Growing processor arrays: how and why?

Development in hardware– What?

- Mechanisms inspired by the biological process of growth (and healing)
- Digital logic using FPGAs
- The goal is NOT to mimic biology (or help biologists) but to solve problems in hardware design
- The goal is NOT to grow form, but <u>function</u>! i.e., design systems that use development to <u>execute an application</u> better/more efficiently/with non-standard constraints

Development in hardware– Why?

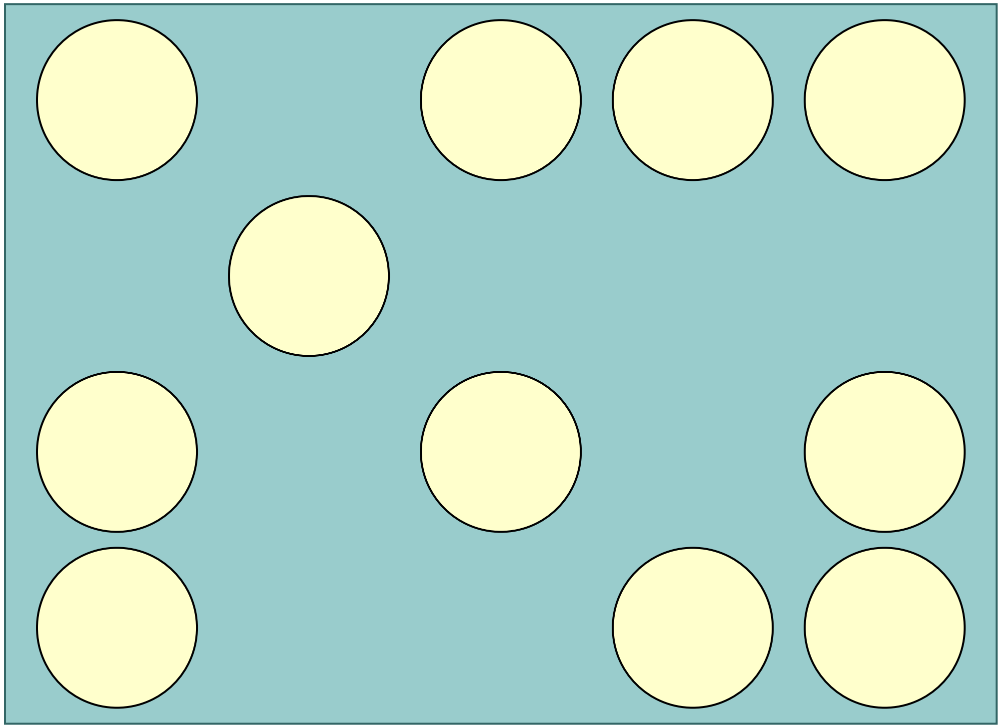
- Complex genotype/phenotype mappings
  - Improves scalability in evolutionary approaches
  - Defines initial structure in ANNs

Development in hardware– Why?

- Complex genotype/phenotype mappings
  - Improves scalability in evolutionary approaches
  - Defines initial structure in ANNs
- Model growth and structural adaptation
  - Self-organization
  - Environmental adaptation

Self-organization

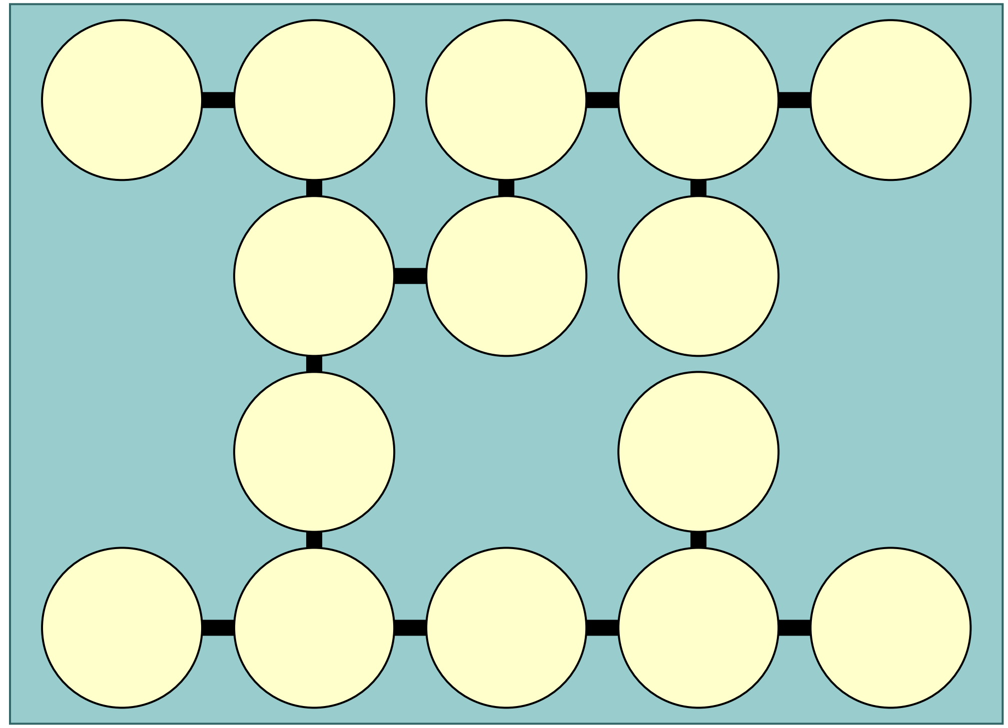
Let us assume that we can make “cells” appear and disappear at will in a surface of silicon.



Self-organization

We can then apply all sorts of nice algorithms:

