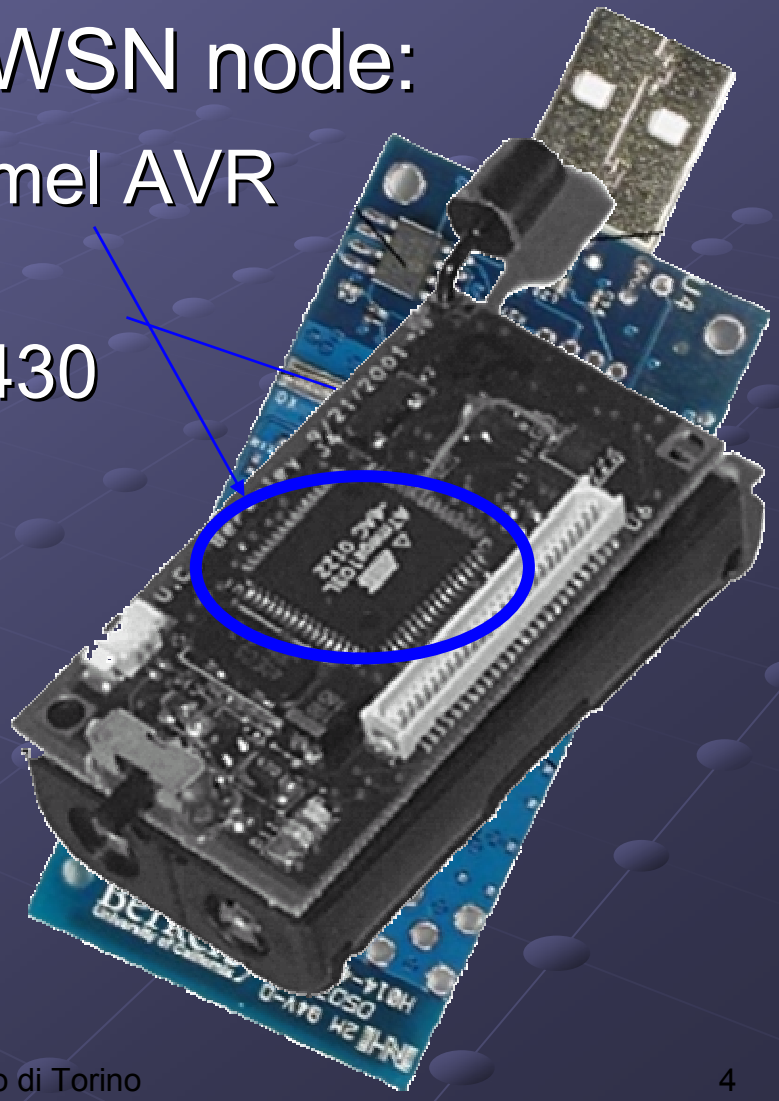
- Monitoring
- Building automation
- Health care, Medical
- Emergency response
- Automotive

Key WSN Requirements

- Flexibility (general purpose design)
- High energy efficiency (battery powered)
- Extremely wide voltage supply range
  - Exhausted battery or energy scavenging
- Fast and inexpensive wake-up
  - event driven power management (not predictable)
- Sporadic high computational load
  - Encryption (security)
  - Aggregation, distributed data processing

Sensor node architecture

- Main components of a WSN node:
  - Microcontroller
  - Memory
  - Radio
  - Sensors / Actuators
  - Power supply
    - Battery (energy storage)
    - Power scavenging



Circuit-level Power Management

Management	Save energy while		Scenario
	Idle	Active	
Clock Gating	X		Idle Time
Power Gating	X		
Dynamic Voltage Scaling		X	Long Deadlines
Adaptive Body Biasing		X	

- DVS can be obtained by:
  - Off-line pre-computed voltage/frequency tables
    - **High delay margins**
  - Evaluated on-line:
    - PowerWise, Razor, Asynchronous, De-synchronization

Closed-loop DVS technique

- PowerWise:
  - Samples, with a high frequency clock, the output of a digital delay line, and arrange voltage supply to deliver required performance
- Razor:
  - Detects timing errors comparing values stored in duplicated slave latches, in which the second is clocked half clock cycle later, restarts the pipeline and arranges voltage supply accordingly
- Asynchronous with Dual-Rail encoding:
  - (Quasi) delay insensitive implementation, that guarantees correctness for (almost) every voltage supply and process variation
- Asynchronous with Bundled Data encoding:
  - A digital delay line output is directly used to generate a local clock signal, resulting in a direct dependence between voltage supply and delay period

