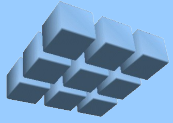


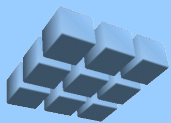
# **Hardware / Software Tradeoffs in Multicore Architectures**

Steven A. Guccione  
Cmpware, Inc.

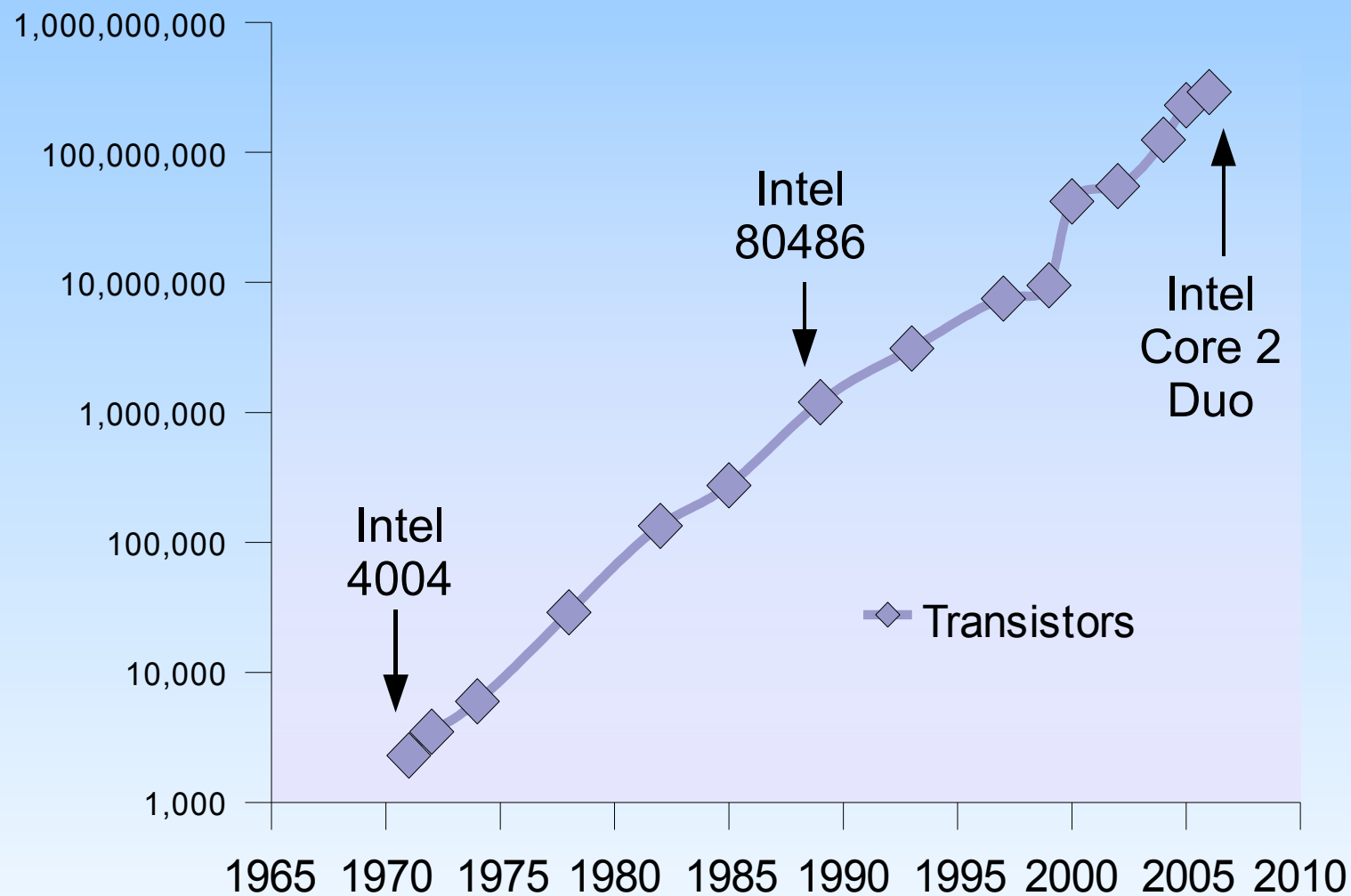


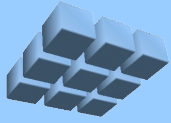
# CPU Transistor Budgets

- Approx. 1B transistors available (2008)
- 30+ years of *Moore's Law*
  - 8 -> 16 -> 32 -> 64 bit CPUs
  - Floating Point
  - Superscalar
  - Caches, buffers, caches and more caches
- Decreasing 'ROI' for new transistors
