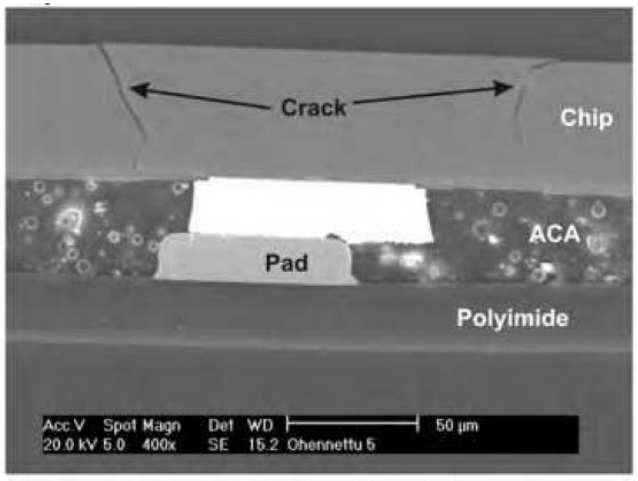
Beyond classical chip design

+



=

Beyond classical chip design

Matthias Függer, Attila Kinali

Beyond classical chip design lecture 1

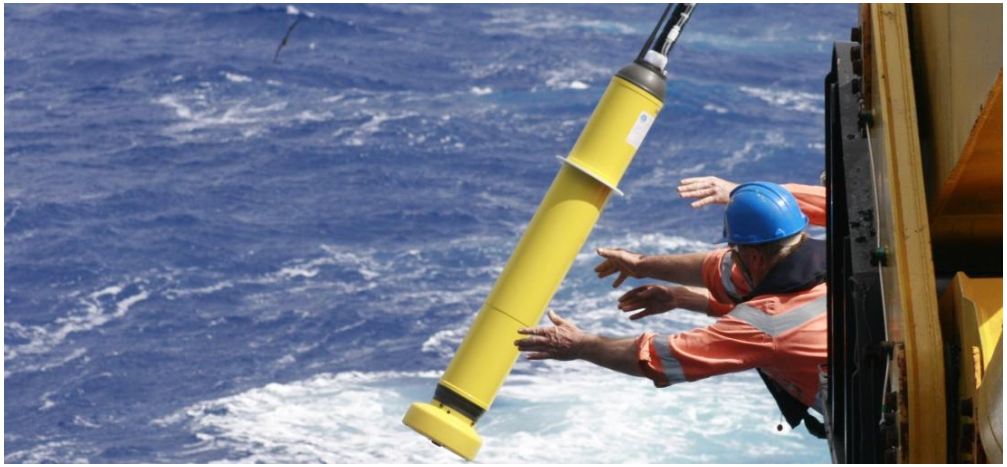
Motivation

Study & discuss methods that come into play when classical approaches fail.

Why beyond?

Missions that require

- high resilience to failures
- high performance
- low power consumption

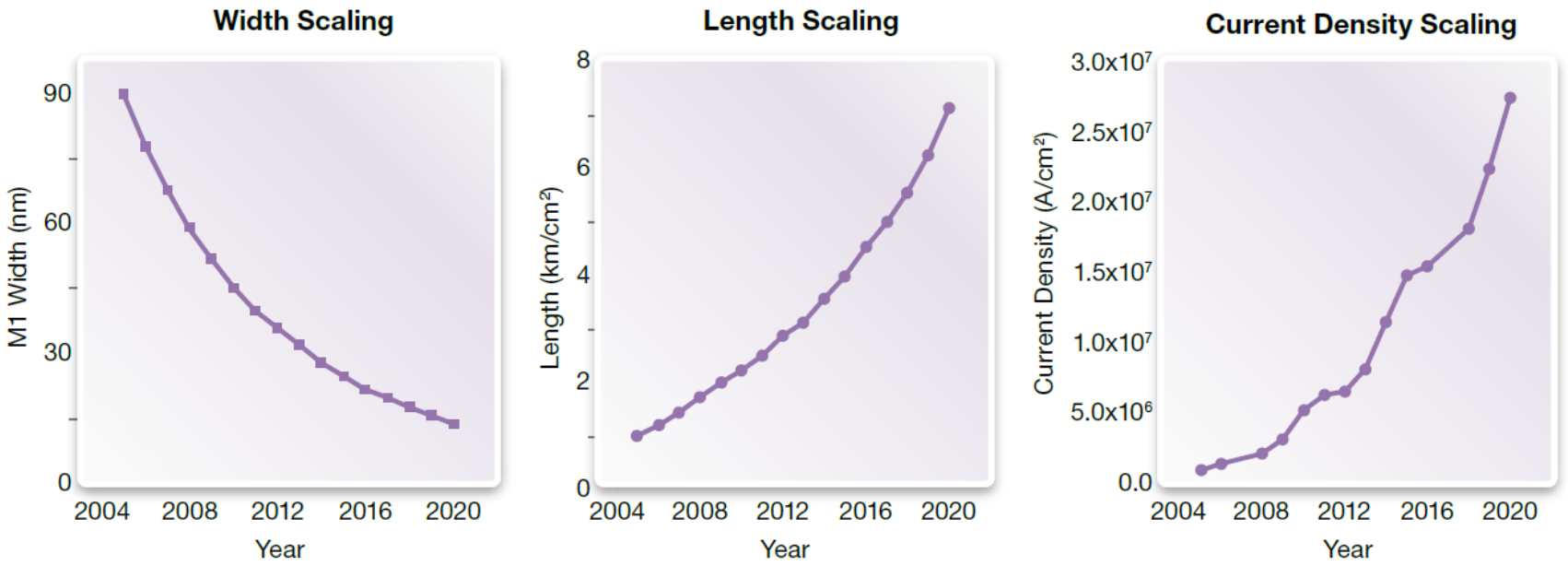


[Airbus & ESA & CSIRO]

Adversaries

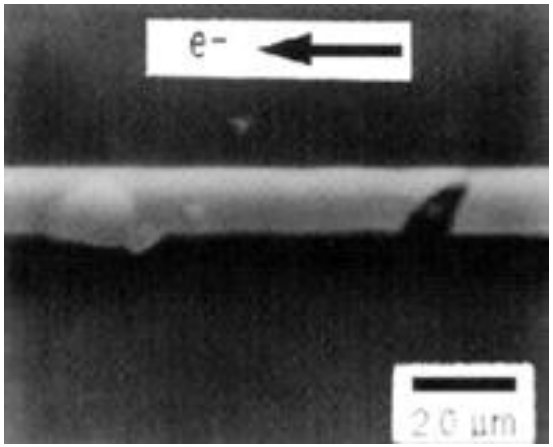
- during mission
- during production

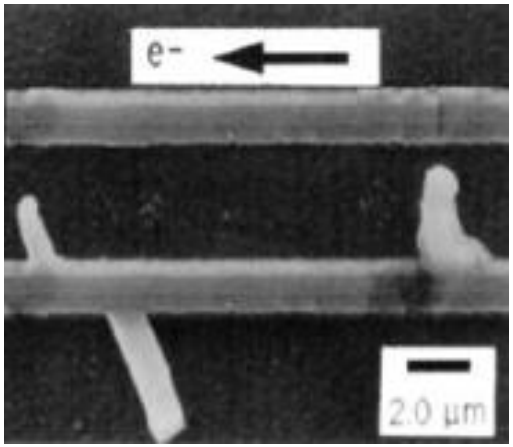
During mission: Scaling



[Synopsys, scaling & current trends]
from www.synopsys.com/Tools/Verification/CapsuleModule/CustomSim-RA-wp.pdf