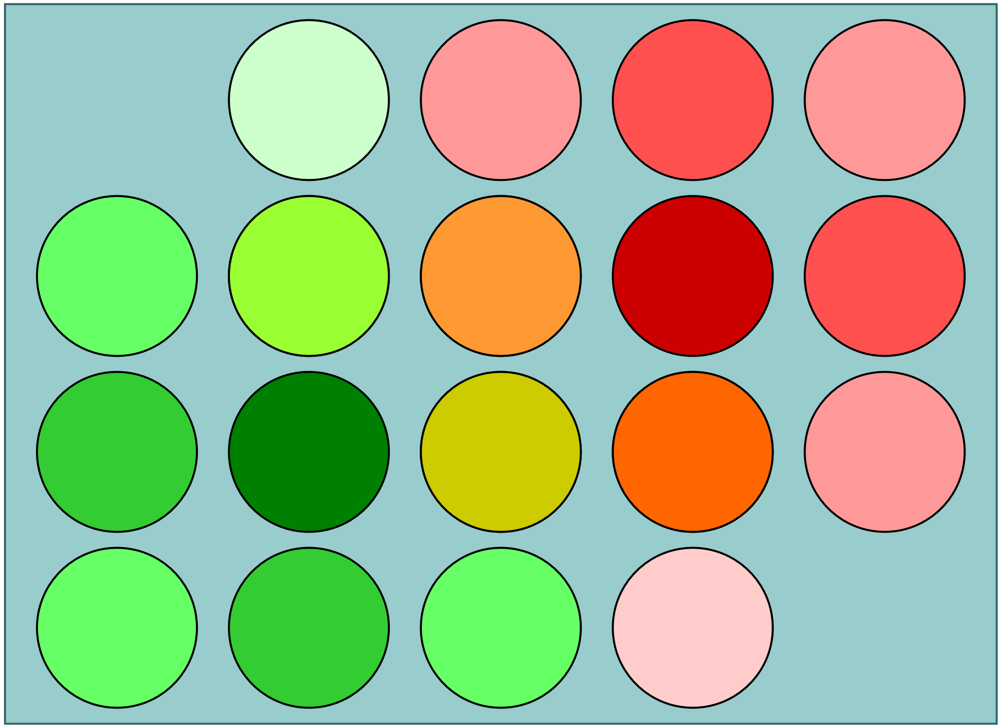
- L-Systems

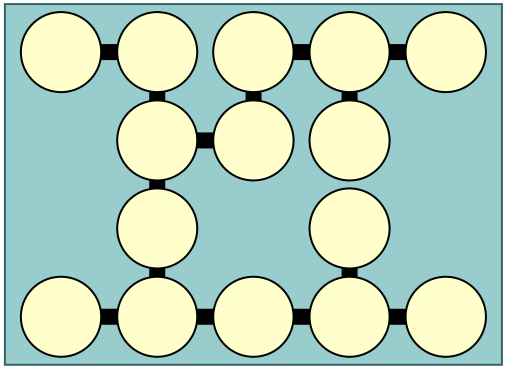


Self-organization

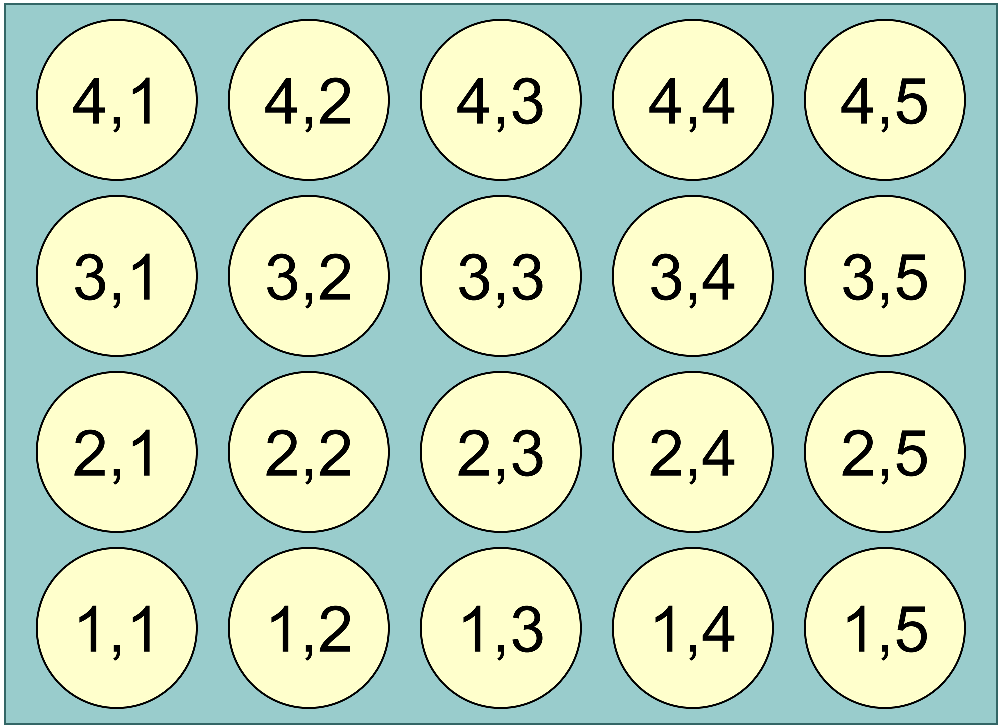
We can then apply all sorts of nice algorithms:

- L-Systems
- Gradients



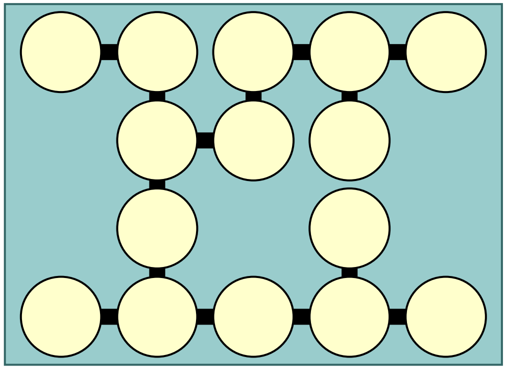


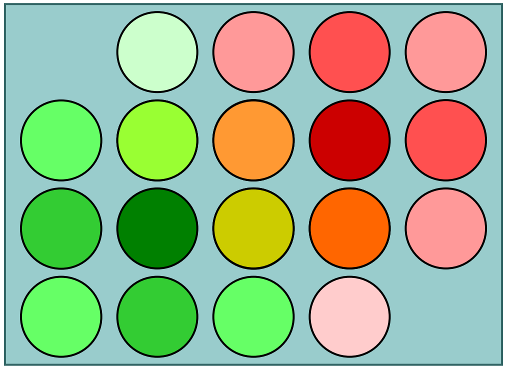
Self-organization



We can then apply all sorts of nice algorithms:

- L-Systems
- Gradients
- Coordinates

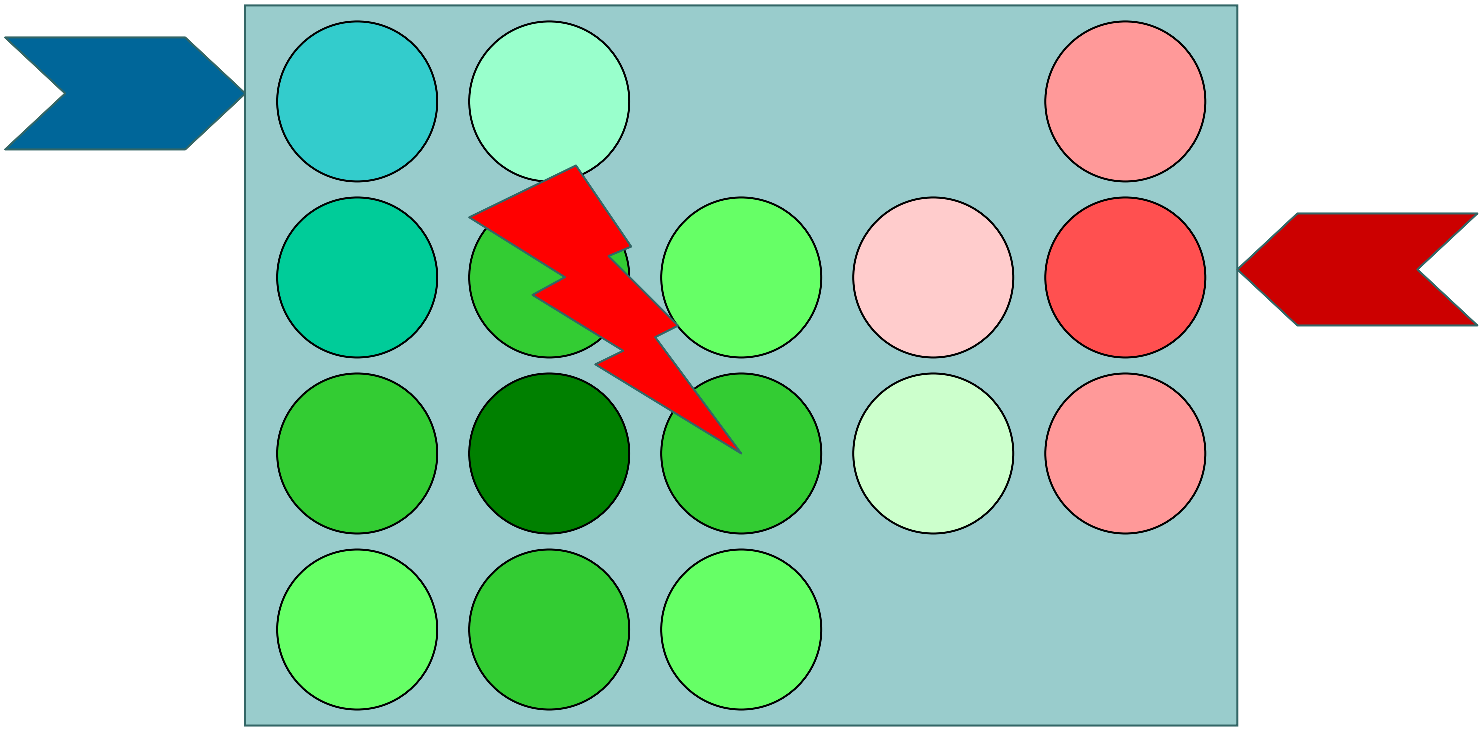




Environmental adaptation

Self-organization is hard to justify for silicon!

…unless growth and structural adaptation cannot be represented in a genome: they are influenced by environmental variables.