  - Little performance boost in last generation of single core CPUs



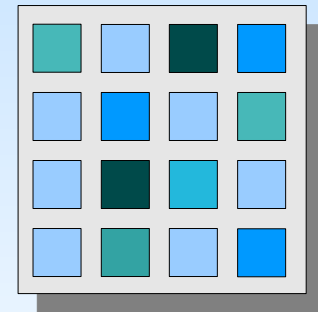
# Transistors Per Year

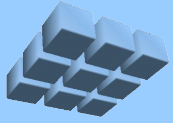




# CPUs in the New Millennium

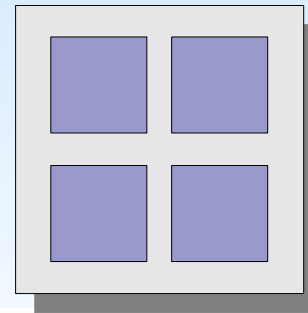
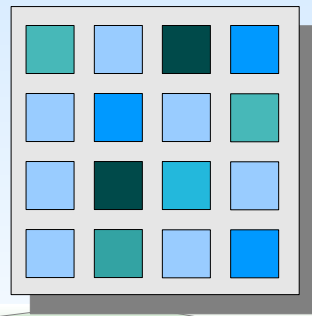
- Single core architectures
  - Clock speeds plateauing (4 GHz)
  - Heat dissipation problems (100+ W)
  - Design costs (\$\$\$)
- Multicore architectures
  - Increase performance
  - Reduce power consumption
  - Simplify design

