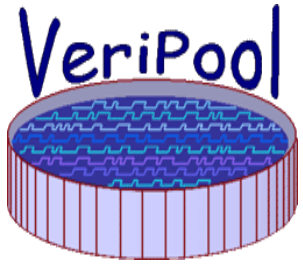


Verilator:
Open Simulation - Growing Up

http://www.veripool.org/papers

Wilson Snyder

Cavium Networks

wsnyder@wsnyder.org

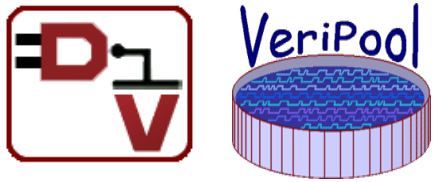
Agenda

- Tenants
- Modernities
- Practicalities
- Finalities
- Q & A

Agenda

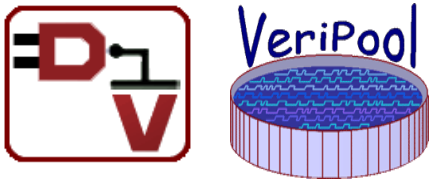
- Tenants
- Modernities
- Practicalities
- Finalities
- Q & A

Verilator Solves A Problem



- Verilator was created to solve a different problem than most simulators
  - Originally CPU design focused
  - Take a Verilog model and merge it with C++/SystemC
  - C++ owns the main loop, not the simulator
  - 100% C++ executable is major advantage
    - Enables easy cross compiling, gdb, valgrind, lots of other tools.
    - Fast simulations into MATLAB (via vmodel)
- So, Verilator compiles Verilog into C++
  - Matches synthesis rules, not simulation rules
  - Time delays ignored (a <= #{n} b;)
  - Only three state; unknowns randomized (better than Xs)

Performance

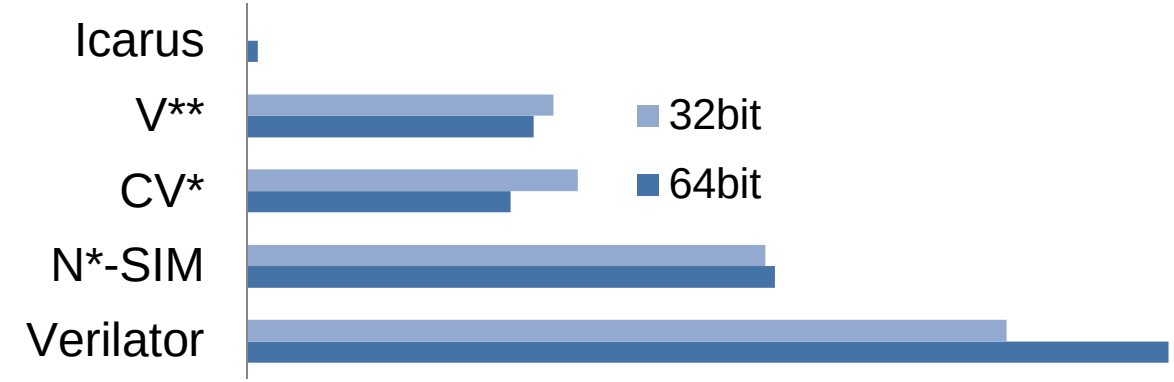


- Booting Linux on MIPS SoC

- Testbuilder-Based Unit Test

Why so close?
8% in Verilog
92% in C Test Bench
Oh well!

- Motorolla Embedded CPU



As in all benchmarks,
your mileage will vary

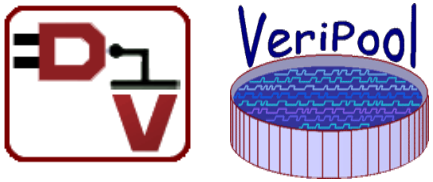
Story HIDDEN

- A lot of hiring here in Boston, interviews every few days, I ask about simulators or environments and every few weeks someone mentions and we use this open source simulator you probably haven't heard of called verilator…

Verilator User Base

All trademarks registered by respective owners.
Users based on correspondence; there is no official way to determine “users” since there’s no license!