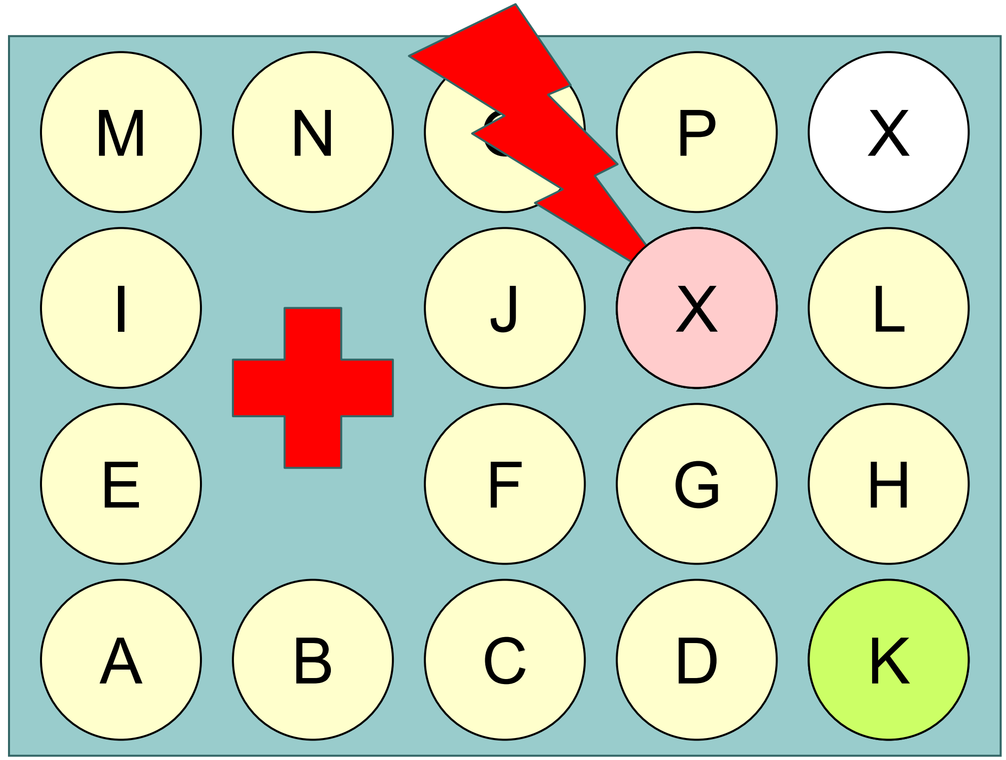
Fault tolerance

- Faults at fabrication are increasing.

  Self-organization is back!

- Online faults are increasing

  Self-organization is back!

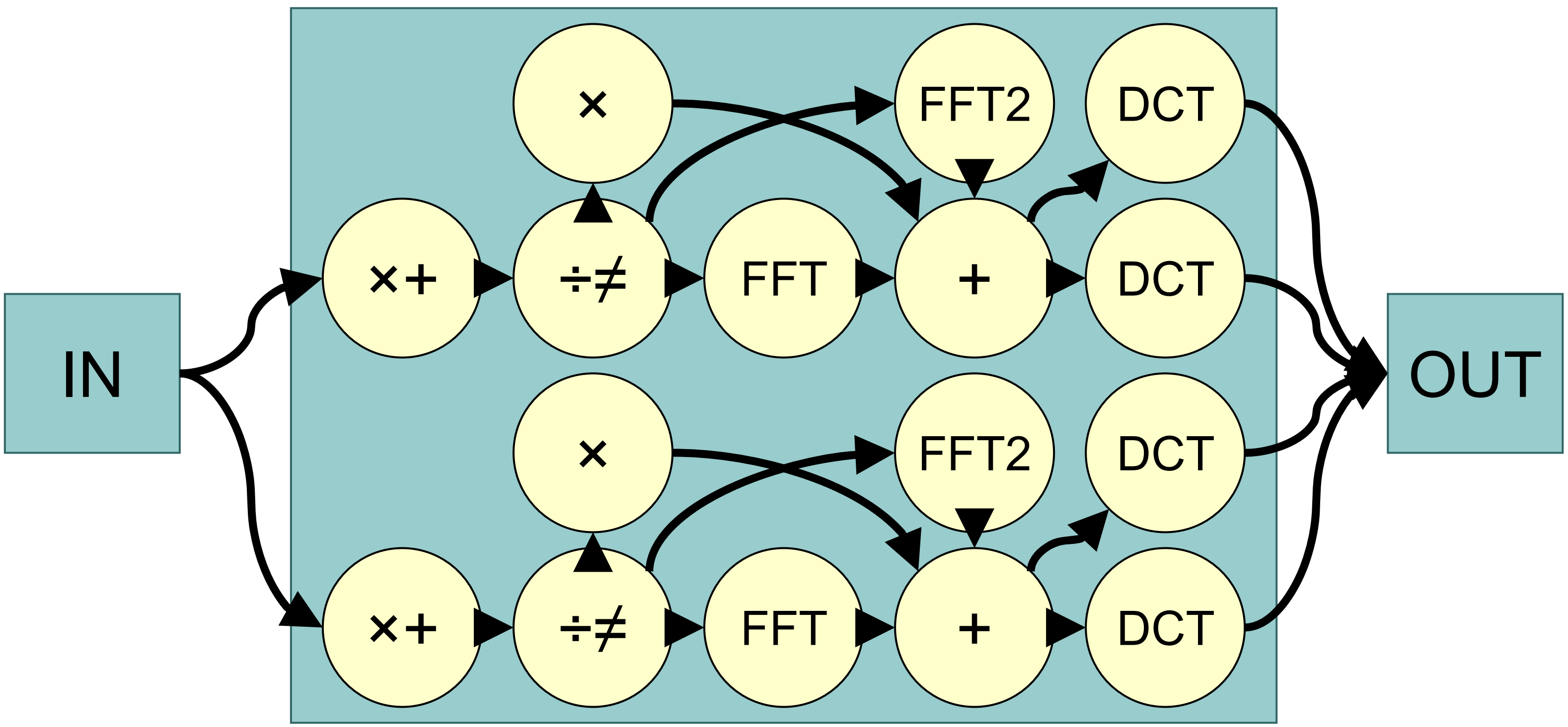


Similar mechanisms can be used for development and for self-repair (stem cells + differentiation!).

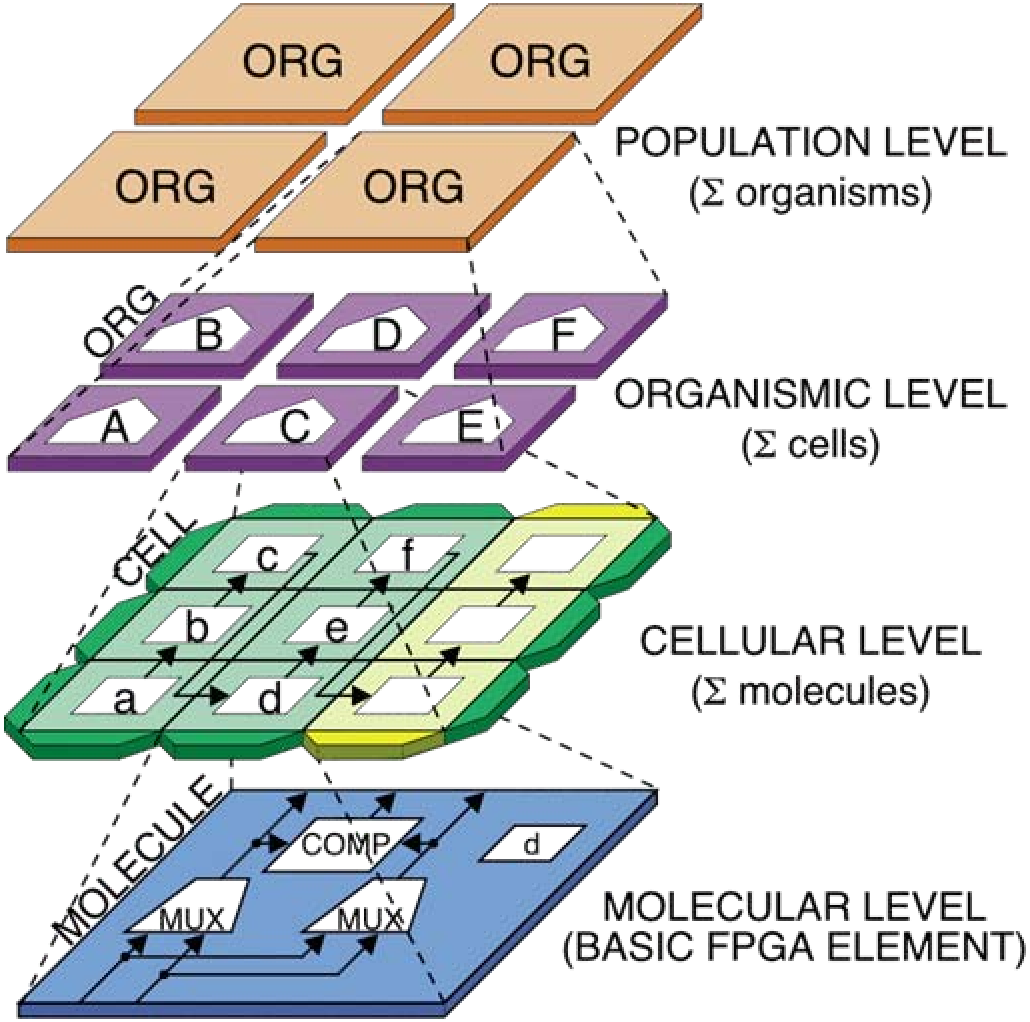
Fault tolerance = environmental adaptation.