... and its effects

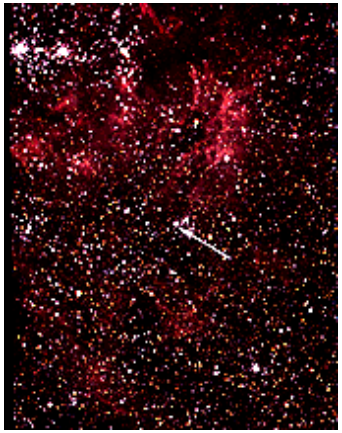




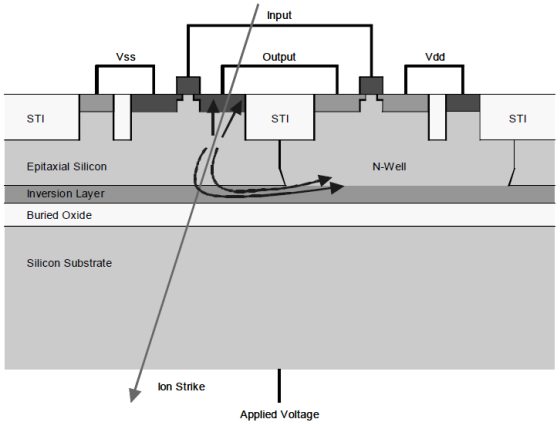
[Synopsys, electromigration]

During mission: radiation

[NASA, before]

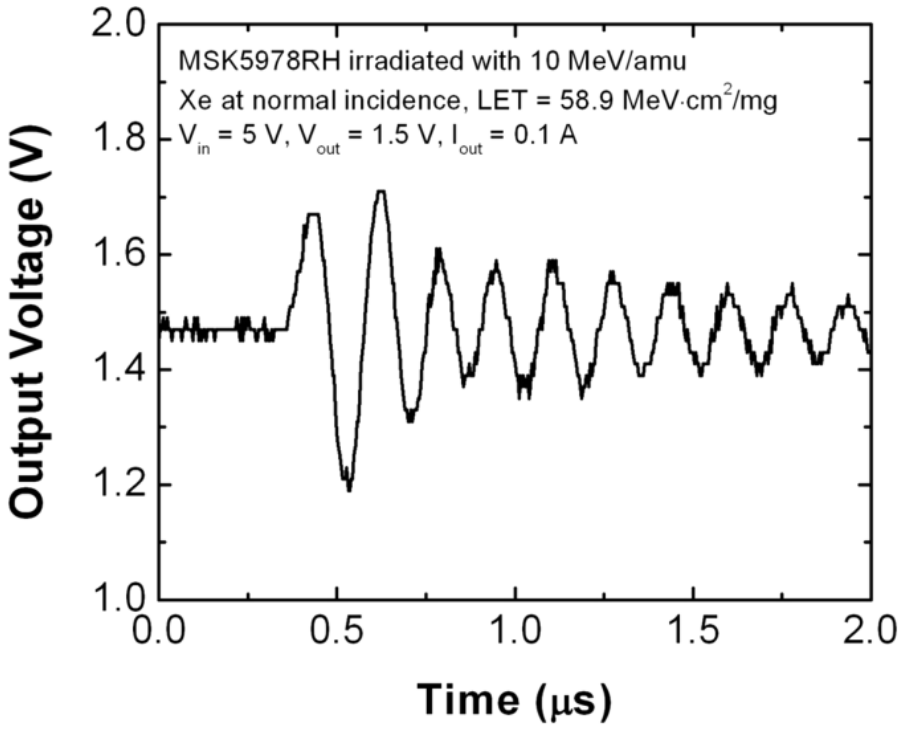


[NASA, after supernova]



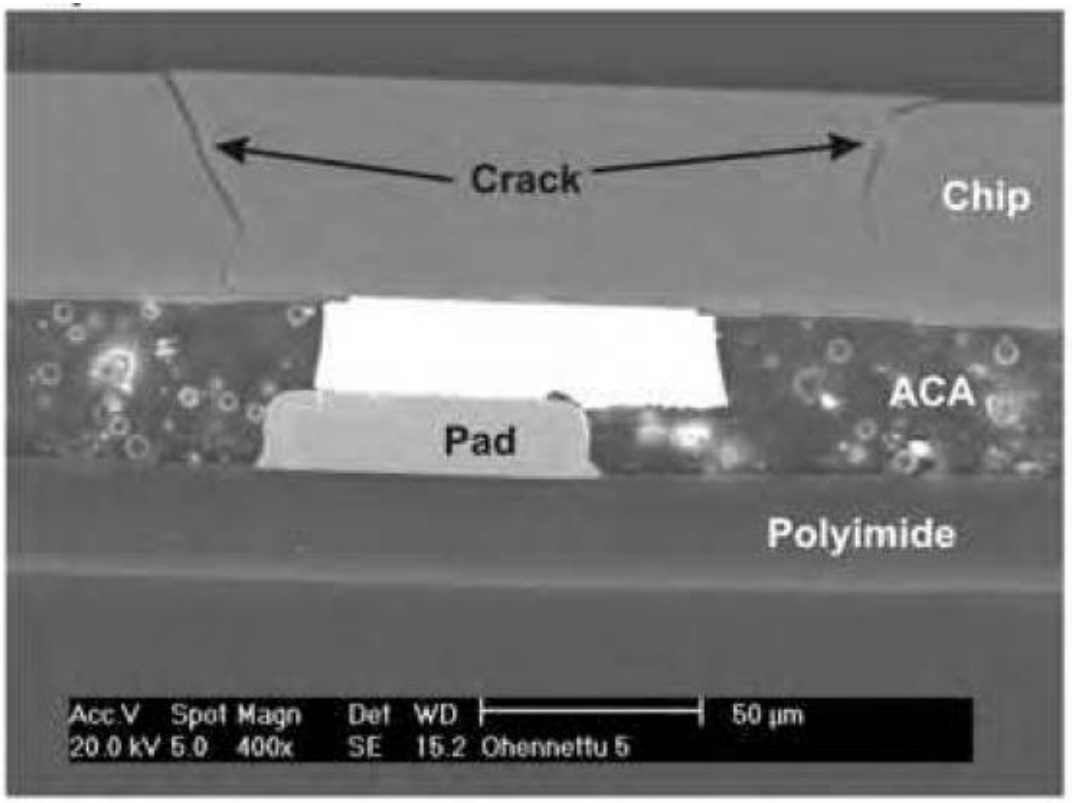
[Mavis, Eaton 2007]

… and its effects



Voltage regulator MSK5978RH
[O'Bryan et al, NSREC 2013]