De-synchronization

Desynchronize

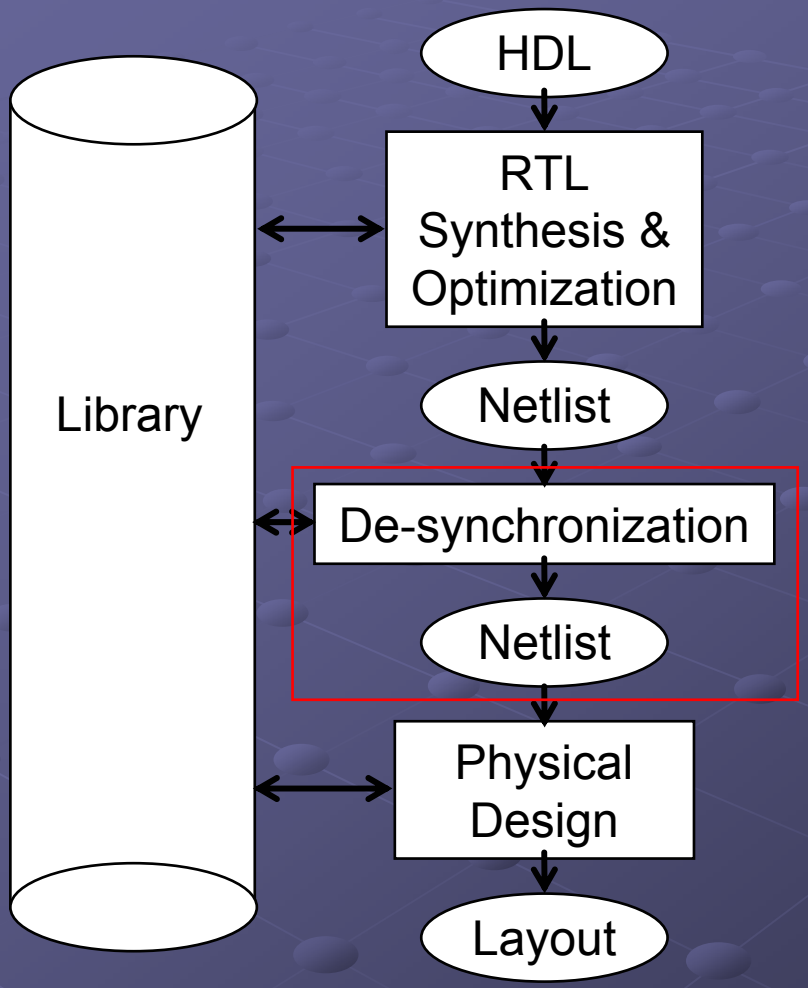
Synchronous

Asynchronous

CLK

~~CLK~~

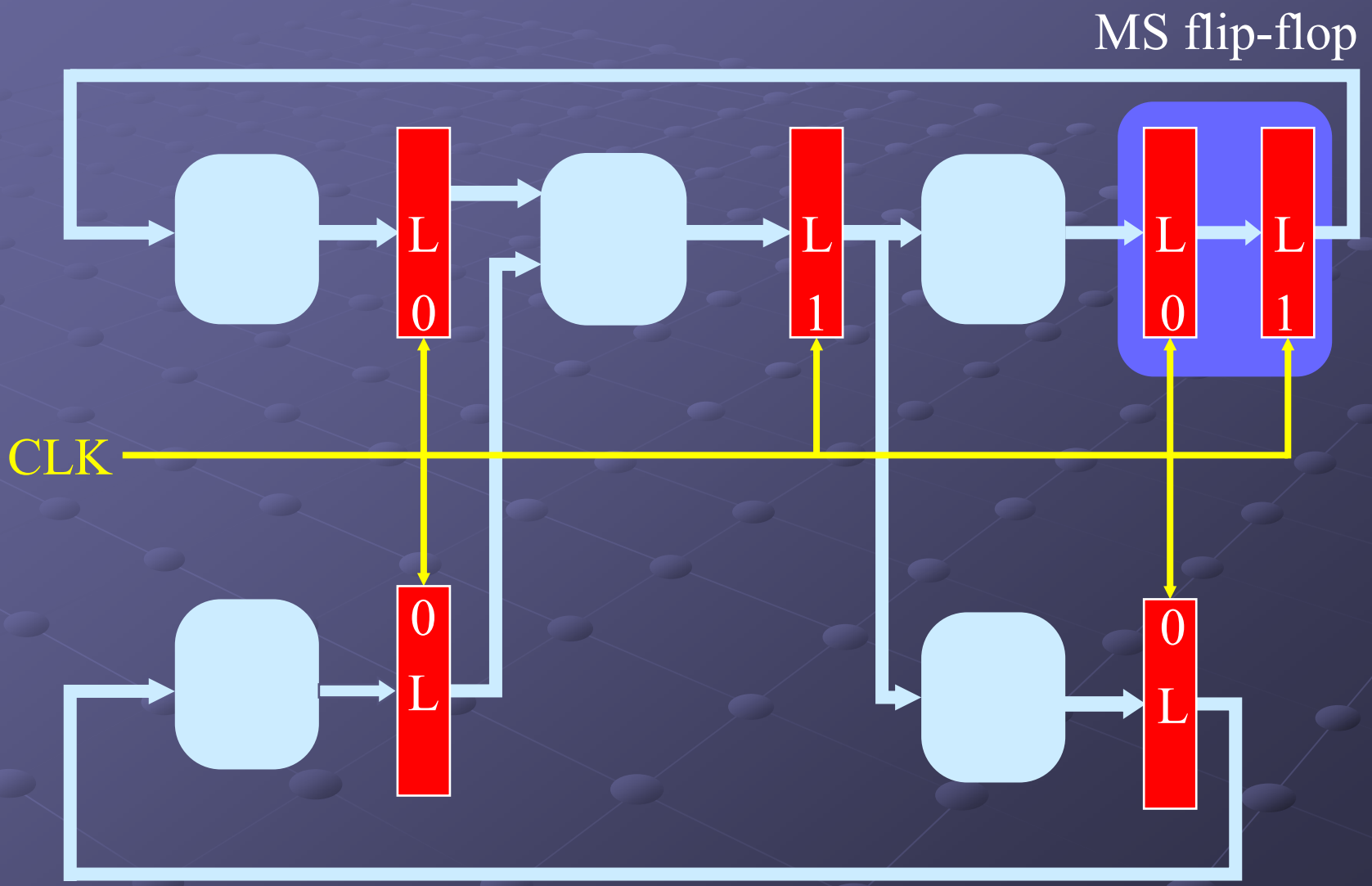
Design Flow



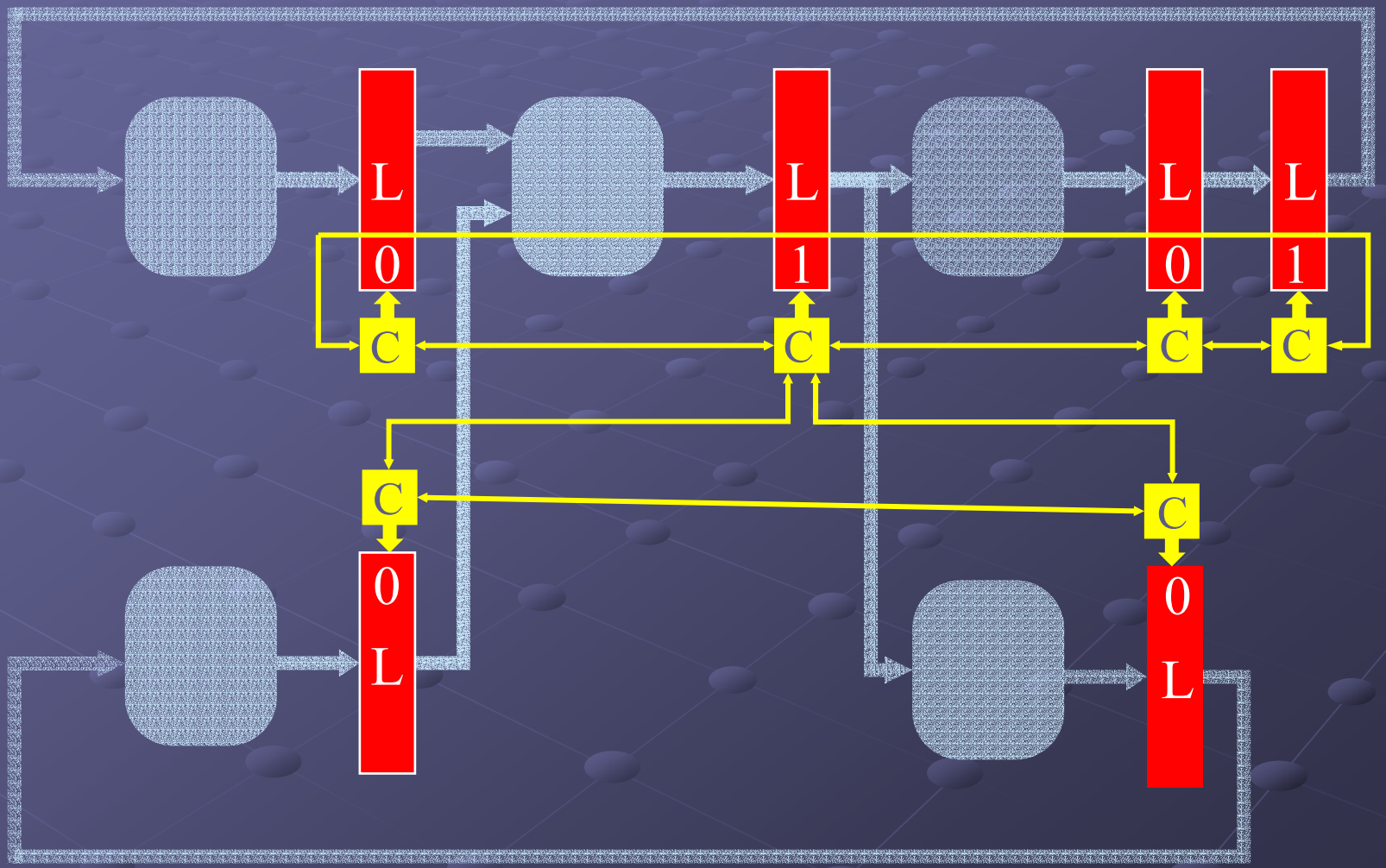
Obtain asynchronous implementation from synchronous specification:

- Think synchronously
- Design synchronously
- De-synchronize (automatically)
- Test synchronously
- Run asynchronously

