Defect Tolerance

Shawn Blanton
ECE Department
Carnegie Mellon University

NSF Workshop on NanoComputing
October 17, 2002

Background

Defect tolerance – why is it needed?

- Nano-systems composed of self-assembled devices are expected to have significant defect densities.
- Thus, the likelihood of producing defect-free nano-systems is extremely low.
- It will therefore be necessary to create methodologies that will allow nano-systems to operate in the presence of a multitude of defects.

Definitions

- A defect is a physical deformation.
- A defect does not necessarily have to affect the behavior of a nano device.
  - A defect has to be activated to cause an internal system error.
  - The internal error must lead to an external error to cause a system failure.
- A defect tolerant system is one that operates (sufficiently) in the presence of defects.

Traditional Approaches

- Hardware redundancy
- Software redundancy
- Time redundancy
- Information redundancy

Traditional Approaches cont'd

- Hardware redundancy via structural design
  - Self checking circuits
  - Arithmetic codes
- Hardware redundancy via duplication
  - NMR (space shuttle uses N=5)

Traditional Approaches cont'd

- Software Redundancy
  - use different programs/algorithms to compute the same task and compare results
- Time Redundancy
  - re-compute task and compare results (possibly using different hardware/software)
- Information Redundancy
  - backup information
  - use of ECC

Existing Approaches Fall Short

Hardware Redundancy

- Self-checking circuits, arithmetic codes, etc. typically assume a single error.
- Duplication assumes the duplicated sub-systems are initially defect free.

Existing Approaches Fall Short

Software Redundancy

- Using different programs or algorithms on the same highly-defective hardware simply won't help.
- Effective only if the defects are all transient or intermittent in nature.

Existing Approaches Fall Short

Time Redundancy

- Completely useless given the presence of permanent defects.

Existing Approaches Fall Short

Information Redundancy

- Similar to hardware redundancy, in that the nature of the likely errors is highly constrained.
- However, it is worth exploring if assumptions concerning errors can be relaxed.

Nano-Systems

- The fabricated nano-system will be highly defective.
- Nano-system defects are permanent.
  - Techniques based on "known-good" or "mostly-good" components won't help.
- Thus, the only possibility for success is to devise new defect-avoidance or defect-masking methodologies.

Defect Avoidance

- The strategy here is to identify and utilize the "good" devices.
  - Implies that the nano-system can be configured to utilize the good devices.
- Identification = Diagnosis
  - Testing the system is not sufficient.
  - The system devices must be diagnosed to identify which ones are (partially) good and which are bad.

Good Device Identification

- Like the assembly process, a self-diagnosis process will be necessary.
  - Tester cost and speed will make it impossible for the identification process to be directed off-chip.
- The expected device density requires that the self-diagnosis process be scalable.

Leveraging Past Work

- The fact that nano-systems are expected to be regular enables scalable self-diagnosis.
- Test and diagnosis algorithms for regular electronic arrays dates back to the early 1960s.
  - e.g. iterative logic arrays, programmable arrays, etc.
- These approaches are indeed scalable, some have constant complexity.