Computer Architectures for DNA Self-Assembled Nanoelectronics

Alvin R. Lebeck
Department of Computer Science
Duke University

Acknowledgements

People
- Students: Jaidev Patwardhan, Constantin Pistol, Vijeta Johri, Sung-Ha Park, Nathan Sadler, Niranjan Soundararajan, Ben Burnham, R. Curt Harting
- Chris Dwyer, Daniel J. Sorin, Thomas H. LaBean, Jie Liu, John H. Reif, Hao Yan
- Sean Washburn, Dorothy A. Erie (UNC)

Funding
- Air Force Research Lab
- National Science Foundation (ITR)
- Duke University Office of the Provost
- Equipment from IBM & Intel
Current Processor Designs

- Large Complex Systems (millions/billions of transistors)
- Mature technology (CMOS)
- Precise control of entire design and fabrication process
- Lithographic process to create smaller and smaller features.
  - But has limits...
- Cost of facility, high defect rates, process variation, etc.

The Red Brick Wall

- “Eventually, toward the end of the Roadmap or beyond, scaling of MOSFETs (transistors) will become ineffective and/or very costly, and advanced non-CMOS solutions will need to be implemented.” [International Technology Roadmap for Semiconductors, 2003 Edition, Difficult Challenge #10]
The Potential Solution

- Self-Assembled Nanoelectronics
- Self-assembly
  - Molecules self-organize into stable structures (nano)

What nanostructures?
What nanoelectronic devices?
How does self-assembly affect computer system design?

Outline

- Nanostructures & Components
- Circuit Design Issues
- Architectural Implications
- Proposed Architectures
- Defect Tolerance
- Conclusion
DNA Self-Assembly

- Well defined rules for base pair matching
  - Thermodynamics driven hybridization
- Can specify sequence of pairs, forms double helix
  - Synthetic DNA
  - Engineered Nanostructures
  - Inexpensive lab equipment

Adenine (A) ↔ (T) Thymine
Cytosine (C) ↔ (G) Guanine

Strands → Tiles → Structures

DNA-based Self-Assembly of Nanoscale Systems

- Use synthetic DNA as scaffolding for nanoelectronics
- Create circuits (nodes) using aperiodic patterning
  - Demonstrated aperiodic patterns with 20nm pitch
    [FNANO '05, Angewandte Chemie '06, DAC '06]
Nanoelectronic Components

- Many Choices / Challenges
  - Good Transistor Behavior
  - Interaction with DNA Lattice
- Crossed Nanotube Transistor
  [Fuhrer et al. '01]
- Demonstrated Functionalization of Tube Ends
  [Dwyer, et al. '02]
- Other candidates: Ring-gated, Crossed Nanorod, Crossed Carbon Nanotube FETs

Circuit Design Issues

Goal
Construct a computing system using the DNA Lattice and nanoelectronic components.

Proposal
Use DNA tags (sticky-ends) to place nano-components on lattice

1. Regularity of DNA Lattice
   - Easy to replicate simple structures on a moderate scale
2. Complexity of Digital Circuits
   - Large Graph with many unique nodes and edges
3. Tolerating Defects
   - Single-stranded DNA for tags (sticky-ends) may have partial matches (must minimize number of unique tags)
   - Nanotubes may not work as advertised
Balancing Regularity & Complexity

- Array of simple objects
- Unit Cell based on lattice cavity
  - Uniform length nanotubes
  - Minimizes # of DNA Tags => reduces probability of partial match
  - 20nm x 20nm
- Two levels of interconnect
- Complex circuits on single lattice (10K FETS)
- Envision ~9µm² node size: ~10,000 FETs + interconnect
- How to get billions or more?

Self-Assembled System

- Self-assemble ~ $10^9$ - $10^{12}$ simple nodes (~10K FETs)
- Potential: Tera to Peta-scale computing
- Random Graph of Small Scale Nodes
  - There will be defects
  - Scaled CMOS may look similar
- How do we perform useful computation?
Outline

- Nanostructures & Components
- Circuit Design Issues
- Architectural Implications
- Proposed Architectures
- Defect Tolerance
- Conclusion

Implications of Small Nodes

- Node: DNA Grid FETs
  - 3µm x 3µm node
  - Carbon nanotube [Dwyer '02]
  - Ring Gated [Skinner '05]
- Small Scale Control
  - Controlled complexity only within one node
- Limited space on each node
  - Simple circuits (e.g., full adder)
- Limited communication between nodes
  - only 4 neighbors
  - No global (long haul) interconnect
- Limited coordination
  - Difficult to get many nodes to work together (e.g., 64-bit adder)
Implications of Randomness

- Self-assemble interconnect of nodes
  1. Random node placement
  2. Random node orientation
  3. Random connectivity
  4. High defect rates (assume fail stop node)

- Limitations -> architectural challenges

Architectural Challenges

- Node Design
- Utilizing Multiple Nodes
  - Each node is very simple
- Routing
- Execution Model
  - Must overcome implementation constraints
- Instruction Set
- Micro-scale Interface
Outline