Synchronous circuit

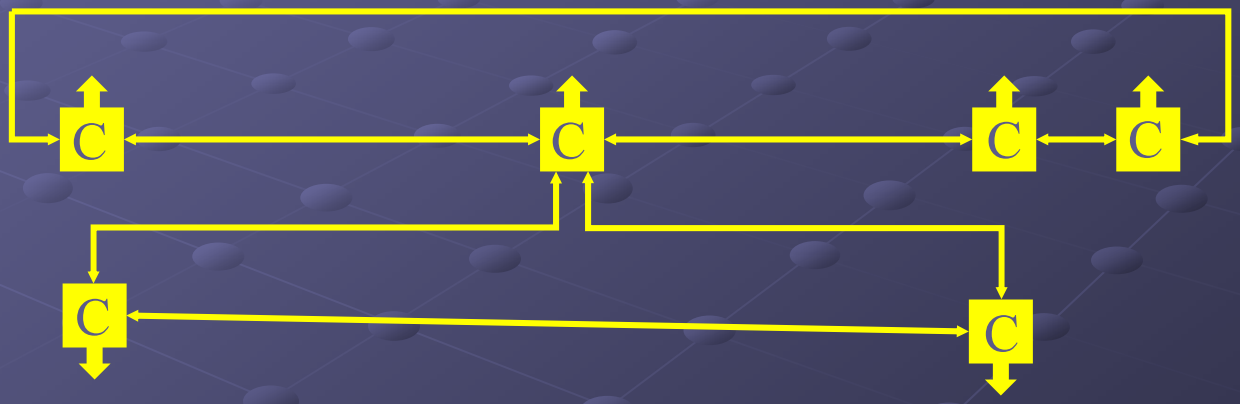


De-synchronization



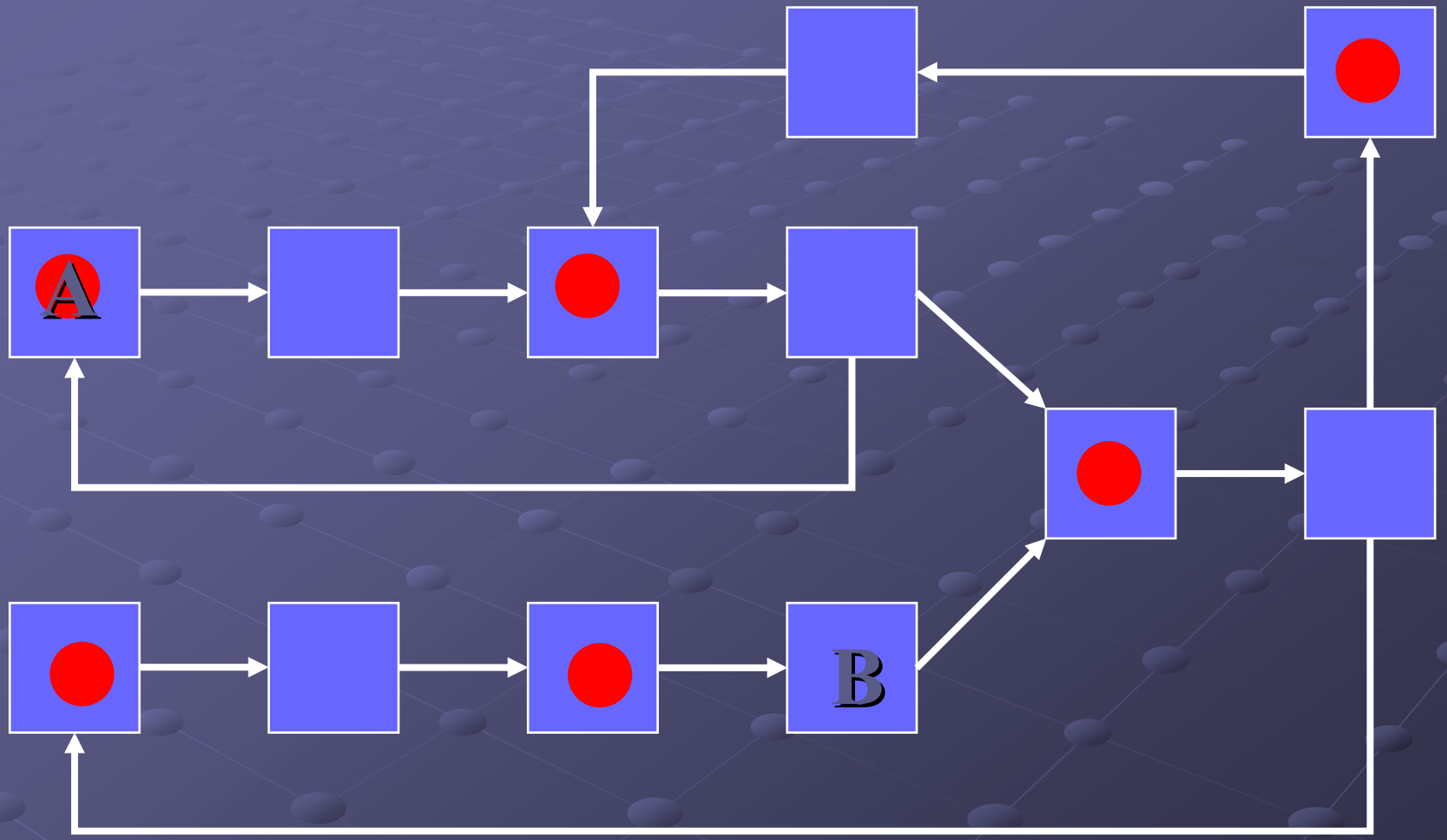
De-synchronization

Distributed micropipeline-style controllers substitute the clock network

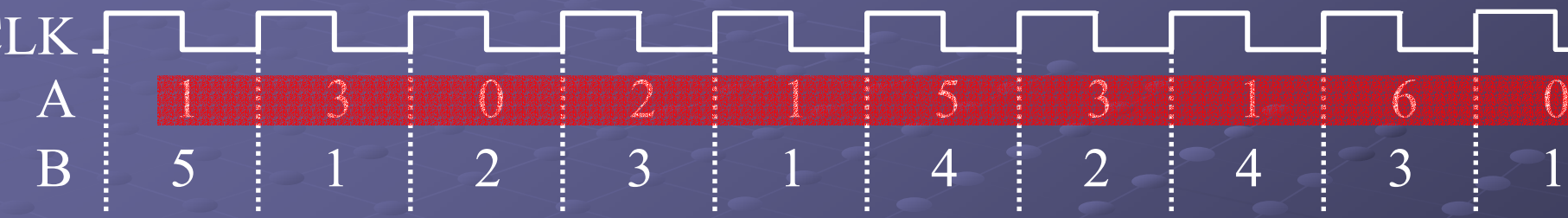


The data path remains intact !

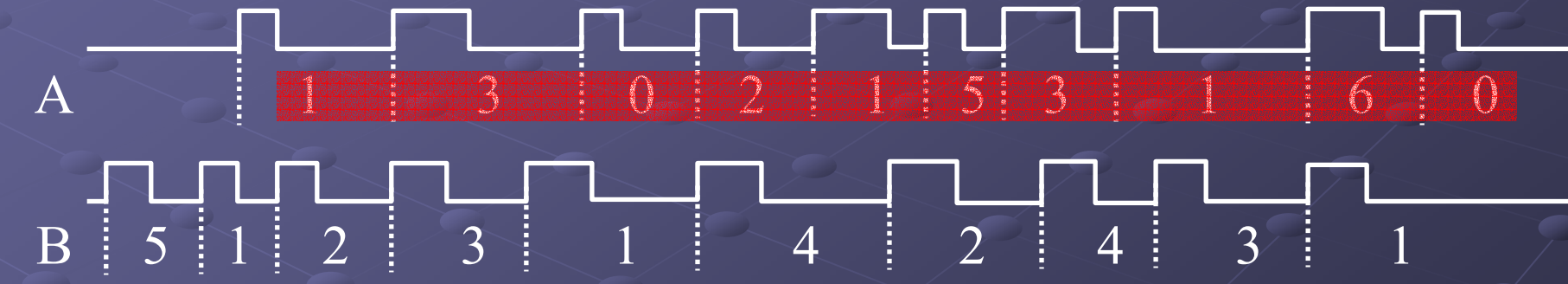
Flow equivalence [Guernic, Talpin, Lann, 2003]



Flow equivalence [Guernic, Talpin, Lann, 2003]

