Chapter 17

Testing Aspects of Nanotechnology Trends
What is this chapter about?

- Introducing testing methodologies for emerging devices
- Providing examples of emerging technologies
  - Resonant tunneling diodes
  - Quantum cellular automata
  - Crossbar array architectures
  - Carbon nanotubes
- Providing testing frameworks/architectures for these devices
Organization

- Introduction
- Resonant Tunneling Diodes and Quantum-Dot Cellular Automata
- Crossbar Array Architectures
- Concluding Remarks
Challenges: Past, Present, Future

- With the downscaling of CMOS we see many challenges in the near future:
  - Developing new materials (e.g., high/low κ dialectrics)
  - New device geometries (e.g., dual-gate, finFETs)
  - Managing power dissipation
  - Economics of commodity manufacturing

- What about longer-term challenges?
  - Development of hybrid systems (CMOS + alternative devices)
  - Dealing with design complexity from integrating $10^{12}$ devices on a chip
Future Challenges (Cont’d)

- What are these “alternative devices”
  - Resonant tunneling diodes
  - Single electron transistors
  - Quantum-dot devices
  - Molecular switches
  - Nanowire devices
  - Carbon nanotube transistors and interconnects

- At this point, it is not clear which emerging hybrid technology will eventually enter production
What Do We Know?

- Future nanoscale devices will have high rates of manufacturing defects.

- Supply voltages will continue to scale down…
  - 0.5V or maybe down to 0.3V
  - Reduced power
  - Reduced noise margins

- Nanoscale manufacturing and reduced noise margins exposes computation to higher error rates.
Defect and Fault Tolerance

- Fact: nanoscale circuits will have higher rates of faults and defects

- Defect tolerance: improve yield
  - Manufacturing defects

- Fault tolerance: improve reliability
  - Runtime (lifetime operation) defects

- Thorough test and high resolution diagnosis
  - Essential for the implementation of defect and fault tolerant schemes
What About the Economics?

- How economically practical will it be to manufacture devices at such a small scale using current fabrication techniques?
- May no longer be feasible to demand precise control over arbitrary structures
- Regular layouts (e.g., array structures) or self-assembly may be more cost effective.

Schmid & Leblebici
Resonant Tunneling Diodes (RTDs)

- Monostable-bistable transition element (MOBILE)
- Implement linear threshold gates – known to reduce circuit logic depth
- If $\Delta I - I_T$ positive, $f$ is charged to logic ‘1’, else discharged to logic ‘0’
Threshold Logic in a Nutshell

- **Definition:**
  \[
  f(x_1, x_2, \ldots, x_n) = \begin{cases} 
  1 & \text{if } \sum_{i=1}^{n} w_i x_i \geq T \\
  0 & \text{if } \sum_{i=1}^{n} w_i x_i < T 
  \end{cases}
  \]

- \(x_i\): binary inputs, \(w_i\): weight of input \(x_i\), \(T\): threshold of \(f\)

- Threshold function – A Boolean function realized by a single threshold gate

- Not every Boolean function, \(g\), is a threshold function

- Determining if \(g\) is a threshold function is an active area of research.
Testing of RTDs

- **Goal**: Develop an automatic test pattern generation framework for combinational threshold circuits
- Fault modeling
- Fault simulation
- Test generation
  - Conditions for fault excitement
  - Calculating $D$-cubes (fault propagation) and singular covers (fault justification)
- Fault Collapsing
- Relating irredundant combinational threshold circuits and test generation
- Model cuts and shorts in a MOBILE
- Equate fault to the single stuck-at fault (SSF) fault model at logic-level
  - Cut on site 2 behaves as input SA0
  - Short on site 7 behaves as output SA1
Fault Simulation

- Any existing Boolean fault simulation paradigm can be used
  - Parallel fault simulation
  - Parallel-pattern single-fault propagation
  - And others …

- How to simulate effect of SA0/SA1 fault in a threshold gate?
  - $x_i$ SA0: Set $w_i = 0$
  - $x_i$ SA1: Set $T = T - w_i$
Theorem 1: Given a threshold function, \( f(x_1, x_2, \ldots, x_n) \), find an assignment on the input variables (excluding \( x_i \)) such that one of the following inequalities is satisfied:

\[
T - w_i \leq \sum_{j=1, j \neq i}^{n} w_j x_j < T \\

T \leq \sum_{j=1, j \neq i}^{n} w_j x_j < T - w_i
\]