Environmental adaptation

- Application self-organizes depending on input stream – structural adaptation



Multi-cellular paradigm



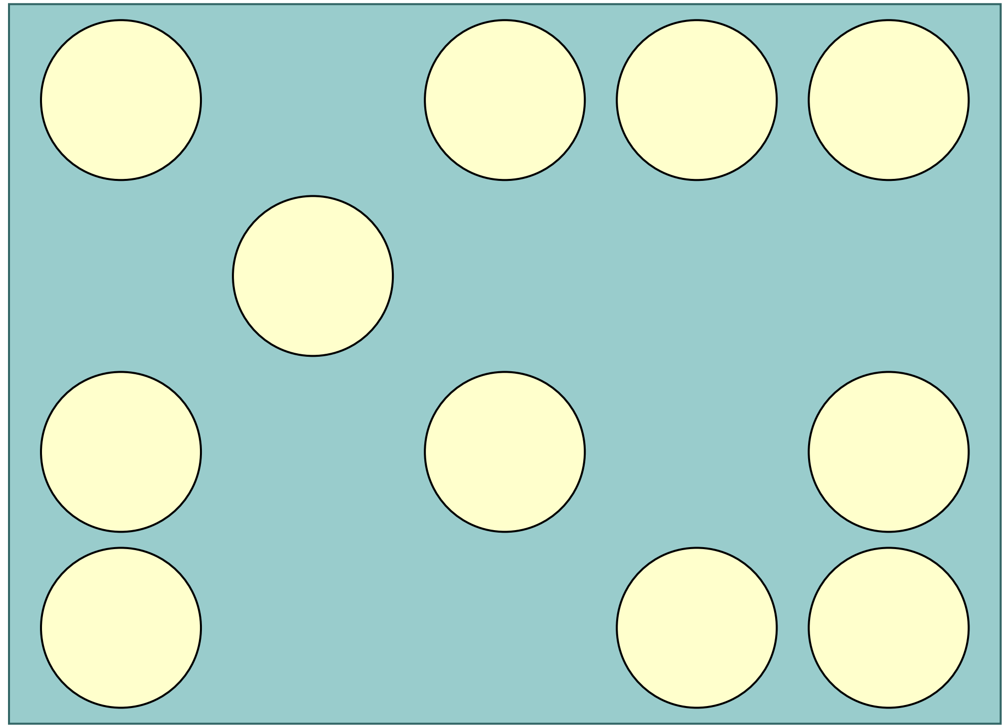
Structural adaptation at the organismic, cellular and molecular level can:

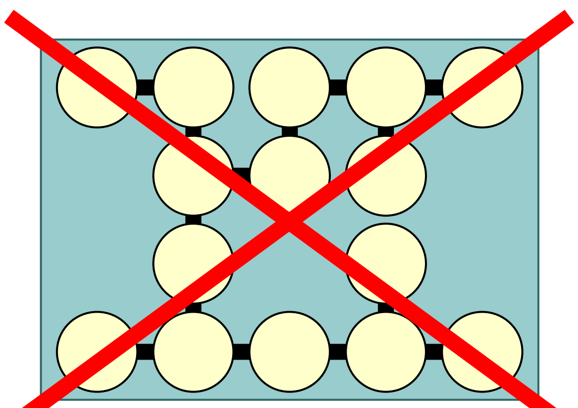
- Increase performance through parallelism.
- Ensure scalability through homogeneity.
- Simplify task partitioning through specialization.

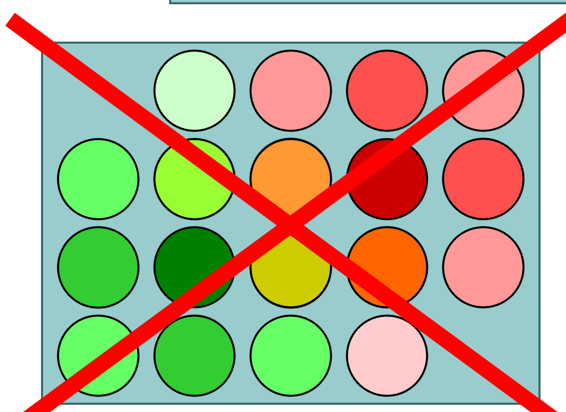
But unless you're IBM it's VERY difficult to prove!

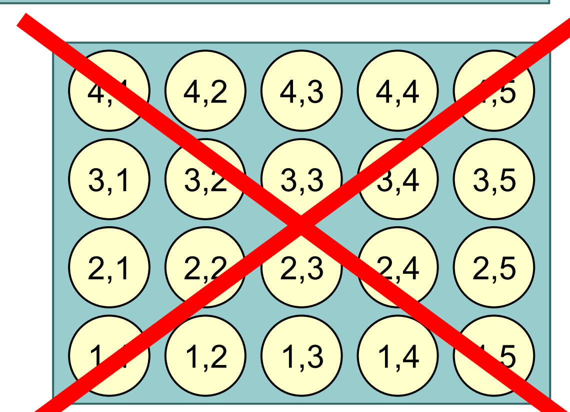
Dynamic hardware

But of course it is NOT easy to make cells appear and disappear at will in a surface of silicon.









Development in hardware– Why?

- Complex genotype/phenotype mappings
  - Improves scalability in evolutionary approaches
  - Defines initial structure in ANNs
- Model growth and structural adaptation
  - Self-organization
  - Environmental adaptation
- Exploit the multi-cellular paradigm
  - Scalable and adaptive massively parallel systems
  - Fault tolerance

Engineering challenges

Several key mechanisms of development are extremely difficult (if not impossible) to implement using silicon-based devices.

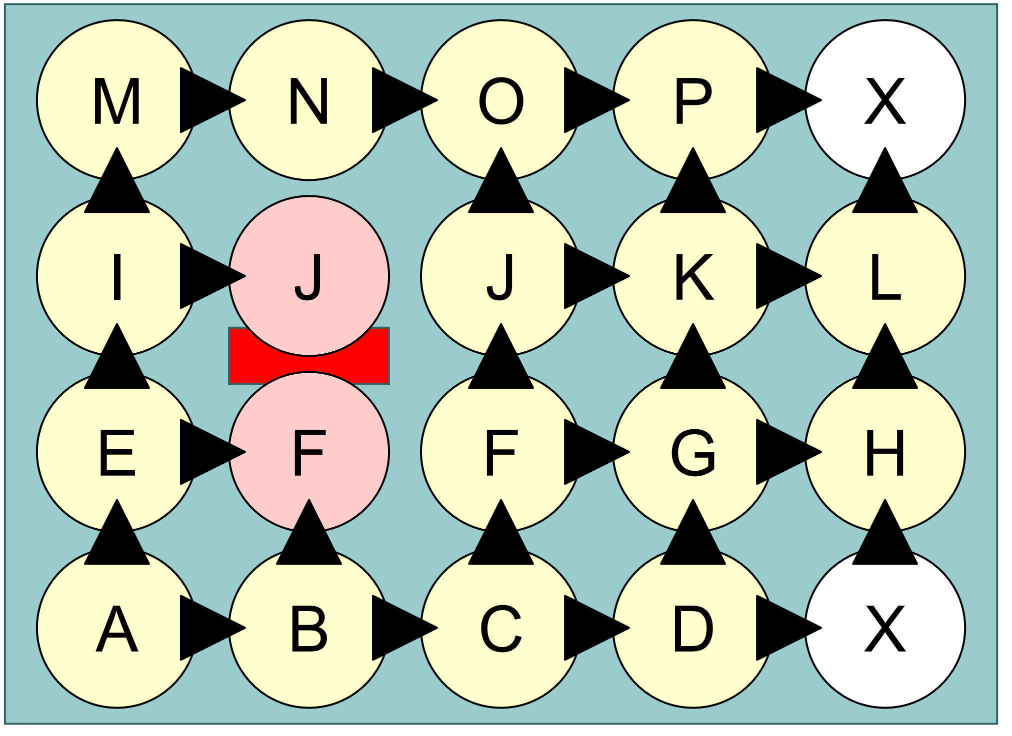
FPGAs have made possible the implementation of developmental processes in an informational (rather than physical) universe.