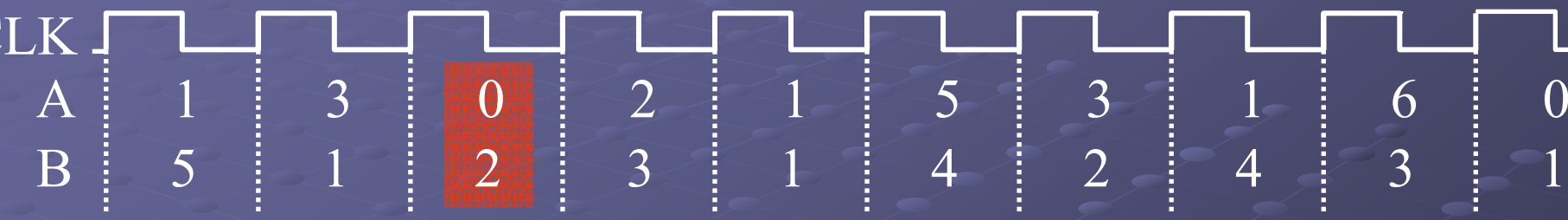


Synchronous behavior

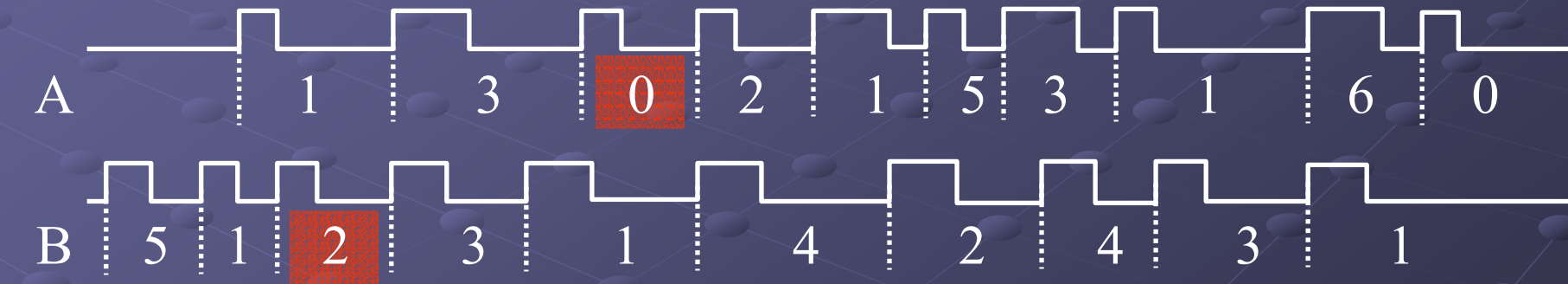


De-synchronized behavior

Flow equivalence [Guernic, Talpin, Lann, 2003]



Synchronous behavior

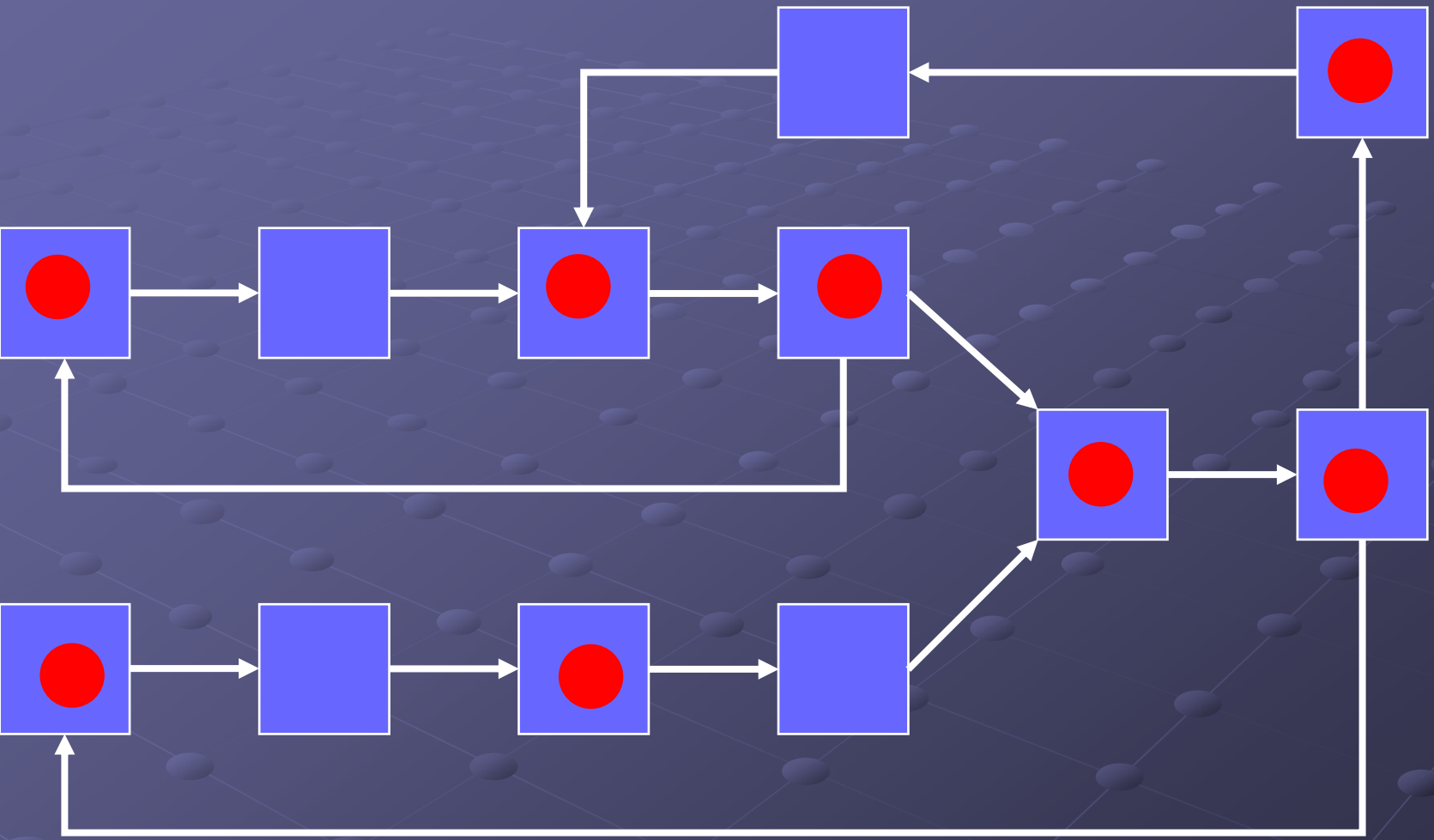


De-synchronized behavior

Theorem:

The de-synchronization model preserves flow-equivalence

Flow equivalence [Guernic, Talpin, Lann, 2003]