- If an assignment exists, then \( (x_1, x_2, \ldots, x_i = 1(0), \ldots, x_n) \) is a test vector for \( x_i \) SA0 (SA1). Otherwise both faults are untestable and redundant.
Test Generation: An Example

One of the following inequalities needs to be satisfied:

\[ 1 \leq \sum_{j=2}^{3} w_j x_j < 3 \quad \text{or} \quad 3 \leq \sum_{j=2}^{3} w_j x_j < 1 \]

- Second inequality not satisfied
- First inequality satisfied and leads to three test vectors for \( x_i \) SA0: 101, 110, and 111
- Test vectors for \( x_i \) SA1 are then 001, 010, and 011
Theorem 2: Given a threshold function, \( f(x_1, x_2, \ldots, x_n) \), if there exists two (or more) inputs \( x_j \) and \( x_k \) such that \( |w_j| = |w_k| \), then test vector to detect \( x_k \) SA0/SA1 can be obtained simply by interchanging the bit positions of \( x_j \) and \( x_k \) in the SA0/SA1 test vectors for \( x_j \), assuming they exist.

Since 110 is a test vector for \( x_2 \) SA0 and \( w_2 = w_3 \), 101 is a test vector for \( x_3 \) SA0.
Fault Propagation and Justification

- D-notation: $0/0=0$, $1/1=1$, $1/0=D$, $0/1=D'$
- Propagation $D$-cubes needed to propagate fault from fault site to primary output
  - Obtained by substituting $D$ or $D'$ in the logic function. For fault to propagate, only cubes that contain $D$ or $D'$ must be activated
- Singular covers used to justify the output values on threshold gates
  - Easily obtained from the cover of a function
Test Generation and Irredundancy

- In Boolean testing, test generation and irredundancy are intricately related
  - If a fault is untestable, then the combinational circuit is redundant and can be simplified
- A similar relationship holds in combinational threshold logic

Original threshold network
System-on-Chip Test Architectures

k SA0 is redundant
Ch. 17 – Testing Aspects of Nanotechnology Trends - P. 18
Fault Collapsing

- To reduce test generation time, need to target fewer faults: fault equivalence and fault dominance

- **Theorem 3**: Given a threshold function, \(f(x_1, x_2, \ldots, x_n)\), if there exists an input \(x_i\) such that \(w_i = T\) and if all other inputs \(x_j\) have positive weights \(w_j\) (i.e., \(w_j > 0\)), then \(x_i\) SA1 is equivalent to \(f\) SA1

- **Theorem 4**: Given a threshold function, \(f(x_1, x_2, \ldots, x_n)\), either an output \(f\) SA0 (SA1) dominates an \(x_i\) SA0 (SA1) or an output \(f\) SA1 (SA0) dominates an \(x_i\) SA0 (SA1)

- Overall conclusion is that output faults dominate input faults and need not be tested
Fault Collapsing (Cont’d)

- Given the fault collapsing results of the previous slide, we have the following theorem for threshold logic testing
  - Analogous to what exists in Boolean testing

- **Theorem 5**: In an irredundant combinational threshold circuit, $G$, any test, $V$, that detects all single stuck-at faults (SSFs) on the primary inputs and fanout branches detects all SSFs in $G$
Quantum Cellular Automata (QCA)

- Four quantum dots arranged in a square
- Two electrons – due to Coulumbic interactions, they occupy opposite corners
- Leads to a binary system
QCA Gates

- Basic gate is the three-input majority gate: 
  \[ M(A, B, C) = AB + AC + BC \]
  - Can be reduced to two-input AND/OR gate
- Majority gate and inverter are functionally complete
QCA Defects

- Types
  - Displacement
  - Misalignment
  - Omission

- Defects result in different logic values in the fault-free (defect-free) and faulty (defective) cases
QCA Defects (Cont’d)

- Interconnect defect modeled using the dominant fault model
- Output of dominated wire is determined by logic value on the dominant wire
- Output equals dominant wire’s logic or its compliment
  - Depends on which cell is displacement and the displacement distance, $\Delta$

