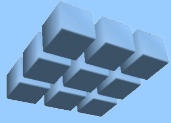


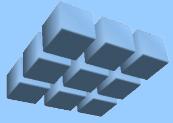
Simulating a Multicore Supercomputer

Steven A. Guccione
Cmpware, Inc.



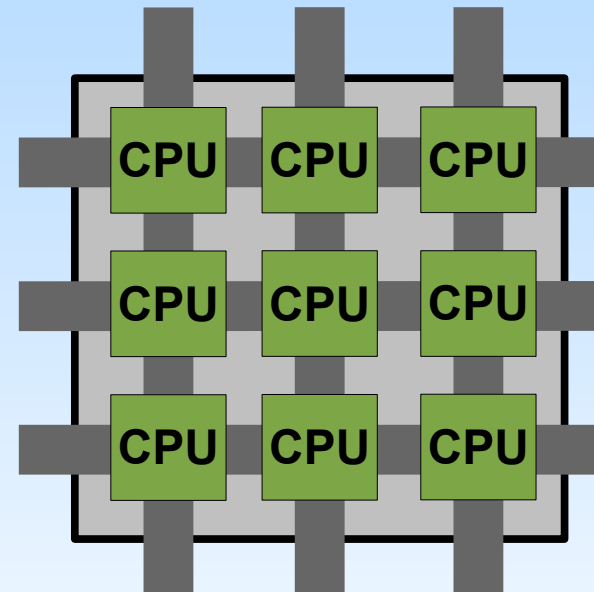
Supercomputers 2006

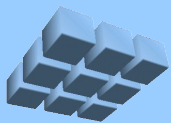
- Mostly multiprocessors
- Few Cray-style vector machines
- Approximately 10,000 – 100,000 processors
- Approximately 100 - 1000 TFLOPS Linpack
- Standard desktop CPUs (80W)
- Power estimate: approx. 1 - 10 MegaWatts
- Power the limiting system parameter
- See: <http://www.top500.org/>



Multicore Processors

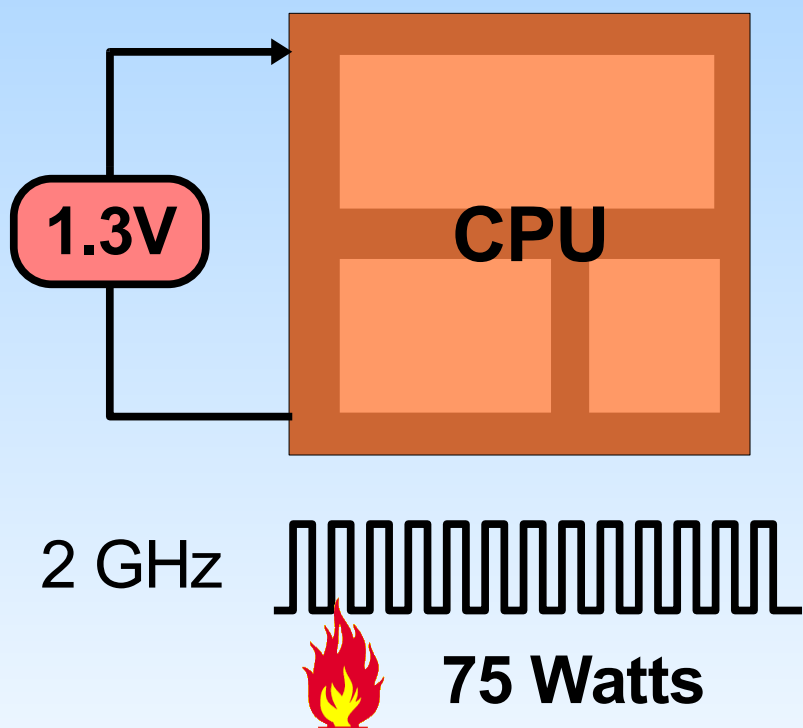
- Thousands of processors on a die possible
- High performance: Thousands of MFLOPs
- Simple HW design
- Simple HW verification
- Flexible
- Scalable
- Reprogrammable
- **Reduces power**