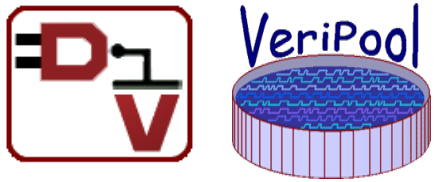
Open Source Upsides



- Cost – iff it does close to what you need
- Open License
  - Required for some applications
    - Example: NXP needed a solution they could provide to software developers, and couldn't contact a license server
    - Example: Running simulations on cloud machines
  - Stronger negotiation position when buying commercial tools

- Source Code Visibility
  - Repurposing and trying new ideas
  - Visibility into everyone's bugs and enhancement reqs
  - Do-it-yourself quick bug turn-around

Open Source Downsides

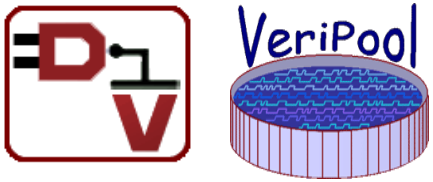


- Support – you're the first level support person
  - There's no guarantees someone else will fix your bug
  - But you could fix it – with commercial tools, you can't
  - If you feed upstream others will likely maintain it!
    - Most open source tools are this model
- Or hire a consultant offering paid support
  - Far more cost effective than a commercial tool
  - You get the exact feature you need
  - They contribute changes upstream
  - Embecosm, etc.

Agenda

- Tenants
- Birth and Exodus
- Modernities
- Practicalities
- Finalities
- Q & A

History



- Verilator born in 1994 at Digital Equipment Corp
  - Verilog was the new Synthesis Language
  - C++ was the Test-bench Language
  - No good tool existed to couple the two
  - So Paul Wasson synthesized Verilog into C++

- Verilator reached Intel as part of ARM group sale
  - Duane Galbi did the second rewrite
  - DEC released Verilator into Open Source

Agenda

- Tenants
- Modernities
- Practicalities
- Finalities
- Q & A

The Big Fixes

- Verilator adds self-tests, not only real designs
  - Now >500 focused tests, send yours!
- Verilator simplifies installation
  - Simplify requirements – e.g. SystemPerl removed (2010)
- Verilator goes multi-platform
  - Linux, Debian, Apple OS, OpenSolaris, Cygwin, MS, GCC, MSVC++, MingGW, …
  - All beyond Linux are by contributors (Thanks!)
- Verilator may be user's only simulator
  - Continual improvement of error messages etc.
  - Added Lint features (2007, 2011)

The Big Fixes (2)

- Verilator parses SystemVerilog
  - SystemVerilog is >6x as complicated as Verilog 2001
  - Different techniques have gotten us to 95% of SV

- Verilator adds complex data types
  - Originally only Verilog array-of-bits
  - Signed numbers (2005), Little endian (2009)
  - SystemVerilog required full data types
    - Data types on each “node” (2012)
  - Many parts still need improvement
    - VCD for SV constructs undefined by IEEE (complain!)

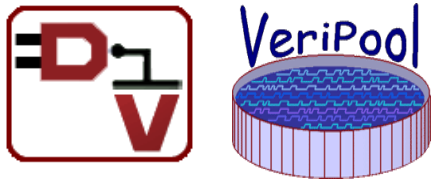
The Big Fixes (3)


- Verilator adds verification constructs
  - Some verification constructs always creep in
  - Dotted references (2006)
  - DPI/VPI in (2010)
  - Model save/restore (2012)
  - Someday everything?
- Verilator needs some event-based simulation
  - Clock gating (2005)
  - Someday full events?
- Verilator VHDL
  - In development!

SystemVerilog Additions

- Preprocessing 100% compliant (2009)
- Byte, chandle, int, longint…, var, void (2010)
- Packages and imports (2010-12)
- Typedefs and enums (2010)
- Packed structures (2012)
- Operator short circuiting (2012)
- More support as contributions continue…

Future Performance



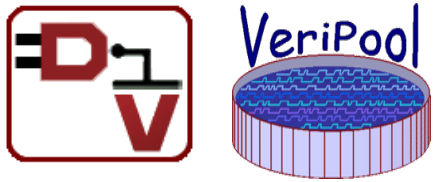
- De-replicate large structures and duplicate logic
- Multithreaded execution & GPUs
  - Hard to avoid communication bottlenecks
  - GPUs – though not great at integer code
  - Some PhD interest, but no patches yet!
- Optimize Caches
  - Most models are load/store limited
  - On large designs, smaller code footprint is faster

TIPS:
1. Buy CPUs with the largest caches you can get, they are generally well worth the premium for ALL simulators.
2. Try using GCC –Os to optimize for size.