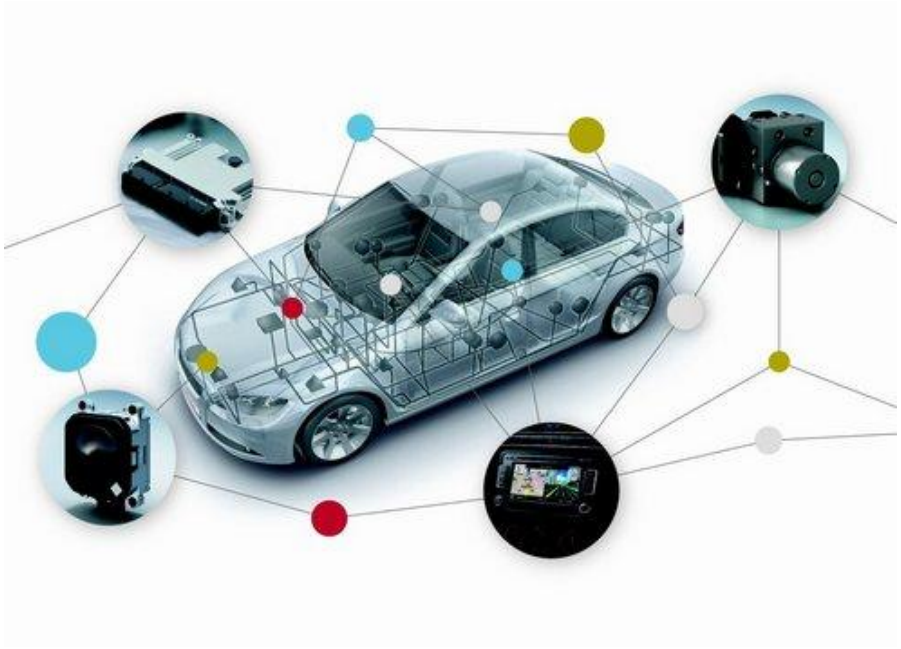
At production



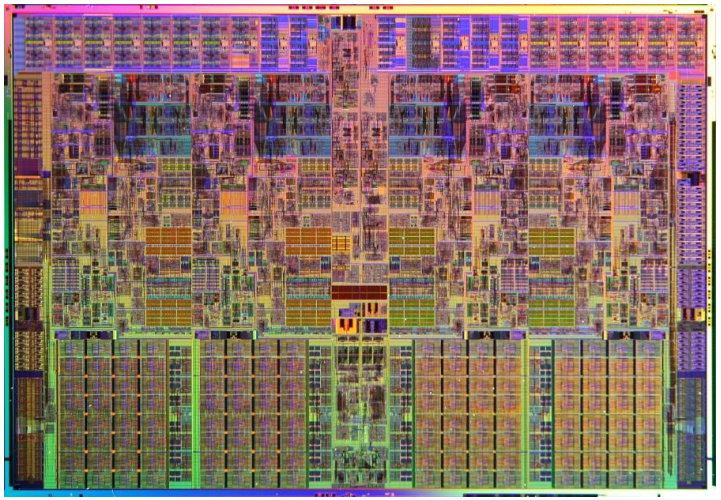
[Laura Frisk, Study of Structure and failure mechanisms in ACA Interconnect using SEM]

Why beyond?

However, also becoming an issue...

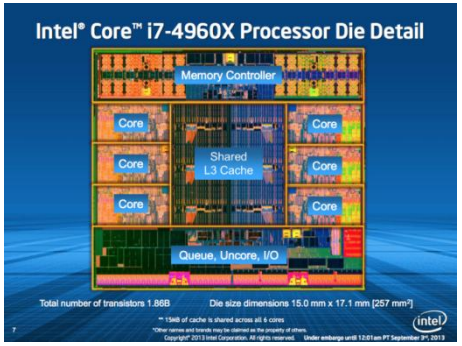


[Bosch, automotive electronics]

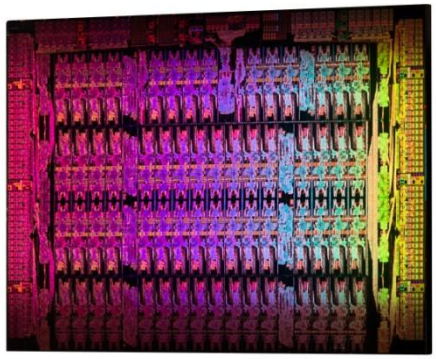


[Intel, Corei7]

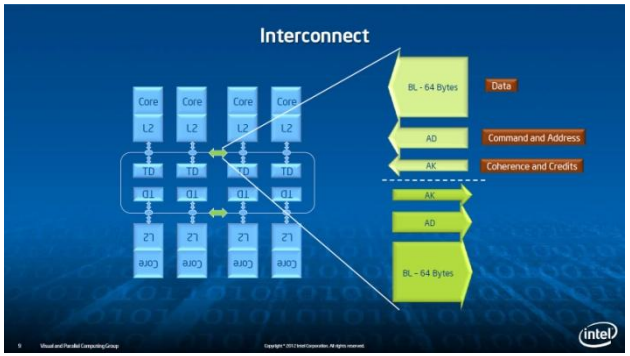
Distributed Systems



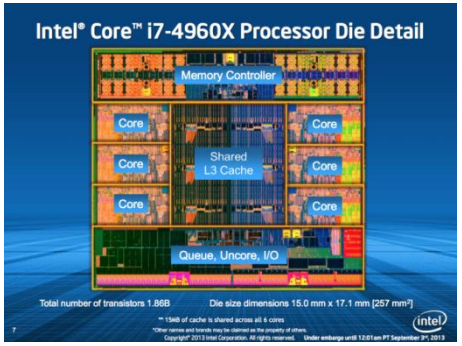
i7 (6)



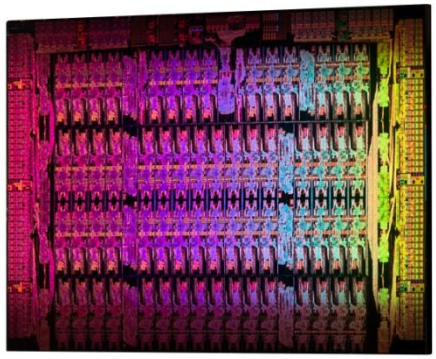
Xeon Phi (60)



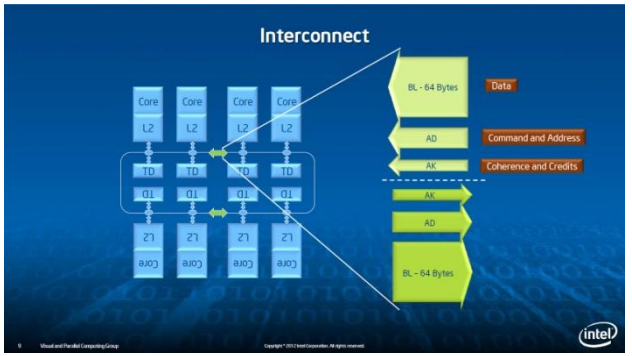
Distributed Systems



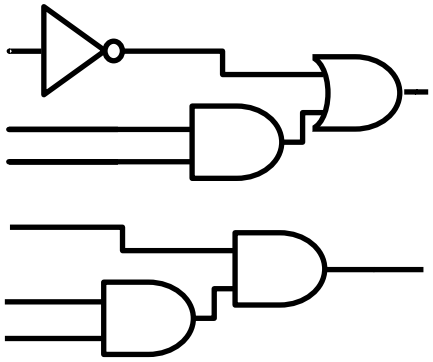
i7 (6)