But…

Several key mechanisms of development are extremely difficult (if not impossible) to implement using conventional devices and systems.

Engineering challenges

Self-organization = self-configuration



≈ self-replication of a partial configuration

Self-replication

- Self-replication by construction

Self-replication

- Self-replication by self-inspection

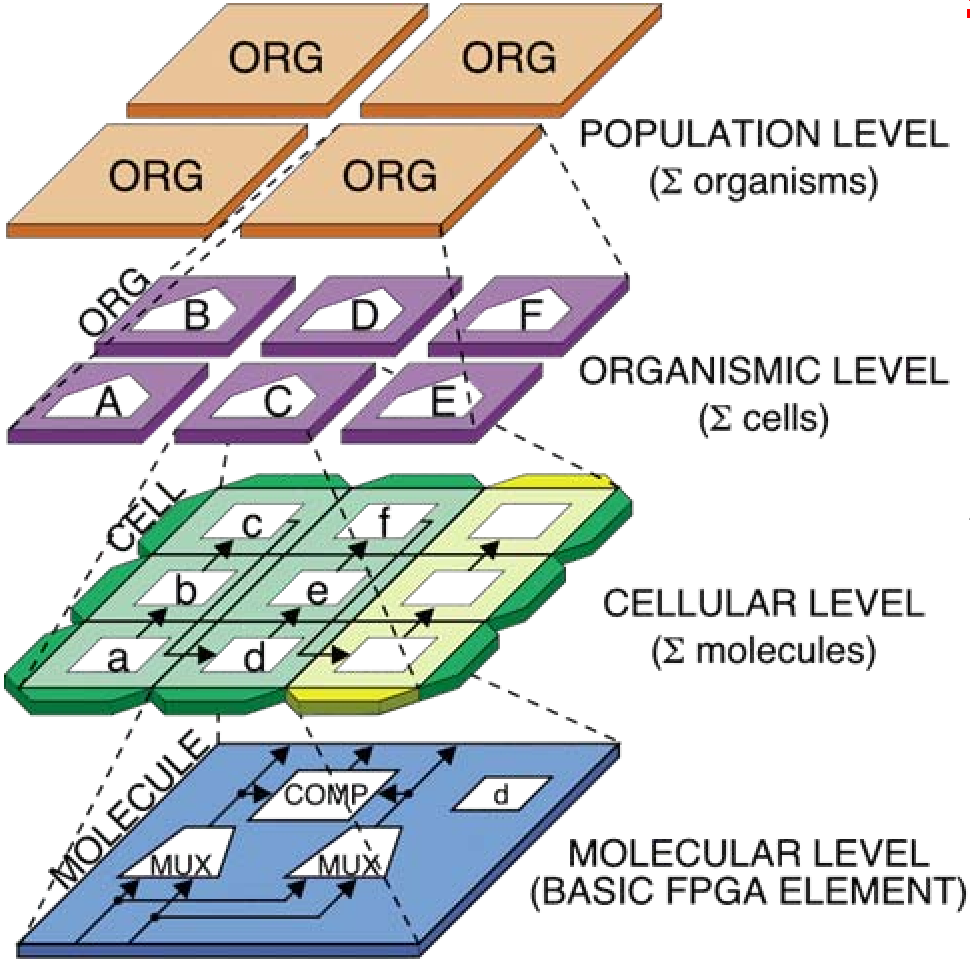
Adaptive systems

- Unfortunately, that's where the real problems begin!!!
- Very practical issues:
  - Granularity – what is a "cell"?

    Can range from a few logic gates to complex processors

    Two sub-issues appear:
    - **Overhead** – must be "reasonable"
    - **Efficiency** – must be high

Multi-cellular paradigm



Assumptions:

1. Processor-level cells can more easily justify and "absorb" the overhead
2. Mechanisms MUST span several levels of complexity to increase efficiency and "spread the cost"

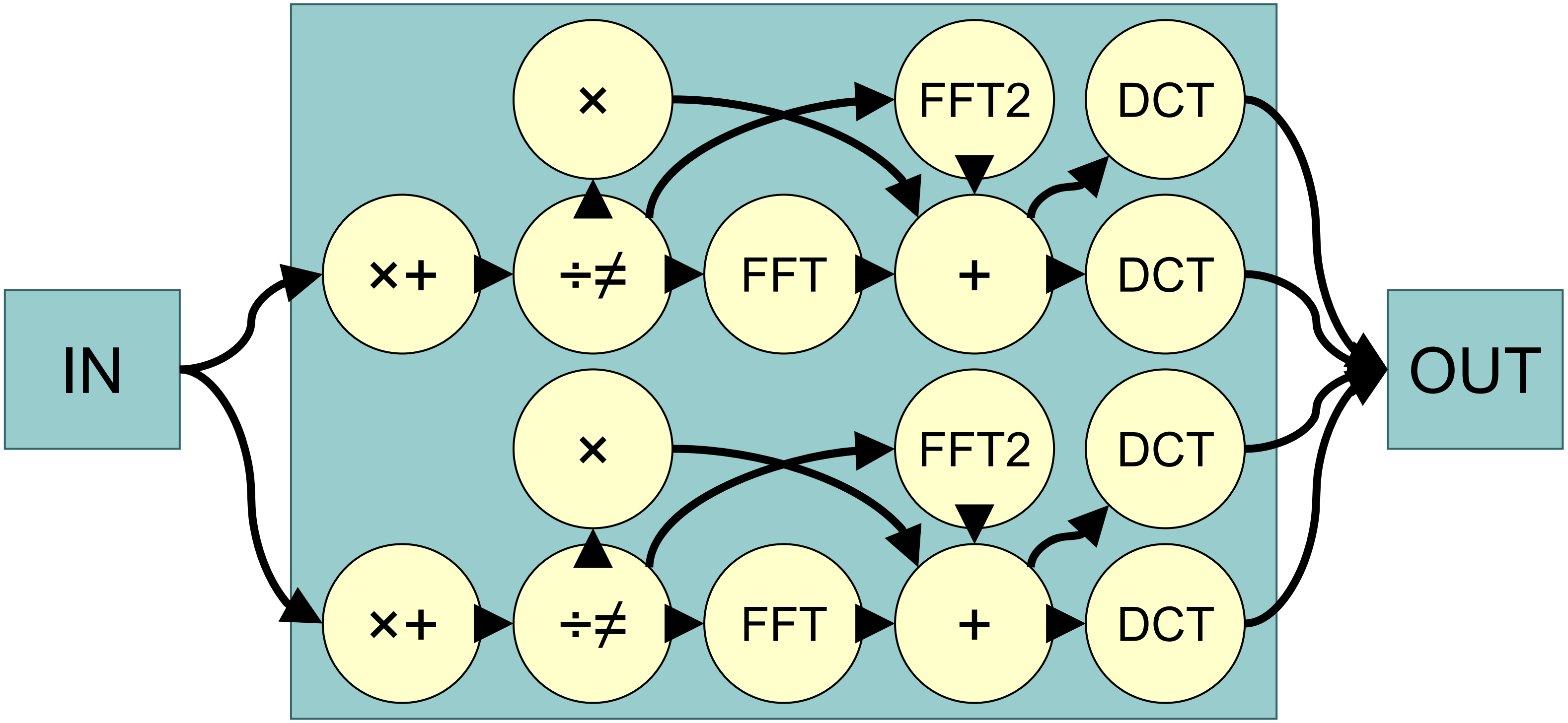
Adaptive systems

- Unfortunately, that’s where the real problems begin!!!
- Very practical issues:
  - Granularity – what is a “cell”?
  - Design – how to go from an application to a system like this?

