A Defect-Tolerant Computer Architecture: Opportunities for Nanotechnology

By: James R. Heath, Philip J. Kuekes, Gregory S. Snider, R. Stanley Williams

SCIENCE, VOL. 280 , 12 JUNE 1998

Reza M. Rad
UMBC

Introduction

- Teramac: a massively parallel experimental computer built at HP-labs
- Contains about 220,000 hardware defects
- Yet it operated 100 times faster than a high-end single-processor workstation for some of its configurations
- The defect-tolerant architecture of Teramac incorporates a **high communication bandwidth** that enables it to easily route around defects

Introduction

- Future nanoscale computers may consist of extremely large-configuration memories that are programmed for specific tasks by **a tutor that locates and tags the defects in the system**

- ***Chemical assembly:*** any manufacturing process whereby various electronic components, such as wires, switches, and memory elements, are chemically synthesized (a process often called “self-assembly”) and then chemically connected together (by a process of “self-ordering”) to form a working computer or other electronic circuit

Introduction

- Some fraction of the discrete devices will not be operational because of the **statistical yields of the chemical syntheses** used to make them,
- It will **not be feasible to test them all** to select out the bad ones
- In addition, the system will suffer an inevitable and possibly large amount of uncertainty in the connectivity of the devices

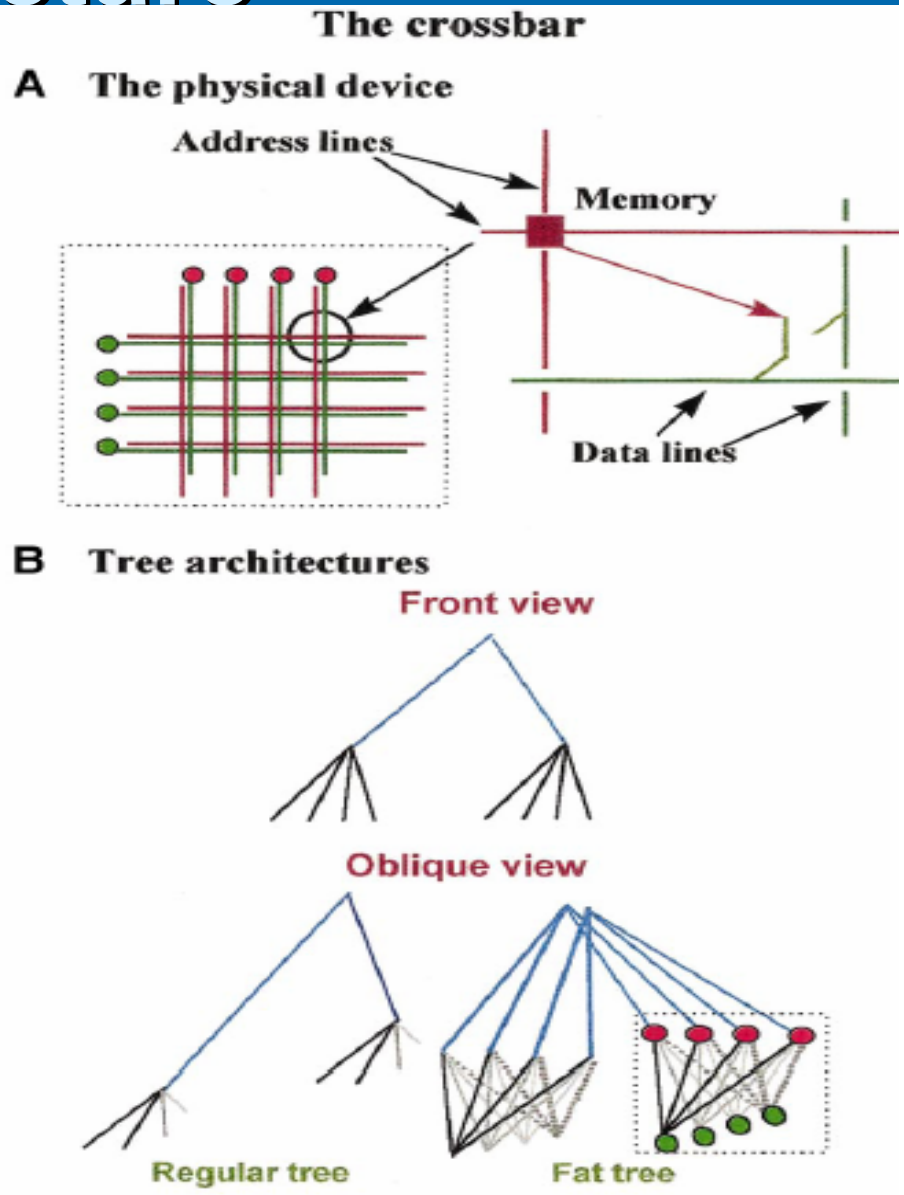
Custom Configurable Architecture

- Teramac contains 864 identical chips (FPGAs) designed and built specifically for Teramac
- The “answers” to the logical functions (the truth tables) are stored in 64-bit Look-Up Tables (LUTs). Each LUT holds the equivalent of 10 logic gates, and there are a total of 65,536 LUTs in the machine

Custom Configurable Architecture

- (**A**) The **crossbar** represents the heart of the configurable wiring network that makes up Teramac

- Between any two configuration bits, there are a large number of pathways, which implies a high communication bandwidth within a given crossbar. Logically, this may be represented as a "fat tree." Such a "fat tree" is shown in (**B**)



Custom Configurable Architecture

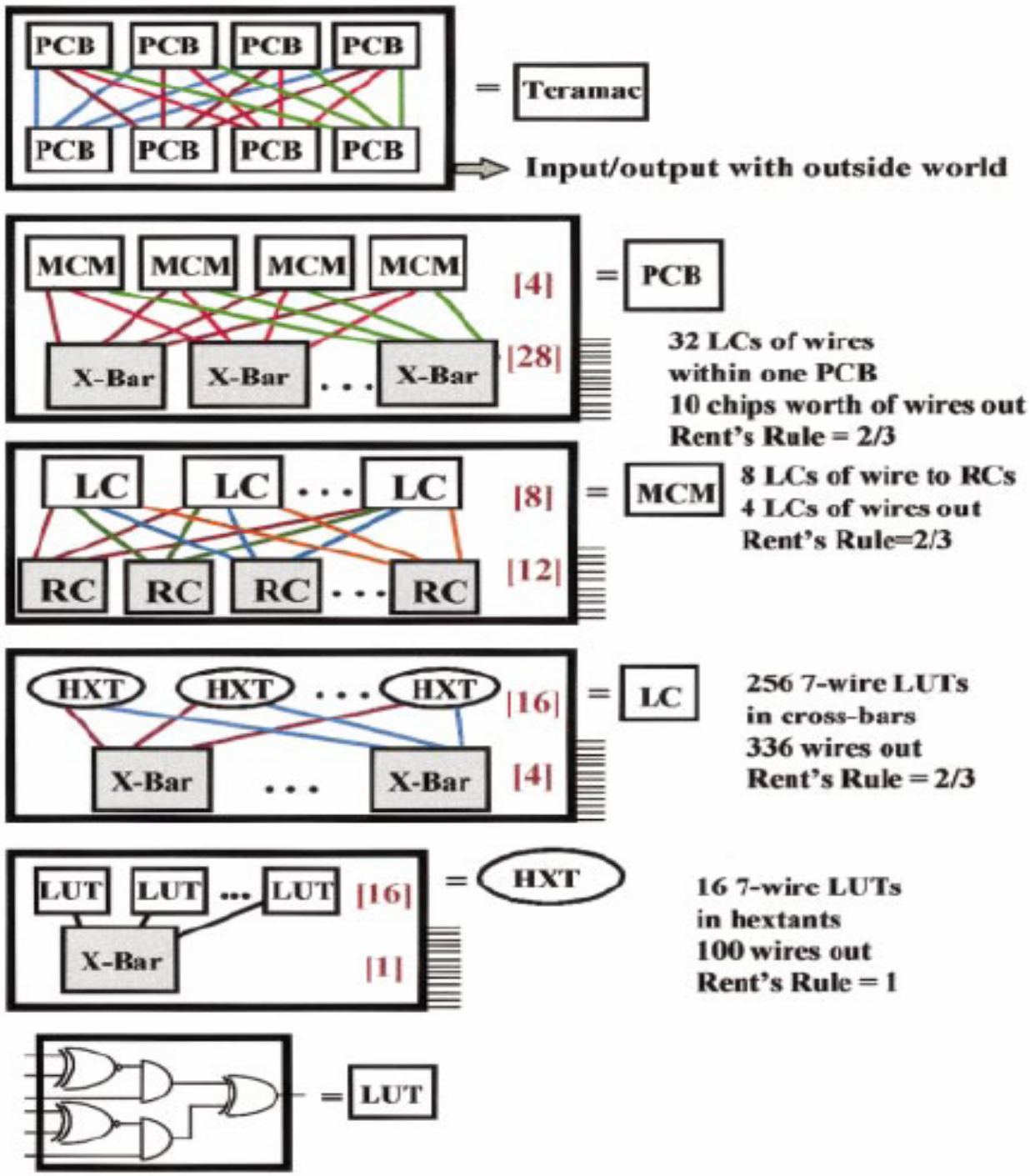
- In the **regular tree architecture**, if the line of communication between a parent and grandparent is broken, then communication to a whole branch of the family tree is cut off

