Verilator 4.0:
Open Simulation Goes Multithreaded

Wilson Snyder
Veripool.org
wsnyder@wsnyder.org
Agenda

• Introduction to Verilator
• Version 4
• Multithreading
• Benchmarking
• Conclusion
• Q & A (Please hold questions until end)
Introduction to Verilator
Verilator is a Compiler

- Verilator compiles synthesizable Verilog into C++
  - Matches synthesis rules, not simulation rules
  - Time delays ignored (a <= #\{n\} b;)
  - Only two state simulation (and tri-state busses)
  - Unknowns are randomized (better than Xs)

- Creates C++/SystemC wrapper

- Creates own internal interconnect
  - Plays several tricks to get good, fast code
Example: Linting a Verilog Module

- Lint check your code

```
verilator -lint-only -Wall Convert.v
```

(Verilator is the only open-source Verilog Lint tool known to the presenter)

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

    initial $display("Hello flip-flop");
    always_ff @ (posedge clk)
        out = data;
endmodule
```

%Warning-BLKSEQ, Blocking assignments (=) in sequential (flop or latch) block; suggest delayed assignments (<=).
Example: Translation

- Translate to a C++ class (also can do SystemC module – not shown)

```cpp
#include "verilated.h"

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

    void eval();
    void final();
}
```

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

    initial $display("Hello flip-flop");
    always_ff @ (posedge clk)
        out <= data;
endmodule
```

- Verilog top module becomes a C++ class (obj_dir/VConvert.h)
- Inputs and outputs map directly to basic C++ types (not objects)
Verilator in Standard IP

- From the world’s most popular cellphone CPU IP:
  ```verilog
  reg clk_en_lat /*verilator clock_enable*/;
  ```
  - Verilation works “out-of-the-box”
  - Thanks!

- Supported by many open-source HW IPs
  - Thanks!

- Under common use internally in other major CPU design centers
  - Thanks!
Verilator Version 4
Thanks to all whom have contributed


Thanks to them, and any we've missed including, and any who whished to remain anonymous.
New in v4

• Single threaded performance enhancements

• Standard runtime option parsing

• Natively write FST files for GTKWave viewing

• Multithreading (later slides)
Multithreading
Multithreading Algorithm Begins

Begin with graph of millions of statement-level “micro tasks”...

\[ \text{Verilator 4.0: Open Source Goes Multithreaded: (ORConf 2018)} \]
Coarsen the DAG

Color nodes into tens of “Macro Tasks”
Merge micro tasks, only macro tasks remain

Merge micro tasks so only macro tasks remain

Typical result: millions of micro-tasks merge into tens of macro-tasks

Runtime synchronization becomes affordable
Schedule macro tasks onto threads

Schedule macro tasks onto threads

• Static assignment, using guess of runtime

• Future improvement to reassign dynamically using feedback

• Every-cycle dynamic runtime assignment was tried but slow! Future research area
Benchmarking
Benchmarks

- Packet Architects 1.6 Tbit Router IP
  - (Thanks to them for allowing publication of this data)

- Verilog compiled out-of-the-box
  - No specialized tuning, no hints

- Your mileage will vary…
Verilator Performance vs. Big 3

Better!

"Big 3" A

"Big 3" B

Verilator v3.926

Verilator v4.000

Verilator 4 threads

Verilator 4.0: Open Source Goes Multithreaded: (ORConf 2018)
Verilator Performance vs. Big 3

But can Verilator do better?

"Big 3" A  "Big 3" B  Verilator v3.926

Better!
Verilator Performance vs. Big 3

Better!

"Big 3" A  "Big 3" B  Verilator v3.926  Verilator v4.000

1.3x

But can Verilator do better?
Verilator Performance vs. Big 3

Verilator v3.926
Verilator v4.000

Verilator 4 threads

Better!
Verilator Performance vs. Big 3

But can Verilator do better?

1.9x on 4 threads
$ lscpu
Architecture:          x86_64
CPU(s):                8
Socket(s):             2
L1d cache:             32K
L1i cache:             32K
L2 cache:              1024K
L3 cache:              16896K
NUMA node0 CPU(s):     0,2,4,6
NUMA node1 CPU(s):     1,3,5,7

Got 4 threads but crossing sockets

$ numactl –C 0,2,4,6 sim_executable

Want 4 threads on same socket

Similar issue may exist getting two hyperthreaded CPUs on same physical CPU. Thus always use numactl.
Verilator Performance vs. Big 3

Better!

"Big 3" A  "Big 3" B  Verilator v3.926  Verilator v4.000  Verilator 4 threads C0,1,2,3  Verilator 4 threads C0,2,4,6

2.5x  1.27x

Verilator 4.0: Open Source Goes Multithreaded: (ORConf 2018)
Verilator Multithreaded Performance on 48 core machine

- 1 thread
- 2 threads
- 4 threads
- 6 threads
- 12 threads
- 24 threads
- 48 threads

5.5x on 12 threads = 46% scaling

Cross-socket penalty
Verilator Gantt Report (4 threads)

Analysis:
- Total threads = 4
- Total mtasks = 45
- Total eval time = 2021242 rdtsc ticks
- Longest mtask time = 614664 rdtsc ticks
- All-thread mtask time = 4212854 rdtsc ticks
- Longest-thread efficiency = 30.4%
- All-thread efficiency = 52.1%
- All-thread speedup = 2.1

Long period of single thread active; Mtask 20 is taking too long. Area for future improvements/tuning

# Threads busy
Conclusion
Conclusion: Adopt Verilator

• Supported
  – Continual language improvements
  – Growing support network for 20+ years
  – Run faster than 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, analog models, gate SDF, etc.
Contributing Back

• The value of Open Source is in the Community!
• Use Forums and Bug Reporting
• 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!
  – Great experience for the resume!
• Advocate
Also at Veripool: 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.
Sources

- Verilator and open source design tools at [http://www.veripool.org](http://www.veripool.org)
  - Downloads
  - Bug Reporting
  - User Forums
  - News (add yourself as a watcher to see releases)
  - These slides at [http://www.veripool.org/papers/](http://www.veripool.org/papers/)