Agenda

- Tenants
- Modernities
- Practicalities
- Finalities
- Q & A

Getting Started



- Download and install
  - Globally: RPMs - Thanks, RPM packagers!
  - or Globally: Download

```
./configure ; make ; make install
```

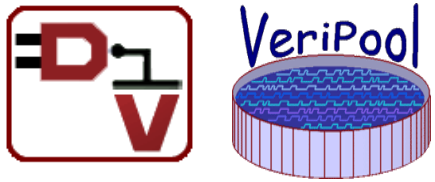
  - or Cad-tool-ish with multiple versions and env var

```
./configure ; make
setenv VERILATOR_ROOT `pwd`
$VERILATOR_ROOT/bin/verilator …
```

- Follow example in "**`verilator –help`**"
  - Create test Verilog file, top level C wrapper, Verilate, compile and run
- Simple run to see warnings

```
verilator –lint-only –f input.vc top.v
```

Example Translation



- Verilog top module becomes a Class.
  - Lower modules are inlined and generally opaque
- Inputs and outputs map directly to bool, uint32_t, uint64_t, or array of uint32_t's

```
module Convert;
   input clk
   input [31:0] data;
   output [31:0] out;

   always @ (posedge clk)
      out <= data;
endmodule
```



```
#include "verilated.h"

class Convert {
   bool      clk;
   uint32_t  data;
   uint32_t  out;

   void eval();
}
```

Calling the model

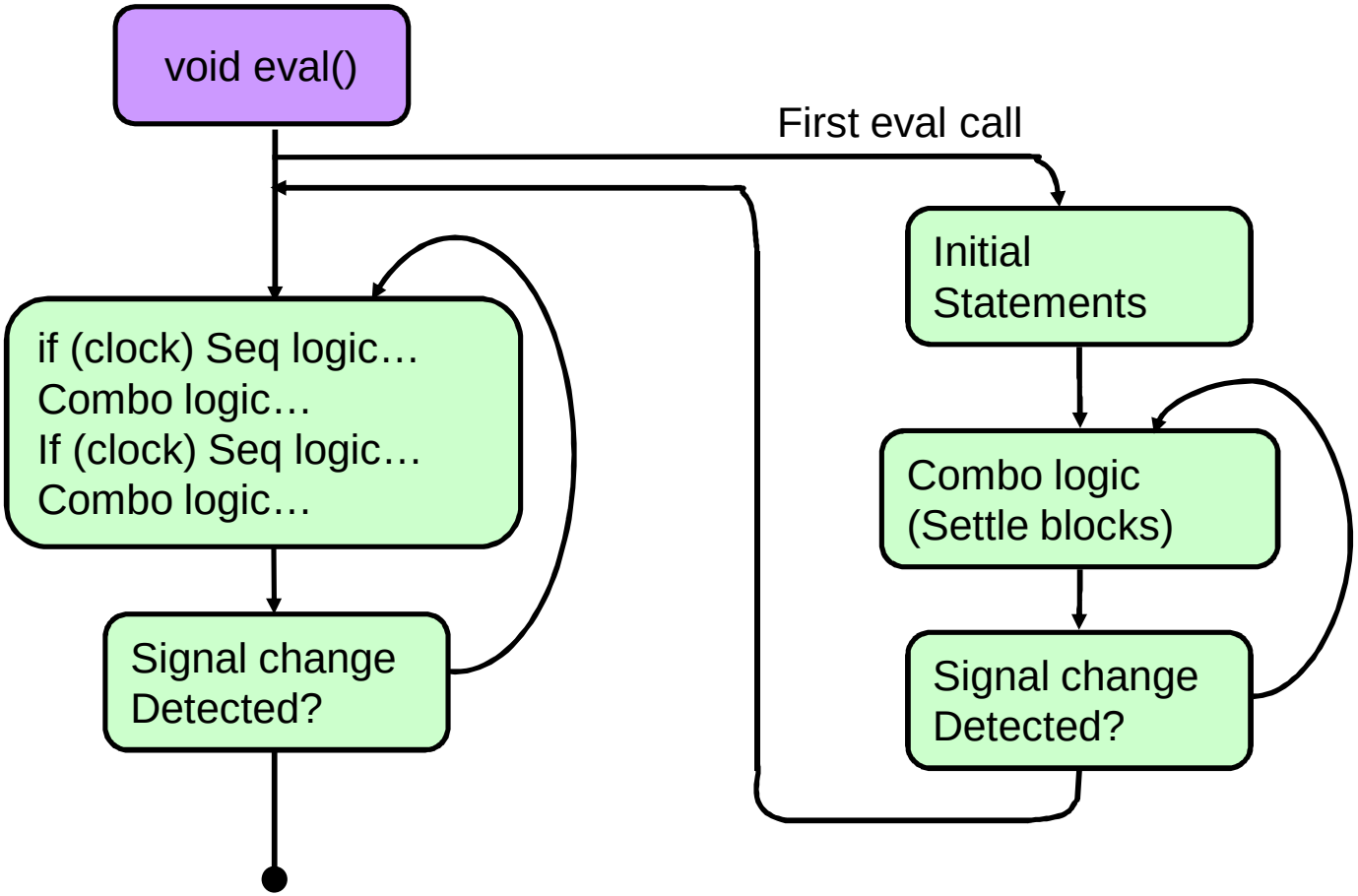
- Application calls the Verilated class in a loop
  - Verilator doesn't make time pass!
    - The key difference from most simulators

```
int main() {
    Convert* top = new Convert();
    while (!Verilated::gotFinish()) {
        top->data = …;
        top->clk = !top->clk;

        top->eval();

        … = top->out();

        time++;  // Advance time…
    }
    top->final();
}
```