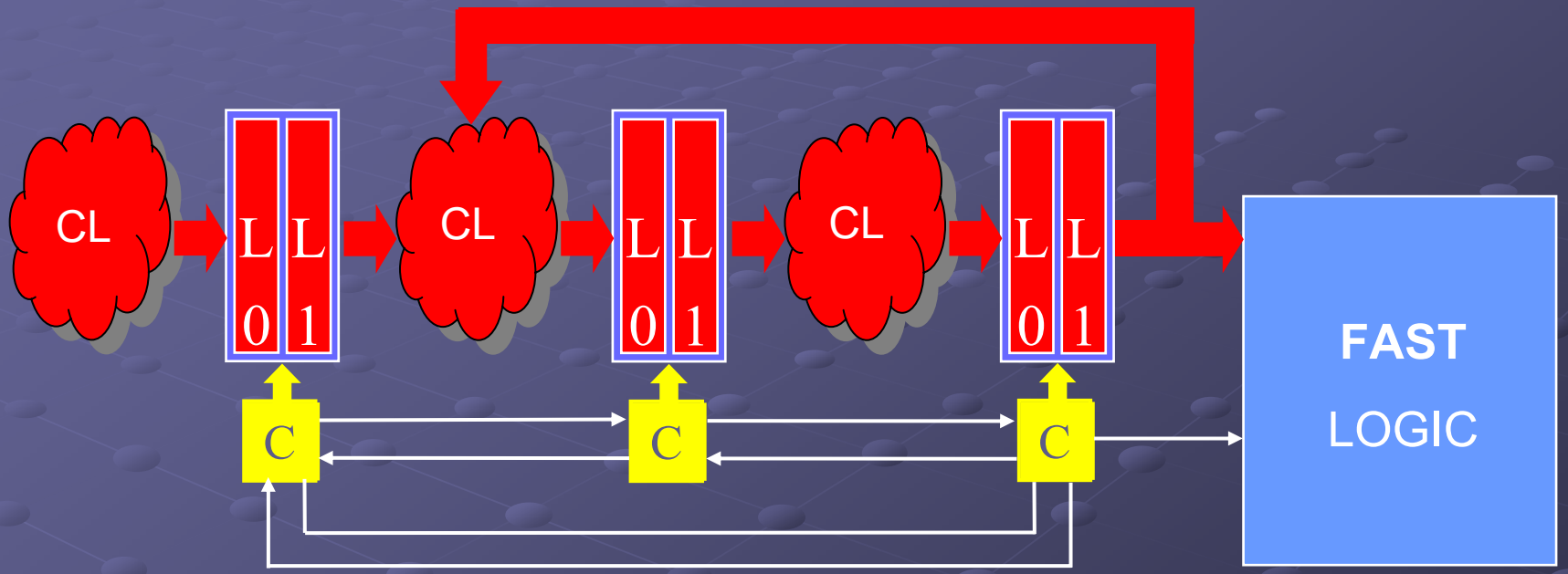
De-synchronization Benefits

- For the end user:
  - Reduced electromagnetic emission
  - Process Variation tolerance
  - Enables **partial** average case design,
    - wrt process & environment variation (not wrt data-dependent delay)
  - The resulting circuit will be:
    - Ready for frequency and voltage scaling
    - Inherently more robust to delay variations
    - Virtually no performance or area overhead wrt synchronous
- For the designer
  - Conventional EDA Tools and design flow
    - Limited design time and effort, fully automated
    - Re-use legacy designs

Asynchronous advantages *not* offered by de-synchronization

- Fine-grained power management
  - The desynchronized circuit inherits the synchronous clock gating
- Fine-grained pipelining
  - The pipeline structure is not changed
- Data-dependent delays
  - Could be exploited by using a datapath with completion detection (work in progress)
- Robustness with respect to uncorrelated local variability
  - Would require completion detection

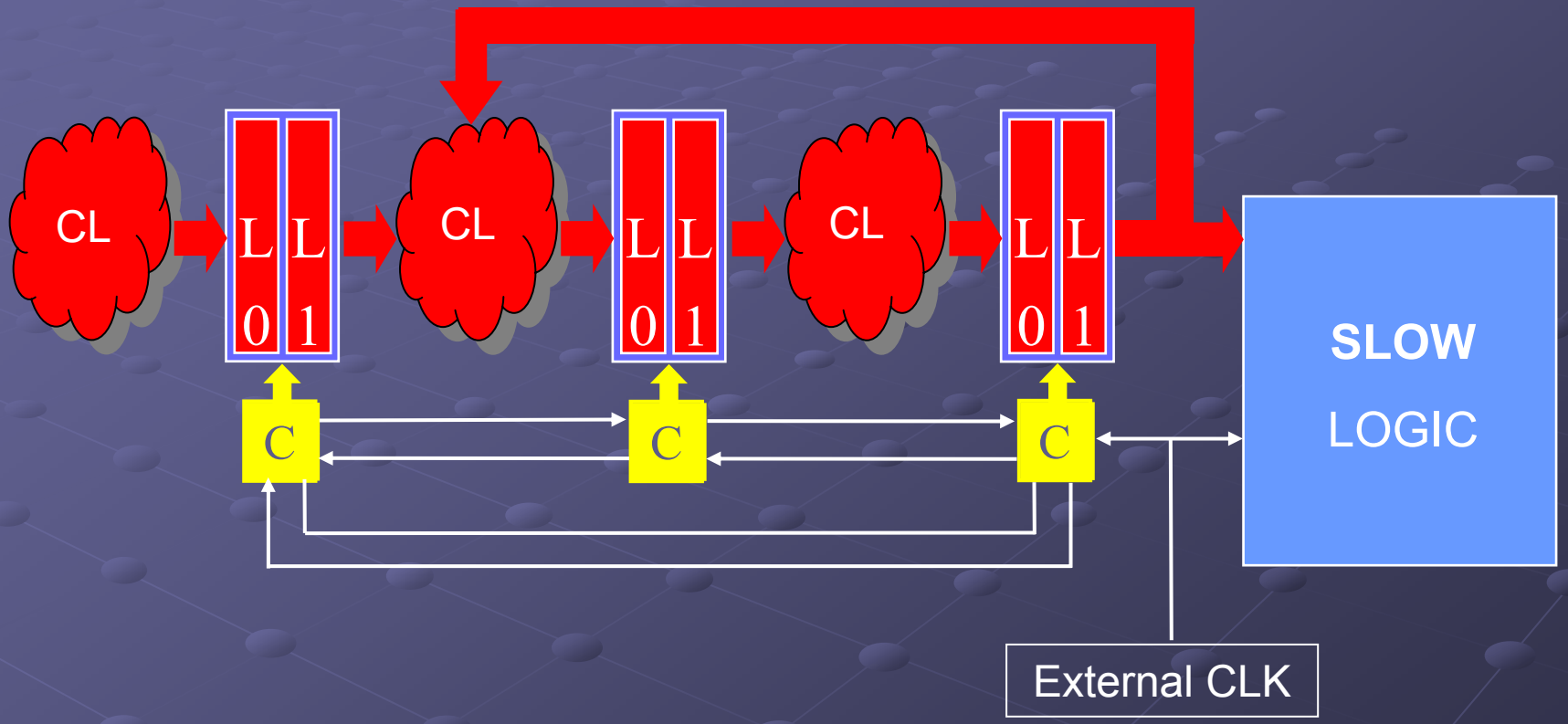
Synchronous Logic Interfacing



Data path (not modified)