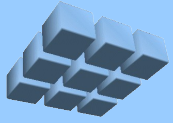




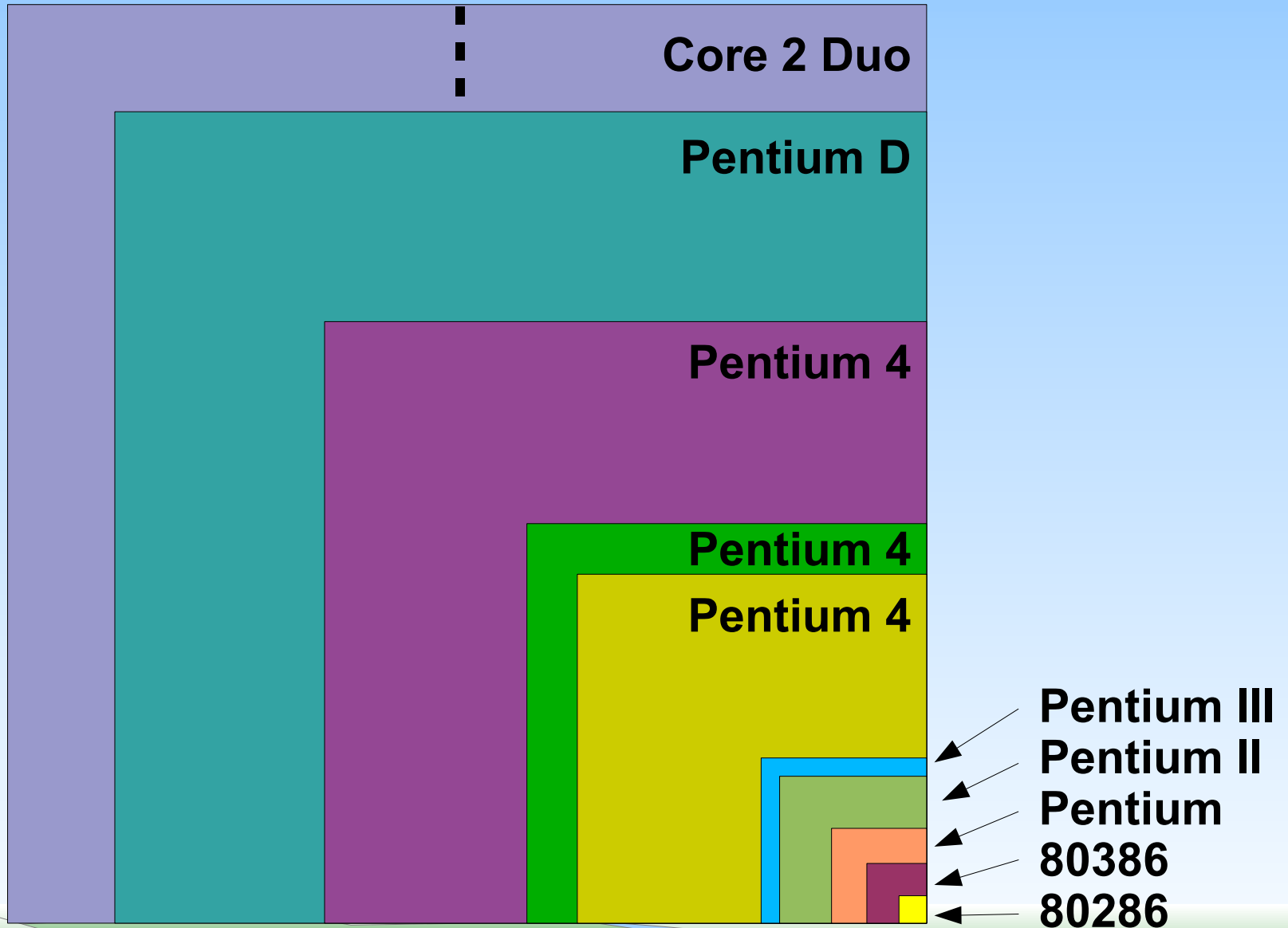
# Multicore Architecture

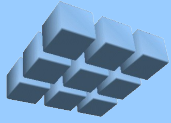
- A new set of design parameters:
  - Core size
  - Number of cores
  - Core functionality
  - Core to core communication
- **Q:** *Do you want dual Pentium 4 or 1,000+ 80386 cores?*





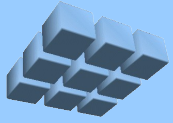
# Relative CPU Sizes





# Massively Multicore

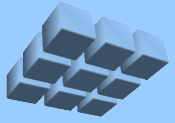
- Smaller cores provide higher total MIPS
  - Using thousands of CPUs in parallel challenging
  - Software and tools issues
  - Memory and IO bandwidth questions
  - Application dependent
- 1,000 cores breaks software -- *but so does dual core*
- **Q:** How small should a core be?