System-on-Chip Test Architectures Ch. 17 – Testing Aspects of Nanotechnology Trends - P. 24
Testing of QCA

- **Goal**: Develop an automatic test pattern generation framework for QCA circuits
  - Target QCA defects in a majority gate
  - Target QCA defects on interconnects
- Ensure each gate receives a complete test vector set that detects *all* detectable defects.
Motivational Example

Need to target 14 single stuck-at faults (SSFs) to detect all SSFs

- one possible complete test set shown

System-on-Chip Test Architectures

Ch. 17 – Testing Aspects of Nanotechnology Trends - P. 26
Motivational Example (Cont’d)

- $M_1$ receives input vectors 000, 001, 011, 100, 110
- Though the vectors are a complete SSF test set, they are not a complete test set for detecting all simulated defects
- If we add 0100110 to original test set, then $M_1$ also receives 010 – now a complete defect test set
Motivational Example (Cont’d)

- Consider bridging fault between inputs \( C \) and \( D \)
- \( C \) dominates \( D \) but unknown whether \( D=C \) or \( D=C' \) in faulty case
- Need vector to detect two of four possible conditions
  - \( D \ SA0 \) w/ \( C=0 \) or \( D \ SA1 \) w/ \( C=1 \)
  - \( D \ SA1 \) w/ \( C=0 \) or \( D \ SA0 \) w/ \( C=1 \)
Motivational Example (Cont’d)

- First (second) condition tested with 0001011 (0010011) that are already in test set
- No vector in test set for third or fourth condition
- 0000011 and 0011011 detect third and fourth conditions, respectively
- Example shows that a SSF test set cannot guarantee detections of all simulated defects in QCA circuits. Additional test generation may be required
Targeting Defects in Majority Gate

- Use CNF of a majority gate during SAT-based ATPG
  \[ M_{CNF} = (A' + B' + F)(A' + C' + F)(B' + C' + F) \]
  \[ (A + B + F')(A + C + F')(B + C + F') \]
- Has nine minimal SSF test sets, containing four vectors each
- Three test sets cannot detect all simulated defects
Defects in Majority Gate (Cont’d)

- $M$ currently does not receive a complete defect test set. Can make it complete if it receives 001 or 110 as an input vector as well.
- Must ensure effect of vector gets propagated to output.
- Condition given to SAT-based ATPG: $FSA_1$ with $A=0 B=0 C=1$ OR $FSA_0$ with $A=1 B=1 C=0$.

Scenario: $M$ receives 010, 011, 100, 101

Fault to test: $FSA_1$ with $A=0 B=0 C=1$ OR $FSA_0$ with $A=1 B=1 C=0$.
### Targeting Defects on Interconnects

#### Scenario 1: $\Delta$ is such that $B = A$

<table>
<thead>
<tr>
<th>Fault-free $A \ B$</th>
<th>Faulty $A \ B$</th>
<th>Equivalent Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0</td>
<td>0 0</td>
<td>-</td>
</tr>
<tr>
<td>0 1</td>
<td>0 0</td>
<td>$B$ SA0 with $A=0$</td>
</tr>
<tr>
<td>1 0</td>
<td>1 1</td>
<td>$B$ SA1 with $A=1$</td>
</tr>
<tr>
<td>1 1</td>
<td>1 1</td>
<td>-</td>
</tr>
</tbody>
</table>

Note: Assumption is that $B$ is dominated by $A$.

#### Scenario 2: $\Delta$ is such that $B = A'$

<table>
<thead>
<tr>
<th>Fault-free $A \ B$</th>
<th>Faulty $A \ B$</th>
<th>Equivalent Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0</td>
<td>0 1</td>
<td>$B$ SA1 with $A=0$</td>
</tr>
<tr>
<td>0 1</td>
<td>0 1</td>
<td>-</td>
</tr>
<tr>
<td>1 0</td>
<td>1 0</td>
<td>-</td>
</tr>
<tr>
<td>1 1</td>
<td>1 0</td>
<td>$B$ SA0 with $A=1$</td>
</tr>
</tbody>
</table>

Note: Assumption is that $B$ is dominated by $A$.

