

Combinatorial Architecture for High-Performance Computing and Networking

**Simon Berkovich and Alexander Kuznetsov
Department of Computer Science
The George Washington University**

**GWU Symposium on High-
Performance
Computing**

October 18, 2006

A novel approach to the extraction of parallelism

The allurements and predicaments of parallel processing:

- Productivity of collective efforts

 - Great variability in performance

 - The bottleneck of Amdahl's Law

- Organization of the division of labor

 - The labyrinths of concurrent activities

 - Difficulties in programming

Types of concurrency

Sequential
Program



Instruction-Level
Parallelism



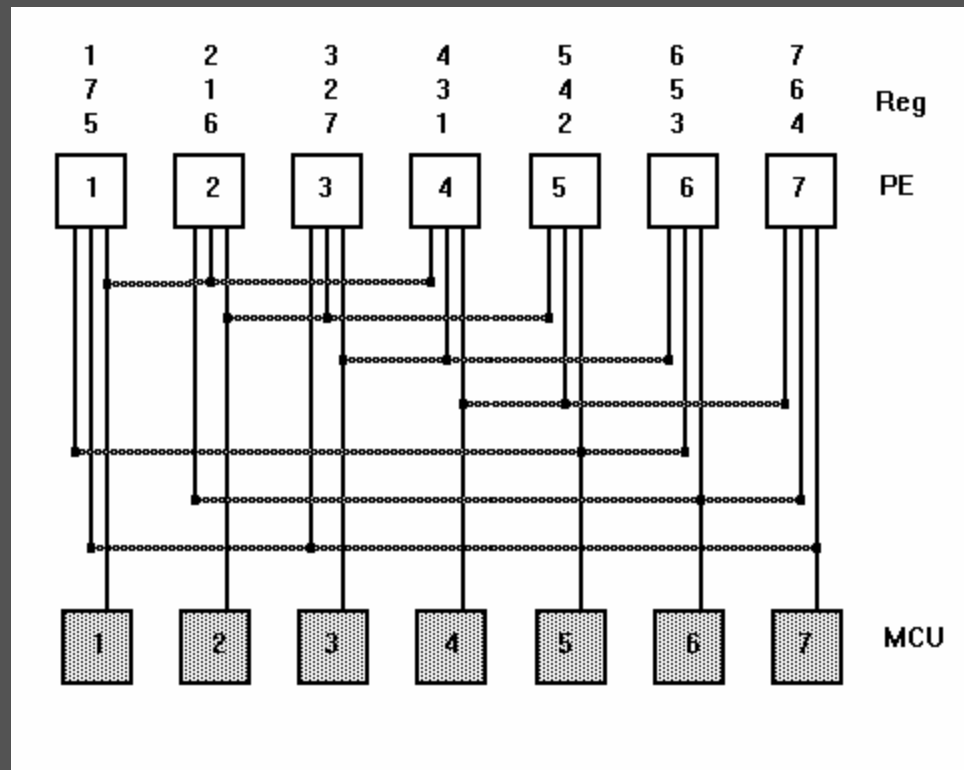
Massive
Parallelism



**The ubiquity of the Instruction-Level Parallelism:
Superscalar, pipelining, VLIW**
**The major problem:
Hardware and software complexity**

Combinatorial Architecture

An example of a (7,7,3,3,1) arrangement



Instructions are split according to the pairs of register operands:

ADD R1 R7 ==> PE #1
 MULT R2 R6 ==> PE #2
 DIV R4 R5 ==> PE #5

Combinatorial designs

A combinatorial design is a mathematical structure representing a collection of subsets of a given set satisfying a certain selection condition.

The suggested architecture uses a particular type of combinatorial designs, the Balanced Incomplete Block (BIB) designs, which are characterized by a distribution of elements showing a pairwise balance property.