# FPGA CPUs

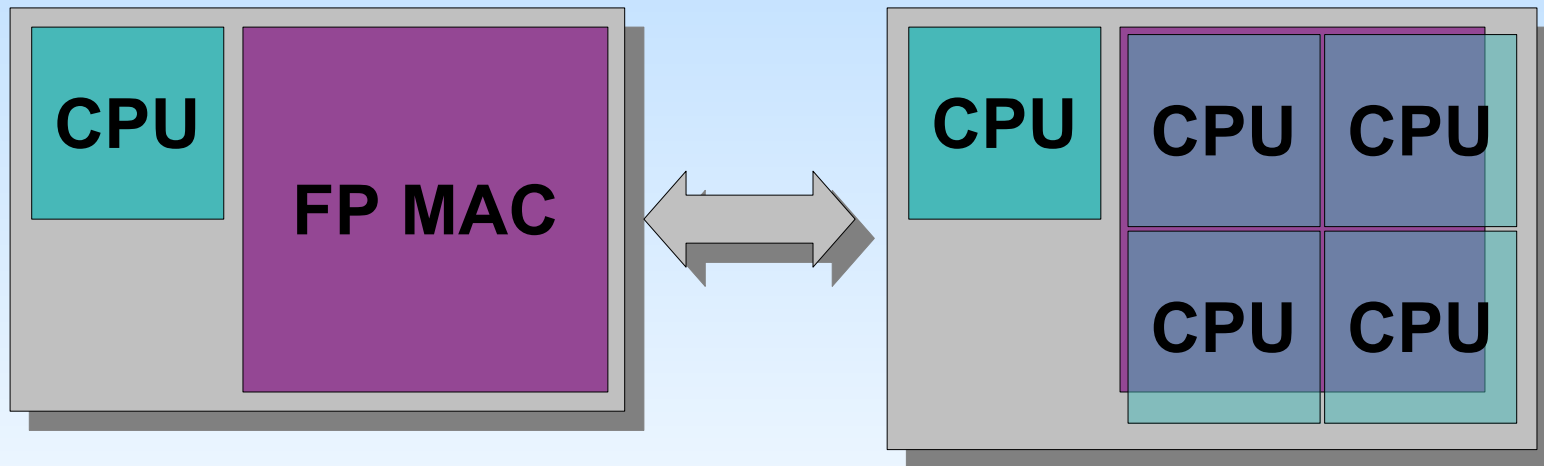
- Currently simple 1980s-style RISC CPUs
- Beginning to implement floating point
- Some multicore experimentation

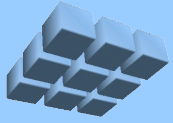
<i><b>Component</b></i>	<i><b>LUTs</b></i>
CPU	800
FP Add	1,312
FP Multiply	1,380
FP Mac	2,772
FPU support*	850



# Relative FPGA Core Sizes

- FP MAC almost 4x larger than CPU
- **Q:** CPU + MAC or 5x CPU?
- **Q:** 50x CPU + MAC or 250x CPU?

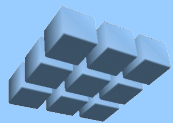




# SW vs. HW FP

- Simple floating point code
- Run on *Cmpware PowerPC* simulator
- HW FP instructions vs. soft FP emulation

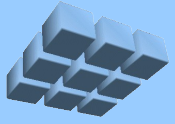
```
int main(int argc, char *argv[]) {  
    int i;  
    float a = 10.5;  
    float b = 3.25;  
    float c = 0.0;  
  
    for (i=0; i<1000; i++)  
        c = c + (a * b);  
  
} /* end main() */
```