Xeon Phi (60)

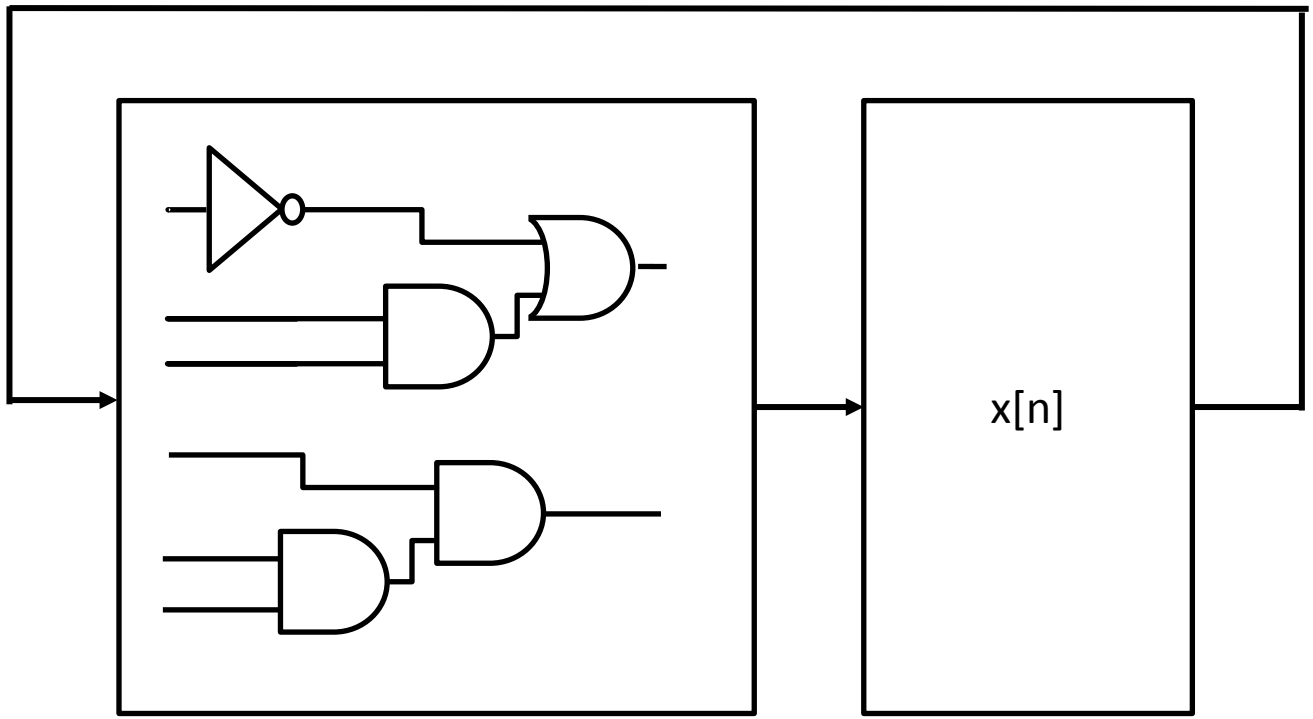


at all scales...

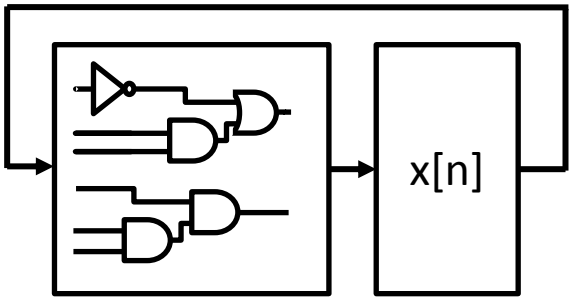


Modeling

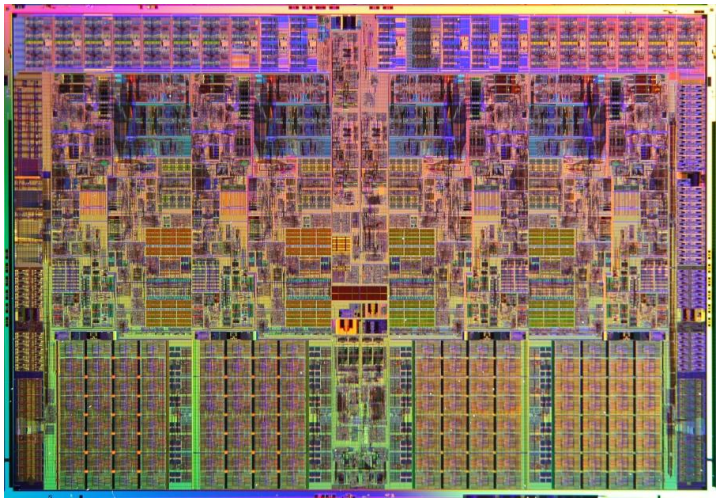
Circuit model



Modeling

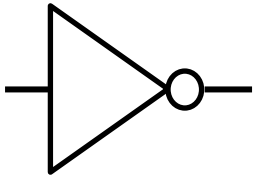


?
=

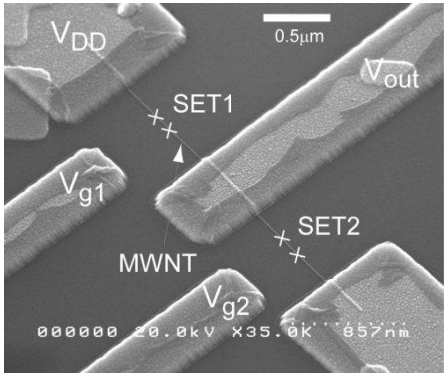


[Intel, Core i7]

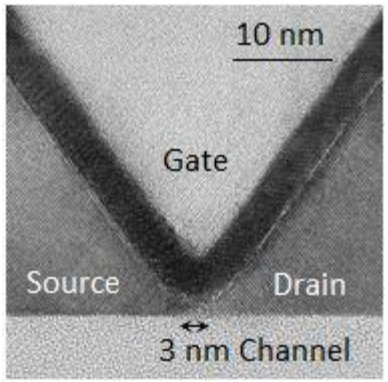
Modeling challenges



=

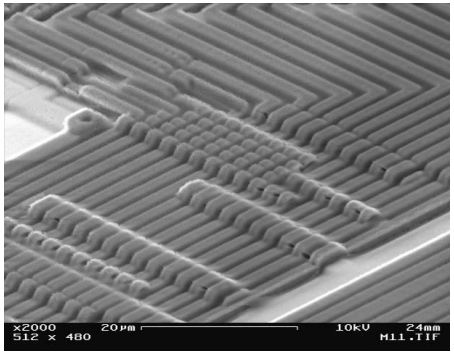


[Nanotech, SET]

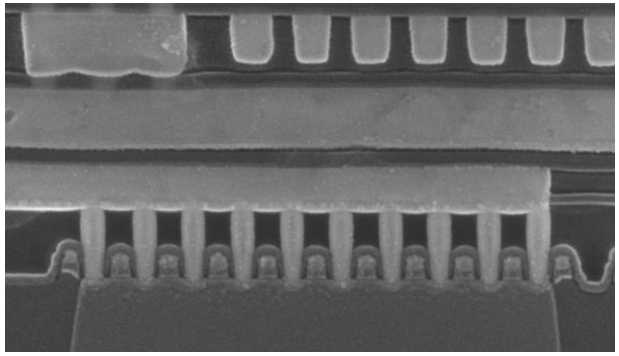


[Aist Japan]

=



[Optics Rochester, Intel Pentium]



[Chipworks, Apple A6]

In this course...