How do you design a system that can exploit a developmental approach?

Design

- How do you design a system where the application self-organizes depending on input stream

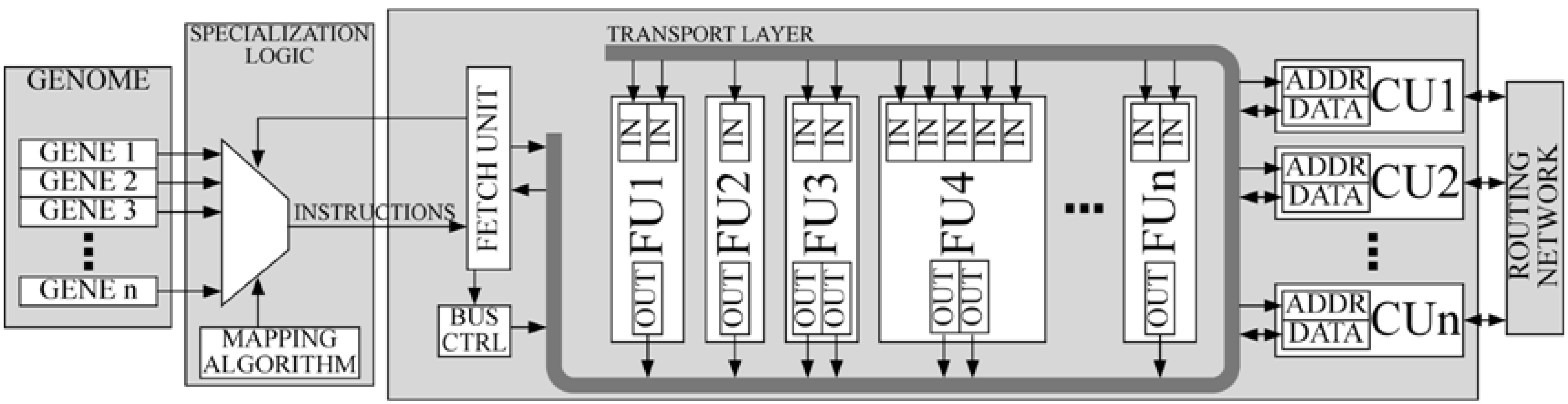


Design choices

- The system can (should?) adapt to application at several levels: molecule, cell, organism. E.g.:
  - Molecule: FPGA self-configuration
  - Cell: Application-specific processing elements
  - System: Self-organization of the PEs
- Choices:
  1) Make them universal, losing some bio-inspired design properties and falling back on “conventional” parallel processing issues
  2) Make them specific (structural adaptation) and try to find ways to integrate development in the design (avoiding hand-design)

Design

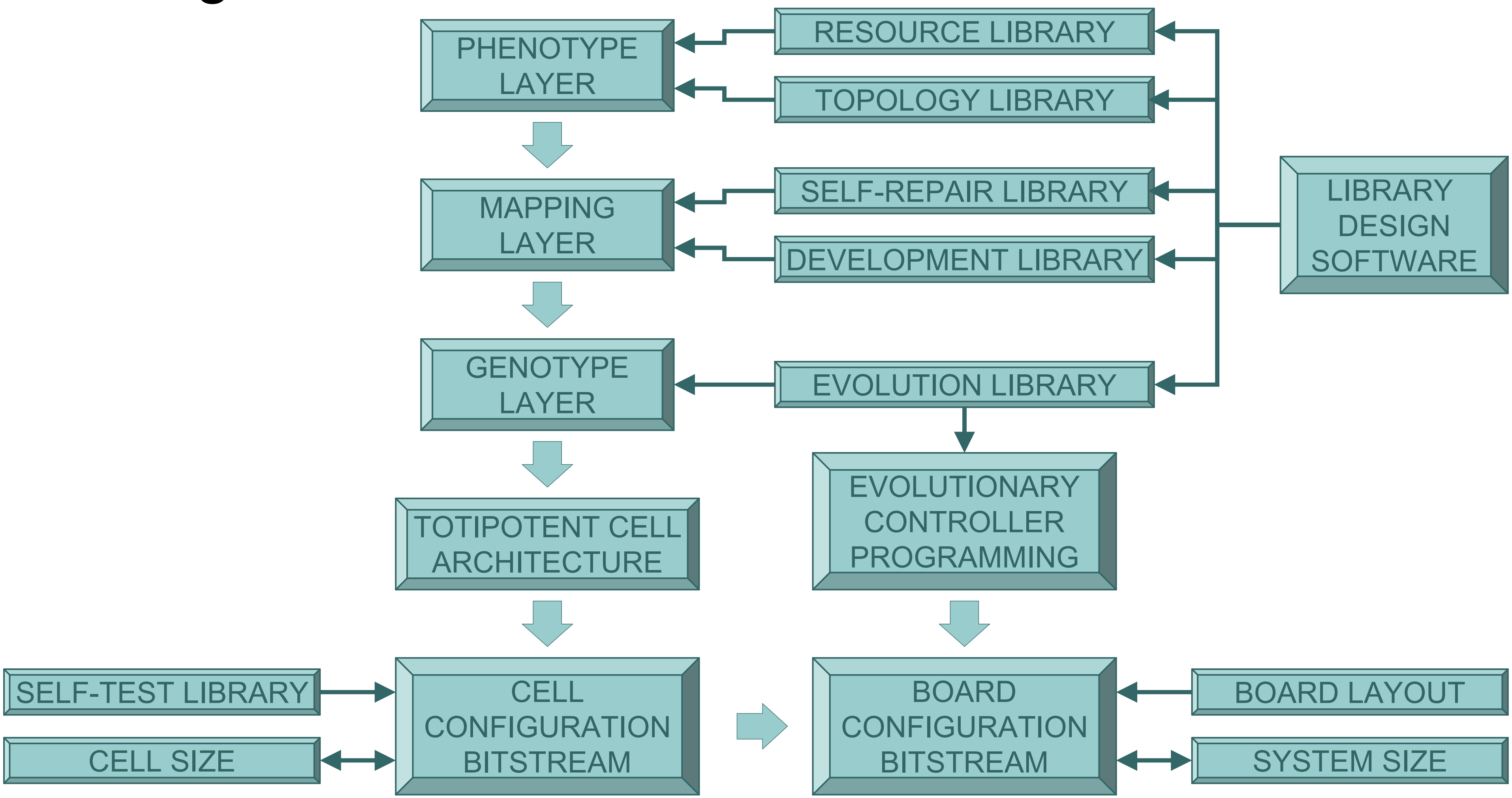
- Molecules: unlikely to be specific in silicon
- Cells: hard, but can be done as application-specific processors (e.g. MOVE processors)



- Organism: can be done (I assume), but VERY hard (hardware/software codesign, parallel processing)

Design

- Design environment



Adaptive systems

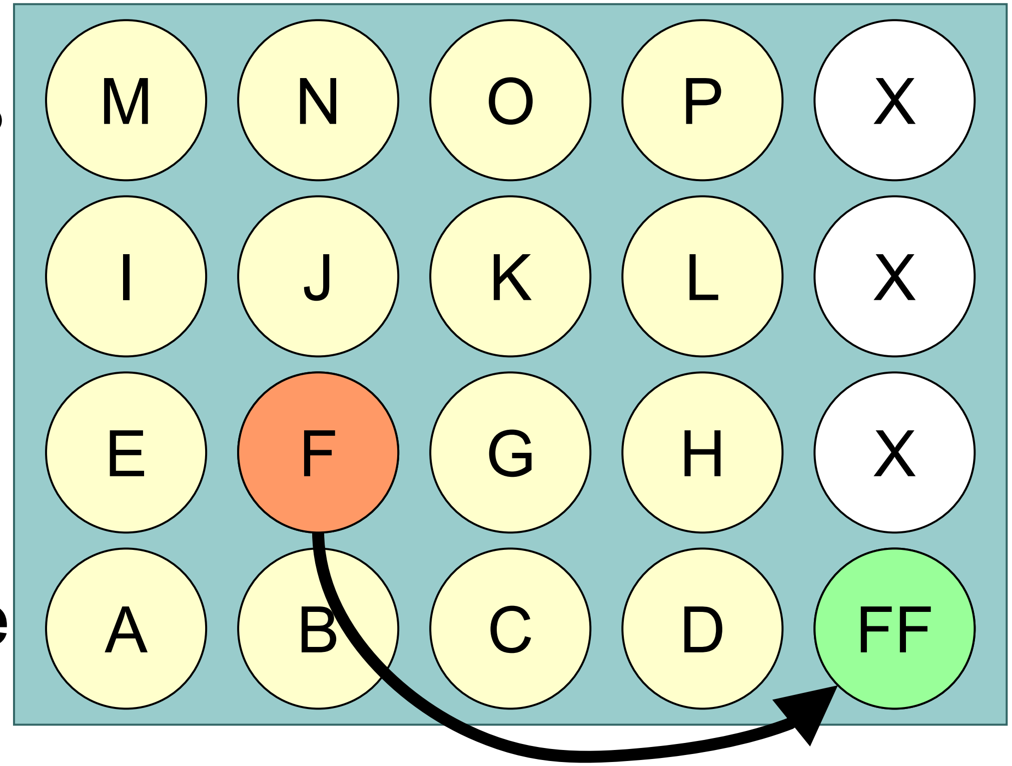
- Unfortunately, that's where the real problems begin!!!
- Very practical issues:
  - Granularity – what is a "cell"?
  - Design – how to go from an application to a system like this?
  - Execution – how does it run?