- Scenario that occurs depends on displacement distance $\Delta$ and/or number of cells displaced
- In reality, which scenario occurred will not be known
- Need to have vectors that detect both scenarios
Defects on Interconnects (Cont’d)

Scenario: $M$ receives 010, 011, 100, 101

Fault to test: $F$ SA1 with $A=0 \ B=0 \ C=1$ OR
$F$ SA0 with $A=1 \ B=1 \ C=0$

- Assume bridging fault between inputs $A$ and $B$. Not known if defective cell is on wire $A$ or wire $B$. Also $\triangle$ is unknown
- Need to test for four conditions: two result from $A$ dominating $B$ and two result from $B$ to $A$
Defects on Interconnects (Cont’d)

B SA0 with A=0 or B SA1 with A=1
B SA1 with A=0 or B SA0 with A=1
A SA0 with B=0 or A SA1 with B=1
A SA1 with B=0 or A SA0 with B=1

- If test set contains vectors that can satisfy above vectors, then bridging fault is completely tested
- Assume QCA layout of circuits is available a priori to get bridging fault list
- Otherwise with n lines, there are n(n-1) possible bridging faults involving two wires
  - 2n(n-1) conditions then need to be satisfied. But 50% of these conditions are already satisfied given any complete SSF test set!
Top-Down Circuit Design

>100 steps are required before final circuit is complete

Complete control over circuit placement and functionality

- oxidation
- optical mask
- stepper exposure
- photoresist coating
- photoresist removal (ashing)
- process step
- acid etch
- spin, rinse, dry

System-on-Chip Test Architectures Ch. 17 – Testing Aspects of Nanotechnology Trends - P. 35
**Bottom-Up Fabrication**

- Use bottom-up assembly as an alternative to top-down
  - Rely on self-assembly for defining device characteristics
  - Easier (less costly) fabrication process.
  - Requires fabrication regularity
    - Lends itself more easily to a *reconfigurable* architecture

**BUT...**

- This creates new challenges:
  - Can no longer arbitrarily determine device/wire placement.
    - Leads to higher defect rates
  - Fabrication may be restricted to simpler (less robust) structures
    - *e.g.*, 2-terminal vs. 3-terminal devices
Crossbar Configuration

bistable junction
Wired-OR Logic

- We can use the array to program OR logic
- All the crosspoints programmed into the “ON” (low resistance) state can be logically OR’ed together
  - Each row output NW serves as a wired-OR for all of the column inputs programmed into the “ON” state.
  - Assume non-driven NWs are pulled down to Ground
  - ON devices pull output NWs to high voltage

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ A + C \]

\[ C + D \]
The Hybrid Approach

- Without the fine control of individual devices, defects will become more widespread
  - Defect rates will be orders of magnitude higher than what we see today
  - Some systematic means of defect/fault tolerance will be needed

- What about a hybrid top-down/bottom-up approach?
  - Self-assembly structures are add-ons to CMOS subsystem
  - Nanoscale circuitry is interconnected via CMOS circuitry.
  - CMOS subsystem can serve as a reliable medium for connecting nanoscale blocks and other computation.

- Following are a few examples of hybrid systems...
nanoPLA (DeHon et al.)

- Programmable interconnect architecture built from hybrid components.

- nanoPLAs are the main nanocircuitry building block
  - Built from a crossed set of N-type and P-type nanowires
  - The nanoPLA is composed of 2 stages:
    - Stage 1: create product terms (ptems) with wired-OR logic fed into inverter (logical NOR)
    - Stage 2: generate outputs of PLA by taking wired-OR of selected pterms and inverting
  - NOR-NOR logic $\rightarrow$ AND-OR logic similar to PLA
  - nanoPLA also serves as switching block for routing
nanoPLA Block

from DeHon, FPGA’04

System-on-Chip Test Architectures Ch. 17 – Testing Aspects of Nanotechnology Trends - P. 41
**CMOL: Molecular CMOS** *(Likharev et al.)*

- CMOL is designed using the same crossbar array structure as the nanoPLA.
- The main difference is in how the CMOS/nanodevices are interfaced.
- Angling the nanoarray avoids the need for precise alignment.