Balanced Incomplete Block Designs:

$(b, v, r, k, ?)$

Example: $(12, 9, 4, 3, 1)$

sets

| | B_1 | B_2 | B_3 | B_4 | B_5 | B_6 | B_7 | B_8 | B_9 | B_{10} | B_{11} | B_{12} |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|----------|----------|----------|
| X_1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| X_2 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| X_3 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| X_4 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| X_5 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| X_6 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| X_7 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| X_8 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| X_9 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |

elements

$$b \cdot k = v \cdot r$$

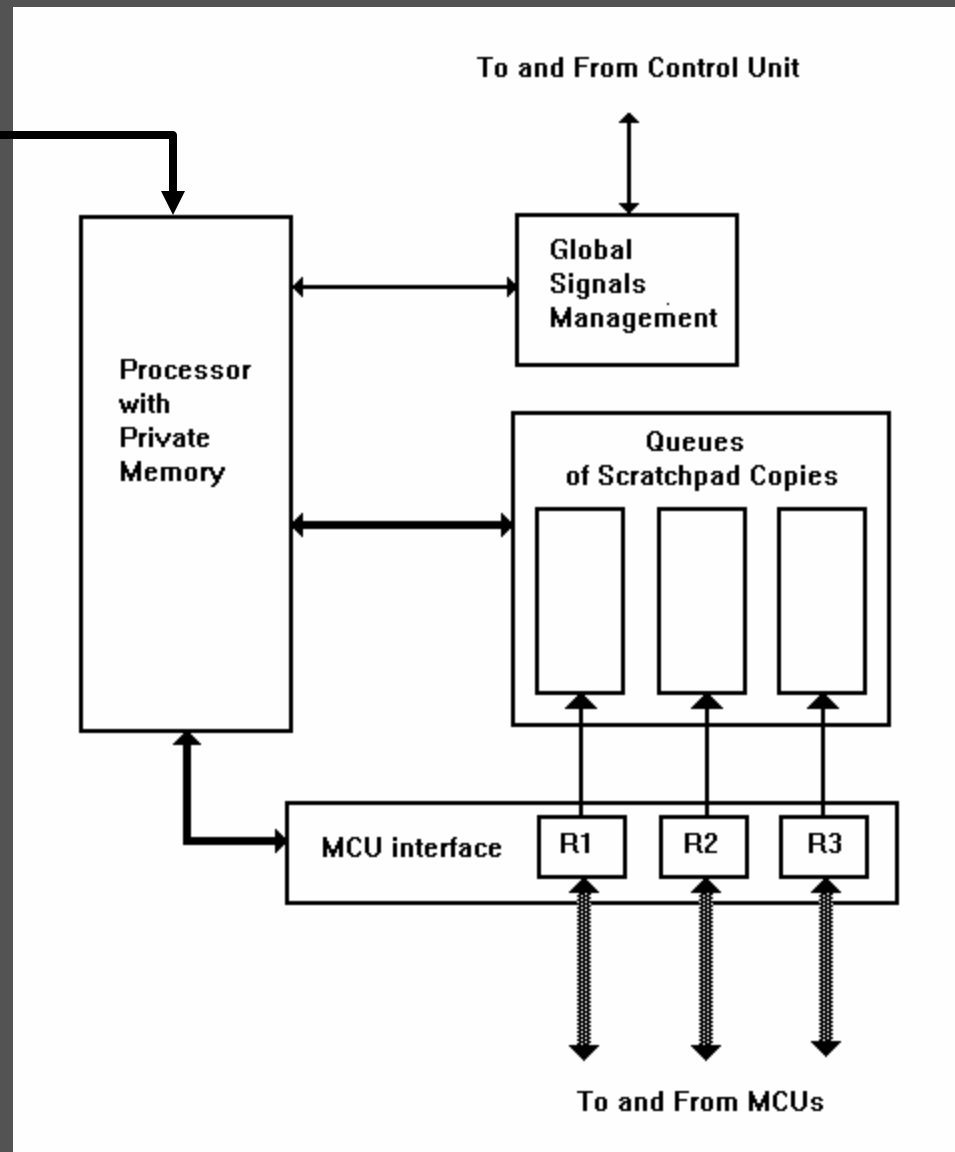
$$r \cdot (k - 1) = ? \cdot (v - 1)$$

Three radical construction solutions

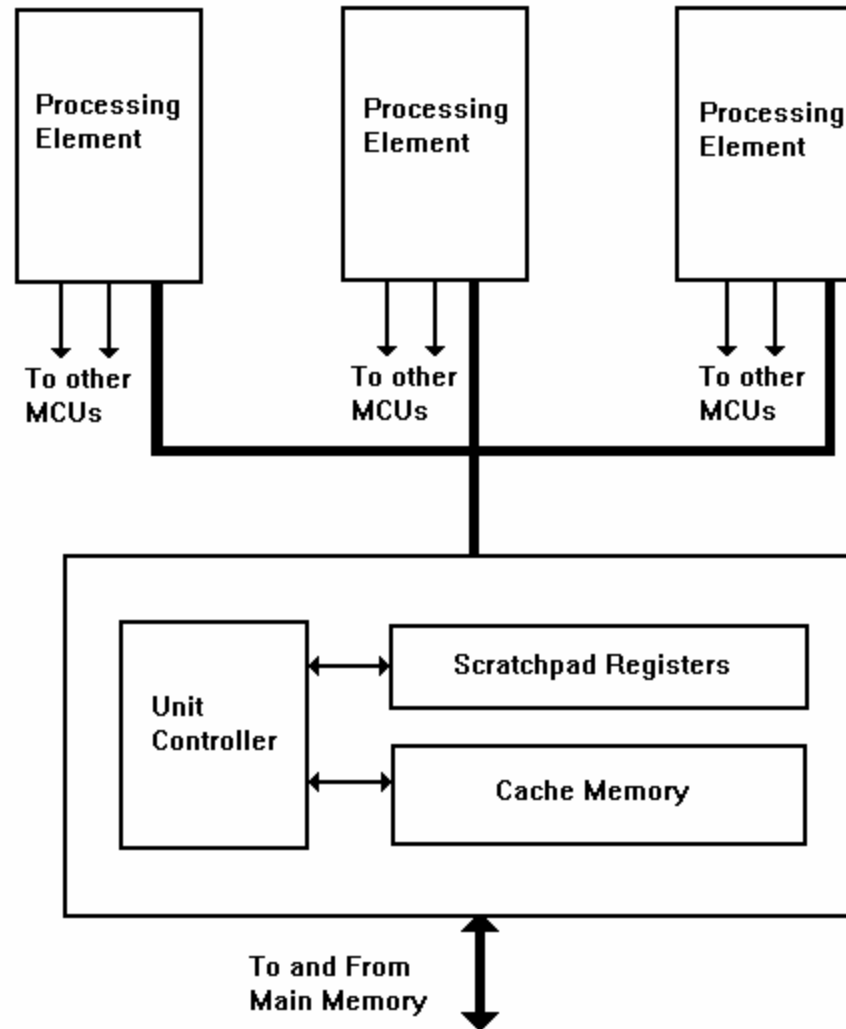
- 1 -- the whole memory system is divided into separate chunks, which are combined at the top of the hierarchy through the registers linked according to the BIB combinatorial design
- 2 -- sequential consistency of partitioned programs is assured by arranging the exchange of operands through queues associated with the scratchpad registers
- 3 -- switching operations for information flows beyond the immediate register interactions - for indirect loading from main memory and for packet routing between input and output ports - exploit the BIB property of complete set coverage by subsets having a common element