- Nanostructures & Components
- Circuit Design Issues
- Architectural Implications
- Proposed Architectures
  - Defect Isolation & Structure
  - NANA [JETC '06]
  - SOSA [ASPLOS '06]
- Defect Tolerance
- Conclusion

Nano-scale Active Network Architecture

System View
- Large-scale fabrication ($10^{12}$ nodes, $10^8$ cells)
- Via provides micro-scale interface, Multiple Node Types
- First Cut: Understand issues
**Defect Isolation/Structure**

- Grid w/ Defects $\rightarrow$ Random Graph
- Reverse path forwarding [Dalal '78]
- Broadcast on all links except input [Nanoarch '05]
  - Forward broadcast if not seen before
  - Implement fail-stop nodes [Nanoarch '06]
- RPF maps out defective regions
  - No external defect map
  - Can tolerate up to 30% defective nodes
- Distributed algorithm to create spanning tree
- Route packets along tree
  - Up*/down*
  - Depth first
- How do we compute?

**NANA: Computing on a Random Graph**

- Perform 3 operations: Add, Add, Multiply
- Search along path for correct blocks to perform function
- Execution packets carry operation and values
- Proof-of-concept simulations
**NANA: Execution Model & ISA**

- Accumulator based ISA
- Carry data and instructions in a “packet”
- Use bit-serial processing elements
  - Each element operates on one bit at a time
  - Minimize inter-bit communication

```
|--------|------|------|------|---------------------|---------------------|------|-----------------|-----------------|------|
```

---

**NANA: System Overview**

- Simple programs
  - Fibonacci
  - String compare
- Utilization is low
- Divide $10^{12}$ nodes into $10^6$ cells
- Peak performance potentially higher than IBM Blue Gene and NEC Earth Simulator
- Need to use more nodes!

---

© 2006 A. R. Lebeck

Duke Computer Architecture

21

© 2006 A. R. Lebeck

Duke Computer Architecture

22
Self-Assembled System

- Self-assemble ~ $10^9$ - $10^{12}$ simple nodes (~10K FETs)
- Potential: Tera to Peta-scale computing
- Random Graph of Small Scale Nodes
  - There will be defects
  - Scaled CMOS may look similar
- How do we perform useful computation?

Node Interconnect

- Group many nodes into a SIMD PE
- PEs connected in logic ring
- Familiar data parallel programming

Self-Organizing SIMD Architecture (SOSA)

- Nodes Grouped to form SIMD Processing Element (PE)
  - Head, Tail, N computation nodes (k-wide bit-slice of PE)
- Configuration: Depth First Traversal of Spanning Tree
  - Orders nodes within PE (Head $\rightarrow$ LSB $\rightarrow$ ... $\rightarrow$ MSB $\rightarrow$ Tail)
  - Orders PEs
- Many SIMD PEs on logical ring
  $\rightarrow$ familiar data parallel programming abstraction
SOSA: Instruction Broadcast

- Instructions broadcast to all nodes
- Instructions decomposed into three “microinstructions” (opcode, registers, synch)
- Can reach nodes/PEs at different times (5 before 9)

SOSA: Instruction Execution

- Instructions execute asynchronously within/across PEs
- XOR parallel within PE vs. Addition serial within PE
- ISA: Three register operand, predication, optimizations, see paper for details…
Two System Configurations

- One Large System
- Latency
- Space sharing
- Multiple “cells”
- Throughput

Outline

- Nanostructures & Components
- Circuit Design Issues
- Architectural Implications
- Proposed Architectures
  - SOSA [ASPLOS ’06]
  - Node Design
  - Evaluation
- Defect Tolerance
- Conclusion
SOSA Node

- Homogeneous Nodes
  - Specialized during configuration
- Asynchronous Logic
- Communication
  - 4 transceivers (4 phase handshake)
  - 3 virtual channels (inst bcast, ring left & right)
- Computation
  - ALU
  - Register (32-bits: 32x1 or 16x2)
  - Inst Buffer
- Configuration
  - Route Setup
- Subcomponent BIST
  [nanoarch '06]

© 2006 A. R. Lebeck
Duke Computer Architecture

SOSA Node

- VHDL
  - ~10K FETs
- Area \( \approx 9\mu m^2 \)
  - Custom layout tools for standard cells
- Power \( \approx 6.5 \text{ W/cm}^2 \)
  - Semi-empirical spice model [IEEE Nano '04]
  - 1ns switching time
  - 88% devices active
  - 0.775\( \mu \text{W} \)/node
- Modern proc > 75 W/cm\(^2\)