*From Ma et al., Trans on Nano, 2005*
Applications of CMOL

- **Memory:**
  - Estimate 1 Terabits of data could be stored on CMOL based memory chip about 2 X 2 cm²

- **FPGA-like Architectures:**
  - Configurable logic block (CLB) composed of inverter and 2 pass transistors connected to nanowire crossbar
  - Program when inverter is disabled
  - Pass transistors are pull-up resistors
  - Wired-OR gates formed within CMOS cell
  - Arbitrary Boolean function formed by connecting 2 or more CMOS cells
Mapping a 32-bit Adder in CMOL

- Many nanodevices per CMOS cell
- Gates with high fanin and/or high fanout can be formed
- Extra devices can act as “spares” for reconfiguring around defects
- Simulations show 99% yield for a 32-bit adder even when ~20% of devices are defective

black dots = defects, green circles = used devices, blue, red circles = interface pins
What Do We Need to Support Reconfiguration?

- The previous examples show architectures built from reconfigurable fabrics
  - Reconfigurable molecular switches provide static fault tolerance for these devices.
- What about testability and an efficient means of enabling the reconfiguration?
  - Previous examples assume some defect detection mechanism as a preprocessing step that creates and stores a defect map
  - Locating all faults is very time consuming
  - Storing the defect map may require large area overheads.

Can we eliminate the requirement of storing the defect map?
**BIST and Defect Tolerance for Crossbar-based Architectures**

- High defect rates, a common problem among nanoscale devices
- Efficient test and defect tolerance methods are required
- Fault tolerance techniques for soft and transient errors are needed
- Defect Avoidance:
  - Avoiding defects during configuration of the device
  - Needs defect location information (i.e., Defect Map)
  - Identifying defect location for such large architectures is very challenging and time consuming
Defect Map-based Method Challenges

- There will be different defect maps for different chips
- Very high defect rates increase the defect map size
- Storing defect map on chip seems to be infeasible
- Defect map should be updated for post manufacturing defects
- New test methods and dynamic defect tolerance schemes are required to alleviate the above challenges
Limits of Static Reconfiguration

- Reliable computation requires systems to be resilient to both static errors and transient faults.
- Static reconfiguration most appropriate for static defects.
  - nanoPLA, CMOL only designed to handle static faults
  - Simultaneous configuration and test approach avoids defect map but still focused on static faults

What about dynamic faults?

- Allow for dynamic reconfiguration
- Use redundant execution
- Compute in the presence of faults
Simultaneous Configuration and Test (SCT)

- SCT is a defect avoidance mechanism that does not require a defect map
- Performs functional testing
  - Fault models are not yet well defined
- Minimizes test time while employing minimum test hardware

Architecture:
- Island style array of crossbars
- Reliable CMOS scale interconnects assumed
- CMOS BIST circuit used

- SCT can be extended to routing and interconnect resources
Components of BIST Circuit

- $m$-input configurable look-up table (LUT)
- $m$-bit counter and $i$-bit Comparator
SCT Procedure

I. **Partitioning** (into $m$-input functions)

II. **Place & Route** (on block array)

III. **SCT:**
   a. Configure and test each $m$-input function using functional test method
   b. Store failing blocks and their functions

IV. **Defect Avoidance:** Having list of available blocks, failed blocks and unimplemented functions, return to II. Repeat the procedure until all $m$-bit functions are configured on nanoblocks (i.e., LUTs).
**SCT (Step III):**

- Configure function $f_i$ on crossbar block (LUT)
- Configure function $f_i$ on LUT of BIST
- Configure the paths
- Run the m-bit counter ($2^m$ patterns are applied)
- Compare the results
Parallel SCT

- BIST circuit is very small
- SCT run time reduces when using multiple BIST circuits
- Functions can be configured and tested in parallel
- It requires programming devices with higher bandwidth
- Pass/Fail information obtained for several blocks in parallel
**Parallel SCT**

- Several BIST circuits (N)
- Configure and test N functions in parallel
- Failed blocks will be avoided in next phase
CNFET Transistor Layout

- Ideal CNFET circuits vs. 32nm CMOS
  - 13X EDP gain [Deng et al., ISSCC 2007]
Perfect CNFET Inverter Layout

- **P+ doped Semiconducting CNTs**
- **N+ doped Semiconducting CNTs**
- **Vdd**
- **Gnd**
- **64nm = 4\(\lambda\)**
- **4nm**

**Input**

**Gates**

**Output**

**PFET**

**NFET**
Carbon Nanotube FET Circuits: Barriers

- **Device level**
  - Excellent CV/I
  - VLSI process compatible
  - Ultra-small