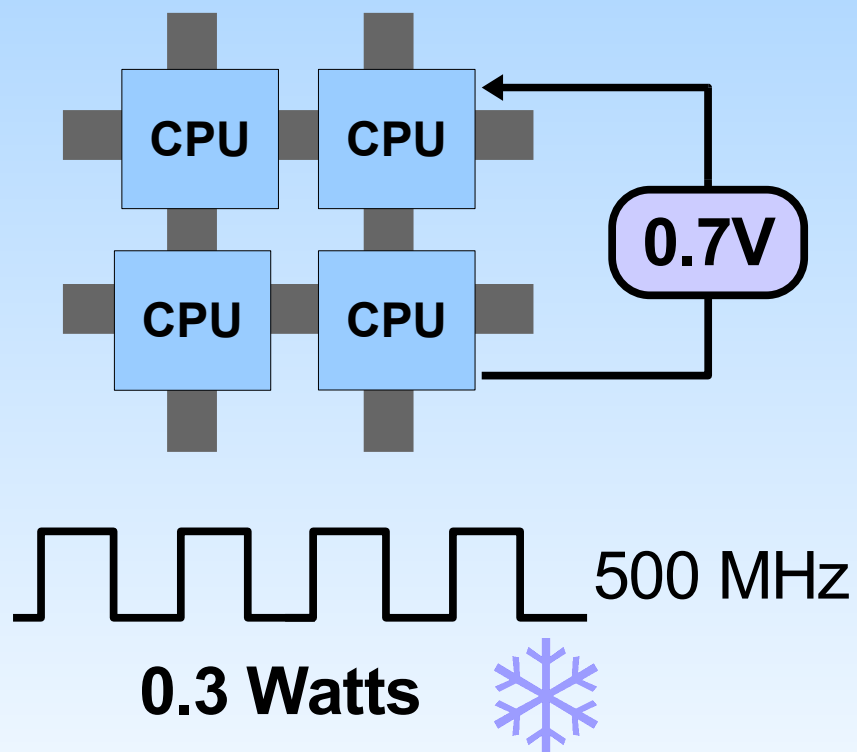


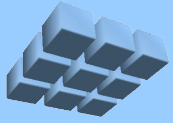
Power and Multicore

Desktop CPU



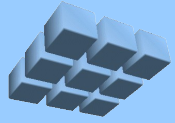
CMP Architecture





Multicore Multiprocessor Supercomputer (MMS-1)

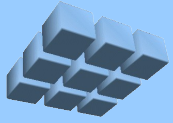
- Initial architecture:
 - Standard 32-bit floating point support
 - SIMD control
 - Mesh (2D)
 - Matrix 'macro' instructions
- Cores per device
 - Technology dependent
 - Independent of device architecture
- Simulated with the ***Cmpware CMP-DK***



Cmpware Modeling Approach

- Models and simulates at a higher level
- Components are processors and channels
- Simpler to model, modify, mix and match
- Follows the ongoing trend in CAD tools