How can an array of processors use this kind of mechanisms and do so efficiently?

Execution

- A cell self-replicates… what does that MEAN?

1) Create whole array at startup (losing some differentiation options)
2) Dynamically create/ destroy cells at runtime



- Can you justify the time required for self-replication?
- A (sequential) cell at runtime has a STATE. Do you want to replicate that state or the initial state?
- For self-repair, do you recover the state? How?

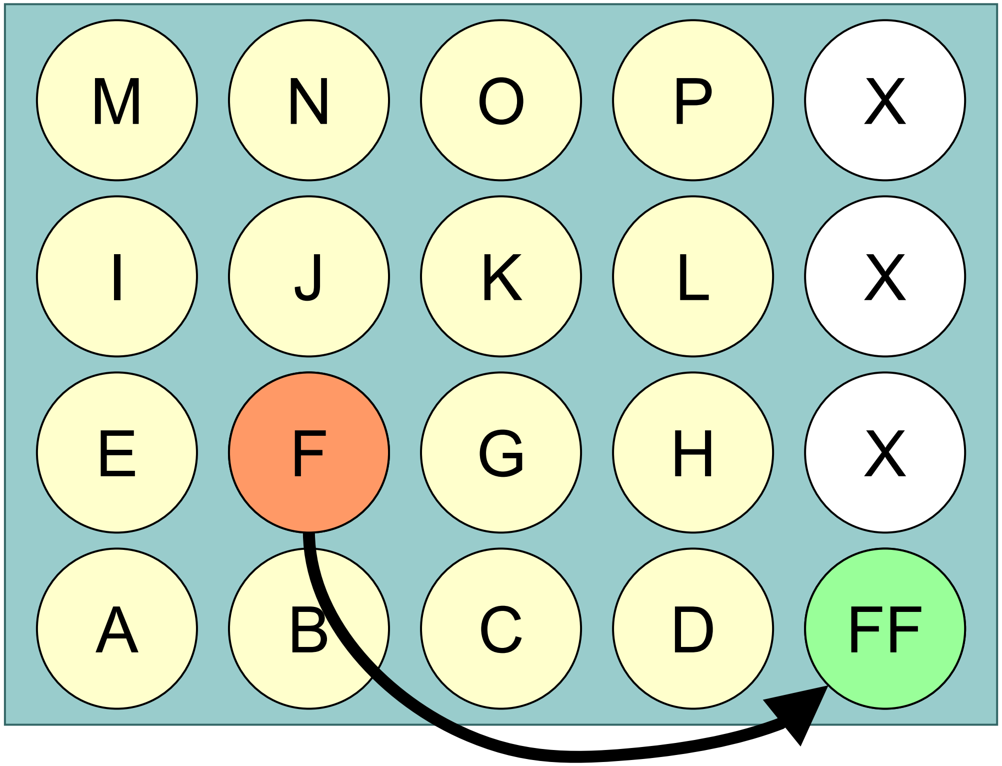
Adaptive systems

- Unfortunately, that’s where the real problems begin!!!
- Very practical issues:
  - Granularity – what is a “cell”?
  - Design – how to go from an application to a system like this?
  - Execution – how does it run?
  - Connectivity – how do the cells communicate?

How can you set up and preserve a communication network through self-reorganization?