Handshaking line

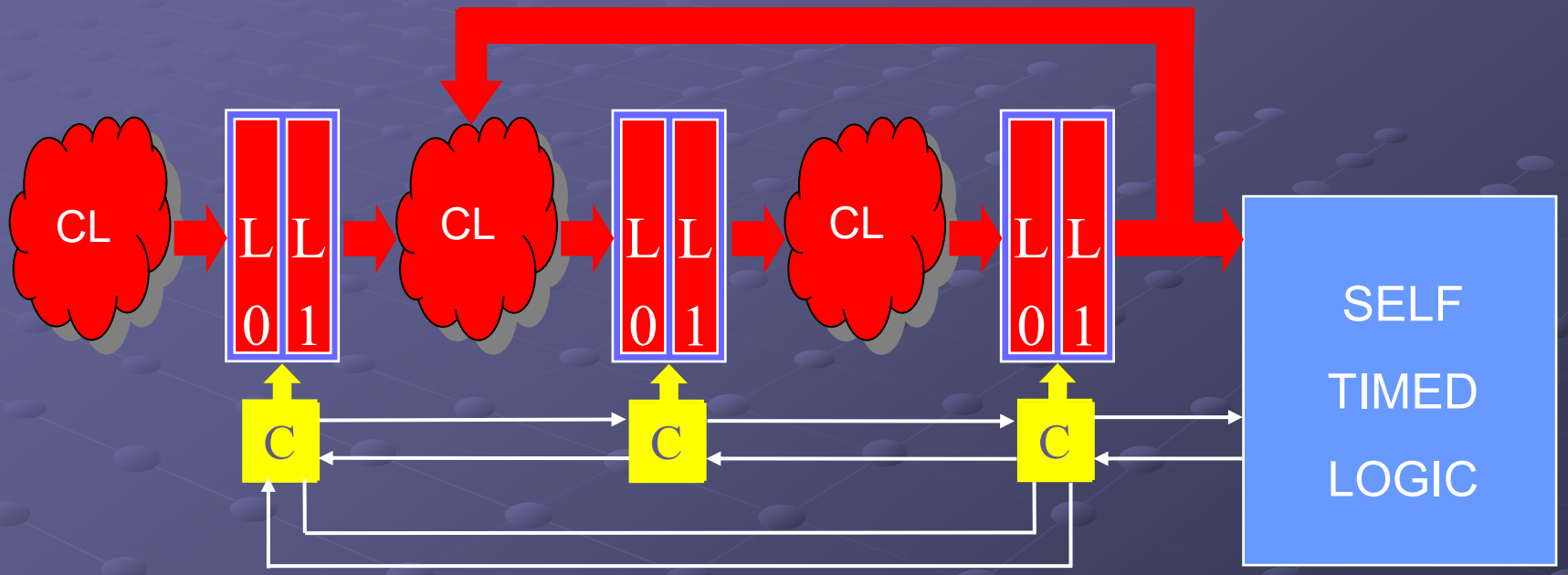
Synchronous Logic Interfacing



- Synchronized with an external slower clock

  -Just low EMI

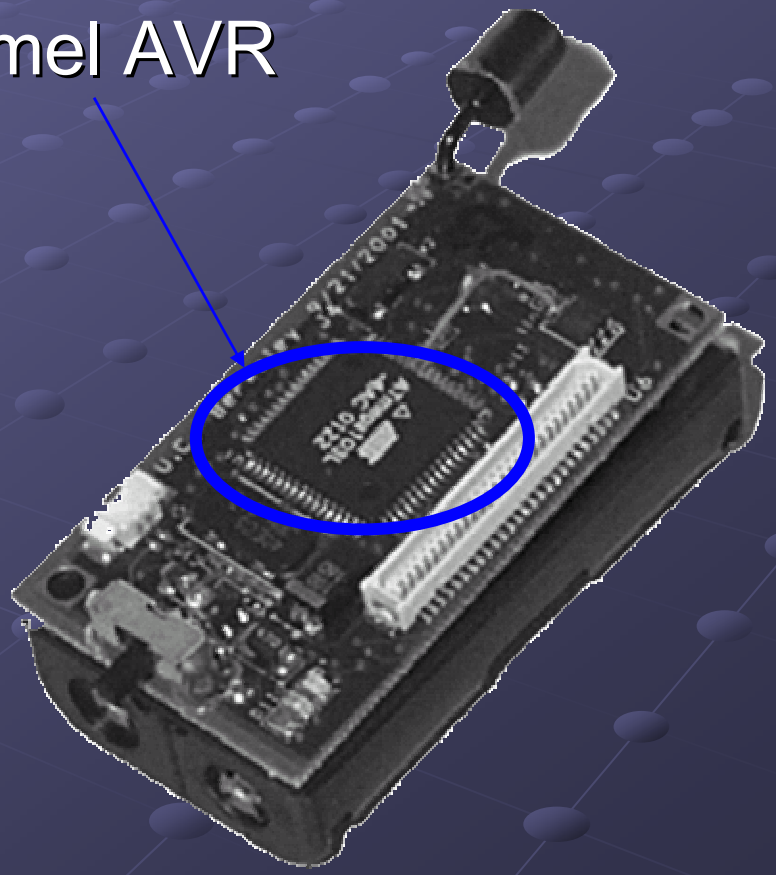
Synchronous Logic Interfacing



- Example: SRAM with Completion Detection

Sensor node architecture

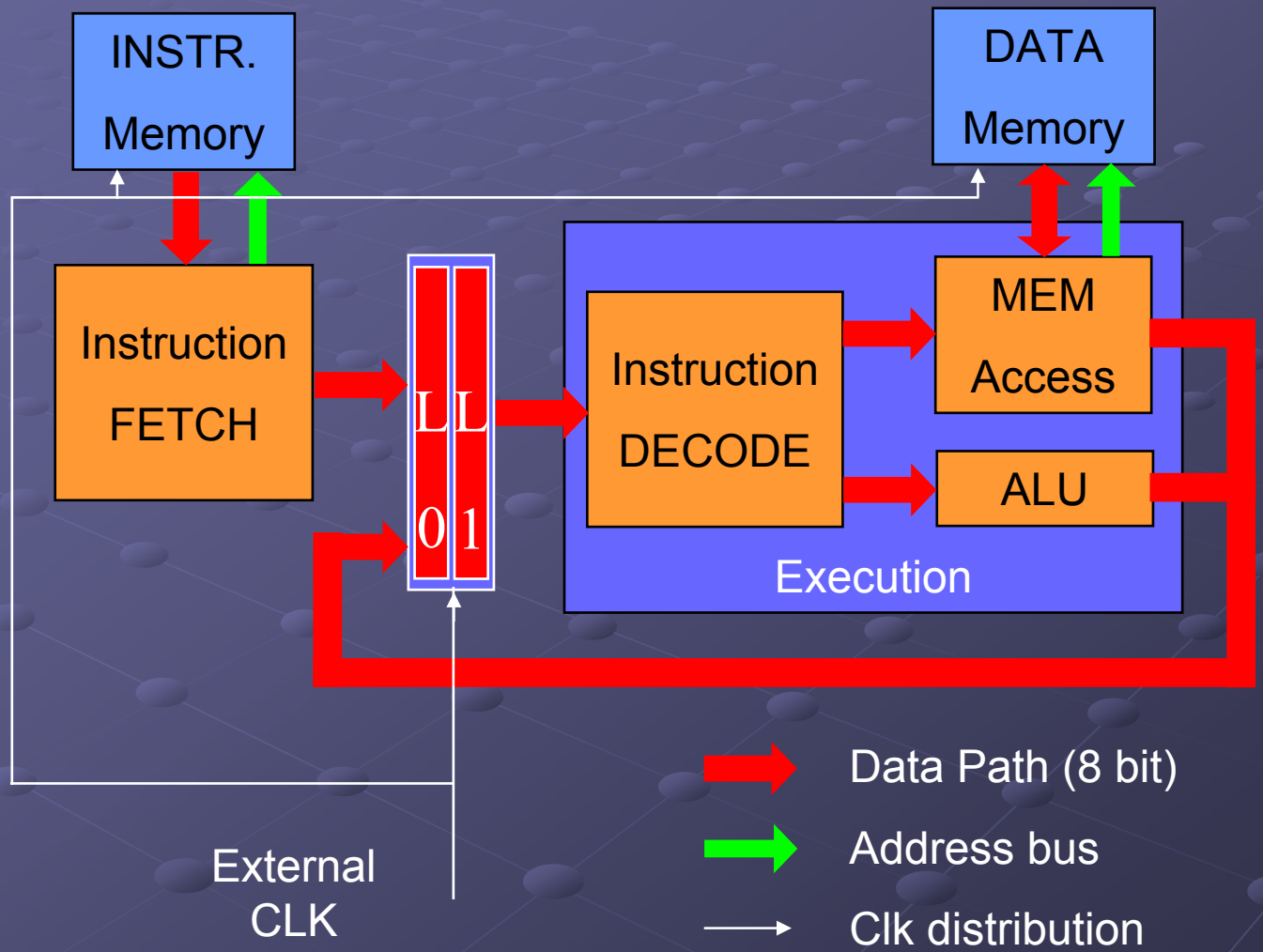
- Main components of a WSN node:
  - **Microcontroller**
  - Memory
  - Radio
  - Sensors / Actuators
  - Power supply
    - Battery (energy storage)
    - Power scavenging



Our Case Study

- Application independent 8 Bit CPU architecture:
  - Atmel AVR Instruction Set (like MICA2 - MICAZ) from OpenCores.org, implemented with a 130nm technology
  - Toolchain and lots of software are ready to use
    - nesC, TinyOS, TinyDB, Surge, Tossim
- Aggressive Energy management enabled by de-synchronization, using:
  - Dynamic Voltage Scaling
  - zero wake-up time (No CLK, no wait for PLL to restart)

