Reconfigurable Nanoelectronics and Defect Tolerance

Seth Copen Goldstein
Carnegie Mellon University
seth@cs.cmu.edu

HLDVT
11/13/03

Technology Shifts

- Size of Devices ⇒ Inches to Microns to Nanometers
- Type of Interconnect ⇒ Rods to Lithowires to Nanowires
- Method of Fabrication ⇒ Hammers to Light to Self-Assembly
- Largest Sustainable System ⇒ 10^1 to 10^8 to 10^{12}
- Reliability ⇒ Bad to Excellent to Unknown

Size Matters

As we scale down:
- Devices become
  - more variable
  - more faulty (defects & faults)
  - numerous
- Fabrication becomes
  - More constrained
  - More expensive
- Design becomes
  - More complicated
  - More expensive

Combination of Hans Moravac + Larry Roberts + Gordon Bell
WordSize*ops/s/sysprice
doubles every few months?
nanometer

From Gray Turing Award Lecture
Size Matters

As we scale down:
- Devices become more variable
- more faulty (defects & faults)
- numerous
- Fabrication becomes more constrained
- more expensive
- Design becomes more complicated
- More expensive

 Requires:
- Defect tolerant architectures
- Higher level specification
- Universal substrate

Defect Tolerant Architectures

- Features:
  - Regular topology
  - Homogenous resources
  - Fine-grained?
  - Post-fabrication modification
- Example from today: DRAM
  - Requires external device for testing
  - Requires external device for repair
- Logic? FPGA
  Key is redundancy

Programmable Switches

FPGA

Interconnection network

Universal gates and/or storage elements
Reconfigurable Computing

General-Purpose Custom Hardware

Compiler

Logic Blocks

Routing Resources

Reconfigurability & DFT

• FPGA computing fabric
  - Regular
  - Periodic
  - Fine-grained
  - Homogenous

• Programs \Rightarrow \text{circuits}

DFT tool flow

- Fabric
- Tester
- Defect Map
- Defect Aware Place-and-Route
- "Hard" Config.

- "Soft" Config.
- (Async.) Circuit Netlist
- Defect Unaware Place-and-Route

Place & Route

Place & Route

Aides defect tolerance

int reverse(int x)
{
    int r = 0;
    for (k = 0; k < 64; k++)
    {
        r |= x & 1;
        x >>= 1;
    }
    return r;
}

int func(int* a, int* b)
{
    int j, sum = 0;
    for (j = 0; *a > 0; j++)
    {
        sum += reverse(*b);
        *a++;
    }
    return sum;
}
Finding The Defects

- Need to discover the characteristics of the individual components, but
- Can’t selectively stimulate or probe the components
- Download test machines

≈ Built-In Self-Test

Configure testers vertically

Configure the device to test itself!

≈ Built-In Self-Test

Configure testers horizontally

Combine data And Identify fault!

Configure the device to test itself!
What if more defects?

Configure testers vertically

Configure testers horizontally

Configure the device to test itself!

High Defect Rates: Our Algorithm

- Need more complex testing methods as defect rate increases
- Key insight: Testers need to return more than binary information

2 key ideas:
- More powerful test-circuits
  - More than binary info; e.g. approximate defect counts
- More powerful analysis techniques
High Defect Rates: Our Algorithm

- Eliminates remaining defects
- Deterministic; no mistakes

Fabric → Probability-Assignment Phase → "Probabilistic" Defect Map → Defect-Location Phase → Defect Map

How It Works

See Mahim Mishra's Talk at ITC '03
(Can be found at www.cs.cmu.edu/~phoenix)

Reconfigurable Computing

General-Purpose Custom Hardware

int reverse(int x)
{
    int r=0;
    for (k=0; k<64; k++)
        r |= x&1;
    x = x >> 1;
    r = r << 1;
}

int func(int* a, int* b)
{
    int j, sum=0;
    for (j=0; *a>0; j++)
        sum += reverse(*b);
}