© 2006 A. R. Lebeck
Duke Computer Architecture
Evaluation Methodology

- Custom event simulator
  - Conservative 1ns time quantum (switching time)
  - 2 bits per node (16 registers, 16 + 2 for 32-bit PE)
- Nine benchmarks
  - Integer code only – no hardware support for floating point
  - Matrix multiplication, image filters (gaussian, generic, median), encryption (TEA, XTEA), sort, search, bin-packing
- Compare performance to four other architectures
  - Pentium 4 (P4) (real hardware)
  - Ideal out-of-order superscalar (I-SS) 10GHz, 128-wide, 8K ROB
  - Ideal Chip Multiprocessor (I-CMP) 16-way ideal
  - Ideal SOSA (I-SOSA) no communication overhead, unit inst latency
  - Extrapolate for large SOSA systems (back validate)

Matrix Multiply (Execution Time)

- Hand optimizations (loop unrolling, etc.)
- Better scalability than other systems (crossover < 1000)
- Still room for improvement
TEA Encryption (Throughput)

- Used in XBOX
- shift, add and xor
- 64 bit data blocks
- 128-bit key
- Pipelined on 64 PEs
- Configure Multiple Cells of 64 PEs
- Single Cell poor
- 200X better than P4 in same area

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Encryptions/sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>P4 @ 3 GHz (100mm²)</td>
<td>3.9 M/sec</td>
</tr>
<tr>
<td>I-SS</td>
<td>73.62 M/sec</td>
</tr>
<tr>
<td>16-CMP</td>
<td>1180 M/sec</td>
</tr>
<tr>
<td>SOSA (1 cell ~ 0.019mm²)</td>
<td>0.175 M/sec</td>
</tr>
<tr>
<td>I-SOSA (1 cell)</td>
<td>27.7 M/sec</td>
</tr>
<tr>
<td>SOSA (5400 cells, 100mm²)</td>
<td>940 M/sec</td>
</tr>
<tr>
<td>I-SOSA (5400 cells)</td>
<td>72300 M/sec</td>
</tr>
</tbody>
</table>

Outline

- Nanostructures & Components
- Circuit Design Issues
- Architectural Implications
- Proposed Architectures
- Defect Tolerance (not transient faults)
- Conclusion
Defect Tolerance

- Simple Fail Stop model
- Encryption gracefully degrades
- MXM < 10% degradation up to 20% defective nodes

Node Failure Modes [Nanoarch '06]

- Exploit modular node design
  - VHDL BIST for communication & configuration (all stuck-at faults)
  - Assume software test for compute logic
- Configuration logic is critical

© 2006 A. R. Lebeck
Evaluation

- Simple node model in C
- Model network with 10,000 nodes
- Vary transistor defect probability from 0%-0.1%
  - Map defective transistors to defective components
- Average 500 runs per data point
- How much do we benefit by node modularity?
  - What device defect probability can it handle?

Results: Usable Nodes

- Hybrid failure mode can tolerate a higher device failure probability
  - Three orders of magnitude greater than typical CMOS designs (10^{-4} vs. 10^{-7})
• Hybrid increases the number of reachable nodes
  – More nodes with functioning compute logic reachable and usable

Fail-Stop Summary

• Test logic detects defects in node components
• Modular node design enables partial node operation
• Node is useful if
  – It can compute
  OR
  – It can improve system connectivity
• Hybrid failure mode increases available nodes
  – Can help tolerate a device failure probability of $1.5 \times 10^{-4}$ (1000 times greater than typical CMOS designs)
**SOSA Summary**

- Distributed algorithm for structure & defect tolerance
  - No external defect map
- Configuration groups nodes into SIMD PEs
- High utilization w/ familiar programming model
- Ability to reconfigure
  - One system for latency critical systems
  - Multiple cells for throughput systems
- Limitations: I/O bandwidth, general purpose codes, FP, transient faults

---

**Conclusion**

- Future limits on traditional CMOS scaling
  - Multicore, etc. -> tera/peta scale w/ 1M nodes
- Defects, cost of fabrication, process variation, etc.
- High performance, low power despite randomness and defects
Duke Nanosystems Overview

DNA Self-Assembly

Nano Devices
Electronic, optical, etc.
[Nanoelects 2006]

Circuit Architecture
[FNANO 2004]

Logical Structure & Defect Isolation
[NANOARCH 2005]

NANA - General Purpose Architecture
[JETC 2006]

Large Scale Interconnection
[NANONETS 2006]

SOSA - Data Parallel Architecture
[NANOARCH 2006, ASPLOS 2006]

Generic Filter (Execution Time)

- 3x3 generic filter (Gaussian & Median similar)

© 2006 A. R. Lebeck
Circuit Architecture

- Unit Cell based on lattice cavity
  - Place uniform length nanoelectronic devices
  - Reduces probability of partial matches
  - Two layers of interconnect
- Achieve balance between
  - Regularity of DNA lattice
  - Complexity required for circuits
  - Defect Tolerance
- Node: DNA Lattice with CNFETs

Fail-Stop Transceivers

- Minimize test overhead
  - Reuse node hardware during test
- Hardware Test
  - Send '0' and '1' in a loop
  - If data returns, enable component
  - If data does not return, component remains disabled
- Similar principle for configuration logic
- Modular design enables graceful degradation

© 2006 A. R. Lebeck
Duke Computer Architecture