Functional Verification of the SiCortex Multiprocessor System-on-a-Chip

Oleg Petlin, Wilson Snyder
wsnyder@wsnyder.org

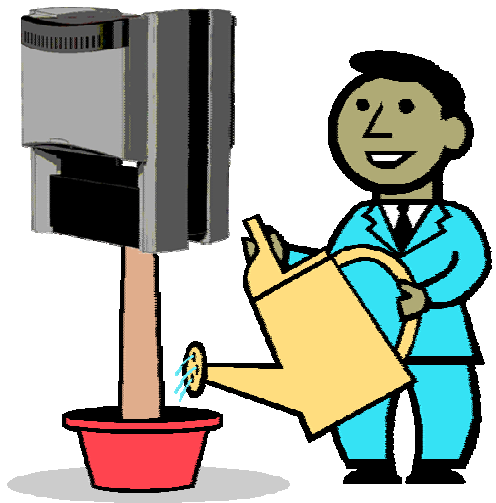
June 7, 2007

Agenda

- What we've built
- Verification challenges
- Verification productivity
- Substitution modelling
- Co-verification
- Verification statistics
- Conclusions
- Q&A

What We've Built

- Complete computer system
  - Rethought how a cluster should be built
- Custom processor chip
  - Reduced power
  - Maximized memory performance
  - Integrated high performance interconnect
- Software
  - Open Source: Linux, GNU, and MPI



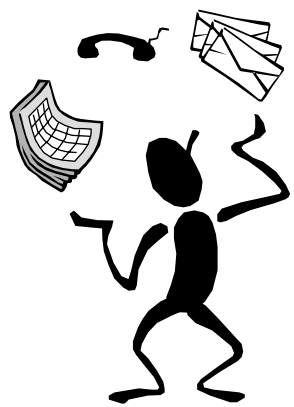
Our Product: SC5832

5832	Gigaflops
7776	Gigabytes DDR memory
972	6-core 64-bit nodes
2916	2 GByte/s fabric links
500	GByte/s bisection bandwidth
270	GByte/s PCI-E bandwidth
18	KW
1	Cabinet

There’s a small version too…

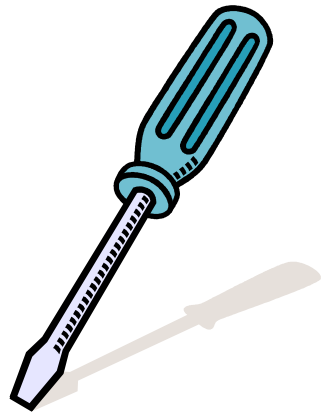
Verification Challenges

- High design complexity
  - 1.2 million lines of RTL -> 198 million transistors
- Both purchased IP and internal developments
- High programmability - 6 CPUs and DMA
- Co-verification of Linux kernel and device drivers
- Control the overall verification cost
  - Verilog simulator speed and license limitations
  - How to find more bugs for less w/o compromising the quality of verification?
  - Project deadlines



Verification Productivity

- Verification Tools
  - Languages, libraries, and simulators
    - C++, STL
    - SystemC verification library and TLM
    - OSCI SystemC
    - Cadence Incisive



  - Open source productivity tools (http://www.veripool.com)
    - Vregs (Register documentation -> headers & verification code)
    - Verilog-Mode (saves writing 30% of the Verilog lines !)
    - SystemPerl (saves writing 40% of SystemC lines !)
    - Verilator (Verilog RTL -> C++/SystemC cycle-accurate model)

Verilator and Incisive

Verilator + OSC	**Incisive**
Full SystemC	Full SystemC
Synthesizable Verilog-2005	Fully Verilog-2005 compliant
C++ Interface	PLI/VPI compliant interface
Two-State	Four-State (0,1,X,Z) and strengths
Cycle accurate	Timing accurate (thus required for PLL, PHY and gate simulations)
Limited PSL assertions	Full PSL assertions
Line and Block coverage	Block, FSM, expression coverage
Waveforms, GDB/DDD	Waveforms, source debugger
Faster simulations (2-5x)	Slower simulations
Limited support	Excellent customer support
Free	Not quite ☺

Verification Productivity (cont.)