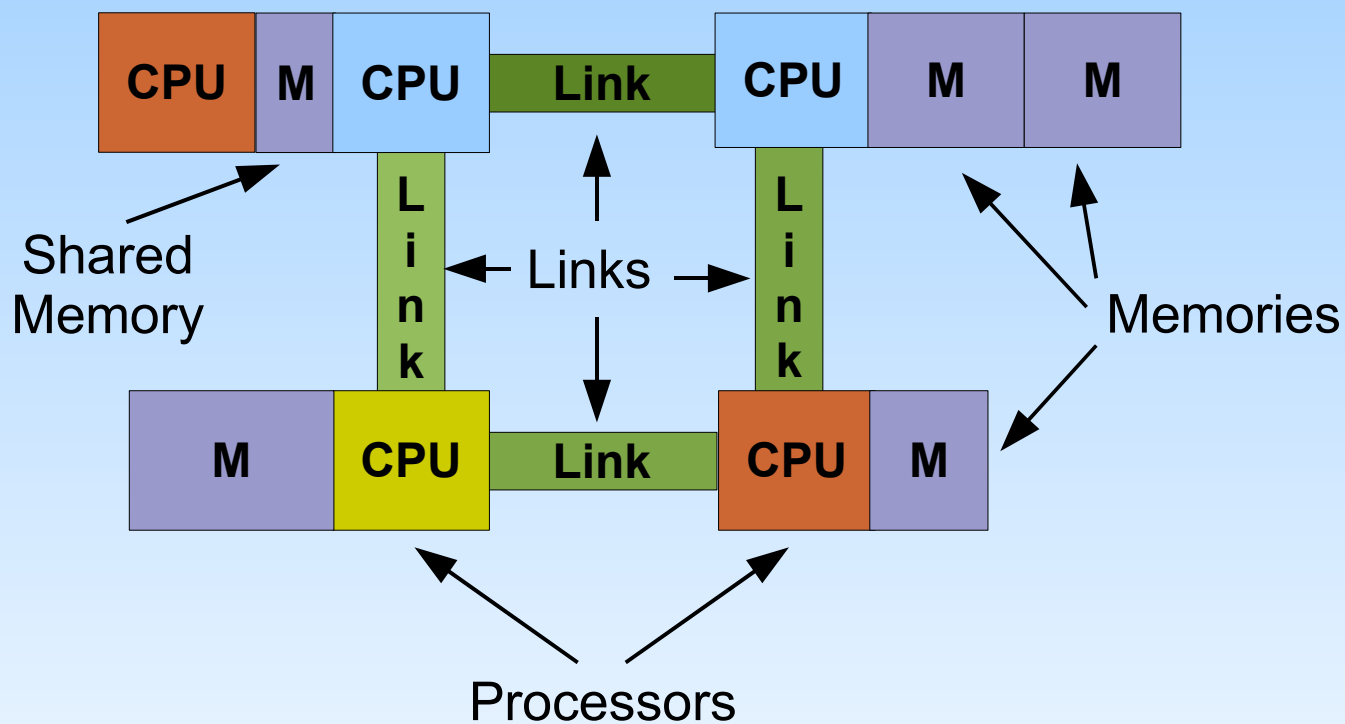
	<u>SPICE</u>	<u>Schematics</u>	<u>RTL</u>	<u>Cmpware</u>
Date:	1970s	1980s	1990s	2000s
Elements:	Transistor	Gate	Register	Processor
Connection:	Wire	Wire / Bus	Bus	Link

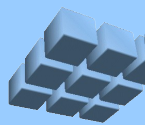


The Cmpware CMP-DK

- Models multiprocessor and interconnection architectures
 - Quickly build and program multiprocessors
 - Redefine multiprocessor / network in seconds
 - Simulation speed of 2M+ instructions / sec
- Executes software from standard tools
- Useful for software development / benchmarking
- Complete *Eclipse* development environment

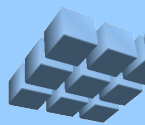
Cmpware CMP-DK Building Blocks





Cmpware CMP-DK Building Blocks (cont.)

- Processors
 - Anything with inputs, outputs and a clock
 - Traditional microprocessor structures supported
 - Standard CPU models available (MIPS, Sparc,...)
 - Simple to implement
- Communication links
 - Variable bit width
 - Synchronized or unsynchronized
 - Buffered or unbuffered (FIFO)

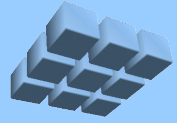


Cmpware CMP-DK Building Blocks (cont.)

- Memories
 - Variable size
 - Big or Little Endian supported
 - Multiple banks supported
- Network
 - Just a collection of links
 - Supports different links (heterogeneous network)