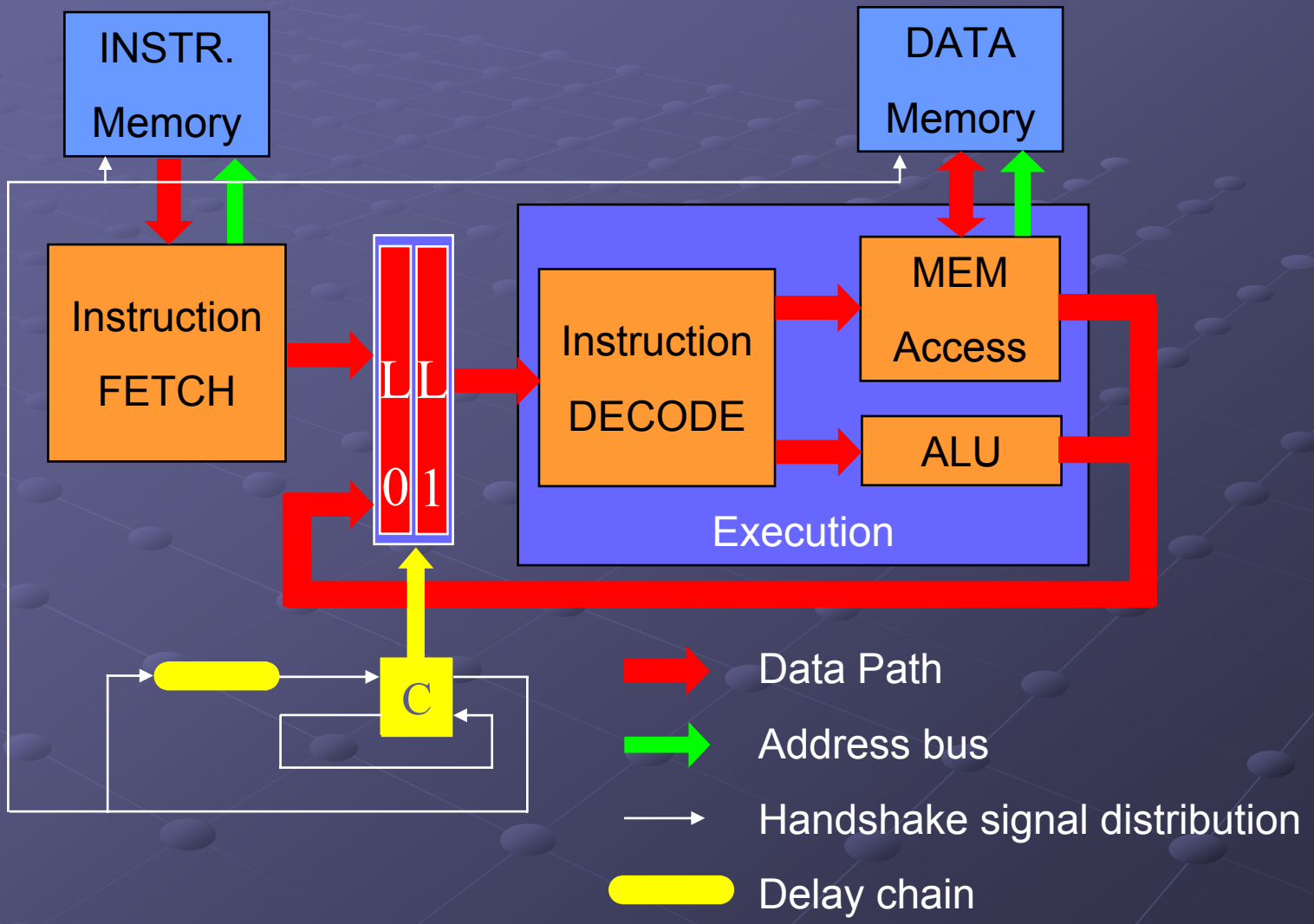
Typical AVR architecture



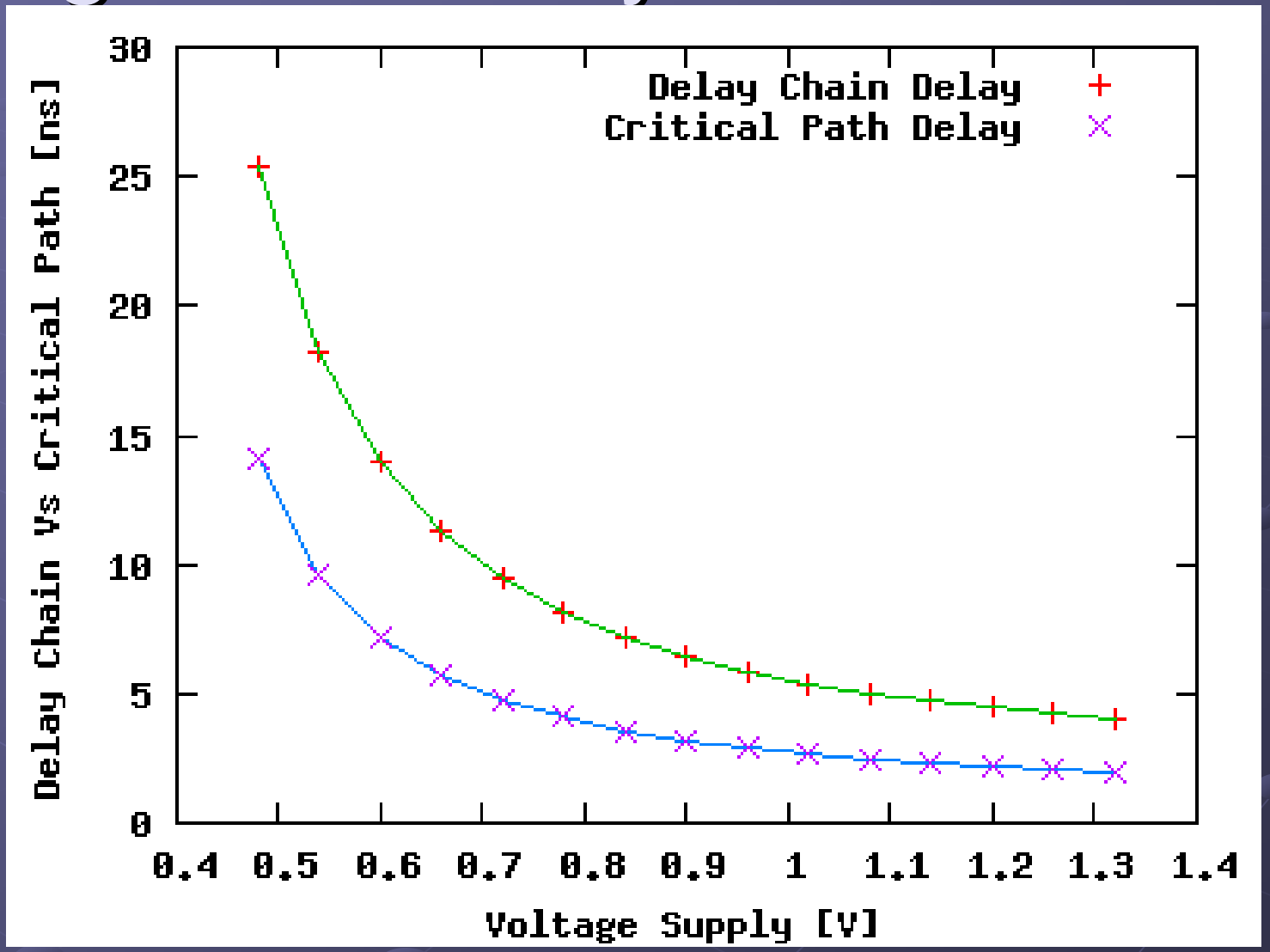
Design Choices

- Main target is energy efficiency (vs speed)
  - Large delay margins (100%) to increase robustness at low voltage supply
- AVR core is really small (~4500 gates), hence we used a **Single** controller
  - Reduced area overhead
  - No electro magnetic emission reduction