- In a **fat tree** each single-parent node is replaced by several nodes, and communications between levels of the tree occur through crossbars that connect multiple nodes at each level

Rent's Rule

- Rent's rule is an empirically derived guideline that may be **used to determine the minimum communication bandwidth** that should be included in a fat-tree architecture
- **Rent's rule** states that for the realistic circuits, the number of wires coming out of a particular region of the circuit should scale as a power of the number of devices (*n*) in that region, ranging from *n*1/2 to *n*2/3
- For the crossbars of Teramac, exponents ranging between 2/3 and 1 were used, and thus significantly more bandwidth than required by Rent's rules was incorporated into the fat tree

The logical map of Teramac



Defect Tolerance

- For Teramac, the entire machine was designed to be defect tolerant
- Each multichip module (MCM) had 33 layers of wiring to interconnect a total of 27 chips, 8 used for their LUTs and 19 for only their crossbars
- Each printed circuit board (PCB) had 12 layers of interconnects for four MCMs
- **Adding defect tolerance to the system essentially involved avoiding those configurations that contained unreliable resources**

Defect Tolerance

- Only 217 of the FPGAs used in Teramac were free of defects
- The rest (75% of the total used) were free of charge, because the commercial foundry that made them would normally have discarded them
- Half of the MCMs failed the manufacturer's tests, so they were also free
- Out of a total of 7,670,000 resources in Teramac, 3% were defective

Defect Tolerance

- If Teramac is physically damaged (a chip is removed, or a set of wires cut, for example), it can be reconfigured and resume operation with only a minor loss in computational capacity
- Teramac was connected to an independent workstation that performed the initial testing
- The testing process can be separated into running configurations that measure the state of the CCC, and a set of algorithms that are run on these measurements to determine the defect

Defect Tolerance

- LUTs were connected in a wide variety of configurations to determine if a resource (switch, wire, or LUT) was reliable or not.
- If any group failed, then other configurations that used the resources in question in combination with other devices were checked.
- Those resources found in the intersection of the unreliable configurations were declared bad and logged in a defect database

Defect Tolerance

- Once the defect data base had been established, computer architectures could be loaded onto Teramac
- As perfect devices become more expensive to fabricate, defect tolerance becomes a more valuable method to deal with the imperfections
- Any computer with nanoscale components will contain a significant number of defects, as well as massive numbers of wires and switches for communication purposes

Lessons for Nanotechnology

- The first lesson is that it is possible to build a very powerful computer that contains defective components and wiring, as long as there is sufficient communication bandwidth in the system to find and use the healthy resources
- In Teramac, wires are by far the most plentiful resource, and the most important are the address lines that control the settings of the configuration switches and the data lines that link the LUTs to perform the calculations

Lessons for Nanotechnology

- The Teramac paradigm is to build the computer (however imperfectly), find the defects, configure the resources with software, compile the program, and then run it