- revisit classical chip design
- clockless chip design
- timing
- distributed algorithms on chip
- fault-tolerance, self-stabilization

Course modalities

2 VO (Tue 10.15 – 12) + 2 UE (Thu 14.15 - 16)

weekly lectures, weekly exercises

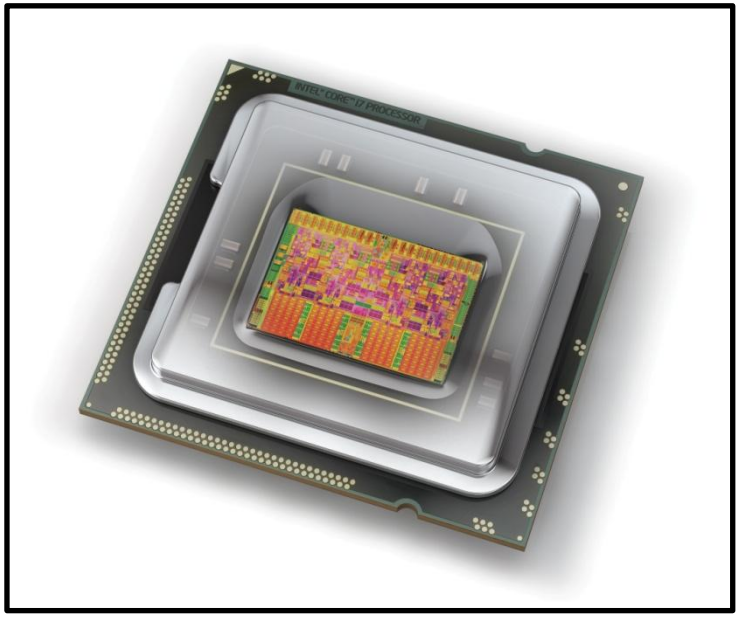
final grade: exercises + oral test

see web page for up to date information

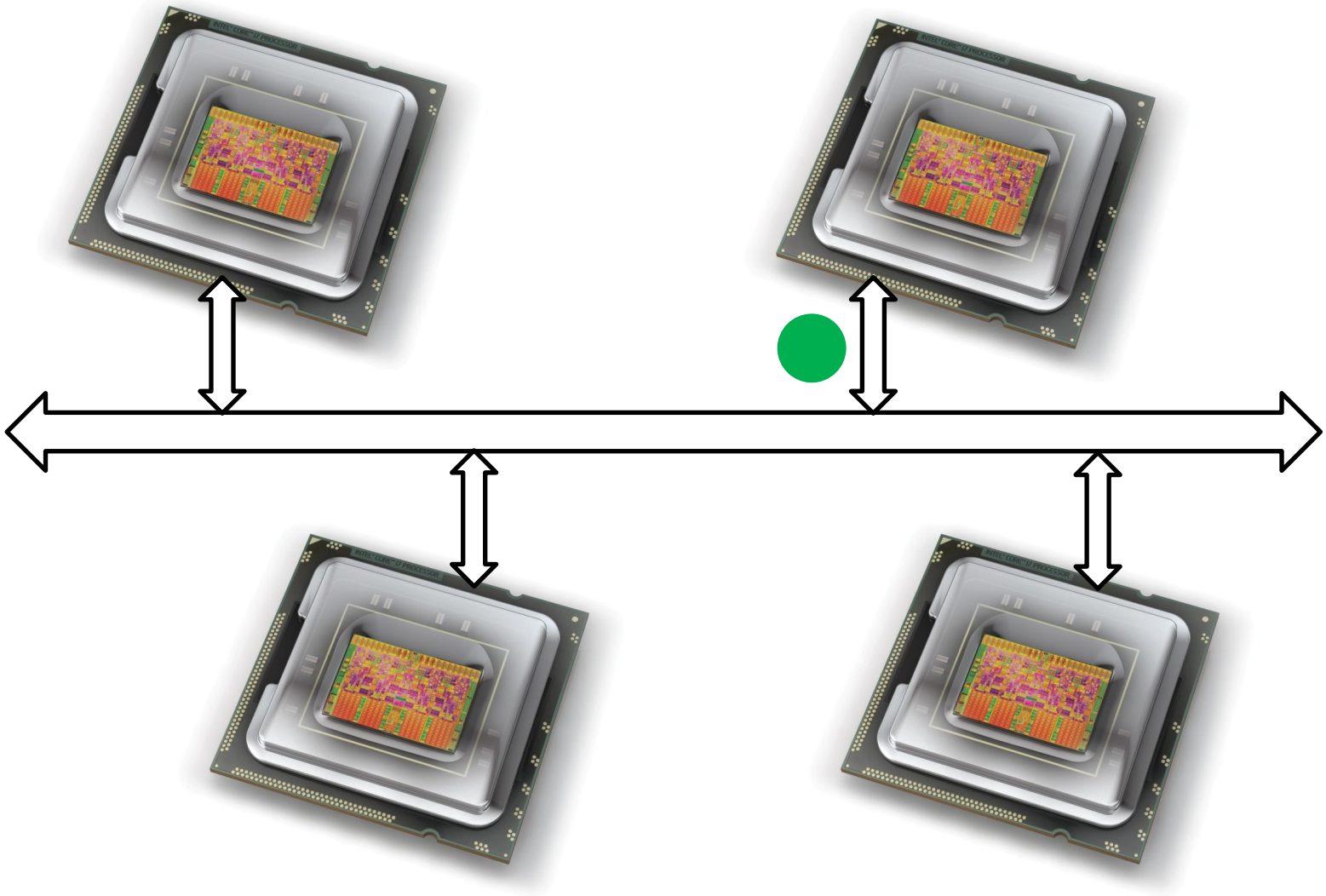
Beyond classical chip design
lecture 1.5