- Code Reuse
  - C++ encapsulation, inheritance
  - Verification infrastructure
  - Test components
- Test Writing Methodology
  - Every test is a class that inherits the test base class
  - The test base class specifies the execution order for a set of virtual methods
  - Chip-level tests are constructed from subchip tests
- Regression Testing
  - Hourly, nightly and weekly runs with random seeds
  - Background random runs
  - Automated web-based reporting system (next slide)

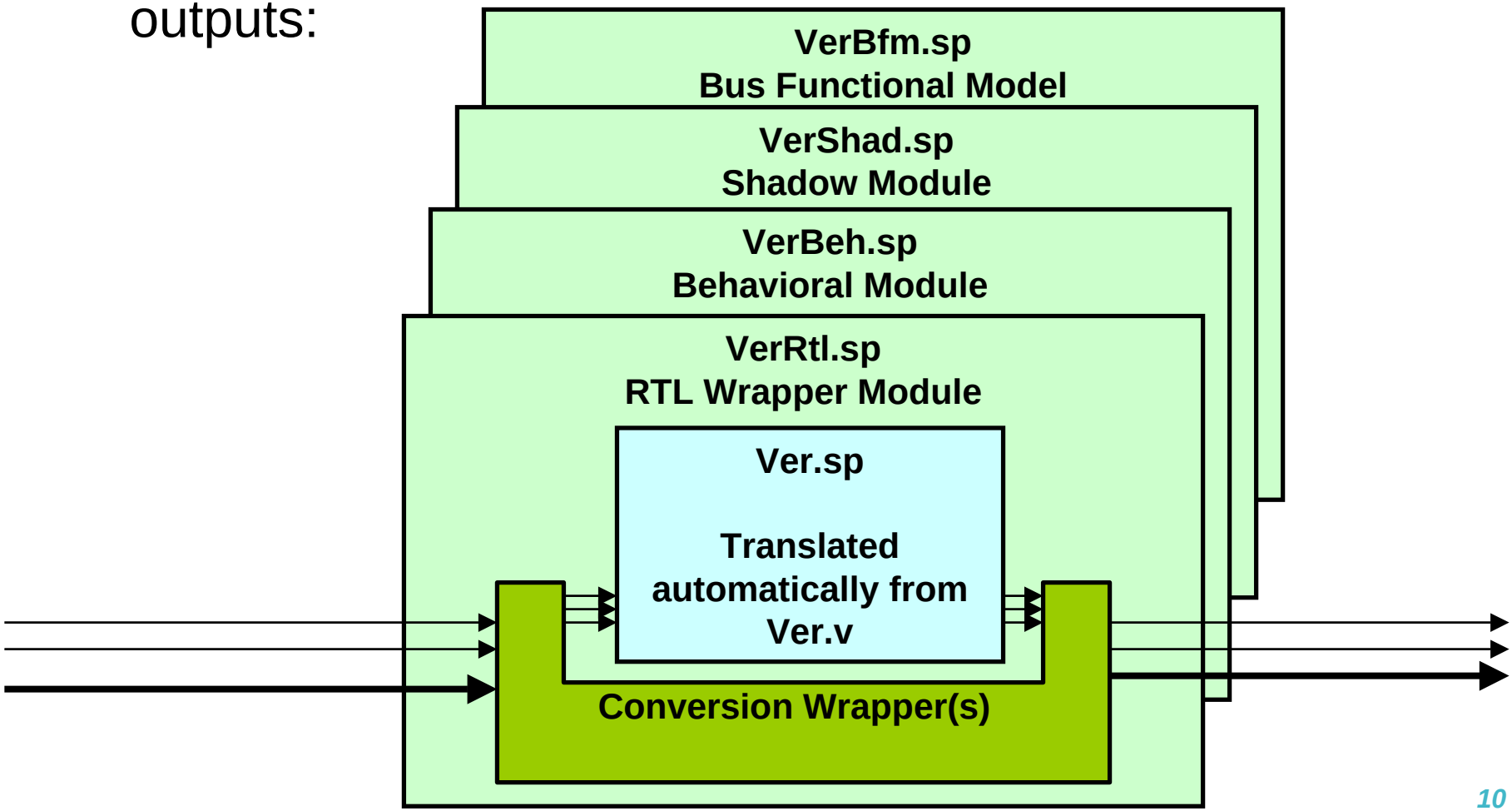
Tracking all Tests

- All tests were tracked in a database with web front-end:
  - Did this test ever work, and when?
  - What versions did the test work in?
  - What changes were made?