Advantages of Reconfigurable

- Flexibility of a processor
- Performance of custom hardware

Near

You have to
- Store and
- Address
  the configuration
Heart of an FPGA

Switch controlled by a 1-bit RAM cell

The cost of the FPGA: Increased Area-Delay Product

Universal gate = RAM

data in
control

Switch controlled by a 1-bit RAM cell

Key Component: Reconfigurable Switch

http://www.chem.ucla.edu/dept/Faculty/stoddart/research/mv.htm
Key Component: Reconfigurable Switch

Reconfigurable Molecular Switch: Eliminates overhead for reconfigurable computing

The Molecular Electronics Advantage: A Reconfigurable Switch

- Each crosspoint is a reconfigurable switch
- Can be programmed using the signal wires

Eliminates overhead found in CMOS FPGAs!

Fabrication Is Different

- Devices & wires alone are not useful
- Key to nanoscale computing is bottom-up assembly

Building a Computing Crystal
Building a Computing Crystal

Implications

- Devices only at cross-points (only 2 terminal devices?)
- Only regular structures
- Defect-tolerance required
- Functionality must be added post-fabrication
  → Reconfigurable!

Abstraction Layers Still OK?

- Applications
- Algorithms
- Programming Languages
- Intermediate Representations
  - ISA
  - Microarchitecture
- Circuits
- Devices
- Tools

Breaking Abstractions

- Applications
- Algorithms
- Programming Languages
- Intermediate Representations
  - ISA
  - Microarchitecture
- Circuits
- Devices
- Tools
- Reconfigurable Fabric
E.g., Asynchronous Design

- Tolerant of
  - Layout
  - Parametric variation
  - Variable Latency Operations
- Low-power
- Natural implementation for dataflow
- Attacks other problems (e.g., clock skew, distribution, ...)

E.g., ISA has to go?

- Current ISA hides too much
  - Good for
    - Forward compatibility
    - Human oriented assembly
    - Ad hoc additions
  - Bad for
    - Exploiting resources
    - Managing new constraints, e.g., power
    - Exploiting compiler
    - Verification
- What can replace ISA?
- In general, how do we eliminate barriers without increasing complexity?

Another Consideration: Power

```
Power \alpha 1/Parallel

P \alpha CV^2F
F \alpha (V-V^{th})^{5/4}, \text{ for CMOS}\frac{V}{V}
In relevant range: F \alpha V
So, P \alpha CF^3
Also, 1/A \alpha T^n, early VLSI theory result \(1 \leq n \leq 2\)
If we fix T, then F^{-1} \alpha T, F \alpha A^{-1/n}
If n=2, P \alpha CA^{-3/2}
If C \alpha A, P \alpha A^{-1}
```

Power densities too high to keep junctions at low temps

Source: Irwin & Borkar, De In tel
Reconfigurable Computing

General-Purpose Custom Hardware

int reverse(int x)
{
    int k, r = 0;
    for (k = 0; k < 64; k++)
        r |= x & 1;
    x = x >> 1;
    r = r << 1;
}

int func(int *a, int *b)
{
    int j, sum = 0;
    for (j = 0; *a > 0; j++)
        sum += reverse(*b);
}

Bryant's Law

• Processor verification is always 10 years behind
• What to do?
Don’t use processors!

Spanning 10-orders of Magnitude

1 Program

Compilers
Phoenix
Theory
Architecture
10 Billion Gates

Circuits From Compilers

1. Program
2. Split-phase Abstract Machines
3. “soft” P&R on indiv configs
4. Placement on chip

Certifying Circuits!

Computations & local storage
Unknown latency ops.
Conclusions

• Reconfigurable Computing is inevitable
  - Cost of manufacturing
  - Defect tolerance
  - Fabrication constraints

• X-point (e.g., Molecular) switches are ideal for reconfigurable device

• EN: New constraints, but huge potential
  - Billions of devices per cm²
  - Ultra-low power
  - Faster design time
  - Easier verification

Conclusions - 2

• New Abstractions are required
  - Defect tolerance
  - Spatial Computing
  - Asynchronous Circuits

• Abstraction Requirements:
  - Tool friendly not human friendly
  - Support parallel research activities
  - Promote interdisciplinary research