SOFTWARE ARCHITECTURE
For MOLECTRONICS

John H. Reif
Computer Science Dept
Duke Univ.

In Collaboration with:
Allara, Hill, Reed, Seminario, Tour, Weiss

DARPA Moletronics Program
**BLACK BOX ARCHITECTURE:**

Moletronics *Processing Array*  
Contains basic logic gates

Surrounded by:  
*Reconfigurable Interconnects*

**INPUTs:**  
\[ X = \text{vector of } m \text{ data inputs} \]
\[ Y = \text{vector of } l \text{ control inputs} \]

**OUTPUT:**  
\[ O(X,Y) = \text{vector of } n \text{ outputs} \]

*Initial Conditions of System:*  
- random interconnects  
- meaningless output  
  for a given data input
KEY MOLECTRONICS SOFTWARE

DESIGN CONSTRAINTS:
♦ Moderate Number (say 1000) of Inputs

♦ Large Number $2^{1000}$ of Possible Single-bit Output Values

♦ Very Large Number $>10^{100}$ of Possible I/Os

♦ Extremely Large Number $2^{1000000}$ Possible Functions which can be I/O Combinations

♦ Very Large Percent of Defective Fabricated Components
  Key Technical Challenge: Need high fault tolerance

MOLECTRONICS PROGRAMMING FLEXIBILITY:
♦ Rewire interconnect topology using strong fields:
  Re-program undesired & inoperative outputs

♦ New Interconnect Routes Between Nanoparticles
  Use electrochemically induced crosslinking e.g., add pyrrole to form new wire interconnects

♦ Increasing Memory Capability:
  Attach During Assembly:
  Add More DRAM Elements:
  Semiconductor Nanoparticle/controllers

♦ Choosing Functional Elements:
  Use “burning out” approach
  Hard-wire specific Elements into Final System
GENERAL MOLECTRONICS SOFTWARE DESIGN CONCEPT:

SELF-RECONFIGURABILITY via TRAINING
(e.g., UCLA/HP Teramac)

[1] TRAIN MOLECTRONICS System to compute function $F(X)$
To Obtain Correct Output $F(X)$ for data input $X$
(e.g., a basic logic function)

Repeatedly:
Vary Control Inputs $Y$
until Stability $F(X) = O(X,Y)$ is achieved
at Control Input vector $Y_0$

[2] VERIFY Training:
verify $F(X) = O(X,Y_0)$
in absence of changes in $Y_0$

♦ MODULAR Functional Training:
  ➢ determine & separate key functional modules to be executed
  ➢ separately train & test modules

♦ ADVANTAGES: to give the correct $F(X)$
  ➢ No detailed system reconfiguration
  ➢ Need not know exact interconnect structure

Methods that may Speed Up Convergence:
➢ Evolutionary Programming Techniques
➢ Simulated Annealing Techniques
➢ Nested Annealing Techniques [Reif]
DYNAMIC ERROR RESILIENCY

♦ Key Software Problem:

PROGRAMMING a moletronic computer to do useful computations when:

➢ Use highly UNRELIABLE components

➢ Some components may be only PARTIALLY functional

♦ Coping With Dynamic Faults:
Programming Needs To Do:

➢ Efficient DETECTION of Faulty Components on an ongoing basis

➢ REPAIR Faults by Bypassing Faulty Components

ERROR-RESILIENT PROGRAMMING TECHNIQUES:

(1) Fault Resiliency Using REDUNDANCY [von Neumann 1950s]

(2) MODULAR Fault Resilient Software Architecture e.g., [Gacs,1989][Gacs,Reif,1990]

(3) Task RE-ASSIGNMENT e.g., [Kar, Nikolaou,Reif,1984]
ERROR-RESILIENT PROGRAMMING TECHNIQUE #1

Fault Resiliency Using REDUNDANCY
[von Neumann, 1950s]

- Transform Digital Circuit with Fault Components Using 3-way Redundancy
- Replicate Logical Components and use Majority Voting

![Diagram showing replication and majority voting](image-url)
ERROR-RESILIENT PROGRAMMING TECHNIQUE #2:

MODULAR Fault Resilient Software Architecture

[Gacs, 1989][Gacs, Reif, 1990]

- Use Hierarchical Structured Fault Detection and Correction
- Decision Making via Majority
ERROR-RESILIENT PROGRAMMING TECHNIQUE #3:

Task RE-ASSIGNMENT

[Kar, Nikolaou, Reif, 1984]

- Re-Mapping Algorithm Uses Decomposition of Task Network
- Re-Mapping to Sub-Network of Reliable Components
Additional Slides on
TESTING METHODOLOGY for MOLECTRONICS
TESTING METHODOLOGY for MOLECTRONICS:

♦ Testing System:
may be Symmetric MultiProcessors (SMPs)

♦ Interface to Testing System:

➢ During Assembly: via I/O leads

➢ After Assembly: via input & output wires

♦ Multiple Testing Stages in Fabrication & Assembly

♦ Components to be Tested: Individually & In Place
COMPONENT I/O Tests:
For each logical component:

- Test if usable truth table output obtained
- Cycle through subsets of inputs to determine truth tables

BUNDLED I/O Tests:

- Decreases Number of I/O testing Combinations
- Increases Likelihood of Overcoming Single Fault Locations

AGGREGATE Fault Testing on Subcircuits:

- May employ Sophisticated Software Routines developed for VLSI testing [Reif, 1993]
MEASURING and MODELING  
ANALOG RESPONSE

◊ 1st Year: using 2D probed configuration may use hybrid on-chip multiplexer

INITIAL VALIDATIONS: characterize & tune:
  ➢ electrode configuration,
  ➢ individual components,
  ➢ signal value/thresholds

[1] APPROXIMATE Numerical Parameters
  ➢ signal thresholds
  ➢ response curves

[2] Develop Software MODELS:
  ➢ for component performance

[3] TUNE MANUFACTURE of Components
  ➢ Goal: INCREASE YIELDs of working Components

[4] Use ITERATIVE REFINEMENT of Above

◊ In Later Years:
  ➢ Full 3D structure NOT always Accessible to Surface Probes
  ➢ Will Use Previously Developed Numerical Software Models