Rev Num	Run History	Rev User	Rev Description
r5333	1 fail	denney	DMA engine support for mandelbrot
r5332		denney	Add PCI express test
r5331	2 pass 2 fail w/mod	wsnyder	Doing something nasty
r5330			
r5329	1 pass	pholmes	New incredible MPI fabric test added

Substitution Modeling

- We allowed many types of modules to be substituted into the same consistent chip model cell, and can compare outputs:



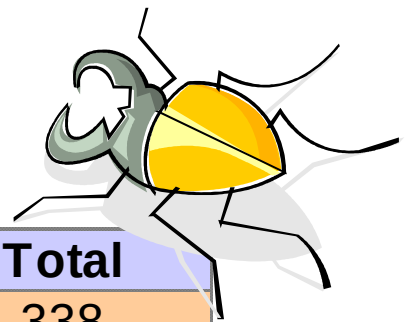
Co-Verification: Booting Linux

- Our major software goal was to boot Linux
  - Linux kernel 2.6 with some modifications
  - Initialization trimmed to 16 million instructions

Model	Simulator	Boot Time
Hardware	N/A	~1 sec
Beh	SimH	50 sec
Beh CPU + rest SystemC	OSC	13 h.
Verilated CPU + SysC	Verilator+OSC	18 h.
Verilog RTL	Cadence NCSIM	140 h.

Verification Statistics

- 20,000 tests
- 5,000+ per night (<20% of the tests required any license)
- 22,000,000 test runs over the last 12 months:
  - 230 compute years
  - 2.1 hours of “real chip” time
- 1,300 critical bugs found, as follows:



Block	HLM	RTL	Total
L2 Cache	304 (90%)	34 (10%)	338
DMA Engine	217 (82%)	47 (18%)	264
FSW Switch	158 (79%)	41 (21%)	199
PCIe-PMI	159 (84%)	30 (16%)	189
CHIP	3 (21%)	11 (79%)	14

Verification Statistics

- 20,000 tests
- 5,000+ per night (<20% of the tests required any license)
- 22,000,000 test runs over the last 12 months:
  - 230 compute years
  - 2.1 hours of “real chip” time
- 1,300 critical bugs found, as follows:

Block	
L2 Cache	
DMA Engine	
FSW Switch	
PCIe-PMI	
CHIP	

Teaser

230 compute years cost us $250K in Opteron Servers.

Our system with 648 cores delivers more than 3x the simulations per dollar, in 1/12th the space. And this isn’t even in the market it is tuned for.

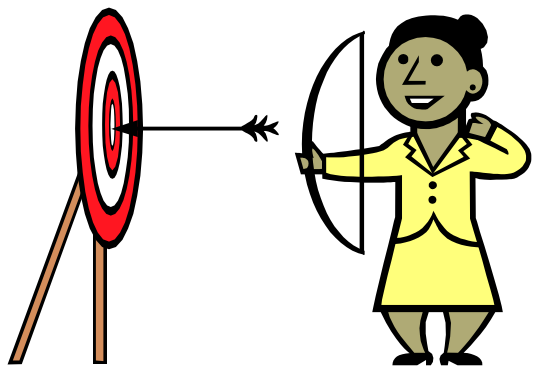
And we’d save $17,000 in power/year!

9 cents /kWH, 24KW for 86 servers drops to 2KW per SC648

How did we do?

- Initial debug went smoothly
  - Dec 28, 2006: Chips arrive
  - Jan 22, 2007: Linux, NFS, Emacs, make….
  - Jan 29, 2007: MPI networking node-to-node.

- Only a few bugs found in Silicon, all with workarounds
  - All due to verification holes
  - Filled now, of course ☺



Conclusions

- Mixing Verilog RTL and SystemC/C++ worked well
- Fast simulation models enabled early software debug
- Our strategy provided for a higher level of control over:
  - Simulation speed
  - Accuracy
  - License usage & overall verification cost
- Open source tools allowed the team to run more tests and find bugs earlier
- Used the best public domain and commercial tools each for what they do best

Public Tool Sources

- The public domain design tools we used are available at http://www.veripool.com
  - Make::Cache - Object caching for faster compiles
  - Schedule::Load – Load Balancing (ala LSF)
  - SystemPerl - /*AUTOs*/ for SystemC
  - Verilator – Compile SystemVerilog into SystemC
  - Verilog-Mode - /*AUTO…*/ Expansion
  - Verilog-Perl – Verilog preprocessor
  - Vregs – Extract register and class declarations from documentation
- The SIMH instruction set simulator is at http://simh.trailing-edge.com