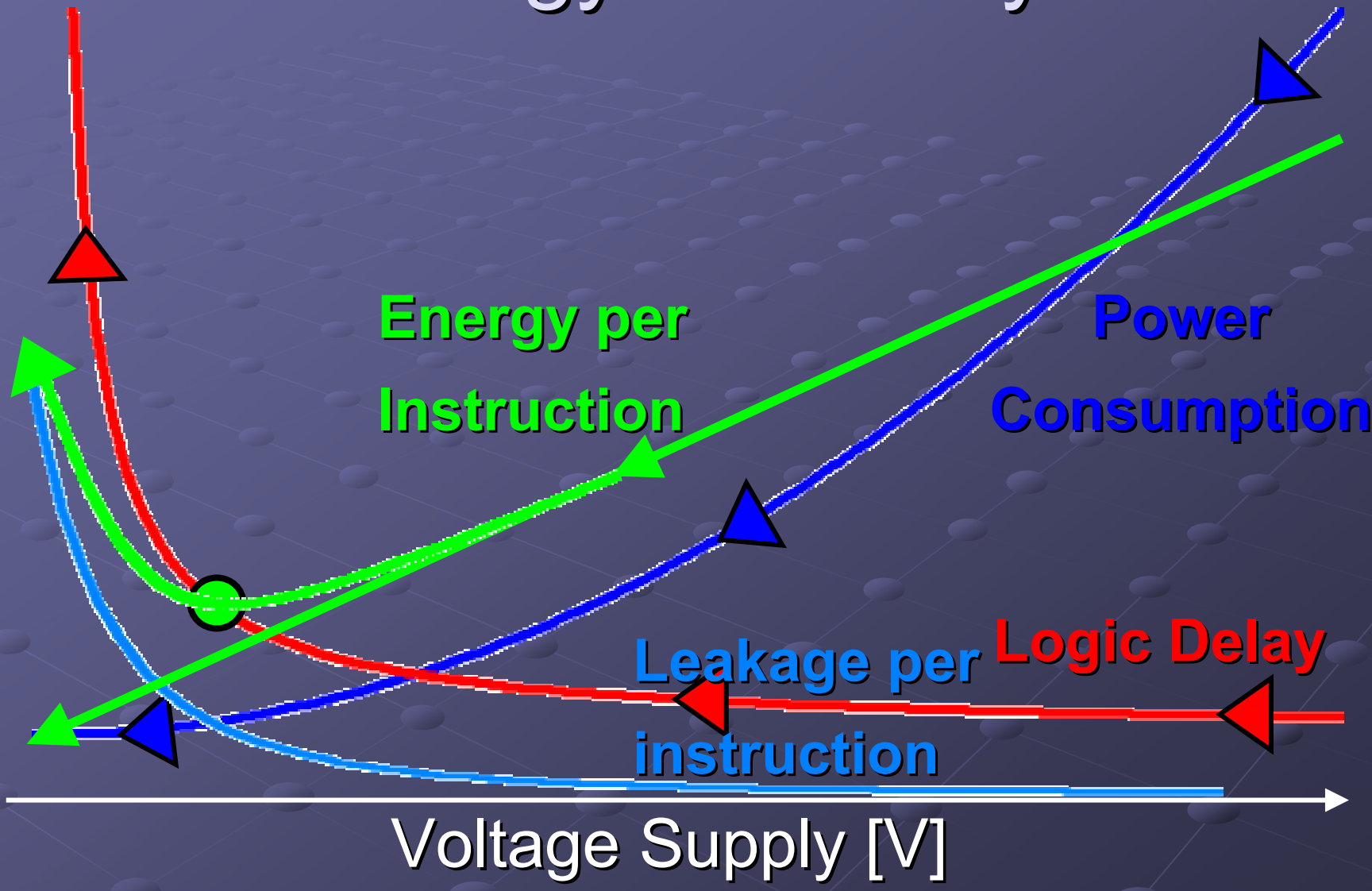
De-synchronized AVR



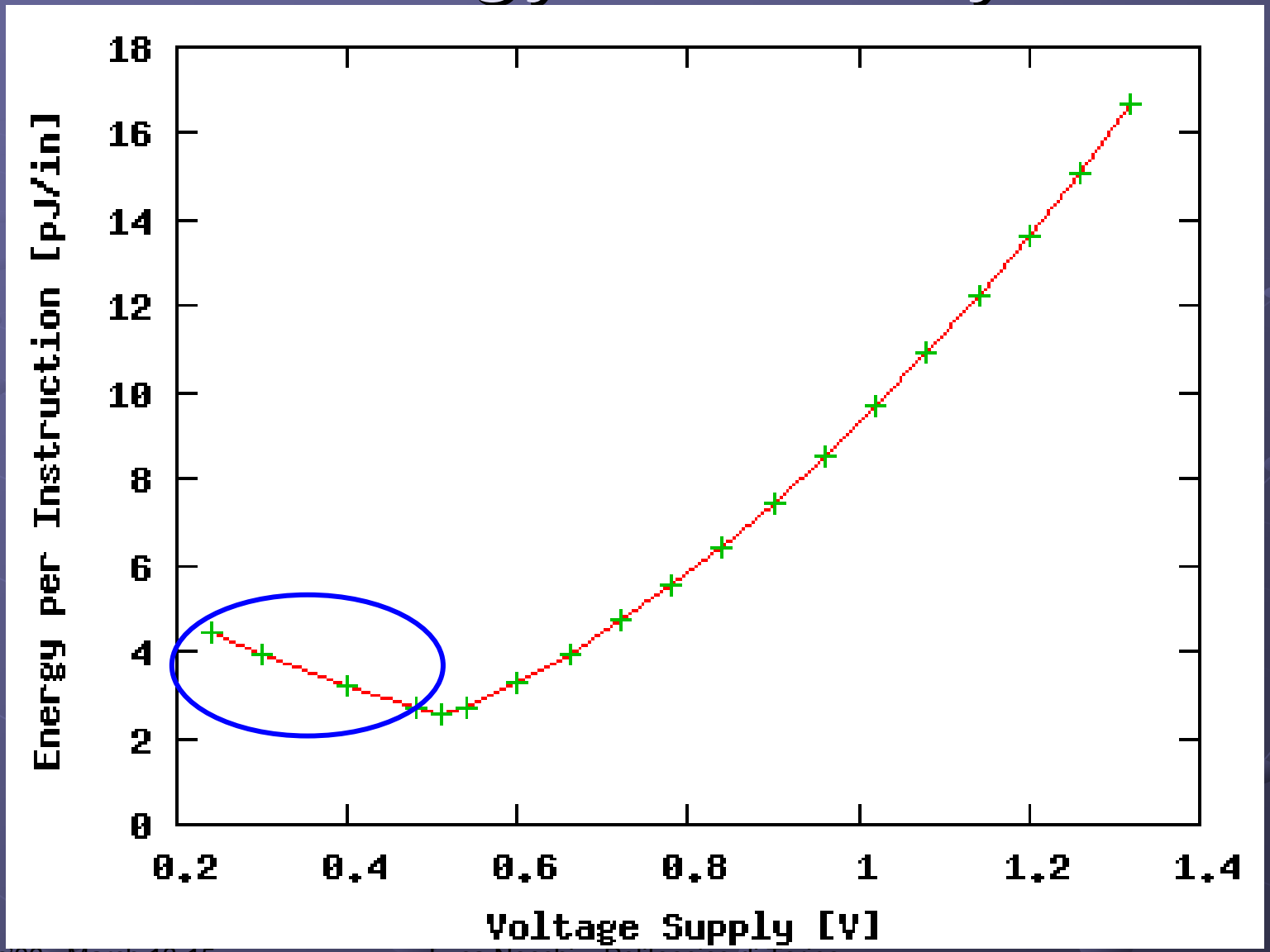
Logic and Delay Line Matching



Energy Efficiency



Energy Efficiency



Some Past Work Comparison

- Philips 80c51 (H. van Gageldonk., 1998)
  - Asynchronous bundled-data implementation of the 8051 ISA, general purpose.
- Lutonium (A. Martin et al., 2003)
  - Asynchronous QDI implementation of the 8051 ISA.
- Snap/le (V. Ekanayake et al., 2004)
  - Asynchronous QDI processor specifically designed for WSN.
- Razor (D. Ernst et al., 2004)
  - Synchronous processor that estimated the best Vdd by dynamically monitoring the delay of the logic using a redundant latching schema.

CONCLUSIONS

- Aggressive Energy management using DVS
  - 14 pJ/Instr @ 1.2 V (170 MIPS)
  - 2.7 pJ/Instr @ 0.51 V (48 MIPS)
- Minimal overhead wrt synchronous counterpart
  - +6% area (due to FF->latch conversion)
  - -20% speed (could be improved by reducing margins)
- Future work:
  - Analysis with other “SPICE-like” simulators (Hsim)
  - Statistical simulations to check robustness wrt process variability (Monte Carlo)
  - Fabrication (?)