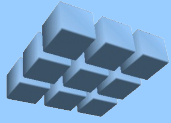
==> *Processors + memories + links =*

Cmpware model

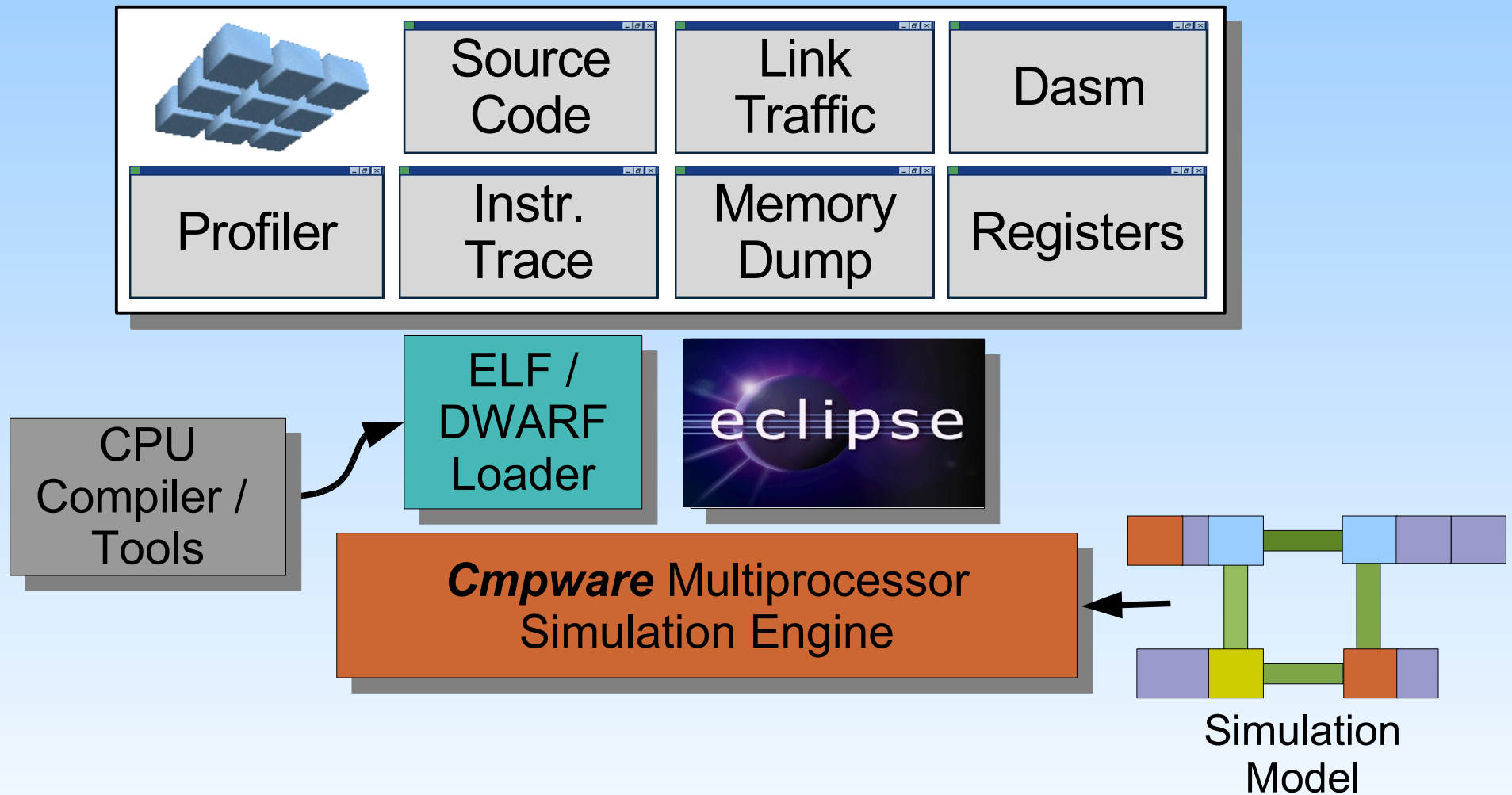


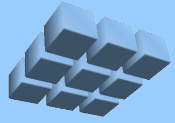
Cmpware CMP-DK Displays

- Standard multiprocessor interface
 - Base on *Eclipse* platform
 - Displays transparently driven by models:
 - Register values
 - Source code variables
 - Memory hex dump
 - Disassembly
 - Source code window
- > custom displays easily added*



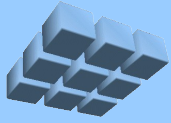
Cmpware CMP-DK





The MMS-1 Design Parameters

- Technology assumptions:
 - 30k gates FPU logic
 - @ 5 transistors per gate
 - @ 500 million transistors per device
 - **==> over 3000 FPU cores per device**
- Raw performance:
 - @ 500 Mhz
 - Power consumption similar to desktop CPU
 - **==> 1.5 TFLOPs per device**



The MMS-1 Model

The screenshot shows the Eclipse SDK interface for the 'Cmpware - Init.dat' project. The main window is titled 'CMP Array' and displays a grid of colored squares representing the array. The grid is 16 columns wide and 16 rows high. The colors of the squares are as follows:

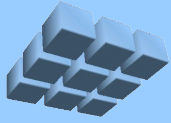
- Row 0: 16 blue squares
- Row 1: 16 orange squares
- Row 2: 16 orange squares
- Row 3: 16 orange squares
- Row 4: 16 orange squares
- Row 5: 16 orange squares
- Row 6: 16 orange squares
- Row 7: 16 orange squares
- Row 8: 16 orange squares
- Row 9: 16 orange squares
- Row 10: 16 orange squares
- Row 11: 16 orange squares
- Row 12: 16 orange squares
- Row 13: 16 orange squares
- Row 14: 16 orange squares
- Row 15: 16 orange squares
- Row 16: 16 orange squares
- Row 17: 16 orange squares
- Row 18: 16 orange squares
- Row 19: 16 orange squares
- Row 20: 16 orange squares
- Row 21: 16 orange squares
- Row 22: 16 orange squares
- Row 23: 16 orange squares
- Row 24: 16 orange squares
- Row 25: 16 orange squares
- Row 26: 16 orange squares
- Row 27: 16 orange squares
- Row 28: 16 orange squares
- Row 29: 16 orange squares
- Row 30: 16 orange squares
- Row 31: 16 orange squares
- Row 32: 16 green squares
- Row 33: 16 green squares
- Row 34: 16 green squares
- Row 35: 16 green squares
- Row 36: 16 green squares
- Row 37: 16 green squares
- Row 38: 16 green squares
- Row 39: 16 green squares
- Row 40: 16 green squares
- Row 41: 16 green squares

The 'Variables' window on the left shows the following data:

r[n]	Name	Value
0	r0	0.0
1	r1	1.0
2	r2	Infinity
3	r3	0.0
4	r4	0.0
5	r5	0.0
6	r6	0.0
7	r7	-1234.5677
8	r8	0.0
9	r9	0.0
10	r10	0.0
11	r11	0.0
12	r12	0.0
13	r13	0.0
14	r14	0.0
15	r15	0.0
16	r16	0.0
17	r17	0.0
18	r18	0.0
19	r19	0.0
20	r20	0.0
21	r21	0.0
22	r22	0.0
23	r23	0.0
24	r24	0.0
25	r25	0.0
26	r26	0.0
27	r27	0.0
28	r28	0.0
29	r29	0.0
30	r30	0.0
31	r31	0.0
32	r32	0.0
33	r33	0.0
34	r34	0.0
35	r35	0.0
36	r36	0.0
37	r37	0.0
38	r38	0.0
39	r39	0.0
40	r40	0.0
41	r41	0.0

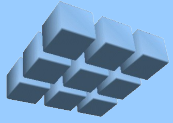
The 'Status' window at the bottom shows the following text:

```
BiccPE0(0,4) selected.  
null(16,15) selected.  
Controller(16,0) selected.  
BiccPE1(15,0) selected.
```



Simulation Experience

- Network and processor modeled in <1 day
- Simulated algorithms at the application level
- Used simple programming tools (assembler)
- Reduced version of processor simulated
 - Paramaterized – change array size dynamically
 - 16 x 16 (= 256) cores used for experimentation
- Custom displays for floating point added
- Custom controller for data loading added



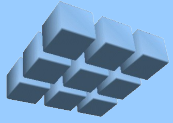
Input / Output Balance

- I/O imbalance in matrix operations identified
- The processor performance vs. I/O:
 - **2D** array = N^2 processors
 - N^3 operations per matrix calculation
 - N cycles per matrix operation

But ...

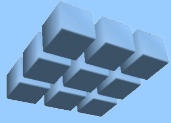
- N^2 data elements per operation (I/O)

==> Processor utilization for Matrix ops = $1/N$



Architectural Explorations

- Linear array (N) appropriate for matrix ops
 - High processor utilization (100%)
 - Balanced I/O
 - Exploration of multithreading
 - More natural $N \times N$ programming model
 - More programming state / reduces I/O
 - Exploration of different networks (3D torus)
 - Examination of SIMD control structures
- ==> Changes to model done in minutes*



Conclusions

- Multicore Supercomputing:
 - A new approach to large-scale computing
 - Lower power / Higher Performance / Inexpensive
 - Custom devices attractive
- Experiences with Simulation:
 - Major system-level deficiencies uncovered
 - New alternatives quickly explored
 - Software application level simulations crucial

==> *Simulation of parallel supercomputer architectures feasible and useful*