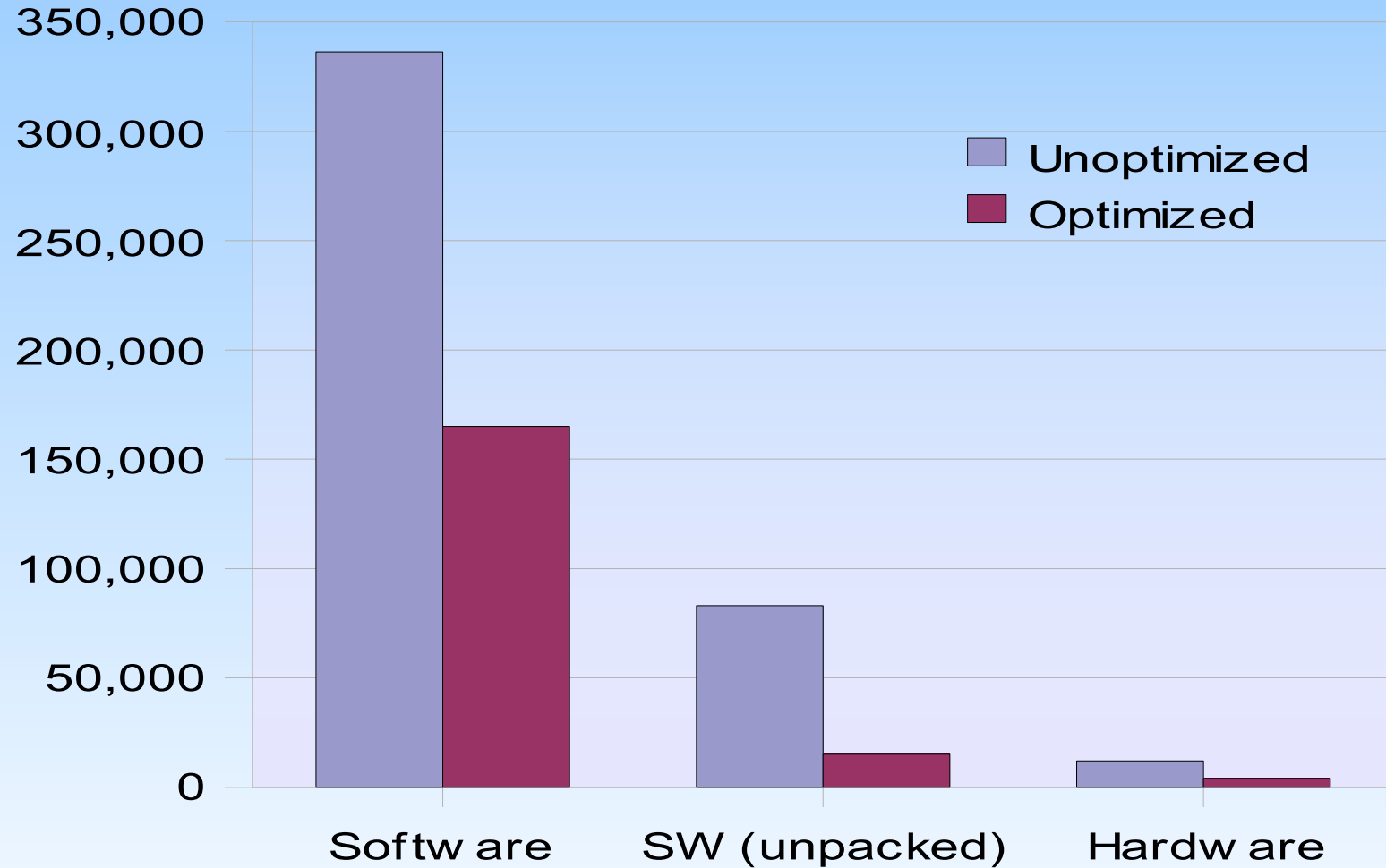
# HW vs. SW Floating Point

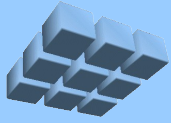
- SW emulation 30x – 40x slower than HW
- ... but most of the time spent packing / unpacking numbers into floating point format
- 'Unpacked' FP only 3.75x slower than HW

	<i>Instructions Executed</i>	<i>Instructions Executed (-O3)</i>
FP Hardware	12,024	4,015
FP SW (unpacked)	83,026	15,073
FP Software	336,254	165,010



# HW vs. SW Floating Point

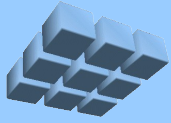




# FP HW in Multicore

- SW FP 3.75x slower FP execution than HW
- ... but permits 5x CPU cores
- Opens up new optimization opportunities
- Performance available for non-floating point applications
- Other applications likely to have an even lower mix of FP / non-FP instructions

**==> *Software FP 'wins' in multicore***



# Conclusions

- CPU + FPU 5x size of CPU
- FPU software emulation optimized to 3.75x speed of HW FPU
- FP software beats FP hardware in multicore
  - Depends on instruction mix
  - Depends on ability to parallelize application
  - Depends on I/O and system parameters
- More simple cores favored
- SW beats special purpose HW in multicore