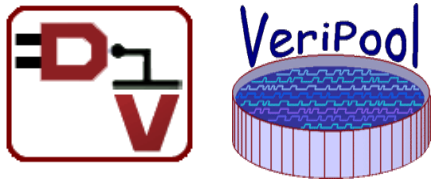
```
class Convert {
   bool      clk;
   uint32_t  data;
   uint32_t  out;

   void eval();
}
```

Verilated Internals

Inside the generated model are two major loops, the settle loop called at initialization time, and the main change loop.



A Bug, Oh no! (and Sorry!)

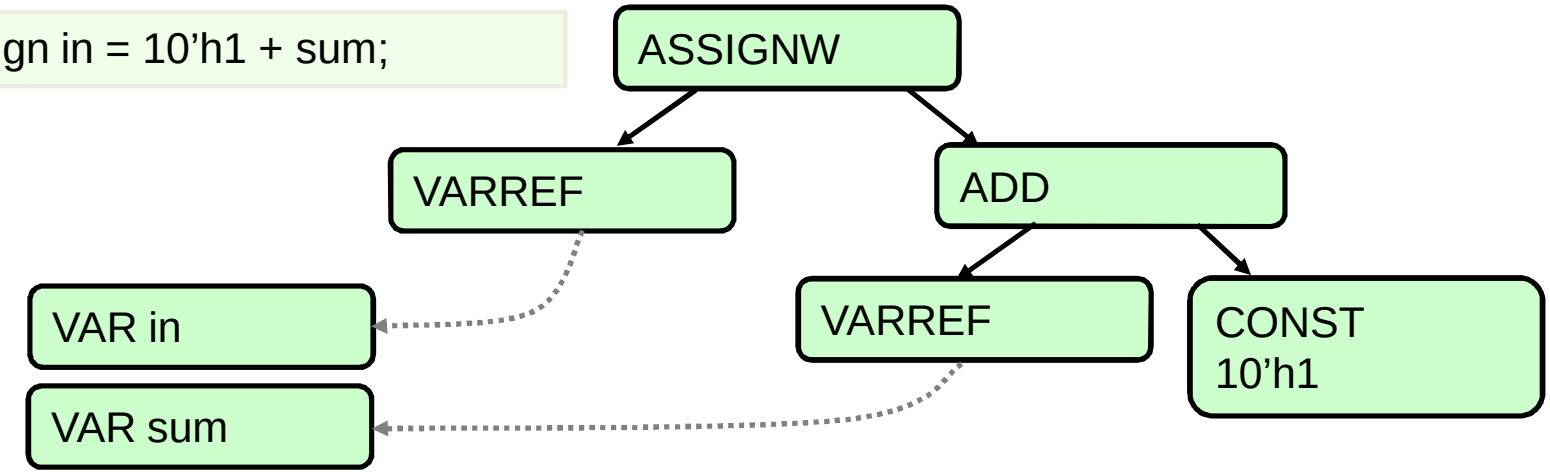


- Try to fix warnings
  - Verilator gets limited testing of warning-disabled cases
  - Last week, user got 2.5x speedup by fixing warning
- Create a test case (see Verilator manpage)
  - Please, please use “test_regress/t/t_EXAMPLE.v”
  - Tests cases can be more important than the code itself
- Check it works on another simulator
  - **`test_regress/t/t_EXAMPLE.pl -vcs/iverilog/nc`**
- Run with –debug
  - **`test_regress/t/t_EXAMPLE.pl -debug`**
- Submit test to Veripool.org and try to fix