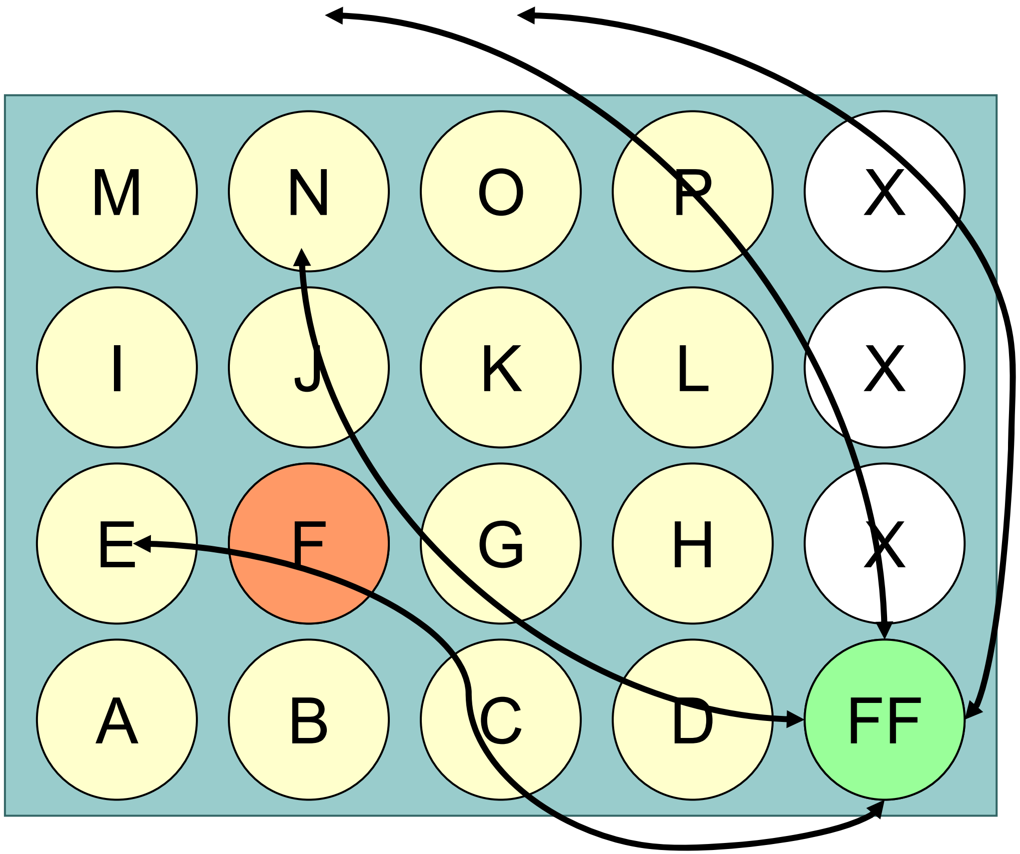
Connectivity

- Growth



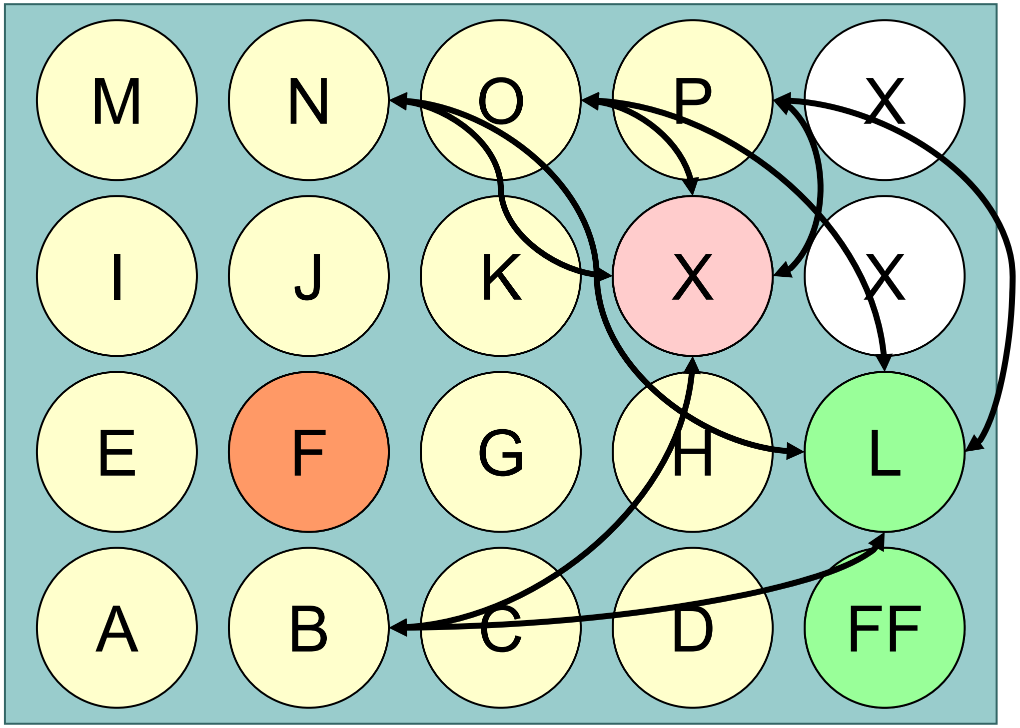
Connectivity

- Growth



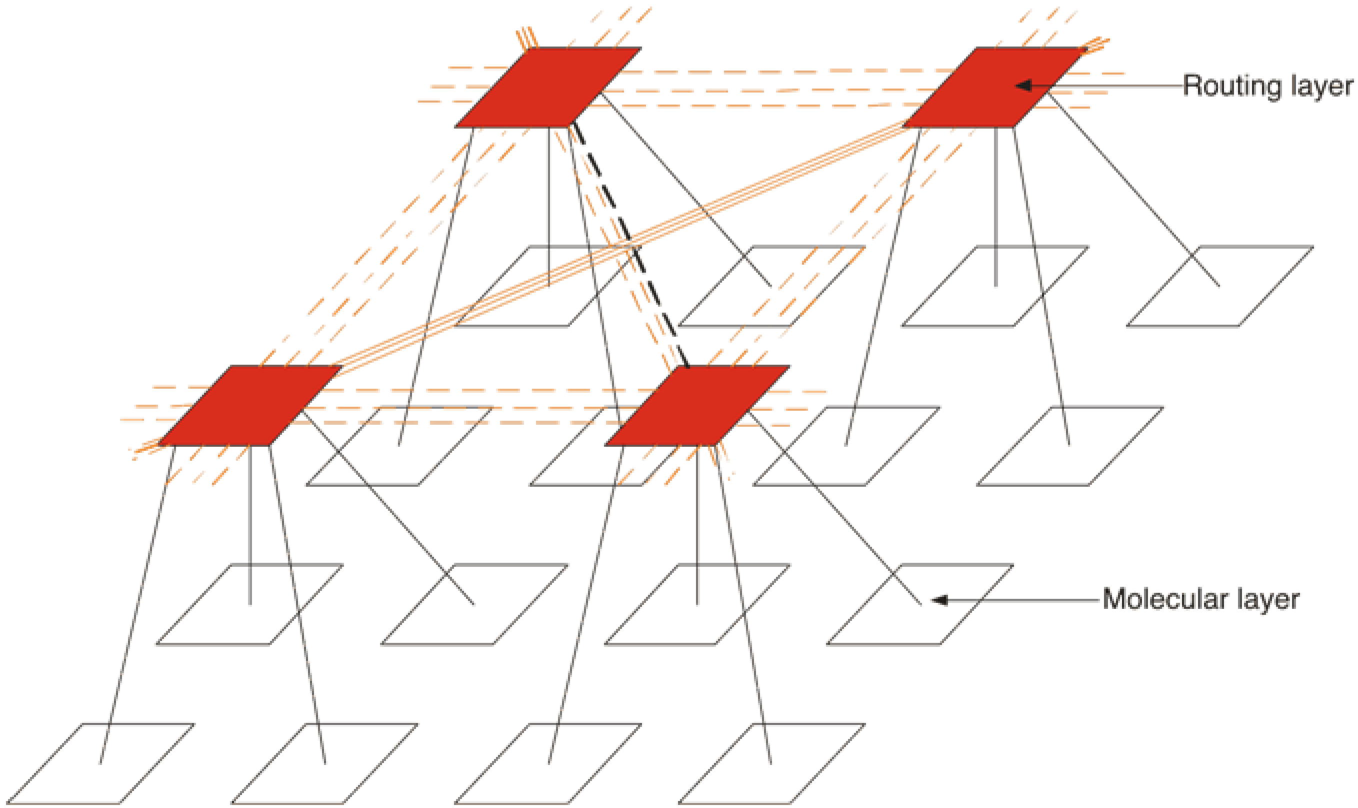
Connectivity

- Growth and fault tolerance



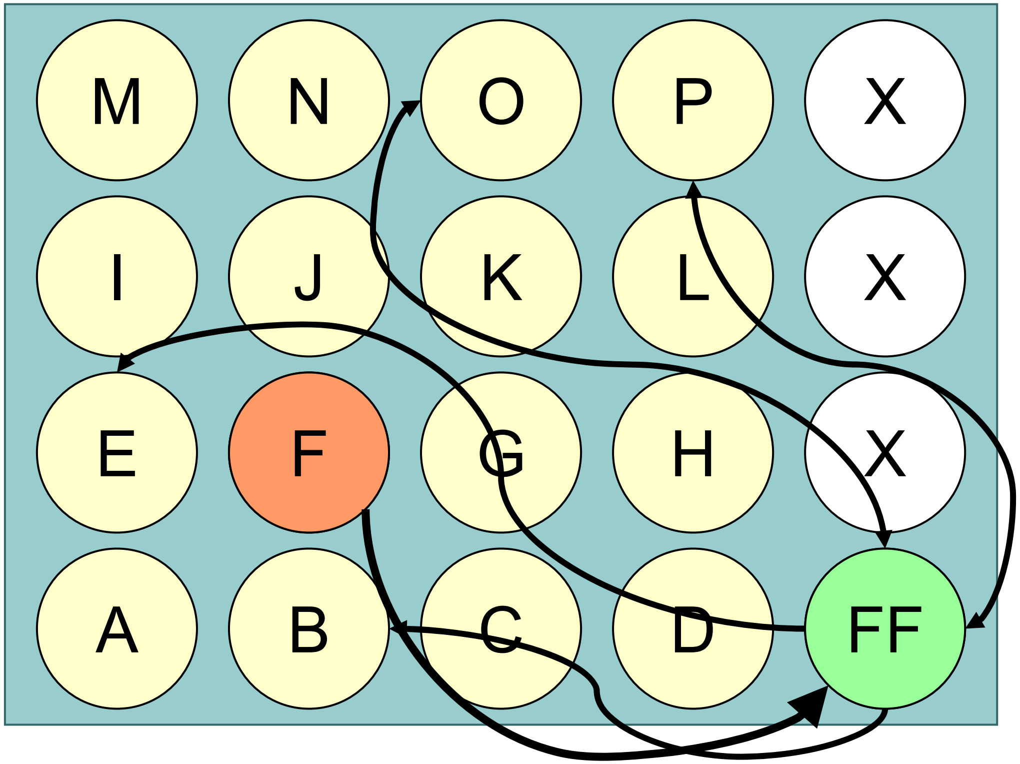
Connectivity

- The POEtic approach



Connectivity

- The POEtic approach



Connectivity

Connectivity

- So you CAN do it!

- But what is “it”?
  - How does a new cell know WHERE to connect?
  - Fault tolerance might be easier, but what about newly created cells?
  - How do they know the address of another cell? Do they need to know?
  - How do you foresee a sufficient number of I/O ports in your cell design?

Adaptive systems

- Unfortunately, that's where the <u>real</u> problems begin!!!
- Very practical issues:
  - Granularity – what is a "cell"?
  - Design – how to go from an application to a system like this?
  - Execution – how does it run?
  - Connectivity – how do the cells communicate?

How can you efficiently fit all this in a circuit?

Conclusions