Structure of the Processing Element

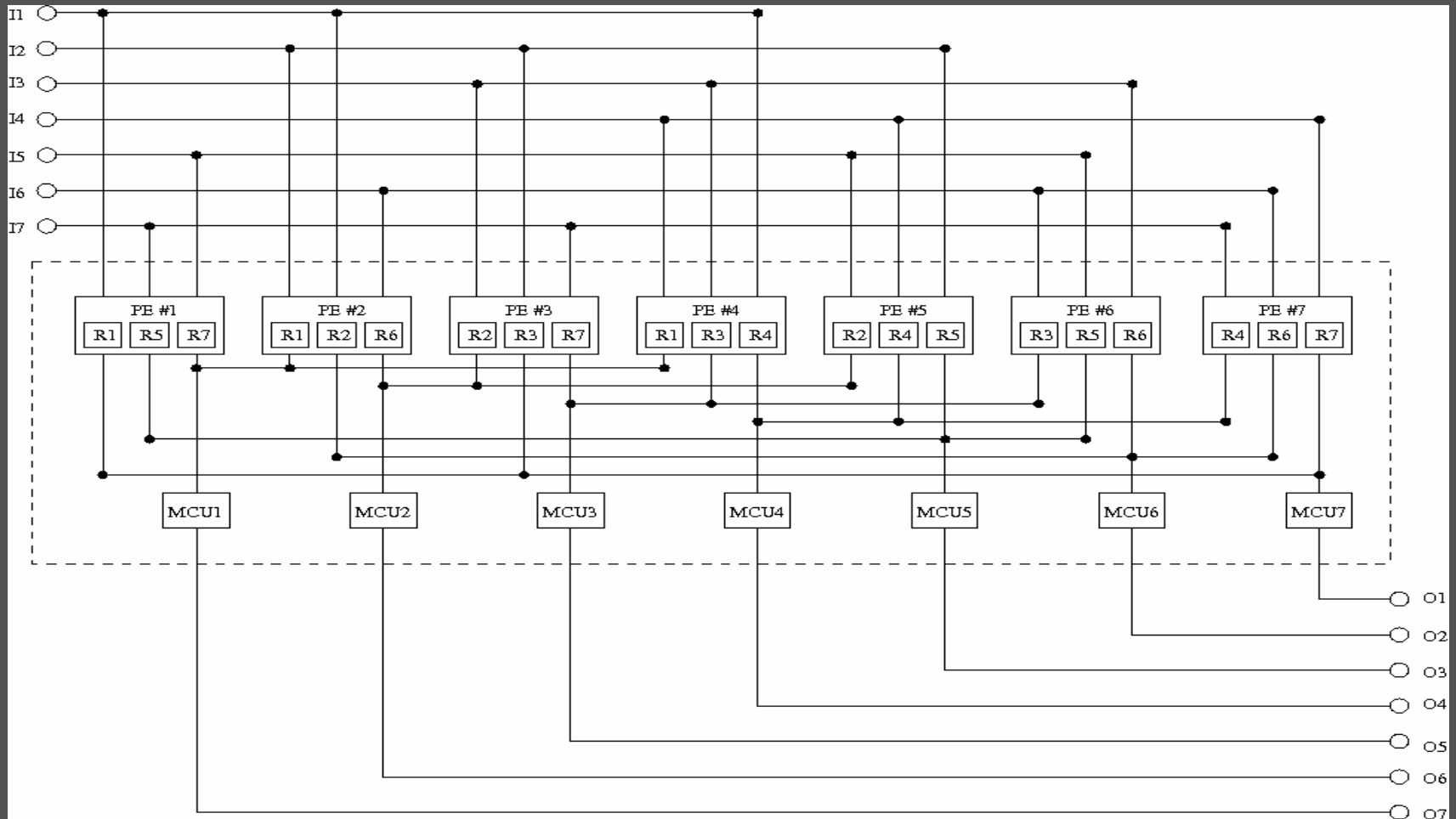
Router
input



Memory connections



Router functionality



Complete set coverage

Indirect addressing and implementation of router operations.

Ind_Load 'R6' R6

This instruction is placed according to PE6 connections in PE 2, 6, 7

Assume that 'R' contains an address in memory block 4

Analogously, suppose a packet is arriving at Input Channel #6, and its destination, yet unknown, is Output Channel #4.

This packet is goes into PE 2,6, and 7. Each of these PEs starts examining the packet. PEs 2 and 6 do not have a connection to the

Output Channel #4, so they dropped this packet.