Self-stabilization

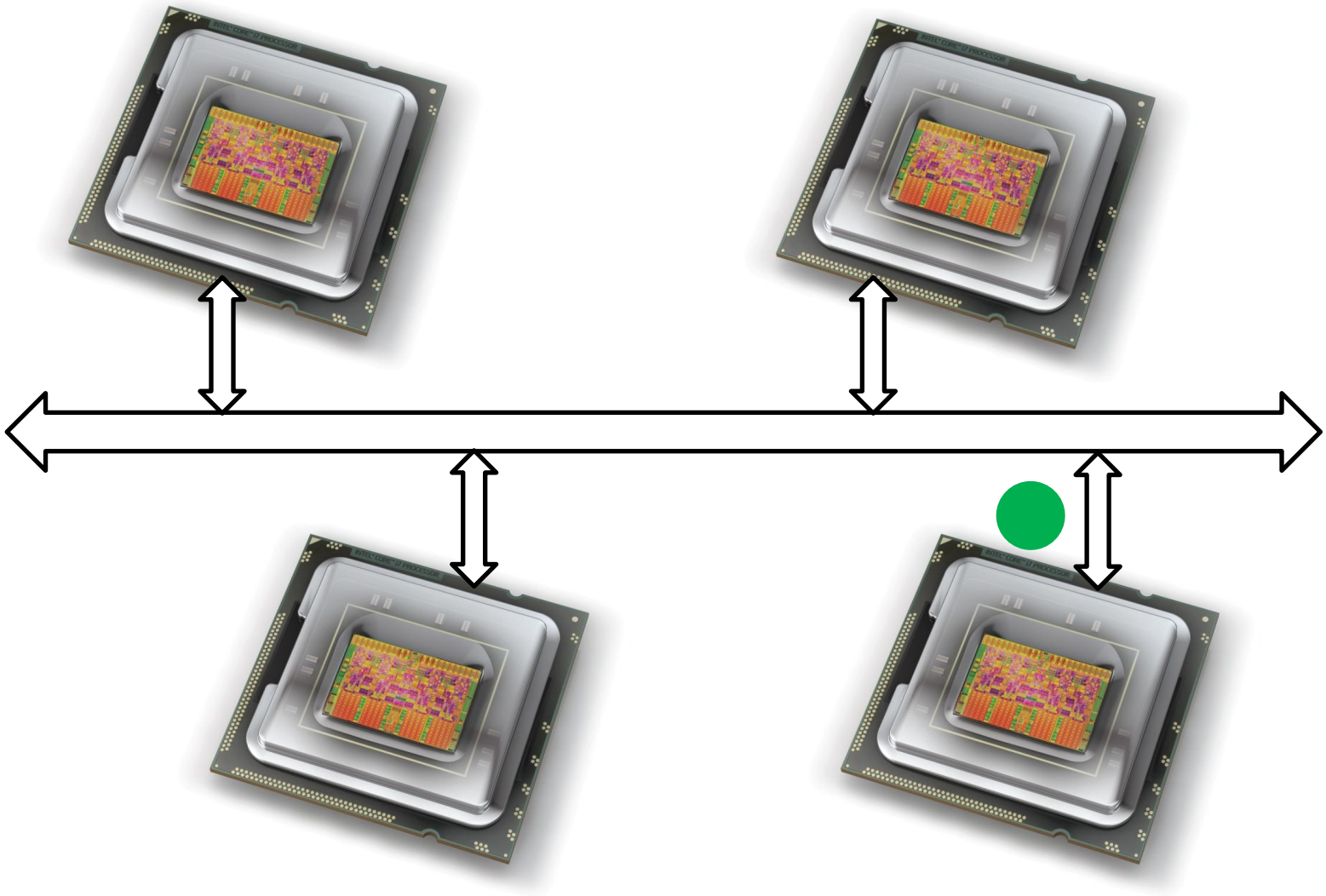
Reliable designs



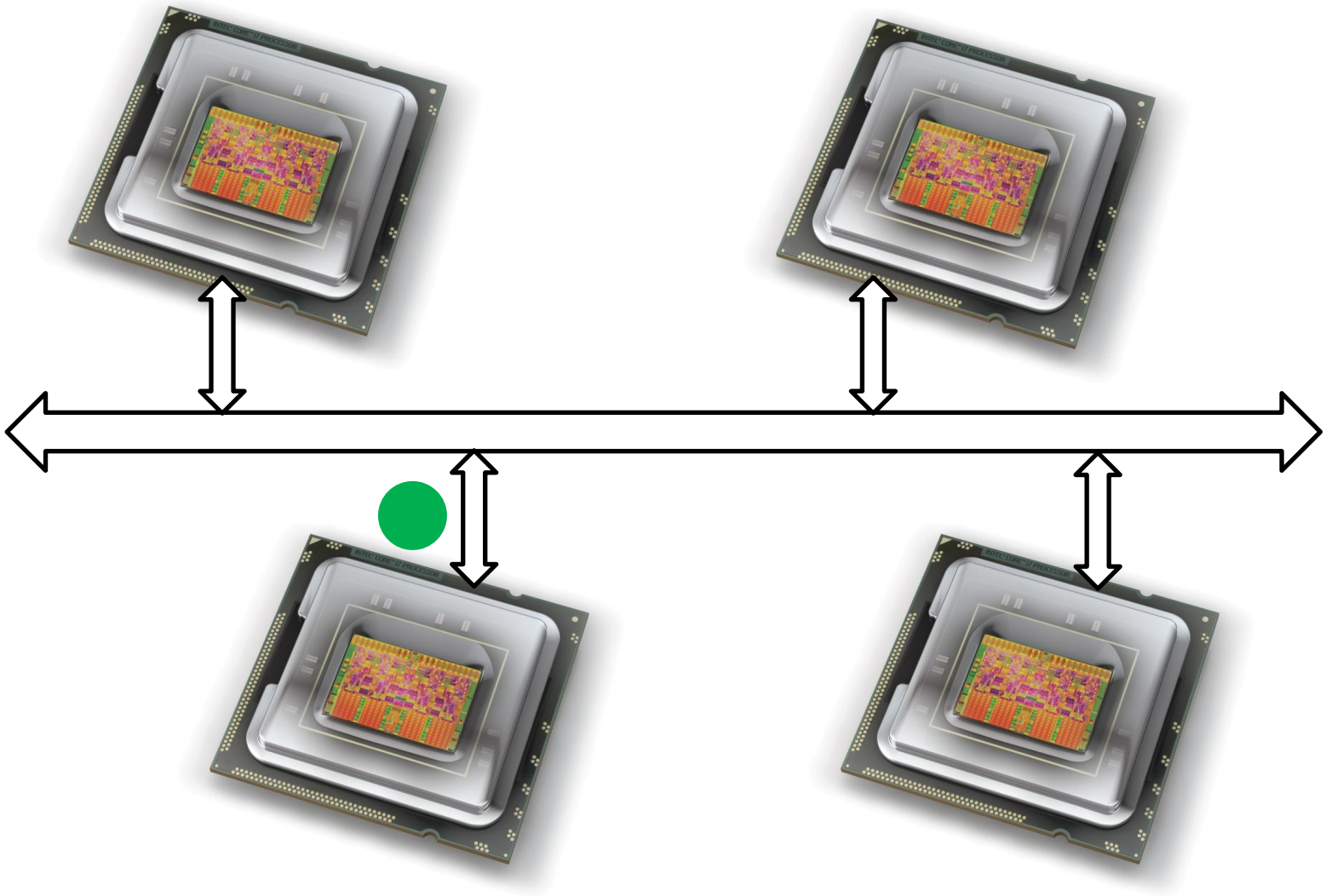
Example problem



Example problem

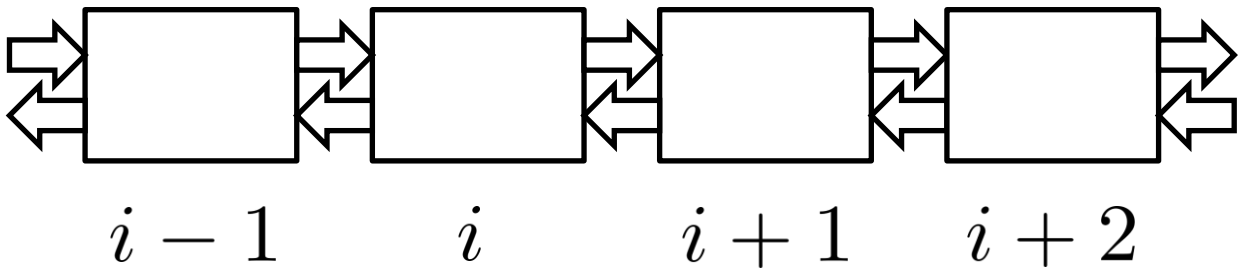


Example problem



Example problem

Token passing system.



- n nodes in a ring, can communicate with neighbours only
- a node can posses a token

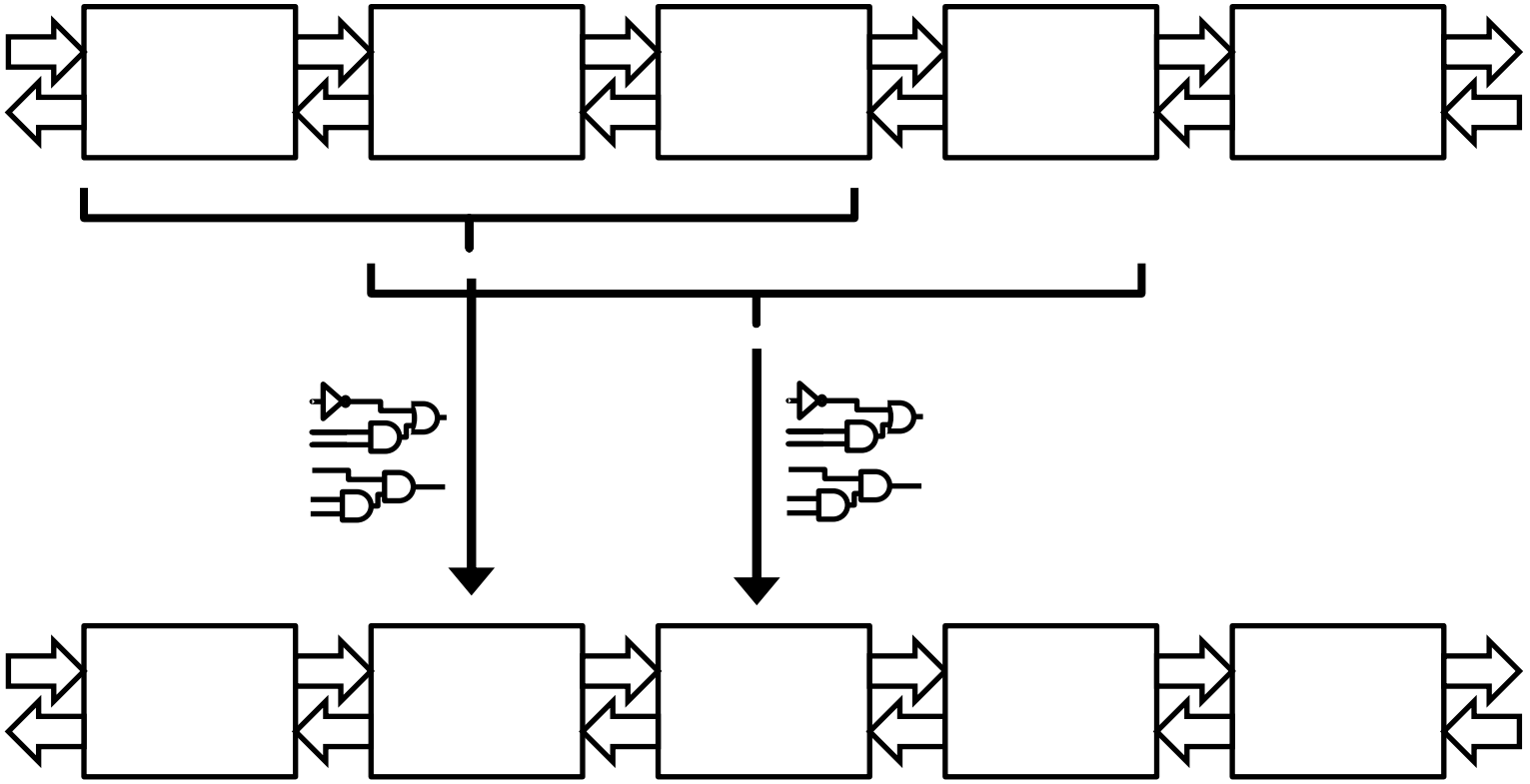
Example problem

Requirements:

- **mutex:** at most 1 node has the token at each time
- **weak fairness:** nodes get the token infinitely often

Example problem

Token passing system.



Example model

System model:

- global state $(x_0, \ldots, x_{n-1}) \in X^n$
- local transition function