- **Circuit level**
  - Misaligned CNTs
  - Metallic CNTs
  - CNT density

*Imperfection-Immune Circuits Required*
CNFET Imperfections: Misaligned CNTs

Wanted: \( A' + B' \)
Got: Short

Wanted: \( A'C' + B'D' \)
Got: \( A'C' + B'D' + A'D' \)

Wanted: \( (A+C)(B+D) \)
Got: \( (B+D) \)
**Misaligned-CNT-Immune NAND Design**

1. Grow CNTs

2. Define gates and contacts

3. Chemically dope P-type region

4. Chemically dope N-type region

5. Etch

- **Formal Approach generalized for arbitrary functions**

System-on-Chip Test Architectures  Ch. 17 – Testing Aspects of Nanotechnology Trends - P. 59
Misaligned-CNT-Immune NAND Design

1. Grow CNTs
2. Define gates and contacts
3. Etch CNTs
4. Chemically dope P-type region
5. Chemically dope N-type region

Formal Approach generalized for arbitrary functions

System-on-Chip Test Architectures  Ch. 17 – Testing Aspects of Nanotechnology Trends - P. 60
Misaligned-CNT-Vulnerable NAND: Pull-up

Implemented Function

Path 1: \( C-D-A-D-C \) : \( fn = A \)
Path 2: \( C-D-B-D-C \) : \( fn = B \)
Path 3: \( C-D-A-D-B-D-C \) : \( fn = A \& B \)
Path 4: \( C-D-C \) : \( fn = 1 \)

Intended Function

\( A \text{ or } B \) or
\((A \text{ AND } B) \text{ or } 1\)

Mismatch

1 \( \neq \) A or B

System-on-Chip Test Architectures
Ch. 17 – Testing Aspects of Nanotechnology Trends - P. 61
### Misaligned-CNT-Immune NAND: Pull-up

Path 1: \(C-D-A-D-C\) : \(fn = A\)
Path 2: \(C-D-B-D-C\) : \(fn = B\)
Path 3: \(C-D-A-D-B-C\) : \(fn = A \& B\)
Path 4: \(C-D-B-UD-A-D-C\) : \(fn = 0\)

### Intended Function
- \(A \text{ or } B\)

### Implemented Function
- \(A \text{ or } B \text{ or } (A \text{ and } B)\)
- \(A \text{ or } B \text{ and } 0\)

\[\equiv A \text{ or } B\]
Misaligned-CNT-Immune Logic Design

- Immune to a large number of misaligned CNTs
- Arbitrary logic function, cell area penalty < 10%
- Formal correctness proof (Details in DAC 2007)
Concluding Remarks

- Emerging nanoscale technologies will enable extremely high levels of devices to be integrated onto a single substrate.
  - How will we make effective use of these huge numbers?
  - Current CAD tools can’t scale to handle so many components.

- Regardless of which device is used, all systems will need some form of built-in defect and fault tolerance.
  - Applying defect and fault tolerance at several levels of abstraction may lead to more cost effective designs.

- Regular, tile-based architectures seem particularly promising
  - Rely on local computation and storage
  - Require limited global communication
Concluding Remarks (Cont’d)

- Hybrid approach appears promising, *BUT*…
  - How do you integrate nanoscale devices into CMOS technology?
  - What kind of constraints will be placed on the architecture?
  - How will interconnect and memory bottlenecks limit defect and fault tolerance capabilities?
Future Directions

- Designs strategies for reliable computation require various means of redundancy.
  
  *What is a reasonable overhead to pay in order to reliably operate with nanoscale devices?*

- Other issues:
  - Asynchronous computation may have a future role
    - May simplify global communication and power issues
  - New fault models may be needed (which may alter our results and design choices)
  - Software support will be needed
    - For managing faults, testing, error recovery
    - For mapping from behavioral description to nanoscale devices.
Future Directions (cont’d)

- Effective use of nanotechnology will require total system solutions
- Require collaboration at all levels
- Design tools and methodologies must change to accommodate new ways of computing.
- Make fault tolerance, reconfigurability, power primary issues at the architectural level
- Explore architectures that exploit benefits of nanotechnology while minimizing pitfalls:
  - “Parallelizing” more algorithms
  - Consider alternative information representation models