AstNode

The core internal structure is an AstNode

```
assign in = 10'h1 + sum;
```



If you run with –debug, you'll see this in a .tree file:

```
1:2: ASSIGNW 0xa097 <e1312> {e29} @dt=0x9
1:2:1: ADD 0xa098 <e774> {e29} @dt=0x9
1:2:1:1: VARREF 0xa099 <e781> {e29} @dt=0x9b0@(w10)  in [RV] <- VAR in
1:2:1:2: CONST 0xa09a <e1556> {e29} @dt=0x9b0@(w10)  10'h0
1:2:2: VARREF 0xa09c <e754> {e29} @dt=0x9b0@(w10)  sum [LV] => VAR sum
```

Source Code Line Number

Data type (10 bits)

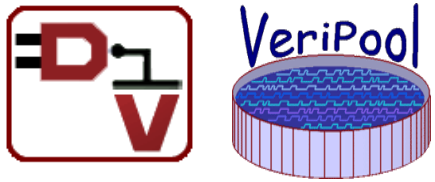
Node Address

LV indicates an lvalue – it sets the variable.

Agenda

- Tenants
- Modernities
- Practicalities
- Finalities
- Q & A

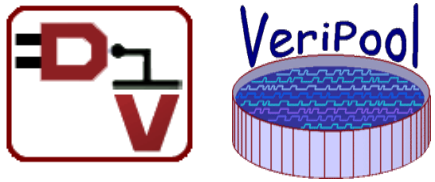
Verilog-Perl Toolbox



- Code shared with Verilator
  - Nearly identical preprocessor
  - Superset of lexical analysis and parser
  - Parses 95% of SystemVerilog 2009
- Vhier
  - Print design hierarchy, input files, etc
- Vppreproc
  - Complete 2009 preprocessor
- Vrename
  - Rename and xref signals across many files

# To	From	Filenames
“a_new”	“a”	“MyMod.v”
“b”	“b”	“MyMod.v”

Verilog-Mode for Emacs



- Thousands of users, including most IP houses
- Fewer lines of code to edit means fewer bugs
- Indents code correctly, too
- Not a preprocessor,
  code is always “valid” Verilog
- ★ Automatically injectable
  into older code.

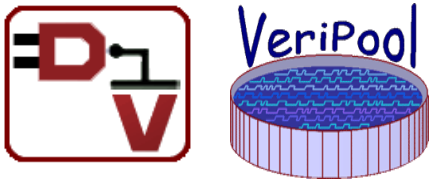
```
  …
/*AUTOLOGIC*/

a a (/*AUTOINST*/);
```

GNU Emacs (Verilog-Mode)

```
/*AUTOLOGIC*/
// Beginning of autos
logic [1:0] bus; // From a,b
logic       y;   // From b
mytype_t    z;   // From a
// End of automatics

a a (/*AUTOINST*/
     // Outputs
     .bus   (bus[0]),
     .z     (z));
```

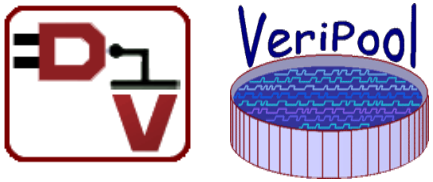
GNU Emacs (Verilog-Mode)

Conclusions



- Why adopt Verilator?
  - Supported
    - Continual language improvements
    - Growing support network for 19 years
    - Run as fast as major simulators
  - Open Source Helps You
    - Easy to run on laptops or SW developer machines
    - Get bug fixes in minutes rather than months
    - Greatly aids commercial license negotiation
  - Keep your Commercial Simulators
    - SystemVerilog Verification
    - Run analog models, gate SDF delay models, etc
    - Reference for signoff
  - 90% of money you would spend on licenses can go instead to computes
    - 2-10x more simulations/$

Contributing Back



- The value is in the Community!
- Use Forums
  - If just to tell us you're using it
- Use Bug Reporting
  - Even if to say what changes you'd like to see
- Try to submit a patch yourself
  - Many problems take only a few hours to resolve yourself; often less time than packaging up a test case for an EDA company!
  - Even if just documentation fixes!
- Advocate

Sources

- Verilator and open source design tools at http://www.veripool.org
  - Downloads
  - Bug Reporting
  - User Forums
  - News & Mailing Lists
  - These slides at http://www.veripool.org/papers/