PE7 does have a connection to channel 4, so it can forward the

packet to the required destination.

Simulation model

to check the sequential consistency of the sophisticated operational logic

The screenshot shows the REBUS Simulation: 7 Processor Model interface. At the top, the title bar reads "REBUS Simulation: 7 Processor Model". Below the title bar is a menu bar with "File", "Action", and "Window".

The main interface is divided into several sections:

- Top Left:** "Rebus Processor stopped at 755 ticks". It shows "Total Instructions: 39" and "Instructions Executed: 282". Below this are "Processors Working" buttons numbered 1 through 7, and "Elapsed Clock Ticks: 2363" with a "Ready..." button.
- Top Right:** "UniProcessor stopped at 2362 ticks". It shows "Total Instructions: 29" and "Instructions Executed: 182". Below this is "Final Speedup 3.1" with a progress bar. To the right is a logo for "Rebus".
- Center:** "Rebus Processor 7" window. It shows a list of instructions with status indicators (MC, WD, LO, PR, BN, ST) and a "Scratchpad Queues" section with registers Reg7 (20), Reg6 (83), and Reg4 (10). The current instruction is "STOP".
- Right:** "UniProcessor" window. It shows a list of instructions and a "Registers" section with values for Reg1 (10), Reg2 (20), Reg3 (147), Reg4 (10), Reg5 (0), Reg6 (83), and Reg7 (20).
- Bottom Left:** "Memory Viewer" window. It displays two memory grids: "Rebus Memory" and "UniProcessor Memory". Both grids show values 19, 20, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, and 0. Below the grids is a slider for "Currently displaying: 136 to 151" with a range from 0 to 511.
- Bottom Right:** Icons for "Memory Editor" and "Instruction Editor".

Next step: hardware model using FPGA

Summary

This talk has presented an innovative computer architecture based on combinatorial interconnections of processor and memory resources.

Arranged in accordance with the so-called Balanced Incomplete Block Designs, this scheme

is characterized by regular distribution of elements pairs over separate units.

The exchange of information is performed by means of direct interaction of replicated

objects rather than by traditional shipping of data from a source to a destination.

The system is simple and robust; it automatically partitions sequential programs into concurrent streams and effectively combines processing and routing capabilities.

Hardware is built of well-developed types of components without complicated functional interactions, like synchronization, branch scheduling, and pipeline hazards.

Software is essentially the same as for conventional serial computers.

With this architecture it is possible to enjoy a universal speedup by factor of 3 and more without any particular efforts just by virtue of unusual interconnections of the operative components.

Intellectual property

S.Berkovich and E.Berkovich

US Patent No. 5,619,680,

“Methods and apparatus for concurrent execution of serial computing instructions using combinatorial architecture

for program partitioning”,

issued April 8, 1997

S.Berkovich and A. Kuznetsov

“Network Router Based on Combinatorial Designs”,

Patent Cooperation Treaty (PCT) international application,

filed February 22, 2006

Reducing to Practice

Washington, DC -- (Oct 10, 2005) -- SNAPP Technologies Corporation (STC) announced today that it has licensed the REBUS project, a combinatorial system architecture that will increase the speed of processing for general applications, from The George Washington University (GWU).

Andrew J. Polcha, CEO SNAPP Technologies Corporation noted, "The wonderful characteristic about this technology is that infinite scalability can be applied without change to application code. No solution can offer a dynamic increase in computational horsepower and include legacy software applications."

Prospective applications

1. Supercomputing

A *FAMILY* of combinatorial designs with
 $m^2 + m + 1 = 7, 13, 21, 31, 57 \dots$ elements

2. Multicore microprocessors

Offsetting the *PHYSICAL SATURATION* of
the integrated circuits technology

3. Router switching fabric

Intelligent managing of *INTENSE PACKET SWITCHING*
in communication networks, like Internet

4. Mission-critical systems

The demand for *HIGHER COMPUTING POWER*
AND FAULT-TOLERANCE is insatiable at any cost