$$\delta_i \in X^3 \to X : (x_{i-1}, x_i, x_{i+1}) \mapsto x_i'$$

- local token predicate

$$P_i \in X^3 \to \{0, 1\}$$

- distributed algorithm/circuit

$$((\delta_i, P_i))_{i \in [n]}$$

Example model

System model:

- schedule, weakly fair schedule [hw]

  $(s(t))_{t\geq 1}$ with $s(t) \in [n]$

- “i makes a transition at time t” <->

  $i = s(t)$

Example model

System model:

- initial global state + distributed algorithm + schedule -> execution $(x(t))_{t \geq 0}$

  where

$$x(t+1) =$$

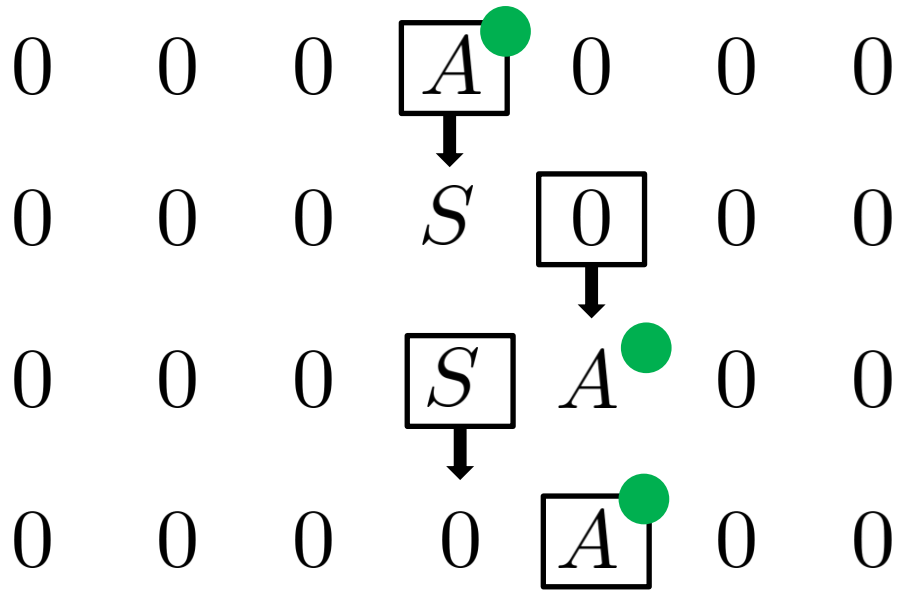
$$(\ldots, x_{i-1}(t), \delta_i(x_{i-1}(t), x_i(t), x_{i+1}(t)), x_{i+1}(t), \ldots)$$

  and $i = s(t+1)$

Example solution

Algorithm:

- ☐ ... enabled = non-trivial transition
- ● ... token



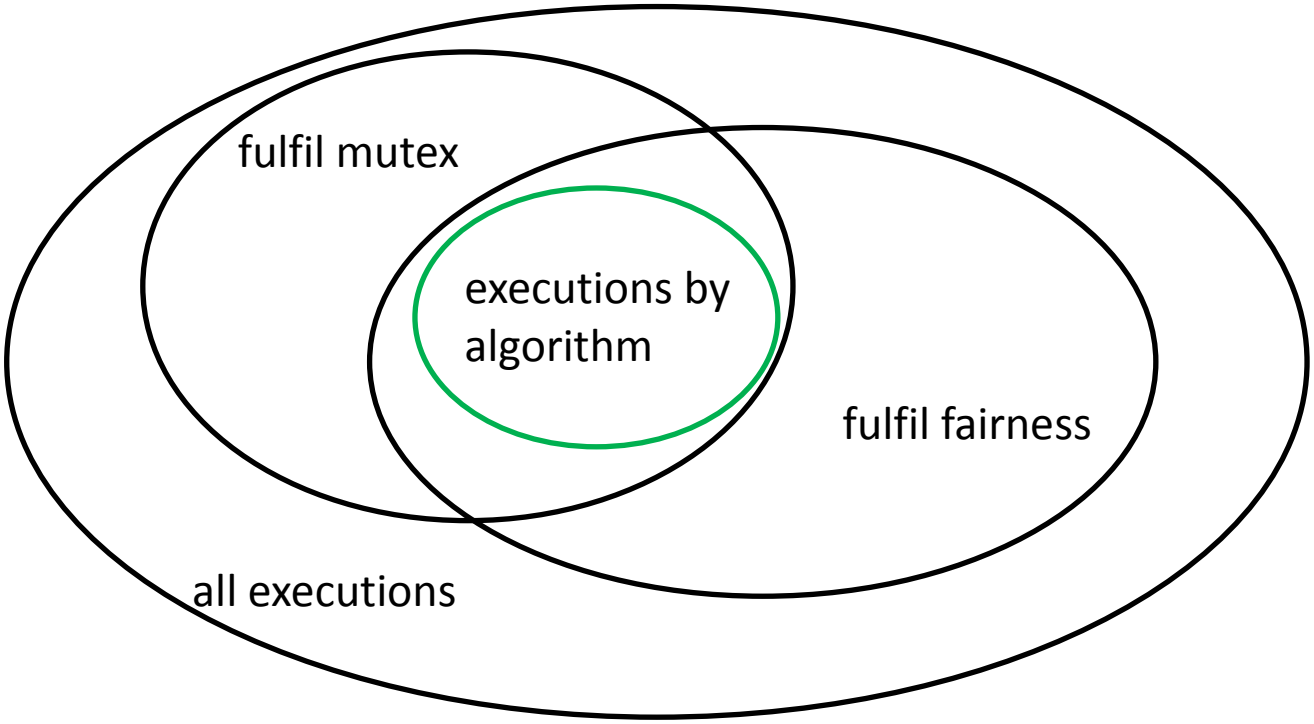
Example solution

Correctness.

Obs 1. When removing stuttering steps, there is only 1 execution.

Nature of requirement

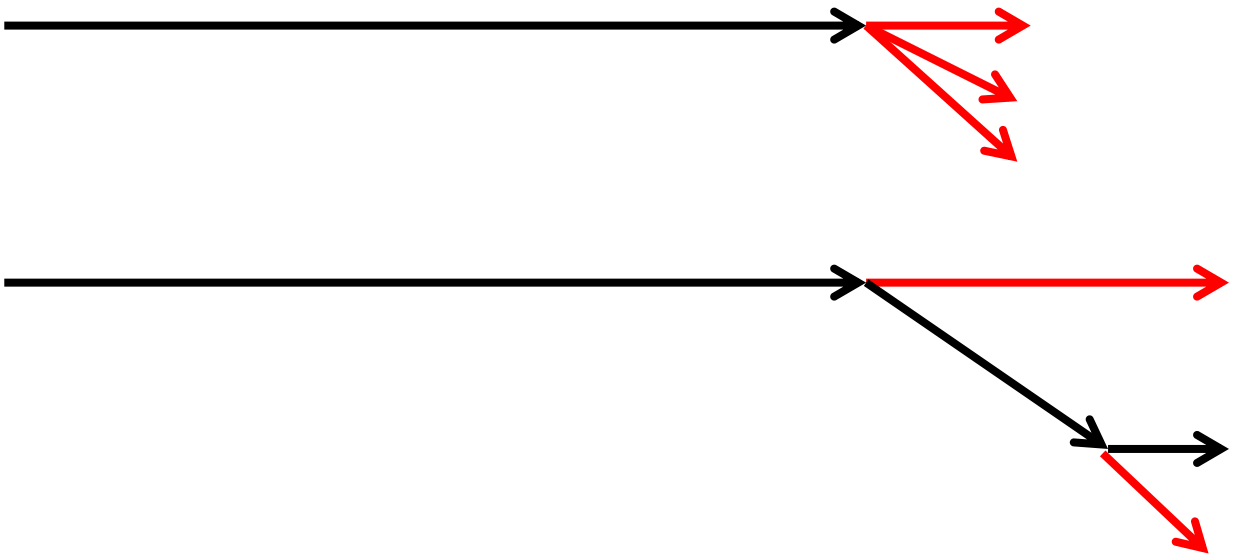
Requirement <-> set of executions that fulfil it



Nature of requirement

violated at time t <->

t-prefix cannot be extended to execution that fulfils requirement



Nature of requirement

violated at time t <->

t-prefix cannot be extended to execution that fulfils requirement

Safety: violated -> violated at some t

Liveness: violated at no time t

Example problem

Safety: mutex

Liveness: weak fairness

proof techniques differ

Here simple proofs [hw]

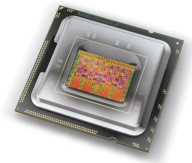
Example solution

Algorithm:

3 states,

uniform,

very simple predicate and transition function

-> 

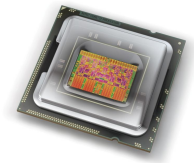
Example solution

Algorithm:

3 states,

uniform,

very simple predicate and transition function

-> 

But wait...

$$\ldots s(t) \in [n]$$